FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Biometric recognition based on the texture along palmprint principal lines

Diogo Santos Martins

Mestrado Integrado em Bioengenharia

Supervisor: Prof. Jaime Cardoso

Co-supervisor: Filipe Magalhães

July, 2011

Abstract

Palmprint recognition has been in the focus of biometric research over the last ten years. Here, a biometric system for palmprint recognition based in the texture of palmprint principal lines is proposed.

Two novel methods for detection of principal lines using graph search algorithms are developed. One of the schemes performs particularly well at detected fully connected lines, which is an advantage over other methods that only employ edge detection algorithms.

The lines detected with the developed scheme are used to extract line textural information using Haralick's features. The concept of texton dictionary is used to find representative textures in principal lines. A palmprint is then represented by the histogram of texton frequencies. The performance of the method is inferior to other documented procedures but, considering that no spacial information is present in histogram representation, and that each histogram contains a small amount of data, the achieved accuracy of 85% highlights the importance of texture in palmprint recognition.

The implemented methods are innovative in palmprint recognition and constitute a framework for future approaches in this field.

Acknowledgements

During the last months, a lot of people contributed in one way or another to this thesis.

I'm specially grateful to my co-supervisor, Filipe Magalhães, for all the help, support and contributions provided during the last months, but most importantly, for his commitment to this project. I want to thank to my supervisor, Prof. Jaime Cardoso, for all the brilliant ideas that shaped this thesis and for accepting me as a student. A special appreciation to Prof. Alexandre Quintanilha for all the incentives and for allowing me to learn science where it crosses its borders.

I have to thank everyone at INESC Porto for being excellent work-mates and providing a great atmosphere.

I really appreciate all the support from my mother, my father, my sister and my grandparents for being with me in the hard times. Without them I would not be writing this now.

A special thank goes to Ana, Joana, Rita and Vera, for company and inspiration in long days and nights of work.

Diogo Martins

I know this thesis like the palm of my hand.
But... oops... didn't know I had this wrinkle here...

Contents

Abbreviations

DoG	Difference of Gaussians
EER	Equal Error Rate
FCM	Fuzzy C-Means
FAR	False Accept Rate
FRR	False Reject Rate
GAR	Genuine Accept Rate
GLCM	Grey Level Co-occurrence Matrix
HMM	Hidden Markov Model
ICA	Independent Component Analysis
LBP	Local Binary Pattern
LDA	Linear Discriminant Analysis
NN	Neural Network
PCA	Principal Component Analysis
ROC	Receiver Operator Characteristic
ROI	Region of Interest
SVM	Support Vector Machine

ABBREVIATIONS

Chapter 1

Introduction

1.1 Overview of Biometrics

Studies of physical and behavioural traits for recognition purposes are known as biometrics. Being specific of an individual, they guarantee one's identity in security control situations. A well known example of a biometric characteristic are fingerprints, being most widely used [1]. It is theoretically impossible to find any two individuals with the same fingerprint [2]. This is a crucial property of a biometric characteristic: to be unique for each person. Other equally important aspects regarding biometric characteristics are universality, as they have to be present in all individuals; and permanence, so they are constant during one's life. Moreover, they should be easy to extract. At present time, there is research activity in a broad range of biometric characteristics which can be divided into physical and behavioural. Physical are, for instance, fingerprints, iris, retinal capillary structure, face, and hand recognition. Examples of behavioural traits are voice and handwriting. Figure 1.1 illustrates several biometric characteristics.

Biometric systems can be used for identification and verification purposes. In all cases there should be a database where biometric features from a set of individuals are stored. In an identification task, the role of the system is to compare an input with all the entries in the database and verify if there is a match, thus detecting the presence of the individual in the database. In a verification task, the algorithm checks if an individual is whom he claims to be. To compare any kind of biometric characteristics it is necessary to represent them in a stable fashion. For instance, it is not feasible to directly compare images from two palmprints, as it is practically impossible to place the hand in the exact same position in different occasions, producing slightly different images that have to be compared in some way. This is the most crucial aspect and can be divided into two tasks:

1. Represent a characteristic trait in reproducible and stable features that resist input variability.

2. Compare such features so users can accurately be recognized.

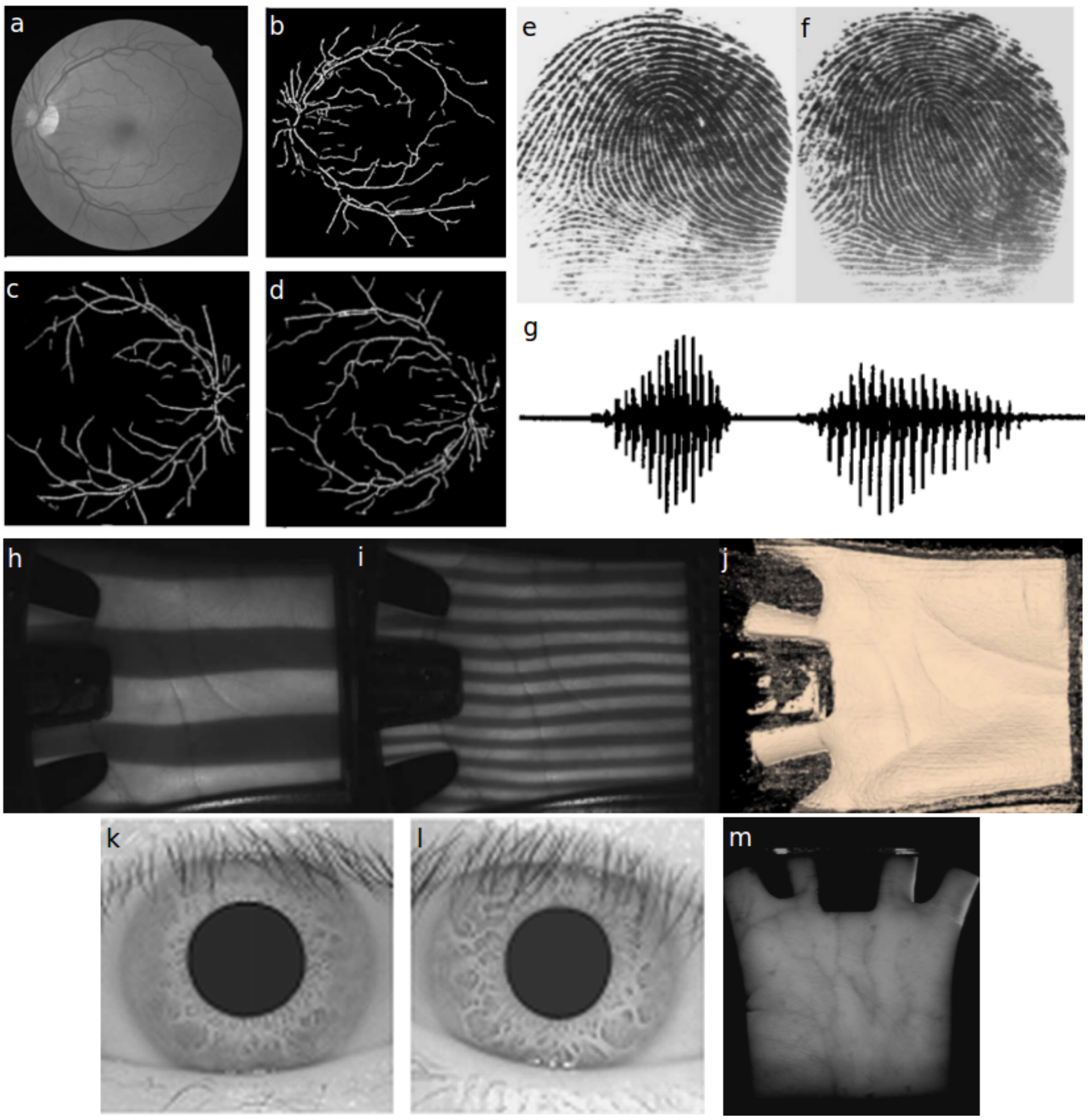

Figure 1.1: **Biometric characteristics.** **a)** Retinal fundus image and **b)** correspondent vascularization. **c)** and **d)** are examples of retinal processed images from different persons. All retinal images were taken from ref. [3]. **e)** and **f)** are fingerprints from twin sisters [4], with noticeable differences to the naked eye. **g)** represents the pressure-time plot of the /eda/ utterance (complete unit of speech) [5], from which spectral information can be retrieved to identify a speaker. **h)** and **i)** depict the illumination system from which 3D palmprint information **j)** is retrieved [6]. **k)** and **l)** are iris from two different persons [7]. **m)** depicts detection of palm veins using infra-red lighting [8].

These two questions are in the core of a biometric system and are addressed by most of the research in the field. Its importance is highlighted in figure 1.2 where the layout of a biometric system is depicted.

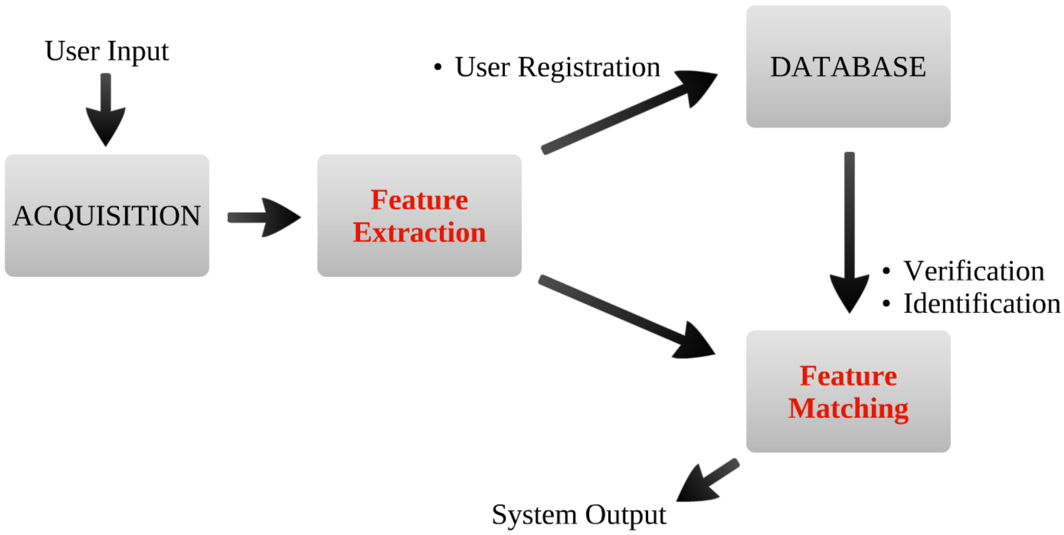

Figure 1.2: **Typical scheme of a biometric system.**

1.2 Palmprint Biometric Recognition

1.2.1 Motivation

During the last years there has been an increasing use of automatic personal recognition systems. Palmprint based biometric approaches have been intensively developed over the last 12 years because they possess several advantages over other systems.

Palmprint images can be acquired with low resolution cameras and scanners and still have enough information to achieve good recognition rates. In this case, the discriminant informations relies in palm lines and texture. However, if high resolution images are captured, ridges and wrinkles can be detected, resulting in an image similar to fingerprints. Forensic applications on latent palmprints typically require high resolution imaging, with at least 500 dpi [9].

According to the classification in [10], palmprints are one of the four biometric modalities possessing all of the following properties:

- universality, which means the characteristic should be present in all individuals;
- uniqueness, as the characteristic has to be unique to each individual;

- Permanence: its resistance to aging;
- Measurability: how easy is to acquire image or signal from the individual;
- Performance: how good it is at recognizing and identifying individuals;
- Acceptability: the population must be willing to provide the characteristic;
- Circumvention: how easily can it be forged.

The other three modalities are fingerprints, hand vein and ear canal. For instance, iris based methods, which are the most reliable, require more expensive acquisition systems than palmprint systems. Face and voice characteristics are easier to acquire than palmprints, but they are not so reliable. Overall, palmprint based systems are well balanced in terms of cost and performance.

1.2.2 Objectives and contribution

The main objective in this thesis is to build a full recognition system, from the pre-processing to the classification phase. It is divided in two main tasks. First, principal line detection, and then, recognition based in textural information of principal lines.

In the principal line detection part, two methods are developed using graph search methods. To our knowledge, graph search line detection is on of the more advanced methods employed in this field. The developed methods can be easily improved and therefore constitute a framework for future work in palm line detection.

In the recognition part, Haralick's features are used for the first time in palmprint recognition. The texture feature extraction method, along with the concept of texton dictionary open a new branch in feature extraction for palmprint recognition purposes.

1.3 Description of Contents

Here a brief summary of the structure of this thesis is presented.

In chapter 2 — State of the Art — a review of palmprint feature extraction and matching in the last decade is presented. The main approaches are discussed and insight about directions for future work is given.

The pre-processing steps used in this work are explained in chapter 3. These are noise filtering and extraction of a region of interest.

One of the major components is the development of line detection methods. Chapter 4 — Principal Lines Detection — describes two different implementations using a graph search approach.

In chapter 5 a method to compare palmprints based in texture of principal lines is developed.

Finally, chapter 6 draws the main conclusions from the work developed in this thesis.

Chapter 2

State of the Art

Comparing two palmprints requires the extraction of useful information that is ideally independent of acquisition conditions, such as hand positioning, palm ageing, illumination and dirt. This process is called feature extraction. Then, to assess one's identity, these features will be compared in a process called feature matching. Most of the research is devoted to these two phases of a biometric system.

Following this research trend, this thesis does not scope the acquisition devices and the operational performance of the palmprint recognition system. It is focused on the algorithmic component of the process. In this chapter, a review of the state of the art in feature extraction and feature matching is presented. Methods used to evaluate performance of different algorithms are also discussed.

The way research is presented in this chapter is somewhat different from a recent review [11], in which a palmprint recognition method is classified into a group according to one of the image processing algorithms used. However, with such scheme it is impossible to organize all methods for palmprint recognition without overlapping the classification, because in each method many techniques are used, and new work often arises from new combinations of techniques previously used.

Here, different techniques are categorized according to the stage of the process in which they are used. First is the preprocessing stage which involves creation of a coordinate system to align palmprint images. Next, is feature extraction, the stage that contemplates more variety in the techniques and methods used. Then, is feature reduction, which is a form of selection of extracted features. Finally, some aspects of the classification stage are presented.

Different algorithms arise from different combinations in each of these stages. Examples are given for each technique, contextualizing the methods used in every stage so the reader can understand the importance of a given stage in the full process. References are often repeated because they serve as example in more than one stage.

2.1 Local *vs* global features

This is a core characteristic of a palmprint recognition methods. To associate extracted features with specific locations in the palm it is necessary to establish a coordinate system to enable valid comparisons between different images. While almost all methods employ local specific features, there were some implementations using global features.

2.1.1 Global features

These are methods that code information retrieved from the whole palmprint at once. Therefore, no spatial information is used, and extracted features are related to the whole palm. Research using global statistical features was short because it compromises performance by discarding spatial information. Such methods can discover what features a palm has, but not where those features are located in the palm.

Examples:

In [12] palmprint images are converted to three wavelet domains, which are sensitive to different orientations, therefore including information about line orientation. Then, features of wavelet sub-bands such as sparseness and energy are used to describe a palmprint. Verification is performed by calculating the difference between features from two palms. A weighted distance scheme was developed for this task.

Invariant Zernike Moments were used by Pang et al. [13] to describe a palmprint. They compare vectors with moments of different orders using euclidean and Norm 1 distances. Higher order moments have more information for personal recognition because they relate to finer details. Li et al. used a Modular Neural Network as a classifier for the same features, instead of simple Norm 1 or euclidean distances [14].

2.1.2 Local features and coordinate systems

Methods including feature vectors associated with spatial location exhibit superior performance. Most of the systems create a square in the middle of the palm [11]. In most cases, the coordinate system is used to extract the region of interest (ROI), and subsequent feature extraction is performed on the ROI. If the coordinate system is well defined, ROI's from different images are aligned — correspond to the same area in the palmprint — and comparison between feature vectors is meaningful with regard to spatial information.

Relevant approaches:

A common approach is to use finger valleys as reference points. Typically the used valleys used are between index finger and mid-finger and between ring finger and last finger. There are numerous approaches to detect such points [15, 16, 17, 18, 19].

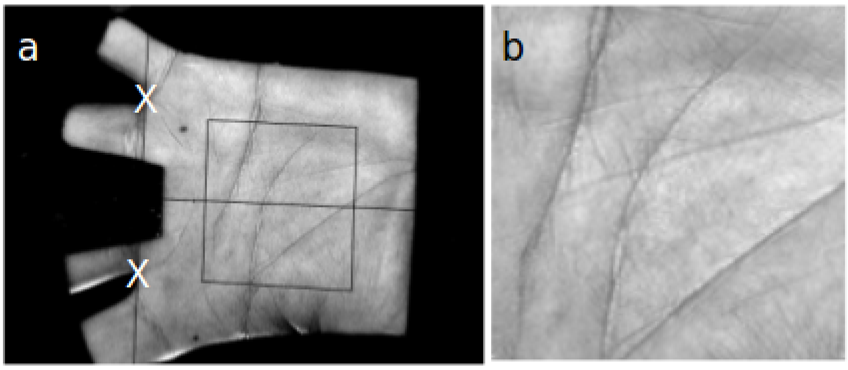

Figure 2.1: **Coordinate system based in reference points in finger valleys.** In **a)**, finger valleys are used as reference points (white X marks), on which a coordinate system is established. The resulting region of interest is depicted in **b)**. Adapted from ref. [15].

In figure 2.1, a squared ROI is extracted based on finger valley reference points. It is defined in such a way that it is centered on the line that crosses the two points. This system is similar in several methods, as reviewed in [11].

The work in [20] defines subregions on a ROI, and elements in the feature vector are associated to the subregions. Of course, the ROI position depends on reference points, which are the three finger valleys between all fingers but the thumb and index fingers. These subregions are based on ellipses (figure 2.2).

A different approach is used in [21], where features are retrieved in concentric circles.

In [22], authors use datum points (characteristic points) in principal lines to define a coordinate system and align images. This was implemented on off-line palmprint images, which are inked and printed on paper, and it's out of date. However, the concept of using lines as a reference for coordinate systems was proposed for on-line palmprint images [23]. This method uses points located on palm lines as reference, which is more suited to account for skin stretching. Methods that use finger valleys as reference are more subject to skin-stretching associated errors. However, there is not enough research yet using this method to support its theory.

2.2 Feature extraction

2.2.1 Line associated features

Visually, lines are the most evident characteristic of palmprints. If one is asked whether two palmprints belong to the same individual, the most natural comparison is made through the position and aspect of lines. Because of their importance, some of the methods developed use edge detectors for feature extraction.

Some of the best performing methods extract features strongly related to line orientation. This is probably one of the most stable features regarding variant conditions at acquisition time. Some examples of these methodologies are detailed below.

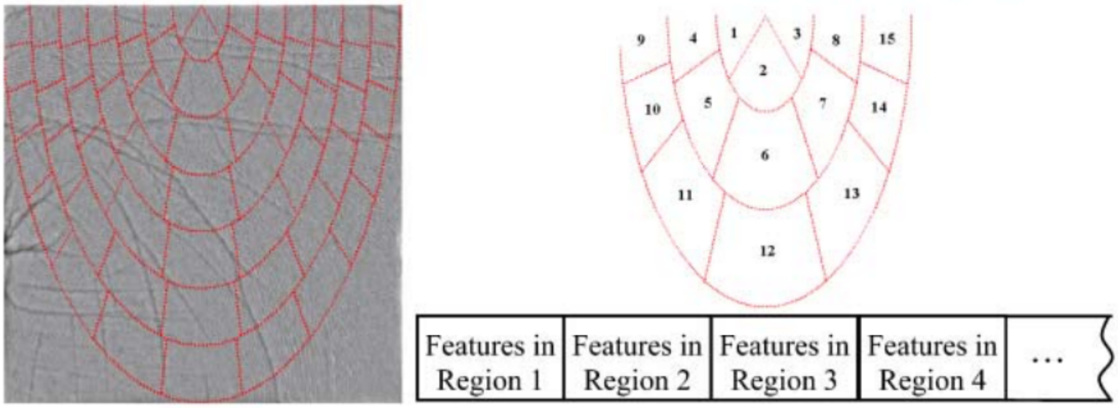

Figure 2.2: **Elliptical coordinate system.** In **a)**, an elliptical coordinate system is defined inside a region of interest. Values in the feature vector depicted in **b)** correspond to the subregions defined in **a)**. Adapted from ref. [20].

Examples:

In [19], sobel operators are used to detect lines. Summation of sobel response over columns and rows originates two histograms, which are used for classification by Hidden Markov Models.

In [24], a filter based approach based on Radon transform is implemented to detect lines. Superposition is used to match palmprints.

In [25], line segments are detected by convolution with 3×3 operators that approximate the shape of a straight line. Because the detected segments are considered to be straight, it is possible to represent them by their end-points. Matching between two images is performed by comparing coordinates of the segment end-points from the two different images.

In [26], lines are detected with canny filter. The extracted ROI is divided in subregions and in each subregion properties associated with canny response and line orientation are kept. Euclidean distance is used for matching.

2.2.2 Gabor filters

In PalmCode approach [15], images are convolved with one Gabor filter. For each location in the region of interest (there are 32×32 locations), Gabor response is converted to a binary format. This can be considered a feature reduction method, as Gabor response will be 1 or 0. Afterwards, hamming distance is used as a classifier. This approach was classified as texture based because Gabor filters are often used as texture discriminators. Because these filters can model lines adequately, subsequent methods used them to detect line orientation [11].

Other examples of work using Gabor filters is found in references [27, 21, 20].

2.2.3 Invariant features

Some computer vision problems require features invariant to rotation, scale and stretching. As image palms are subject to these conditions, invariant features were used in palmprint recognition.

Examples:

Scale Invariant Feature Transform (SIFT) [28] was used in ref. [29] to extract features, although the results are not as good as other methods.

In [14], Zernike invariant moments are used as features. In [30], the feature vector is composed by Hu invariant moments.

In [31], a rotation invariant texture descriptor — local binary pattern (LBP) [32] — is used to extract features.

2.2.4 Wavelets

Wavelets are a mean of extracting useful information from images. There are some examples of articles using wavelets for feature extraction.

Examples:

Low- and high-pass images are originated through Haar wavelets and regarded as features. Dimensionality reduction is performed on sub images using Independent Component Analysis (ICA) [33].

In [34] M-band wavelets are used to decompose the image. L_1-norm and variance of sub bands compose the feature vector.

2.2.5 Image subspace

Approaches to determine statistically relevant subspaces, such as ICA, Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) can be directly applied to images. The result is a feature vector by itself. Although it is not an intuitive approach to identify personal characteristics in palmprints, the purely statistical information is useful.

Examples:

An example of image subspace as features is the work presented by Wu et al. [35]. Images are treated as a point in a multidimensional vector and LDA is used to reduce its dimensionality. In the training phase, a projection is computed using a set of pictures. This determines the Fisher's space for the given training set. Then, in the testing phase, images are projected accordingly to the previous phase, reducing its size while keeping relevant information from pixel values. Later, in the classification stage, euclidean distance is used as matching function.

In [36], images are projected with PCA into a subspace, which the author name “eigenpalms"".

2.2.6 Other

Other features were used for palmprint recognition. Two examples are the use of fractals [37] and phase component of Fourier transform [38].

2.3 Feature selection

In virtually every type of data one can think of, there are some points more relevant than others. Methods to find statistically important components of data have been applied, such as PCA or LDA. The initial data is thus represented in a subspace where important information is more evident.

These methods have been applied to palmprint recognition in different ways. Here, examples of direct application to features extracted from images are shown.

Application of subspaces directly to images was discussed in the section 2.2 — Feature extraction.

2.3.1 Specific property coding

Following Palmcode [15], different approaches using Gabor filters were implemented. In Competitive Code [39], each point in the palm is considered to belong to a line and an orientation is assigned by finding the maximum output of six Gabor filters with different orientations. The fusion code [40], a similar approach is implemented with Gabor filters with four orientations. Another approach is reported in [41], where response of Gabor filters with three different orientations are encoded in three bits.

2.3.2 Subspace

A good example of using subspace approaches is described in [42]. Magnitude from Gabor filters is useful to take information from images. 40 different filters combining eight different directions and five scales were used. As the images are 128 x 128, the total number of features is $128 \times 128 \times 40 = 655360$. AdaBoost, an adaptative machine learning algorithm, and LDA, are used to reduce the number of features. This is done by training the algorithm towards separation of self classes and external classes. The resulting number of features is less than 200. There is no discrimination between which gabor filter are being used, or from where in the palm are features being taken, because they do it all at once.

2.3.3 Clustering

An example of clustering algorithms is a subsequent work to Competitive Code [39]. In this approach Gabor filters with six different orientations are used to extract line orientation. An improvement to this approach using a clustering algorithm was related in [43] where filters with 180 orientations were used to extract features. Through clustering with Fuzzy C-Means algorithm the 180 orientations are clustered into six centroids. Implementing Competitive Code with Gabor filters oriented as the detected centroids results in improved performance while using the same amount of information.

2.4 Classifiers

Euclidean distance is most used measure for matching feature vectors, although other distances have been employed. However, the use of learning methods and more advanced classifiers often results in improved performance. Here, examples of classifiers used for palmprint recognition are given.

In [18], statistical properties calculated over the response of edge detectors are used to feed a Neural Network. In [14], Neural Networks are used to match features based on Zernike moments.

In [19], lines are detected with sobel operators. Histograms of sobel response summed over X and Y axis of the image is used to train an HMM.

Use of SVM's to match features based in wavelet decomposition was stated in [44].

These are representative examples of standard classification procedures in palmprint recognition.

2.5 Performance assessment

2.5.1 Databases

To correctly assess performance of different methods it is advisable to run large scale tests. Here, four examples of palmprint databases are shown.

PolyU Palmprint Database

PolyU [45] is the most widely used low-resolution palmprint database for algorithmic research considering recognition purposes. It is comprised of 7752 images from 386 different users. Users provide either the left or the right hand, but not both. On average, there are 20 samples per user, taken in two sessions. Visually, it is possible to identify more variability between images between different sessions. This happens because there is time lapse between the sessions, which makes the results closer to what happens in operating systems. Due to the use of pegs, hand positions are restricted simplifying preprocessing stages.

CASIA Palmprint Database

The CASIA Palmprint Database [46] is similar to PolyU although smaller in size. There are no pegs to restrict hand positions. Users lay the back of the hand on a dark surface and a fixed CMOS camera is used for image acquisition.

IIT Delhi Touchless Palmprint Database

This database [47] uses a normal camera to take pictures to hands standing in the air, without any support. This increases the variability in hand positions. There are two problems to deal with in such conditions: hands can be at different distances and inclinations relative to the camera, distorting the images. These factors increase the complexity of pre-processing steps necessary to align images before any comparison is made between different hands.

Hand Geometric Points Detection Competition Database

In order to explore influence of hand position in acquisition of geometric features, a database without pegs or any position constraints during acquisition was created [48]. This is a very challenging database because it requires extraction of characteristic points to align images before any further steps. However, positional freedom caused some palms to be partially hidden or severely distorted, compromising its usage in palmprint recognition.

2.5.2 Measures of performance

Genuine matches (or true positives) are matches between palmprints from the same user. Impostor matches (or false positives) are matches between different users. Performance of an algorithm is assessed by how well the system can separate genuine matches from impostor matches in a database. To measure that separation, the system outputs a distance between the palmprints, which should be higher in the case of an impostor match.

There are two tests to evaluate performance:

- Verification
- Identification

In verification tests it is assessed whether the method can prove if an individual is who he claims to be. This is the same as asking the question: *Is this person who he claims to be?* These tests are performed by comparing every possible pair of palmprints in the database, and classifying in genuine and impostor matches. An operating threshold determines what is considered a genuine or an impostor match (if the individual is whom it claims to be). If the distance measure is correct, the system is able to correctly separate genuine from impostor matches.

In identification tests, the system aims to identify an individual in a database. The system should also be able to discriminate whether the individual is present in the database or not. Technically, an input is compared with a set of palmprints in a database (to simulate a user database). The system uses the distance measure to identify the best match (shortest distance) with the entries in the set. That distance has also to be smaller than the operating threshold, to validate identification of the input palmprint and guarantee that unregistered users are not accepted. There are more errors in this test, because in addition to the errors arising from the operating threshold, genuine matches have to correspond to the shortest distance in the whole user set. A greater number of users in the set generates more errors.

Figure 2.3 illustrates palmprint distance measures for a set of genuine and impostor matches. Impostor matches have greater values than genuine matches, which is expected, but there is some overlap between two distributions, leading to classification errors. The operating threshold should be set in order to minimize such errors. The true positives and false positives are on the left of the threshold. True positives arise from the elements in the genuine distribution, while false negatives are the elements in the impostor distribution falling under the threshold value. The same concept is valid for true and false negatives, with the genuine distribution originating false negatives (false rejections).

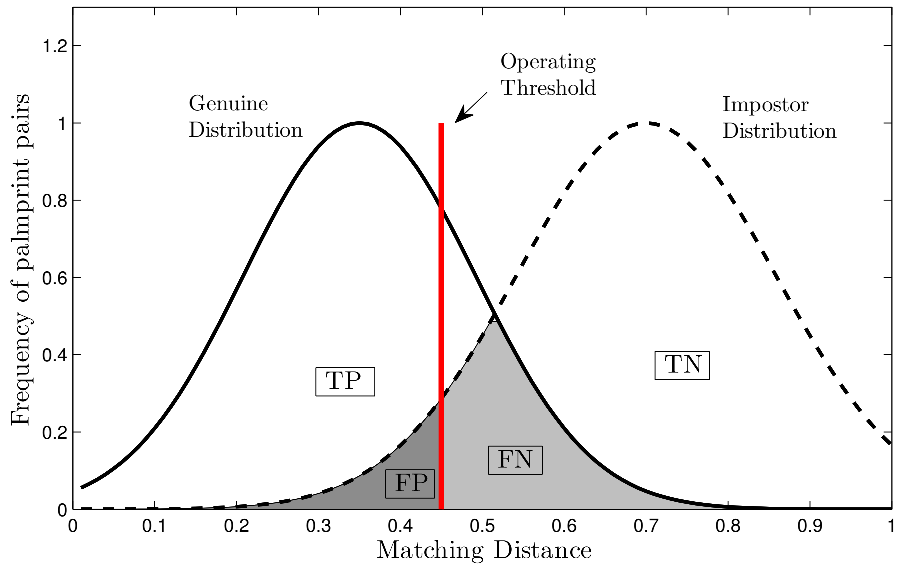

Figure 2.3: **Genuine and impostor distributions as function of a similarity measure.** FP, FN, TP, and TN denominate false positives, false negatives, true positives and true negatives, respectively. The red vertical line represents the operating threshold separating true and impostor matches.

Value of the operating threshold can be set lower, to increase security, or higher to decrease false rejections. A Receiver Operator Characteristic (ROC) curve is created by plotting true and false positive rates for all possible threshold values (figure 2.4). It is possible to assess all relevant rates at once: GAR, FAR and FRR (Genuine Acceptance Rate, False Acceptance Rate and False Reject Rate). Genuine accepts are genuine matches correctly classified. False accepts are impostor matches classified as genuine matches. False rejects are genuine matches classified as impostor. The equal error rate (EER) can be estimated from the ROC curve, and corresponds to the point where FAR and FRR have the same value.

ROC plots have the same interpretation in both verification and identification tests.

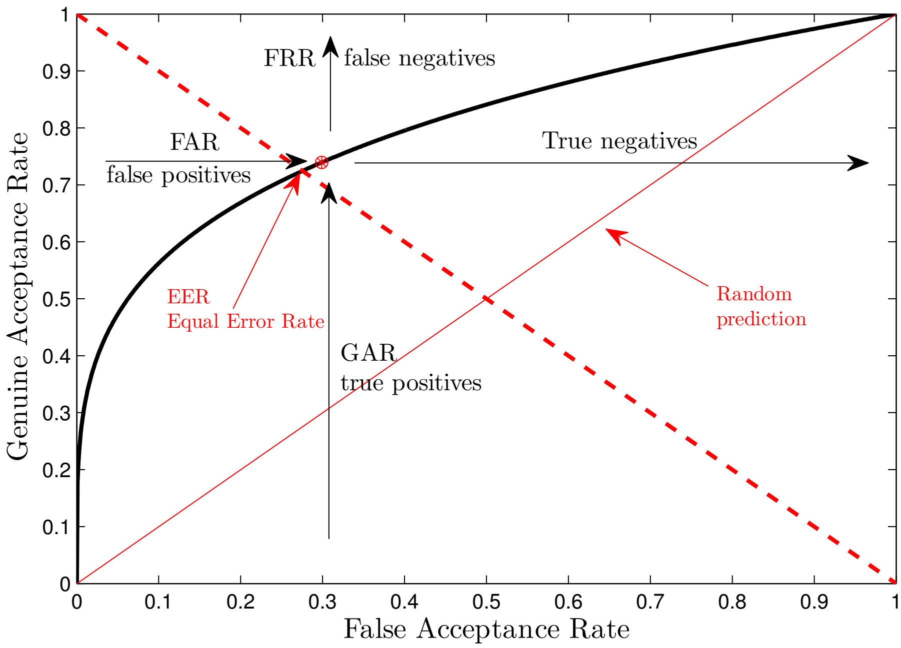

Figure 2.4: **Toy ROC curve.** For all possible operating thresholds, the genuine and false acceptance rates are calculated and plotted (black line). For a given point (red circle) FAR and GAR are represented as the distances to the Y and X axis, respectively. A random matching algorithm is represented by the continuous red line. The intercept between the dashed red line and the ROC curve represents the EER.

Chapter 3

Pre-processing Notes

3.1 Region of interest

Most of the existing works use small regions from the palmprint, wasting a significant portion of the palm. Intuitively, the more area used for feature extraction and matching, the better the recognition. Of course, this might increase computation time. We implemented a method to extract a region of interest (ROI) that maximizes the area used.

The image is converted to binary format using the Otsu's algorithm [49], which is composed of white (ON) and black (OFF) pixels. Then, three morphological operations are performed:

- **Closing**, with a circular structuring element of 8 pixels in diameter. This eliminates holes in the image.

- **Opening**, with a circular structuring element of 5 pixels in diameter. Small objects originated from noise are deleted with this procedure.

- **Erosion**, with a circular structuring element of 8 pixels in diameter, ensuring the region of interest is completely inside the palm.

The option for the size of morphological operations is adequate since the resolution of the image is 180×240, approximately.

At this point the left bottom and top borders of ROI are defined. Now, it's necessary to set the left border of the ROI. This is done by calculating the number of ON pixels in each column. This value will be zero on the left of the image, where no portion of the palm is present. Towards the right side, the number of ON pixels starts increasing because the fingers are present in those columns. The maximum value is present after the fingers zone, and this is where the left border of the ROI should be set. The column chosen for left border is 8 pixels to the right from the first column in the image (counting from the left) to have 86% of the maximum of ON pixels (figure 3.1). The value of 86% was determined empirically.

An example of an extracted ROI is depicted in figure 3.2.

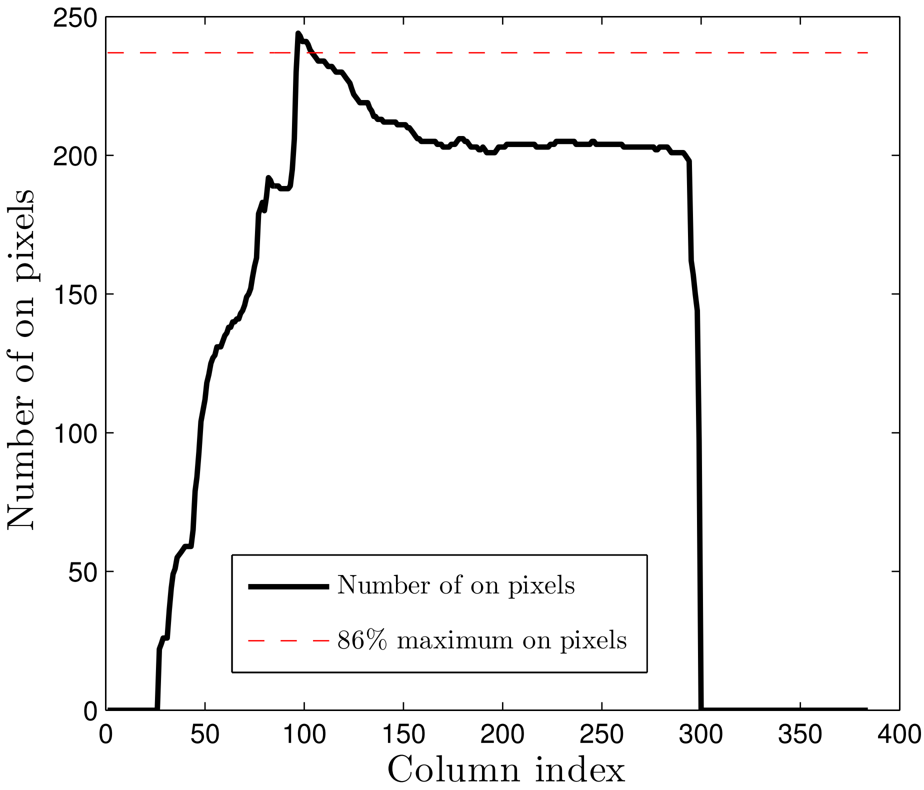

Figure 3.1: **Finding left border of ROI.** In the binary image obtained after morphological operations, the number of ON pixels in each column is represented (black line). The red line is set at 86% of the maximum of ON pixels for a given image. The left border of ROI is placed 8 pixels to the right from the first column (counting from the left) hitting the red line.

Figure 3.2: **Extraction of ROI.** Original image is in **a)**. Through morphological operations, the mask in **b)** is computed. The region used further in this thesis is represented in **c)**.

3.2 Extracting mid-frequency information

There are two types of noise that can affect performance of methods developed: high frequency noise and low frequency noise. High frequency noise arises from the resolution limit of the camera and corresponds to strong local variations in pixel values. A pixel with high intensity inside a palm line is an example of high frequency noise. Low frequency noise is originated from the three-dimensional shape of the hand, which makes some areas of the palm to be illuminated differently.

To reduce both types of noise in input image I, two Gaussian filters are used for convolution, adequate to respond to different frequencies. This methodology is called Difference of Gaussians (DoG) and is commonly used as an edge detector, but in this case its use is as a band-pass filter. The filtered image I_F is calculated as follows:

$$I_F = (I * G^{3\times3}_{\sigma=0.5}) - (I * G^{20\times20}_{\sigma=20}), \tag{3.1}$$

where $G^{3\times3}_{\sigma=0.5}$ is a 3×3 Gaussian operator with $\sigma = 0.5$ which eliminates high frequency noise. Convolution with a Gaussian operator of size 20×20 and $\sigma = 20$ detects low frequency information which is subtracted.

Finally, image is mapped to the range $[0,255]$:

$$I_F = 255 \cdot \frac{I_F - min(I_F)}{max(I_F) - min(I_F)} \tag{3.2}$$

Chapter 4

Principal Lines Detection

Edge detection - is a universal task in computer vision and algorithms for this purpose exist since long time ago. There are a multitude of different algorithms suitable for a different range of problems. At the time being, a search for “Edge Detection” on IEEE Xplore engine retrieves over 18.000 results. However, only very few were applied to palmprint images. Some are standard edge detection techniques, such as the Canny filter [50] used in ref. [51], or the Sobel operator used in ref. [52]. In other works new methods for palm line detection are developed [53]. Line detection usually uses detected edges to reconstruct lines, as for instance, the Hough transform [54] used in ref. [55]. In this thesis, two methods based in graph-search for edge following are implemented and tested.

Principal lines are among the most stable features in palmprints, considering natural ageing processes and changes in acquisition conditions. Therefore, it is of great interest to detect them accurately. Actually, there were previous attempts to detect principal lines [56], as they can be used in several ways to improve palmprint recognition. It is possible to directly compare the shape of palm lines by superposition methods [57]. Some features can be extracted directly from palm lines [52] and be used to compare different hands.

One of the mandatory processes in a significant number of recognition systems is the establishment of a coordinate system, which is based in defined geometric points. Usually, these points are the valleys of the fingers [11]. However, this procedure might be inappropriate since the reference points are external to the palm and skin stretching allows a wide range of palm positioning. Using reference points inside the palmprint — based in principal palm lines — would be more efficient, as it would diminish the errors associated with skin stretching[23].

One of the main objectives in this thesis is to develop a method that avoids coordinate systems, which are subject to alignment errors. The proposed approach (Chapter 5) takes textural information from regions around the detected principal palm lines, discarding spatial information as the extracted texture has no association with a specific position in the palmprint. Also, orientation of the lines is not included in any fashion in the feature vector. This procedure will give strong insight on the importance of line texture as a feature for palmprint recognition.

4.1 Related work in palmprints

In this section, two existing methods for palm line detection are detailed: gaussian derivatives as used in [58]. Gaussian derivatives are a standard edge detection algorithm, used in several edge detection problems. Modified finite radon transform (MFRAT) was specifically created for palm line detection. These two were chosen for being representative examples of edge detection algorithms.

Gaussian Derivatives

Calculating derivatives of an image is a standard procedure to detect edges —lines in this case. Here is a summary of the methodologies found in ref. [58]. First, image is smoothed by 1-D Gaussian filtering with variance σ_s, which has influence on the smoothness of the lines that can be extracted. The smoothed image I_s is calculated as follows:

$$I_s = I * G_{\sigma_s} \tag{4.1}$$

where "$*$" denotes the convolution operation, "I" is the original image and "G_{σ_s}" is a one-dimensional Gaussian function. Then, first and second order derivatives are calculated by a sliding window that performs convolution with derivatives of a one-dimensional Gaussian function. With this procedure line detection is direction specific, thus it is necessary to repeat the method in several directions, which are 0° 45° 90° and 135°. These are derivatives of 1-D Gaussian functions with variance σ_d, which has selectivity on width of detected lines. The mathematical formulation is as follows:

$$I'_\theta = I_s * ({}^{\theta}G'_{\sigma_d})^T \tag{4.2}$$

$$I''_\theta = I_s * ({}^{\theta}G''_{\sigma_d})^T \tag{4.3}$$

where "^{T}" is the transpose operation, "$*$" is the convolution operation and "${}^{\theta}G'_{\sigma_d}$" and "${}^{\theta}G''_{\sigma_d}$" are the first and second-order derivatives in direction "θ" of a one-dimensional Gaussian function with variance "σ_d".

Edges are detected by looking for the zero-cross points of the first derivative, which indicates the presence of an edge. At that point, the second derivative is used as a measure of intensity of the detected edge. A line in a palmprint digital image is represented by low intensity pixels surrounded by high intensity pixels, which can be detected like edges. The first-order derivative is negative as the filter passes from the surroundings into the line, and is positive as the filter goes out of the line into the surroundings again. This obligates existence of a zero cross point. For a stronger line, the first-order derivative will produce more negative values before the line and more positive values after the line. This implies that a strong line has a high second-order derivative. This is why it is used as a measure of line intensity. Check figure 4.1 for a graphical explanation.

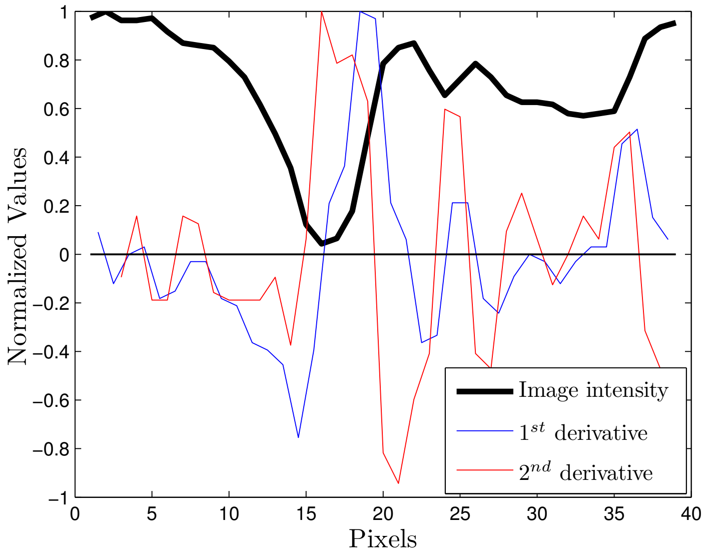

Figure 4.1: **Detecting lines with first and second order derivatives.** The dark black line represents a slice of a palmprint image. The slice transversally crosses a palm line at pixels 13 to 19. As palm lines are represented by darker pixels, the slice's intensity profile has a valley when it crosses the line. In the center of the line, first order derivative is zero while second order derivative has its maximum value.

As derivatives are sensible to noise and false edges are detected, there is the need to find a threshold to correctly separate lines from background. However such procedure results in broken lines because some of its segments fall under the threshold. In ref. [58] hysteresis thresholding was used to overcome this issue. It consists in applying two thresholds on the value of the second derivative, T_{high} and T_{low}. Lines above T_{high} are considered strong lines and are always kept. Lines above T_{low} are kept only if they contact strong lines. T_{low} is the minimum of the non-zero values, which means it is considering all detected lines. T_{high} results from applying Otsu's method to the non-zero values.

Modified Finite Radon Transform

In this work authors extract lines using a modified finite Radon transform (MFRAT). The Radon transform consists in projecting a bi-dimensional function, such as an image, into several 1-D projections [59]. The set of 1-D projections is performed in the range of 0° to 180°. A projection is the summation of image intensity along all parallel axis to an imaginary line with the angle specified. Despite the mathematical description above, the proposed MFRAT looks more like a filter than a transform. In fact, it is not possible to restore the original image from the output. It consists in sliding windows that convolves with the image. If the center pixel of a window falls in a line, and the rest of the mask is aligned with that line (low pixel values), the filter output will be minimum, indicating presence of a line. Thus it is necessary to have enough filters to represent all possible line directions. There are 12 sliding images, each for a different angle. Six of them are represented in figure 4.2.

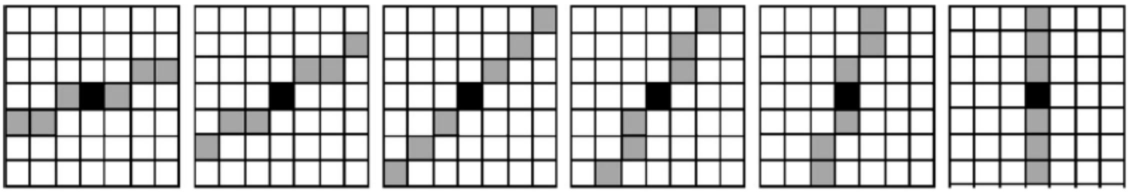

Figure 4.2: **Windows used in ref. [60]**. Only windows from 15° to 90° are shown.

As this sliding window screens the image pixel by pixel, two new images are generated. One is the energy image, which has the value of the minimum summation for every angle window. The other is the angle image for which the angle corresponding to the lowest energy value is kept. It is important to mention that the average of each window is subtracted before the filtering operation occurs, causing the method to be sensible to line direction and shape rather than differences in pixel values.

The angle image is used to split energy image into two sub-images. Energies associated with line angles between 0° and 90° are stored in one sub-image and energies associated with line angles between 90º and 180º are stored in another sub-image. It is considered that principal lines are in one of the two sub-images. Radon transform is used to find the sub-image with more line intensity, and that will be considered the principal lines sub-image. This has an inherent drawback, since principal lines can be present in both images, as the angle chosen for division is independent of the images.

4.2 Shortest paths as edge detector

4.2.1 Context and concept

To understand what a shortest path is it is necessary to introduce some graph concepts. A graph is a representation of a set of nodes, which are connected by a set of arcs. In the case of image processing, we consider the pixels of an image the nodes of a graph, where arcs conceptually connect pixels to each other. Usually, each pixel is connected to its eight neighbours.

According to graph theory, if arcs connecting nodes have a weight associated, the graph is called an weighted graph. The shortest path between two nodes is the path formed by the arcs summing the least weight possible. If one properly uses image properties to weight arcs, shortest paths will exist over the edges.

This concept was introduced because of its ability to detect connected edges even in the presence of noise[61]. Other methods would produce fragmented lines, which are more difficult to interpret computationally.

There are other approaches to avoid segmented lines detection, like active contours (also known as snakes). With this approach, a line is initialized. Then, external and internal forces operate to guide the line towards the edge. Internal forces regard line shape and length, to approximate detection to the shape and size of the expected output. External forces attract the line to the edge. This approach is useful in the detection of objects, because the initial line can be closed around a probable location of the object[62].

4.2.2 Description of the algorithm

In the image derived graph, the distance between two pixels is dependent on characteristics of those pixels. Because palm lines are regions of the image with low intensity level, the weight of an arc is set to be lower if pixel values are lower. This results in shortest paths to exist on darker regions such as palm lines, thus making shortest path algorithm work as a line detector. This concept was already used for staff line detection in musical scores [63], yielding promising results.

The concept described for line detection works best if the shortest path is computed between two nodes on the edges of a line. However, it is harder to detect the ends of a line than the line itself, making it necessary to screen for the shortest path over a reasonable area.

Because of this, the implemented algorithm searches for the minimal path between a set of nodes A and a set of nodes B. Practically, A and B are the first and last rows of pixels in a rectangular image.

The graph search problem in line finding is much simpler (lines have to keep the same direction, can't go back and forth), the implementation of known algorithms [64, 65] is simplified. Costs of several paths are computed row by row, from A towards B. There are three conceptual elements involved:

- Weight matrix $W(i,j)$, which is a matrix of the same size as the image that assigns a weight for each pixel, based in, for example, grey level intensity. In this method the weight matrix are the pixel values themselves;
- Cost matrix $C(i,j)$ that stores cost values of several paths;
- direction matrix $D(i,j)$ that has information about the change of direction of each path in every pixel.

The weight of a path between two pixels is calculated from the weight matrix. It is defined as the geometric mean between the vales of the two pixels, multiplied by the euclidean distance separating the two pixels. The geometric mean gives a lower value to the path linking a dark to a bright pixel than to the path linking two intermediate pixels.

For a pixel $P(i,j)$, where i iterates over rows from top to bottom, and j is the index of the pixel in that row, three paths are considered: the path arising from the pixel immediately above: $P(i-1,j)$, the path arising from the top-left pixel $P(i-1,j-1)$ and from the top-right pixel $P(i-1,j+1)$. The cost for each of these three paths is the weight of the considered path linking the previous pixel to the current one, plus the cost of the already calculated path in the pixel where the path is coming from: $Cost = W(i,j) + C(i-1, j+d)$, where $d = \{-1,0,1\}$ for the top-left, top-center and top-right pixels, respectively. The minimum of these three possible values for $Cost$ is stored in $C(i,j)$ and the corresponding direction value d in $D(i,j)$.

In this way, the value of each element in $C(i,j)$ is the cost of the shortest path linking the correspondent pixel to the first row in the image.

As the algorithm runs over the rows, cost values increase as they correspond to the sum of all weights associated to pixels forming growing paths. For a pixel j in a row i, there is only one path, the cost of that path is $C(i,j)$ and the indexes of pixels forming that path can be retrieved from the direction matrix D. In the last row, the path starting from the pixel with the smallest value in the cost matrix C is the shortest path in that image. There are no horizontal connections between pixels, only vertical and diagonal, limiting paths to be inclined at a maximum of 45°. Also, paths are either top to bottom or bottom to top (is the same as the paths are symmetric) but they can't support both directions. This means that paths will be straight forward linking top and bottom edges, without turns. It is guaranteed that the global minimum is found. Figure 4.3 illustrates the output of the described algorithm.

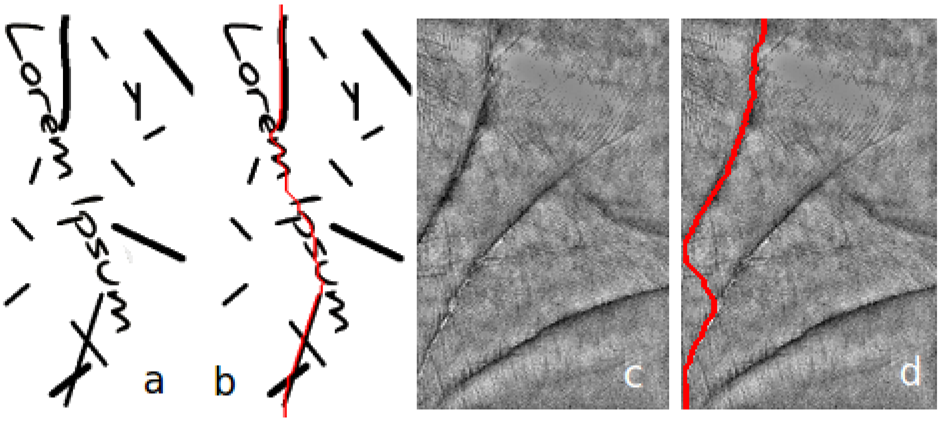

Figure 4.3: **Short paths in images.** Images are treated as a graph and the shortest path **(red lines)** between top and bottom of each image is computed. Arcs connecting dark pixels have a lower weight, forcing the shortest path to exist over long vertical and dark portions of the image. **a)** original toy image and **b)** corresponding shortest path. **c)** original palmprint image and **d)** corresponding shortest path.

4.3 *Method I* — Shortest paths in subregions

Using shortest path to detect line has the advantage of detecting a long, straight lines, which are characteristics of principal lines in palmprints (figure 4.3). However, it is not trivial to define where to compute the path. To solve this problem it was proposed to compute several shortest paths in subregions of the image. In this method, the image is divided in squares of arbitrary size and two shortest paths are computed in each subregion: the vertical path, linking top and bottom edges, and the horizontal path, linking left and right edges. Figure 4.4 depicts the output from this implementation including square subregions of two different sizes.

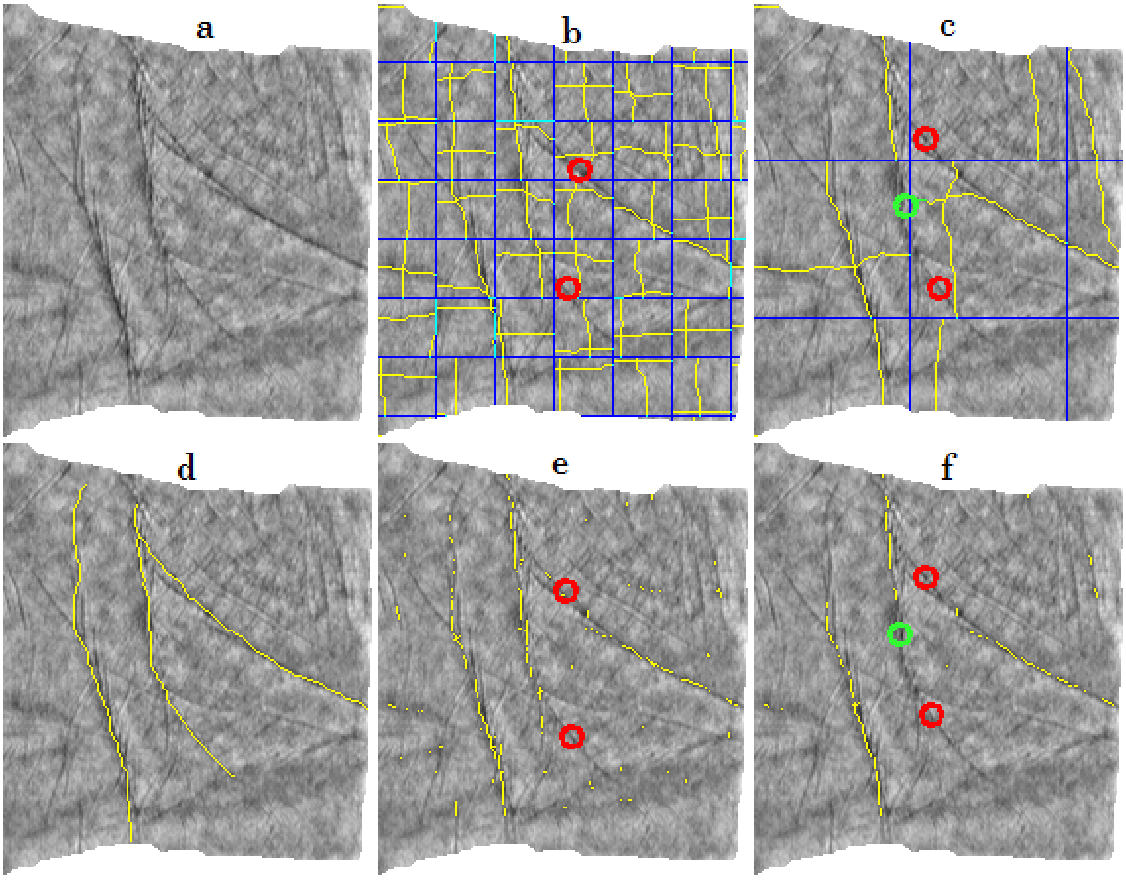

Figure 4.4: **Computing shortest paths in subregions of an image. a)** is the original input image and **d)** has principal lines marked by hand. **b)** and **c)** are divided in squares 30 and 80 pixels wide, respectively. Horizontal and vertical shortest paths **(yellow)** are computed in each of the squares. A threshold was applied to keep the 15% (in **e**) and 30% (in **f**) darkest pixels as principal lines. Red circles mark line segments that cannot be detected as they are inclined by nearly 45°. Green circles indicate a line segment that was not detected using 80 pixel wide squares because the vertical path in that square is detecting a stronger line. Note that this problem did not occur with the 30 pixels wide squares.

As a shortest path will always be computed even if there is no line inside a square, numerous portions of paths that do not correspond to any line will exist — see figure 4.4 b) and c). Thus it is necessary to eliminate such path segments. Since lines are the darker pixels of the image, this property can be used to differentiate the valid pixels in the shortest paths. A threshold is applied to keep only a percentage of the darker pixels in computed shortest paths of an image. Ideally, this procedure can eliminate all extra segments while keeping pixels over the palm lines.

The two parameters in this method are:

- Subregion size (width of the square subregions in pixels)
- Percentage of darkest shortest path pixels to keep (a threshold)

There are various drawbacks inherent to this implementation, and some consequences can be predicted.

First, if subregions are too big and two lines with similar orientation are placed inside the same square, it is not possible to detect both lines. Using smaller squares can solve this issue. However, smaller subregions mean more shortest paths will be computed resulting in an increased number of extra segments that correspond to no line. It might be complicated to separate the extra segments from the meaningful detections.

Second, if lines are oriented at 45°and cross subregions through adjacent borders, neither the vertical nor the horizontal path are able to track them adequately.

To issue the mentioned cases without diminishing subregion size, which could produce additional detection noise, a solution is proposed. It consists in the combination of two modifications: horizontal and vertical shift of the square subregion by half its size and the rotation of the subregion by 45°. This results in four different divisions of the original image:

- Original subregions
- Shifted subregions
- Rotated subregions
- Shifted and rotated subregions

The shortest paths computed in both horizontal and vertical directions for the four subregion set-ups mentioned above are put together. In figure 4.5, it is illustrated how this procedure withdraws the mentioned issues. As referred earlier, a threshold is applied to eliminate extra segments that correspond to no line.

Because palmprint images have some noise, threshold application cannot eliminate all pixels that have no line correspondence. Likewise, some pixels over lines are wrongly eliminated. This results in broken lines and lost pixels in the images. To issue this problem, the detected lines are subject to morphological dilation, which reconnects lines, followed by morphological thinning till all segments are one pixel wide. Finally, segments shorter than 5 pixels are removed.

4.4 *Method II* — Dynamic tracking shortest paths

Determining shortest paths is an adequate way to find where a line is but not at verifying existence of a line. This happens because a shortest path will always be detected, whereas there is a line or not. Shortest paths should be used as a procedure to detect continuous line segments. However, in the first implemented method (shortest paths in subregions), a threshold is applied to eliminate extra segments, which brakes up the lines. Then, lines are put together again by morphological operations.

One of the main advantages in using graph search methods is the detection of fully connected segments.The previous method required the use of a threshold, which broke line segments, and

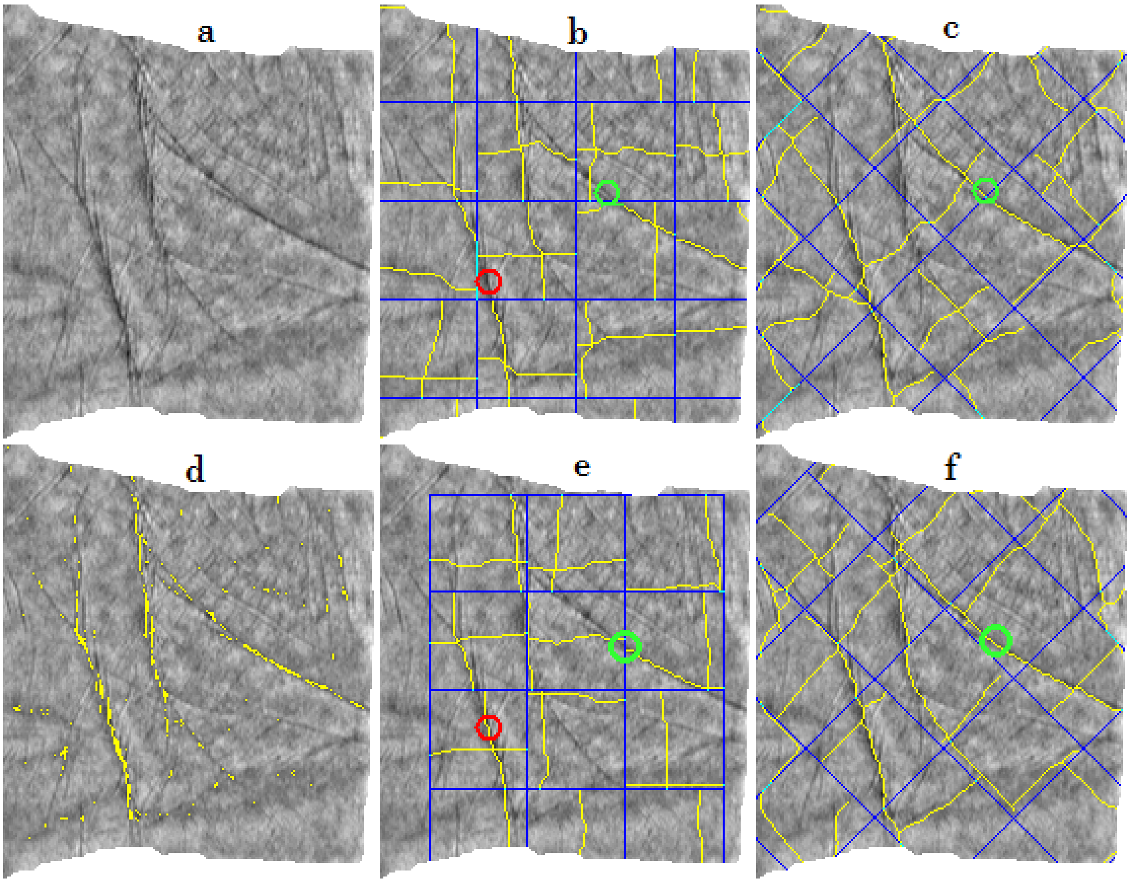

Figure 4.5: **Computing shortest paths in subregions of an image. a)** is the original input image. **b)** illustrates computation of vertical and horizontal shortest paths in 50 pixel wide squares. To diminish the number of line segments that are not detected, the squares are **c)** rotated by 45°, **e)** shifted by half its size and **f)** rotated by 45°and shifted by half its size. Red circles point a segment that was detected after shifting the squares and green circles point a segment that could only be detected in rotated squares.

a subsequent reconstruction by morphological operations. This is a redundant procedure. To overcome this situation, a new method which requires no thresholding was implemented.

This method is based in the idea of tracking lines with small subregions. Given an already detected portion of a line, its continuation can be detected by placing a new subregion with an adequate orientation at the end of the detected segment.

Initial (seed) segments are detected from zones in the palmprint where lines almost always exist. By studying the palmprint database it was possible to identify such areas. In figure 4.6, these areas are in black. After a line segment was detected in these initial search boxes, a new subregion is placed on top of the segment, with according orientation. The new shortest path is forced to start in the same pixel in which the previous path ended.

A problem is how to detect initial segments accurately. One property of principal lines is that they exist in fixed positions in the hand. In most cases, each of the three lines starts at one of the palm borders. Three initial boxes were created to detect the seed segment for each of lines. Now from these seeds, subregions are created and another segment is detected. It is forced that the shortest path in this subregions starts at the same pixel where the previous segment ended. This process is continued till it reaches a border of the palm, meaning the line is detected even after it ended.

Initial segments for the two lines closer to the fingers have the same shape and size. The principal line around the thumb has a different set-up.

- Thumb: It starts at $1/4$ from the top and ends at $3/4$ from the bottom. Therefore, the height of the box is half of the image's. From the right, it starts $1/60$ and its width is $1/6$ multiplied by the images width.

- Two upper palm lines (symmetric): Horizontally, the box starts at $1/8$ the width of the input image and ends at $5/8$, which is slightly after the middle. The height of the box is $1/6$ of the image's height. It starts at $1/60$ from the top and bottom of the images, in order to detect two different lines.

The orientation update after each shortest path detection for the tracking squares is computed taking the first and last pixels of that path, so the next square is aligned and its rows are perpendicular to the axis crossing the first and last pixels. The initial pixel for the next box is the last pixel in the path.

Figure 4.6 illustrates the algorithm described.

4.5 Performance evaluation

4.5.1 Ground truth dataset

In order to compare different methods, principal palm lines of some images were hand-marked. In total, 83 images from different users comprise the ground truth dataset. It was tried to have a balanced number of easy and difficult images. In some of the more difficult images it is hard to

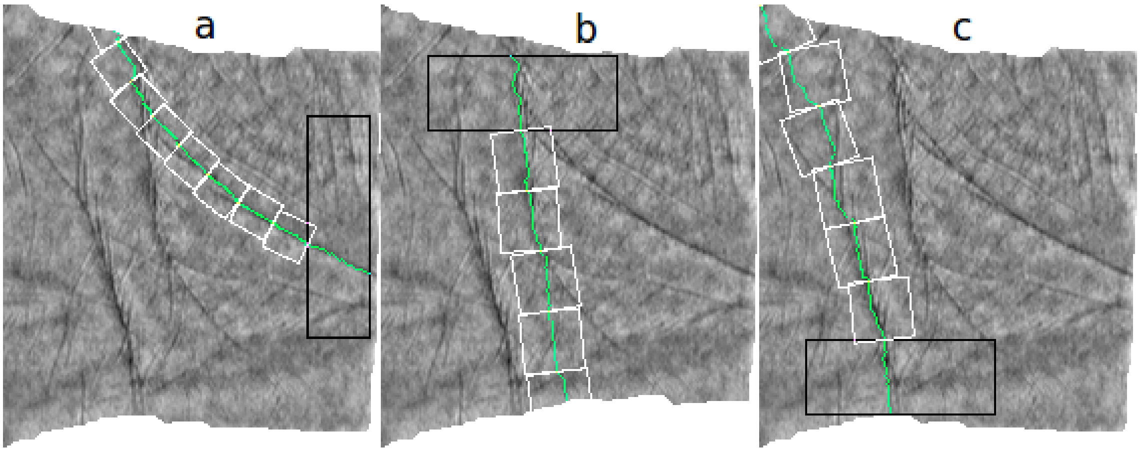

Figure 4.6: **Illustration of method II. a)** shows detection of the line around the thumb. **b)** and **c)** are the detection of the other lines. Initial search boxes are in black, tracking boxes in white, and detected path in green.

define the lines, and the criteria used is subjective. However, this dataset will prove efficient in evaluating algorithm performance because it comprises a reasonable number of samples.

It was considered that all palm prints have three principal lines. This is a contradiction to the findings in ref. [66] which claims that palm prints can have 1, 2 or 3 principal lines. However, it is possible to visually identify three lines in all of them, although some are much weaker and probably not detected by the methods used in those works.

Figure 4.7 shows some examples of the ground truth dataset.

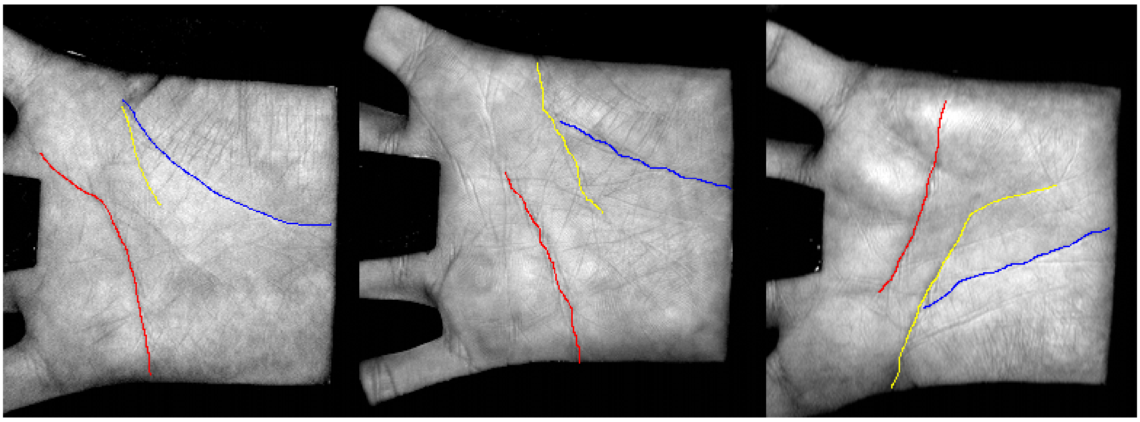

Figure 4.7: **Images in the ground truth dataset.** Hand-marked principal lines used as ground truth to evaluate performance of different methods.

4.5.2 Distance measures

In each image, the distance of detected lines to the ground truth is taken as a performance indicator. Of course, the smaller the distance, the better is the method. Each pixel is considered as a point in a bi-dimensional Cartesian space and euclidean distance between the two sets of points is measured.

Consider two sets of points A and B corresponding to the lines in two images. The distance of any point p in A to the whole set B is the distance of p to the closest point in B. This is also called the *infimum* of the distances between p and all points in B.

The average distance from A to B — $D_{avg}(A,B)$ — is defined as the average of the distances from all points p in A to B.

The Hausdorff distance of A to B — $D_{haus}(A,B)$ — is the distance of the furthest point p in A from B.

An interesting aspect of distances between sets of points is that it is not a symmetric measure. This means that $D(A,B)$ is not that same that $D(B,A)$. It is possible to take advantage of this property to retrieve useful information about algorithm performance.

The distance of an image to the ground truth $D(Detection, GroundTruth)$ is low if all pixels in detected lines correspond to a pixel in the ground truth. But consider the case in which only one pixel was detected and it was near a line in the ground truth. The distance will be low, although this situation cannot be considered a proper detection. This measurement will only be high if the detected lines are wrong, but not if there are undetected lines. This is why $D(Detection, GroundTruth)$ is a measure of wrongly detected lines.

Distance of the ground truth to an image $D(GroundTruth, Detection)$ is low if all lines in the ground truth were detected. But if there are extra detections, which correspond to no line, this distance is still low. This is why $D(GroundTruth, Detection)$ is a measure of undetected lines.

Optimally, both distances are small. Figure 4.8 and table 4.1 illustrates the usefulness of these measures.

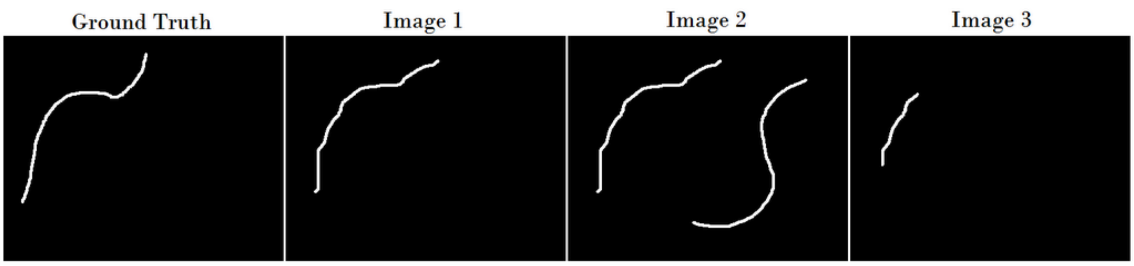

Figure 4.8: **Toy images to illustrate distance measures.** Three images are compared with the ground truth. **Image 1** corresponds to a nearly perfect detection. **Image 2** has a nearly perfect detected segment plus a false detection — false positive case. **Image 3** has a correctly detected segment but misses most of the line — false negative case. The average distance of the ground truth to an image is an indicator of undetected segments, so $D(groundtruth, image3)$ will be high. On the other hand, distance of an image to the ground truth is an indicator of bad detected lines, so $D(image2, groundtruth)$ will be high. Results for this toy example can be found in table 4.1.

4.5.3 Experimental results for Method I

In this method there are two principal parameters that affect the output. These are the size of the square subregions used and the threshold applied. This section shows results for various combinations of these parameters.

Table 4.1: Results for toy images in figure 4.8

	Indicates false negatives		Indicates false positives	
	$D_{avg}(GT,Img)$	$D_{haus}(GT,Img)$	$D_{avg}(Img,GT)$	$D_{haus}(Img,GT)$
Image 1	6	20	6	15
Image 2	6	20	74	187
Image 3	34	115	3	10

Performance with different parameters was evaluated as described in section 4.5. The parameters tested were all 90 combinations using:

- Size of square subregion (in pixels): 20, 30, 40, 50, 60, 80, 100, 125 and 150.
- Amount of (darkest) pixels kept: 2%, 4%, 6%, 8%, 10%, 15%, 20%, 26%, 35% and 50%

The amount of false positives (wrong detections) and false negatives (undetected lines) is depicted in figure 4.9, where both the distance from detection to ground truth and from ground truth to detection are shown. The distance shown is averaged within the 83 samples in the ground truth and corresponding detections. In figure 4.10 the maximum between the two distances is shown, thus including information of both false positives and false negatives. The parameters for which the maximum of the two distances is minimum are the best for line detection. In table 4.2, distance measures for 4 of the best parameter combinations are shown, including both average and Hausdorff distances.

Table 4.2: Summary of best results using four different parameter set-ups

		Indicates false negatives		Indicates false positives	
Square size	Threshold	$D_{avg}(GT,Img)$	$D_{haus}(GT,Img)$	$D_{avg}(Img,GT)$	$D_{haus}(Img,GT)$
40	8%	6.7	39.8	6.7	48.6
50	10%	6.6	38.5	6.7	48.9
80	20%	5.7	35.0	6.8	50.6
100	26%	6.7	38.4	6.2	46.5

4.5.4 Experimental results for Method II

There are some conditions to which performance of this method is subject, such as size of the boxes used to track lines. This is basically the same as changing size of square subregions in Method I. In order to compare performance of this method, all three implemented versions were tested with tracking squares of different sizes.

The three versions described in section 4.4 are:

- Version 1 : Start-point for line continuation at the end of the path in each tracking square.
- Version 2 : Start-point for line continuation at 80% of the path in each tracking square.

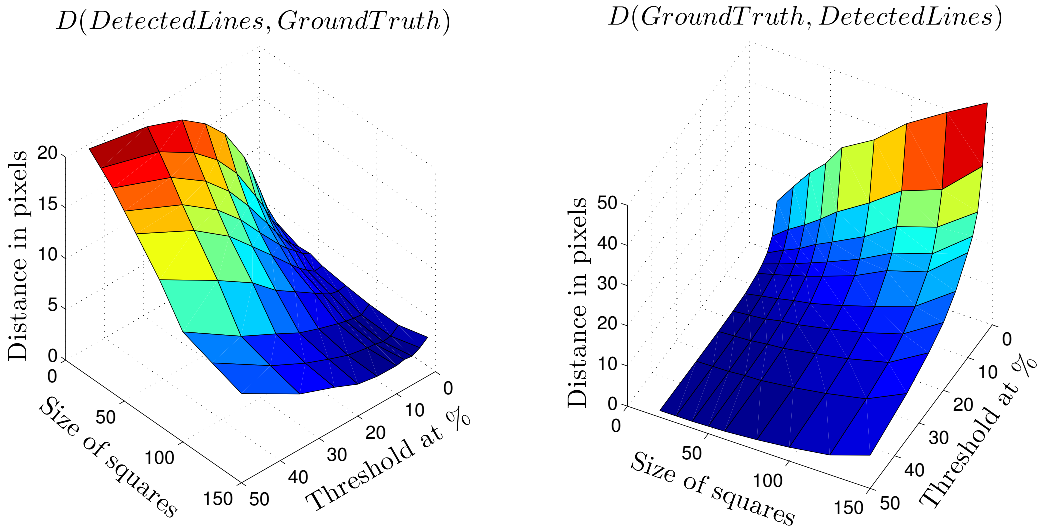

Figure 4.9: **Average distance from ground truth using different square sizes and thresholds.** Detection of lines is sensitive to the size of squares (in pixels) and to the threshold applied. The smaller the squares, the more segments will be detected, therefore a smaller fraction of segments should be kept. A threshold of 10% means that only the 10% darkest pixels will be kept as detected lines. On the left, the distance between detections and ground truth is an indicator of how much segments are wrongly detected. With smaller squares and a more permissive threshold there are a lot of wrong detections — false positives. On the right, the distance between the ground truth and the detected lines is sensible to the amount of lines that are not detected. Using big squares with a tight threshold cannot detect most of the lines, explaining why such combination of parameters results in a big number of undetected segments.

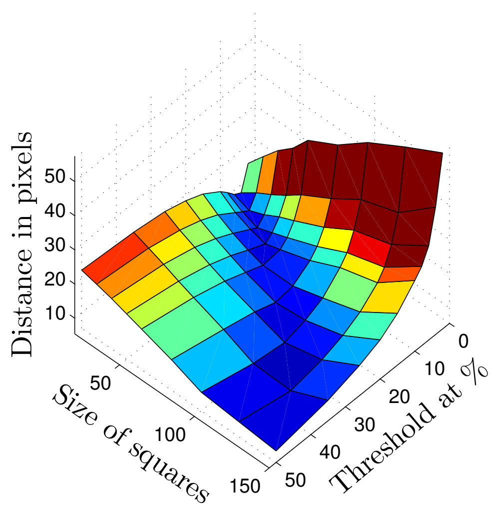

Figure 4.10: **Maximum of the two average distances represented in figure 4.9.** A good performance is achieved if both distance from ground truth to detection and from detection to ground truth are low. In this figure, the maximum of both measures is plotted in function of square size and threshold applied. Bigger squares require more permissive thresholds, such as keeping 50% of the darker segments. Interestingly, good results are achieved independently of square size, as long as an adequate threshold is applied.

- Version 3 : Same as previous but normalizing mean and standard deviation of input image.

The initial search box was kept constant in all experiments. The set-up for orientation was the same as described.

Figure 4.12 and 4.13 show the average and Hausdorff distances, respectively.

Sometimes, when the track of the line is lost, the method is unable to find the line again. In this situations, there will be a bad detection (figure 4.11).

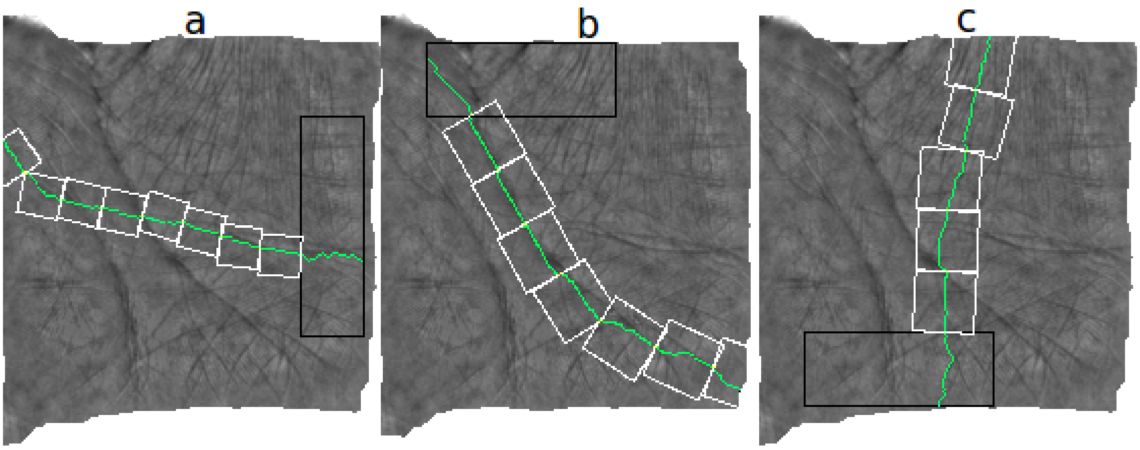

Figure 4.11: **Bad detections with tracking algorithm. a)** shows detection of the line around the thumb. **b)** and **c)** are the detection of the other lines. Initial search boxes are in black, tracking boxes in white, and detected path in green. Tracker squares deviate from line and continue tracking in wrong directions.

4.6 Discussion

Regarding method I

The most interesting conclusion from method I is that the size of the squares works well within a large range $[30, 125]$ (size in pixels). Of course, it is required for the threshold applied to be in conformity. This happens because the smaller the squares, the more extra segments will be produced.

Unfortunately, due to image noise, some lines appear broken and lost dark pixels not associated to lines result in false detections. Because shortest paths have the point of detecting connected segments, it does not make sense to apply a threshold on it. This causes difficulties in identifying three principal lines.

To overcome these difficulties, method II was developed.

Regarding method II

In method 2, three lines are generated in all cases. This makes it easier to regularly detect principal lines.

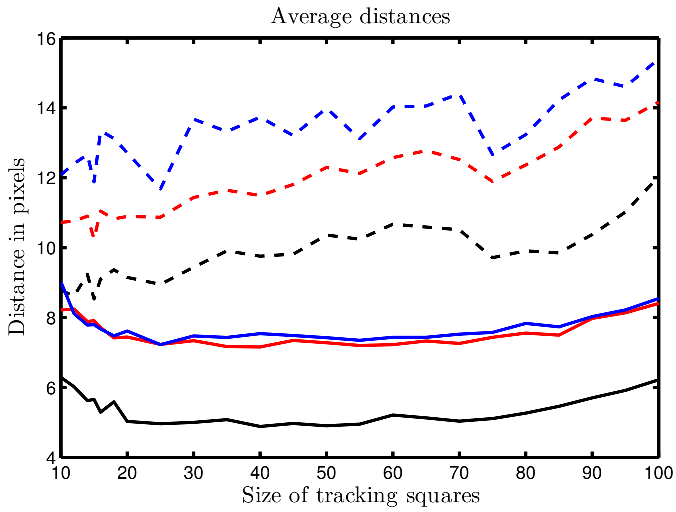

Figure 4.12: **Average distances for tracking algorithm.** Solid lines represent $D(GT, Img)$ and dashed lines represent $D(Img, GT)$, or analogously, false negatives and false positives. This method produces less false negatives but more false positives. A tracker window near 20 is optimal because it is the best compromise between false negatives and false positives. Versions 1,2 and 3 are in blue, red and black, respectively.

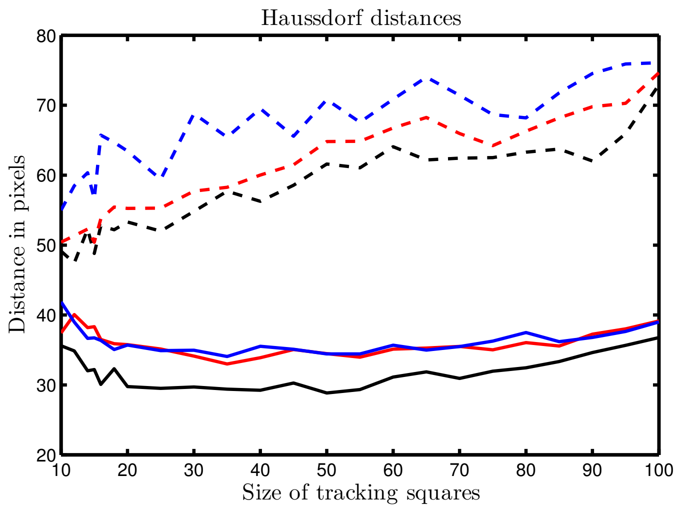

Figure 4.13: **Hausdorff distances for tracking algorithm.** Solid lines represent $D(GT, Img)$ and dashed lines represent $D(Img, GT)$, or analogously, false negatives and false positives. As seen with average distances, this method produces less false negatives but more false positives. Again, a tracker window near 20 is optimal because it is the best compromise between false negatives and false positives. Versions 1,2 and 3 are in blue, red and black, respectively.

However, the obtained results indicate that the performance of this algorithm is close to that of method I — table 4.3. This might be due to the fact that if the line track is deviated from the correct position, the tracking system will not find the line again. This is particularly problematic when the lines of interest are close to each other and detected lines jump from one to the other (figure 4.11). Some spatial rules can be implemented to improve the method [66].

However, less false negatives are produced. Probably, this is because lines are always connected, resulting in less undetected pixels.

It is visible an increase in false positives with bigger square sizes figures 4.12 and 4.13. Maybe because a bigger search window is ineffective in tracking lines accurately, forcing the shortest path to go through wrong directions.

Table 4.3: Comparing method I and II.

	Parameters		Distance for false negatives		Distance for false positives	
Method I	Square size	Threshold	$D_{avg}(GT, Img)$	$D_{haus}(GT, Img)$	$D_{avg}(Img, GT)$	$D_{haus}(Img, GT)$
	60	15%	5.3	33.8	7.5	52.9
	80	20%	5.7	35.0	6.8	50.6
Method II	Square size		$D_{avg}(GT, Img)$	$D_{haus}(GT, Img)$	$D_{avg}(Img, GT)$	$D_{haus}(Img, GT)$
	15		5.0	29.8	9.1	53.3
	20		5.7	32.2	8.5	48.8

Chapter 5

Recognition Based in Principle Line Texture.

5.1 Introduction

In Chapter 2 different methods to extract features from palmprints are described. In this work, the texture of principal lines is used as a global feature of palmprints. The features don't include information about texture localization or orientation of lines.

Principal lines are detected by the *dynamic tracking shortest paths* approach developed in this thesis (Method II in Chapter 4). Texture information is detected using Haralick's features [67] on principal line boundaries. This approach was used in a wide range of problems, such as cancer characterization in medical imaging [68] or segmentation of urban areas in satellite and aerial images [69]. Here, for the first time to our knowledge, Haralick's features are applied to palmprint recognition.

In a training stage, exemplar textures are identified in a set of images representative of different classes. These exemplar textures form a dictionary of textons. A palmprint is represented by an histogram of the frequencies of each texton [70]. This concept is also used for other features rather than texture, and the dictionaries are called bags of visual words [71].

The first two sections in the methodology section describe Haralick's features and the texton dictionary concept in detail. The implementation procedure is described in the methodology section, followed by results and respective discussion.

5.2 Methodology

This section details about the implemented procedure in four stages: feature extraction, building the dictionary of textons, histogram representation and matching and finally, the set-up of a verification test.

Recognition Based in Principle Line Texture.

6	7	7	7
5	1	4	7
1	5	0	3
5	5	3	0

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	2	0	0
1	0	0	0	2	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	2	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0
5	2	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	2

Figure 5.1: **GLCM of a** 4×4 **toy image.** In this example, the GLCM (right) of a toy image (left) is computed. The grey-levels in the input image range from 0 to 7, so the size of the GLMC is 8×8. The displacement vector is $D = (2, -1)$, which means a spatial displacement of two pixels to the right and one to the bottom. In the GLCM, element $P(i, j)$ is the frequency of finding grey-level i and j separated as defined by D. The pairs found in this condition are $(6,4)$, $(5,0)$ (twice), $(1,3)$ (twice) and $(7,7)$. Note the counts are doubled up because the matrix is made symmetric by summing its transpose. This causes (i, j) and (j, i) to be undifferentiated.

5.2.1 Feature extraction

Haralick's texture features

It is not trivial to define what texture is. Different methods have been developed to extract texture information from images. In 1973, Haralick defined texture as the spatial dependencies between grey-level values [67]. Such information is organized in a grey-level co-occurrence matrix (GLCM). The GLCM (denoted P) is computed for a given displacement vector (a spatial dependency) and each element $P(i, j)$ corresponds to the frequency of two grey-level values (i and j) spaced as specified in the displacement vector. See figure 5.1 for a graphical explanation of how GLCM is built.

From the GLCM $P(i, j)$, 13 statistical features were computed that describe textural properties of images are computed. Therefore, an image (or subregion) is represented as a point in a 13-dimensional space. Note that each displacement vector produces a different set of features. Here is the list of used properties:

- Entropy: $f_1 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P(i, j) \cdot log(P(i, j))$,

- Angular second moment: $f_2 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P(i, j)^2$,

- Contrast: $f_3 = \sum_{n=0}^{N_g - 1} n^2 \left\{ \sum_{i=1}^{N_g - n} P(i, i + n) \right\}$,

- Inverse difference moment: $f_4 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \frac{1}{1 + (i - j)^2} P(i, j)$,

- Correlation: $f_5 = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i \cdot j)P(i,j) - \mu_x\mu_y}{\sigma_x\sigma_y}$,
- Information measure of correlation 1: $f_6 = \frac{f_1 - HXY1}{max\{HX,HY\}}$,
- Information measure of correlation 2: $f_7 = \sqrt{1 - exp[-2 \cdot (HXY2 - f_1)]}$,
- Sum average: $f_8 = \sum_{i=2}^{2N_g} i \cdot P_{x+y}(i)$,
- Sum variance: $f_9 = \sum_{i=2}^{2N_g}(i - f_{10})^2 \cdot P_{x+y}(i)$,
- Sum entropy: $f_{10} = -\sum_{i=0}^{N_g-1} P_{x+y}(i) \cdot log\{P_{x+y}(i)\}$,
- Difference average: $f_{11} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i-1) \cdot P(i,j)$,
- Difference variance: f_{12} = variance of P_{x-y} ,
- Difference entropy: $f_{13} = -\sum_{i=0}^{N_g-1} P_{x-y}(i) \cdot log\{P_{x-y}(i)\}$,

where the following notation applies:
N_g is the number of gray levels in the GLCM,
$P_x = \sum_{j=1}^{N_g} P(i,j)$,
$P_y = \sum_{i=1}^{N_g} P(i,j)$,
$P_{x+y}(k) = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P(i,j)$, where $i+j=k$ and $k=[2,3,...,2N_g]$
$P_{x-y}(k) = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P(i,j)$, where $|i-j|=k$ and $k=[0,1,...,N_g-1]$
HX and HY are the entropies of P_x and P_y, respectively,
$HXY1 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P(i,j) \cdot log\{P_x(i)P_y(j)\}$
$HXY2 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P_x(i)P_y(j) \cdot log\{P_x(i)P_y(j)\}$.

Extraction of features along palmprint principal lines

The first step is to define the region around principal lines for further texture extraction. This is performed by extracting subregions of size 22×22 cenetred at pixels along the line spaced by four pixels. These spacing guarantees a reasonable overlap of information between adjacent subregions

Table 5.1: Feature extraction summary.

Abbreviation	d in $D(d,\theta)$	Statistical reduction	Features per subregion
D=1, F=13	1	Mean	13
D=1, F=26	1	Mean + Std	26
D=1, F=52	1	—	52
D=3, F=13	3	Mean	13
D=3, F=26	3	Mean + Std	26
D=3, F=52	3	—	52
D=5, F=13	5	Mean	13
D=5, F=26	5	Mean + Std	26
D=5, F=52	5	—	52

for the given subregion size. Moreover, subregions are rotated according to line angle so that all line segments present in the 22×22 subregions have the same orientation. This ensures texture features have no information about line orientation, enabling results to elucidate about importance of line texture for palmprint recognition.

On average, between 100 and 150 subregions are extracted in the three lines of a palmprint. For each of them, Haralick's features will be computed from 256-levels ($N_g = 256$) GLCM's using different displacement vectors. As grey-level patterns exist in every direction, it makes sense to use displacement vectors that represent spatial dependencies in multiple orientations.

To accomplish this, the displacement vector is defined as $D(d,\theta)$ where d is a distance in pixels and θ is an orientation. Orientations used are always 0°, 45°, 90°and 135°. Features are extracted using three values for the distance in pixels d, which are 1, 3 and 5, for all four values for θ. This results in ($4 \times 13 = 52$) features per subregion, for each d.

In addition, the 52 features are treated in three different ways. First is by taking the mean of each feature over the four values of θ, which results in 13 features. Secondly, by taking both the mean and standard deviation over the four values of θ, resulting in 26 features. Finally, by keeping all 52 features.

This set-up results in nine different ways of taking features from subregions, which are summarized in table 5.1.

5.2.2 Building the texton dictionary

Each subregion is represented by a point in a multidimensional space (13, 26 or 52 dimensions). To build the dictionary, 200 palmprints from 50 different classes are used. As each palmprint possesses an average of 130 subregions, the approximate number of points in feature space is 26000. It is expected that common textures in palm lines (the textons) are disposed as clusters (groups) of points in the feature space.

To identify palmprint principal line textons, K-means clustering algorithm is applied. Given a set of data, K-means algorithm aims to partition the data into K groups. The number of clusters K must be specified and is an important parameter since it will restrict how the groups are divided. It must be as close as possible to the real number of groups in the data. The algorithm is initialized

Table 5.2: Clustering and assignment.

	Distance measure	
Abbreviation	K-means clustering	Texton assignment
Euclidean	Squared euclidean	Euclidean
Cosine	Cosine	Cosine
SoftWeight ($N = 3$)	Cosine	Eq. 5.1 with $N = 3$

with K random centroids and assigns each data-point to the closest centroid, given a distance definition. In an iterative process, the distance of all data-points is minimized by adapting location of centroids. This is performed till a local minima is obtained (K-means does not guarantee to find the global optimum). In the final iteration, the coordinates of centroids represent feature values of textons.

As it is impossible to know beforehand how many centroids exist, we perform clustering for $K = 35, 50, 80, 160$ and using both cosine and squared euclidean distances to assign data-points to centroids.

5.2.3 Texton histrograms and matching

After the texton dictionary is built with K textons, it is necessary to find the frequencies of each texton in palmprint principal lines. This is done by assigning each of the 130 (on average) data-points in a palmprint to the closest centroid. A data-point is a 22×22 subregion represented in the feature space. Euclidean distance is used if K-means algorithm was used with squared euclidean distance. Similarly, cosine distance is used if K-means clustering was performed with cosine distance. The number of data-points assigned to each centroid is divided by the total number of data-points in a palmprint, so that histograms represent frequencies of textons, making this approach useful because the number of subregions in palmprints is variable.

Because some data-points might be in the middle of two or more centroids, and it is not trivial to decide to which of centroids it should be assigned, a soft-weighting scheme [71] that assigns a data-point to the N closest centroids was implemented. It is defined as follows:

$$W_c = \sum_{i=1}^{N} \sum_{j=1}^{M_i} \frac{1}{2^{(i-1)}} sim(j,c), \tag{5.1}$$

where W_c is a vector with K elements $W = [W_1, ..., W_c, ..., W_K]$, M_i represents the number of data-points whose ith nearest neighbour is the centroid c and $sim(j,c)$ is the cosine similarity between data-point j and centroid c. This weighting scheme was tested with $N = 3$.

As the distance measure used to assign each data-point to a texton is the same as used in K-means algorithm, these two parameters are put together in a new parameter called *clustering and assignment*. Table 5.2 summarizes the three tested options for clustering distance and texton assignment.

Palmprints are represented by a frequency histogram with K textons. Therefore, a palmprint can be considered a point in a K dimensional space, in which coordinates are the frequencies of each texton. To match two palmprints, the euclidean or cosine distance in this K dimensional space is taken. Note that the distance in histogram matching is not directly associated to clustering or assignment stages, where the distance measure should be the same.

5.2.4 Verification test set-up

PolyU palmprint database comprises images from 386 classes. To reduce computation time and test an high number of parameters, only 50 classes are used in the test. For each class, the first two images from both sessions are used to build the texton dictionary. The remainders — eight from each session on average — are used in the verification test.

All possible combinations formed by two palmprints are matched. For each pair, the distance matching is computed as described. If the distance between two palmprints is below a certain threshold T and the palmprints are from the same user, it is considered a genuine accept. If the palmprints are from different users, it is a false accept. GAR and FAR are computed for every possible value of T, considering all parameters described earlier. ROC curves and EER's are calculated from GAR and FAR.

5.3 Results and discussion

As described in the previous section, there are multiple parameters to test. They are divided in four groups:

- Nine possible ways to extract features.
- Four different values of K in clustering phase.
- Three cluster and data assignment procedures.
- Two distances for histogram matching.

This results in 216 possible combinations, which are tested in a brute force scheme. The EER is calculated for each of them, and six plots are created to show the relation between two groups of parameters. For example, EER values for different feature extraction ways and number of centroids in K-means, number of centroids and matching distance, and so on. In each of these plots, EER for two parameters is calculated as the average of all possible combinations of other parameters while keeping those parameters constant. For example, EER while using $D = 1, F = 13$ and $K = 50$ is the average of all six combinations of cluster assignment and matching distance parameters.

In figure 5.2, the nine feature extraction methods are tested with the four values of K. The first observation is that the more centroids, the bigger EER, which indicates a poorer performance. This is a bit counter-intuitive, since more textons should represent better a palmprint. Secondly, using

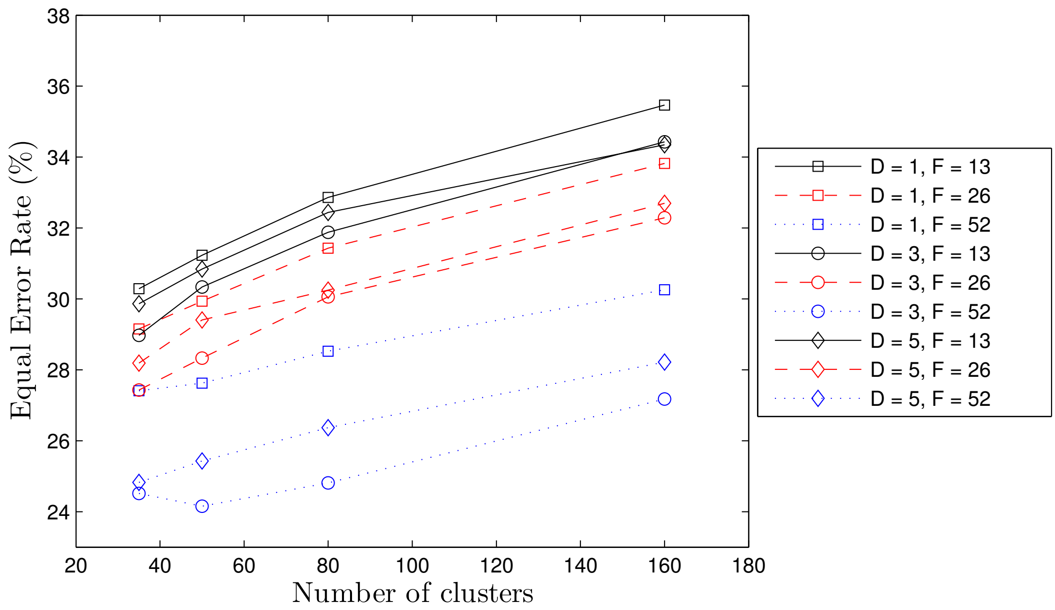

Figure 5.2: **Feature extraction *vs* number of centroids.** Black solid lines represent the mean of 13 features over four directions, red dashed lines represent mean and standard deviation of 13 haralick features (total of 26 features) and blue dotted lines are all features for all four directions concatenated (total of 52 features). Squares are for displacement vector = 1, circles = 3 and diamonds for 5. Using only the mean is bad, the mean+std is slightly better but all 52 is what tells more information. The best combinations is D=3 and F=52.

features from all four directions of displacement vector performs better then reducing them to the mean ($F = 13$) and mean and standard deviations concatenated ($F = 26$). This makes sense since textural information is dependent on direction. Third, distance d in displacement vector $D(d, \theta)$, is slightly better for $d = 3$ than to $d = 5$, while $d = 1$ performs the worst. The best combination appears to be $F = 52, D = 3$ and $K = 50$, but this might not result in the global minimum EER because values are averaged for all combinations of clustering/assignment and matching distance parameters.

In figure 5.3, it is possible to observe the same behaviour for feature extraction parameter. $F = 52$ performs the best, followed by $F = 26$ and $F = 13$. $D = 3$ and $D = 5$ prove to be the best option, with a slightly better performance of $D = 3$. For the clustering and assignment parameters, using both cosine distance in K-means and nearest neighbour assignment is the best option, followed by squared euclidean and euclidean and finally, the soft-weighting scheme with $N = 3$ (which uses cosine distance for clustering).

Euclidan distance might perform worse because values of Haralick's features are in different scales, which unbalances the importance of features. Cosine distance is independent of scaling which is a probable cause for its better performance.

Relation between features and matching distance is depicted in figure 5.8. The same trend for extracted features is observed: matching using cosine distance performs better than with euclidean.

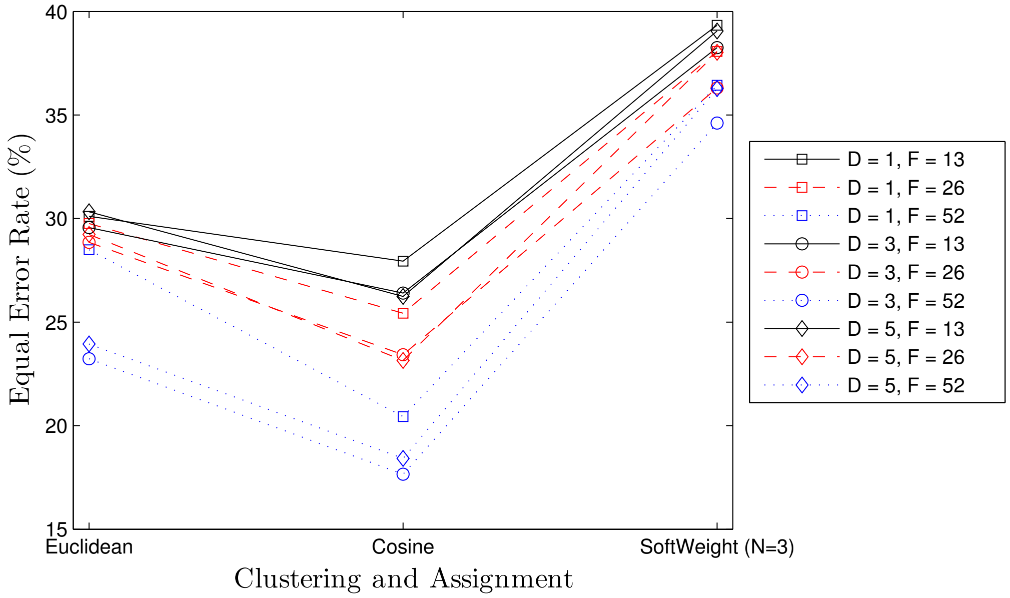

Figure 5.3: **Feature extraction *vs* clustering and assignment.** Black solid lines represent the mean of 13 features over four directions, red dashed lines represent mean and standard deviation of 13 haralick features (total of 26 features) and blue dotted lines are all features for all four directions concatenated (total of 52 features). Squares are for displacement vector = 1, circles = 3 and diamonds for 5. Again, the bigger the number of features the better. Considering the cluster of features and assignment of data-points phase, the best option is to use cosine distance in both cases.

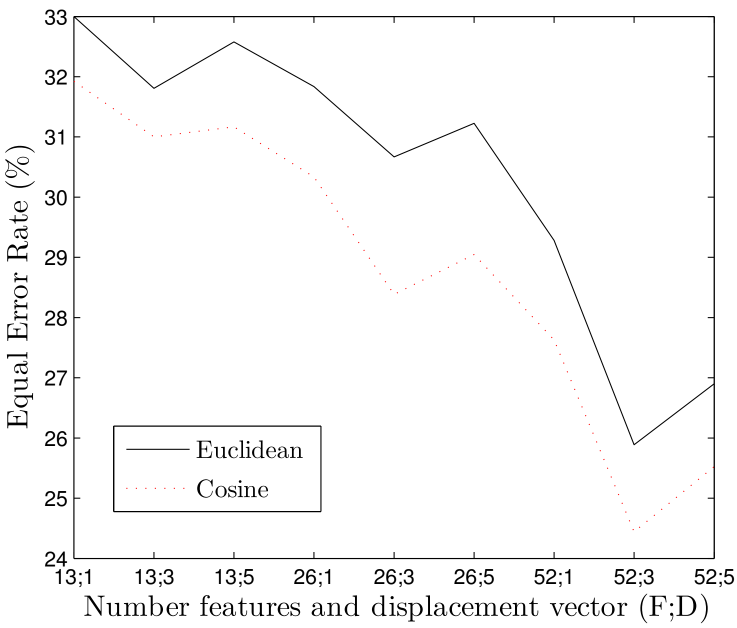

Figure 5.4: **Matching distance *vs* feature extraction.** Black solid line represents euclidean distance in the matching scheme, while red dotted line represents cosine distance. Cosine distance performs better for all nine ways of feature extraction.

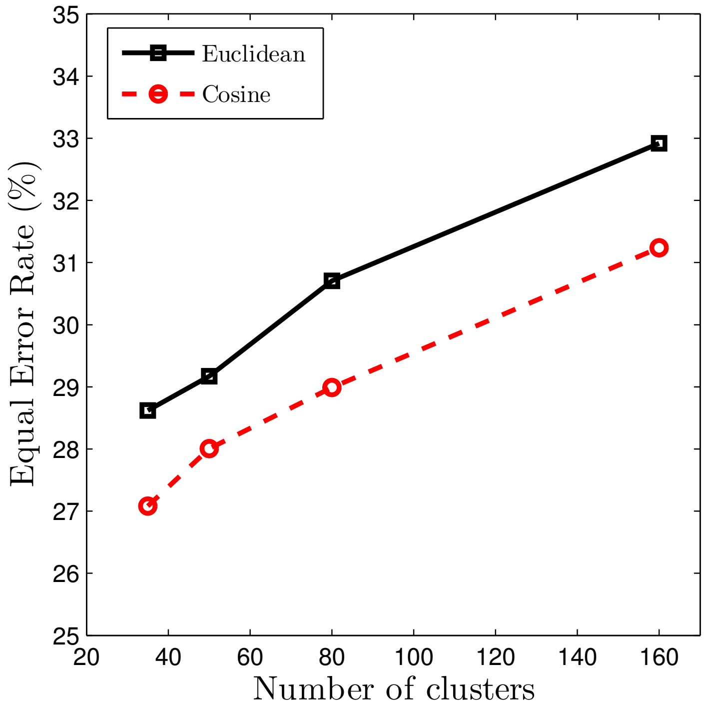

Figure 5.5: **Matching distance *vs* number of centroids.** Black solid line represents euclidean distance in the matching scheme, while red dotted line represents cosine distance. Line depicted in figure 5.2, more centroids mean a worse performance. Matching with cosine distance performs better independently of the value of K.

In figure 5.5, EER for all values of K and different clustering and assignment options are shown. Again, a bigger value of K results in worse performance and cosine distance for matching is the more appropriate.

Observing EER values for different clustering and matching option and matching distances shows that cosine distance is the best option for both clustering/assignment and matching phase (figure 5.6). Matching with euclidean is always worse except if the soft-weighting scheme is used.

Finally, the relation between number of centroids K and clustering and assignment scheme is depicted (figure 5.7). Respective to clustering and assignment, results are in accordance with previous observations: cosine distance performs the best, followed by the euclidean scheme and, worse of all, the soft-weighting scheme.

Interestingly, the soft-weighting scheme is more sensitive to the number of clusters. In all cases, an increase in K results in a bigger EER, but this increase is less significant if euclidean or cosine schemes are used in the clustering and assignment stage.

According to these observations, the best combination of parameters appears to be using $D = 3, F = 52$, a low value for K, cluster and assign points with cosine distance and, finally, match with cosine distance. However, the best possible combination might be masked as soft-weighting scheme has poorer performance with a big number of centroids. Indeed, the best combination of

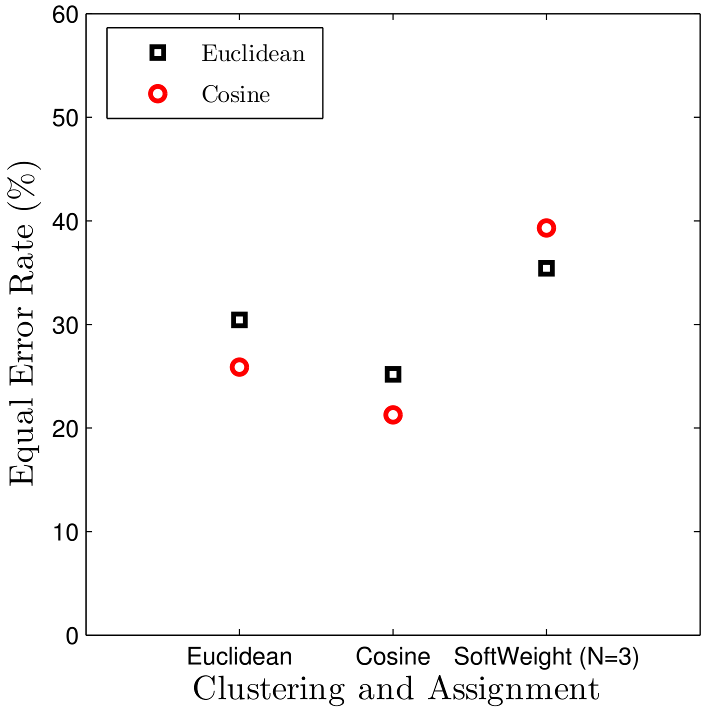

Figure 5.6: **Matching distance *vs* clustering and assignment.** Matching with cosine distance is better if euclidean or cosine schemes are used in clustering and assignment stage. Euclidean distance performs better if the soft-weighting scheme is used.

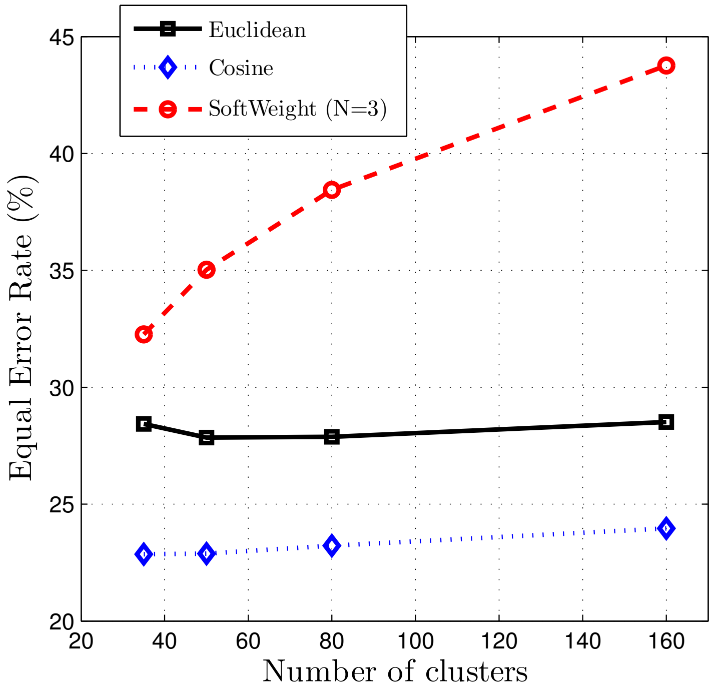

Figure 5.7: **Clustering and assignment *vs* number of centroids.** All three schemes in clustering and assignment stage show poorer performance with a big number of centroids. The more sensitive is the soft-weighting scheme, where the greatest decrease in performance is observed.

parameters is obtained using the maximum number of centroids (160). Figure 5.8 shows a partial ROC plot of the best five combinations, table 5.3 shows its parameters and EER's.

From the best five combinations of parameters some conclusions can be drawn. Cosine distance for matching should be used, as well as in the clustering and assignment stage. It is also important to keep all 52 features from the four directions of the displacement vector. The distance d in $D(d,\theta)$ should be 3. Contrary to previous observations, a bigger value of K yields better results, although it is the less important parameter since the fifth best combination is obtained with $K = 50$.

From these results some general conclusions can be taken. First, when using GLCM's is important to use displacement vectors with different orientations and study the best distance to use. In this case the best results were obtained for $d = 3$.

Table 5.3: Best five parameter combinations.

		Parameters			
Rank	EER	Feature ext.	Centroids	Clust./Assign.	Matching
Best	15.02%	$D=3,F=52$	$K=160$	Cosine	Cosine
2^{nd}	15.45%	$D=3,F=52$	$K=80$	Cosine	Cosine
3^{rd}	15.54%	$D=5,F=52$	$K=160$	Cosine	Cosine
4^{th}	16.25%	$D=5,F=52$	$K=80$	Cosine	Cosine
5^{th}	16.64%	$D=3,F=52$	$K=50$	Cosine	Cosine

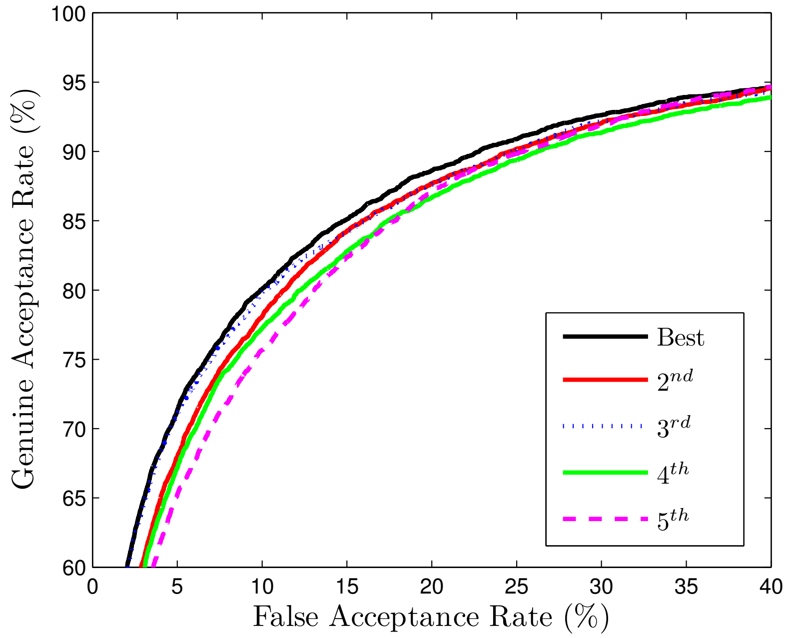

Figure 5.8: **Partial ROC plots for the best five combinations of parameters.** In some segments of the ROC, specially the second and third best, it is not evident which one is better. The third best combination has better GAR for low FAR values, although its EER is bigger. The EER's of all ROC curves are depicted in table 5.3.

Factors influencing construction of texton dictionary are also important. Although the number of clusters appears to have less influence, choosing an appropriate distance measure is critical. The same is valid for assigning a centroid to a data-point.

Regarding histogram matching, cosine distance appears to be more appropriate. However, as this can be considered a classification problem, other methodologies could be used, such as Support Vector Machines (SVM) or Neural Networks.

Chapter 6

Conclusions

There are some key aspects to discuss about this thesis in general. First, it is important to mention that the main objective of this work — implementation of a palmprint recognition system — was successfully accomplished.

The approach was based on texture along principal lines. Therefore, the first part of the work is devoted to principal line detection. Two methods were implemented using graph search methodologies. Although performance of the two methods is similar, the second method — dynamic tracking shortest paths — is of particular interest because it can be easily improved with some easy modifications. Wrong detections often occur when the track of line is lost, but using knowledge of frequent locations of principal lines can reduce the errors. For example, in some cases the predicted lines cross each other — an error that can be easily acknowledged. If some spatial rules are implemented, forcing the algorithm to be repeated in different conditions, the correct lines might be detected.

To date, this was the only application of graph search methods in line detection for palmprint recognition purposes. Unlike other approaches, the developed method can produce fully connected lines which is a major improvement. It's main advantage is that it is easier to integrate with other procedures that require detected lines as input, such as the texture extraction system implemented in chapter 5.

Palm line based recognition systems were developed in the early years of research in the field. However, the line detection schemes used at the time were very basic. The present work might serve as a base for future developments in palmprint principal line detections, enabling more methods based in palm lines to be created.

Regarding the recognition part of the project described in chapter 5, some innovations were introduced to palmprint recognition. First is the use of Haralick's texture features, which were proven to be useful. Secondly, the texton dictionary concept was used, which is a recent technique to reduce and optimize the feature space.

However, the recognition rates are below the performance of other methods described in the literature. This is mainly because the implemented method does not consider spacial information about textures of palmprint lines — the features are globally present in the whole palmprint. Another reason is because each palmprint is represented by an histogram of a reduced number of textons (50 for an EER of 16.64%), built from a reduced number of data-points (130 on average).

There is a lot of room for improvements on this method, specially if spatial information is considered. Also, the use of more powerful classifiers, such as Support Vector Machines (SVM), can be explored. All in all, chapter 5 constitutes a solid base for a new school in palmprint feature extraction and matching.

A particular aspect is that the use of spatial information requires a coordinate system to guarantee alignment of different palmprints. A big improvement to these systems would be the integration of principal lines in the establishment of the coordinate system. In this way skin stretching deformations would be neutralized as principal line position might uncover such deformations. That is a field where the work provided in this thesis might influence future research, as the framework for a better line detection system has been established.

References

[1] A. Jain, Lin Hong, and R. Bolle. On-line fingerprint verification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):302 –314, April 1997.

[2] Davide Maltoni and Rafiaele Cappelli. Fingerprint recognition. In Anil K. Jain, Patrick Flynn, and Arun A. Ross, editors, *Handbook of Biometrics*, pages 23–42. Springer US, 2008.

[3] M. Shahnazi, M. Pahlevanzadeh, and M. Vafadoost. Wavelet based retinal recognition. In *Signal Processing and Its Applications, 2007. ISSPA 2007. 9th International Symposium on*, pages 1 –4, February 2007.

[4] Anil K. Jain, Salil Prabhakar, and Sharath Pankanti. On the similarity of identical twin fingerprints. *Pattern Recognition*, 35(11):2653 – 2663, 2002.

[5] F. Plante, G. Meyer, and W.A. Ainsworth. Speech signal analysis with reallocated spectrogram. In *Time-Frequency and Time-Scale Analysis, 1994., Proceedings of the IEEE-SP International Symposium on*, pages 640 –643, October 1994.

[6] D. Zhang, Guangming Lu, Wei Li, Lei Zhang, and Nan Luo. Three dimensional palmprint recognition using structured light imaging. In *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, pages 1 –6, 292008-oct.1 2008.

[7] Zhenan Sun, Yunhong Wang, Tieniu Tan, and Jiali Cui. Improving iris recognition accuracy via cascaded classifiers. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35(3):435 –441, August 2005.

[8] Yi-Bo Zhang, Qin Li, Jane You, and Prabir Bhattacharya. Palm vein extraction and matching for personal authentication. In *Proceedings of the 9th international conference on Advances in visual information systems*, VISUAL'07, pages 154–164, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] A. Jain and J. Feng. Latent palmprint matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31:1032–1047, June 2009.

[10] Ruud Bolle and Sharath Pankanti. *Biometrics, Personal Identification in Networked Society: Personal Identification in Networked Society*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[11] Adams Kong, David Zhang, and Mohamed Kamel. A survey of palmprint recognition. *Pattern Recognition*, 42(7):1408 – 1418, 2009.

[12] Lei Zhang and D. Zhang. Characterization of palmprints by wavelet signatures via directional context modeling. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(3):1335 –1347, june 2004.

[13] Ying-Han Pang, T. Connie, A. Jin, and D. Ling. Palmprint authentication with zernike moment invariants. In *Signal Processing and Information Technology, 2003. ISSPIT 2003. Proceedings of the 3rd IEEE International Symposium on*, pages 199 – 202, dec. 2003.

[14] Yanlai Li, Kuanquan Wang, and David Zhang. Palmprint recognition based on translation invariant zernike moments and modular neural network. In Jun Wang, Xiao-Feng Liao, and Zhang Yi, editors, *Advances in Neural Networks – ISNN 2005*, volume 3497 of *Lecture Notes in Computer Science*, pages 822–822. Springer Berlin / Heidelberg, 2005.

[15] D. Zhang, Wai-Kin Kong, J. You, and M. Wong. Online palmprint identification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1041 – 1050, sept. 2003.

[16] Wenxin Li, David Zhang, and Zhuoqun Xu. Palmprint identification by fourier transform. *IJPRAI*, 16(4):417–432, 2002.

[17] Chin-Chuan Han. A hand-based personal authentication using a coarse-to-fine strategy. *Image and Vision Computing*, 22(11):909 – 918, 2004.

[18] Chin-Chuan Han, Hsu-Liang Cheng, Chih-Lung Lin, and Kuo-Chin Fan. Personal authentication using palm-print features. *Pattern Recognition*, 36(2):371 – 381, 2003.

[19] Xiangqian Wu, Kuanquan Wang, and David Zhang. Hmms based palmprint identification. In David Zhang and Anil Jain, editors, *Biometric Authentication*, volume 3072 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin / Heidelberg, 2004.

[20] Carmen Poon, David Wong, and Helen Shen. Personal identification and verification: Fusion of palmprint representations. In David Zhang and Anil Jain, editors, *Biometric Authentication*, volume 3072 of *Lecture Notes in Computer Science*, pages 1–40. Springer Berlin / Heidelberg, 2004.

[21] A. Kumar and H.C. Shen. Palmprint identification using palmcodes. In *Image and Graphics, 2004. Proceedings. Third International Conference on*, pages 258 – 261, dec. 2004.

[22] Jane You, Wenxin Li, and David Zhang. Hierarchical palmprint identification via multiple feature extraction. *Pattern Recognition*, 35(4):847 – 859, 2002.

[23] Wei Li, Lei Zhang, D. Zhang, and Jingqi Yan. Principal line based icp alignment for palmprint verification. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1961 –1964, nov. 2009.

[24] De-Shuang Huang, Wei Jia, and David Zhang. Palmprint verification based on principal lines. *Pattern Recogn.*, 41:1316–1328, April 2008.

[25] Dapeng Zhang and Wei Shu. Two novel characteristics in palmprint verification: datum point invariance and line feature matching. *Pattern Recognition*, 32(4):691 – 702, 1999.

[26] Xiangqian Wu, Kuanquan Wang, and David Zhang. Fuzzy directional element energy feature (fdeef) based palmprint identification. *Pattern Recognition, International Conference on*, 1:10095, 2002.

[27] Ajay Kumar and David Zhang. Personal authentication using multiple palmprint representation. *Pattern Recogn.*, 38:1695–1704, October 2005.

[28] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004.

[29] G.S. Badrinath and P. Gupta. Palmprint based verification system robust to rotation, scale and occlusion. In *Computers and Information Technology, 2009. ICCIT '09. 12th International Conference on*, pages 408 –413, dec. 2009.

[30] Jin Soo Noh and Kang Hyeon Rhee. Palmprint identification algorithm using hu invariant moments and otsu binarization. In *Computer and Information Science, 2005. Fourth Annual ACIS International Conference on*, pages 94 – 99, 2005.

[31] Xianji Wang, Haifeng Gong, Hao Zhang, Bin Li, and Zhenquan Zhuang. Palmprint identification using boosting local binary pattern. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 503 –506, 0-0 2006.

[32] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971 –987, jul 2002.

[33] Guang-Ming Lu, Kuan-Quan Wang, and D. Zhang. Wavelet based independent component analysis for palmprint identification. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 6, pages 3547 – 3550 vol.6, aug. 2004.

[34] Qingyun Dai, Ning Bi, Daren Huang, Dvaid Zhang, and Feng Li. M-band wavelets application to palmprint recognition based on texture features. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 2, pages 893 – 896 Vol.2, oct. 2004.

[35] Xiangqian Wu, David Zhang, and Kuanquan Wang. Fisherpalms based palmprint recognition. *Pattern Recognition Letters*, 24(15):2829 – 2838, 2003.

[36] Guangming Lu, David Zhang, and Kuanquan Wang. Palmprint recognition using eigenpalms features. *Pattern Recognition Letters*, 24(9-10):1463 – 1467, 2003.

[37] Yu Tao, Ernest C. M. Lam, and Yuan Y. Tang. Feature extraction using wavelet and fractal. *Pattern Recognition Letters*, 22(3-4):271 – 287, 2001.

[38] K. Ito, T. Aoki, H. Nakajima, K. Kobayashi, and T. Higuchi. A palmprint recognition algorithm using phase-based image matching. In *Image Processing, 2006 IEEE International Conference on*, pages 2669 –2672, oct. 2006.

[39] A.W.-K. Kong and D. Zhang. Competitive coding scheme for palmprint verification. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 520 – 523 Vol.1, aug. 2004.

[40] Adams Kong and David Zhang. Feature-level fusion for effective palmprint authentication. In David Zhang and Anil Jain, editors, *Biometric Authentication*, volume 3072 of *Lecture Notes in Computer Science*, pages 1–43. Springer Berlin / Heidelberg, 2004.

[41] Zhenan Sun, Tieniu Tan, Yunhong Wang, and S.Z. Li. Ordinal palmprint represention for personal identification [represention read representation]. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 279 – 284 vol. 1, june 2005.

[42] Rufeng Chu, Zhen Lei, Yufei Han, Ran He, and Stan Z. Li. Learning gabor magnitude features for palmprint recognition. In *Proceedings of the 8th Asian conference on Computer vision - Volume Part II*, ACCV'07, pages 22–31, Berlin, Heidelberg, 2007. Springer-Verlag.

[43] Feng Yue, Wangmeng Zuo, Kuanquan Wang, and D. Zhang. Fcm-based orientation selection for competitive coding-based palmprint recognition. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008.

[44] Xinhong Zhou, Yuhua Peng, and Ming Yang. Palmprint recognition using wavelet and support vector machines. In Qiang Yang and Geoff Webb, editors, *PRICAI 2006: Trends in Artificial Intelligence*, volume 4099 of *Lecture Notes in Computer Science*, pages 385–393. Springer Berlin / Heidelberg, 2006.

[45] Polyu palmprint database, "http://www.comp.polyu.edu.hk/ biometrics/".

[46] Casia palmprint database, "http://biometrics.idealtest.org/".

[47] A. Kumar. Incorporating cohort information for reliable palmprint authentication. In *Computer Vision, Graphics Image Processing, 2008. ICVGIP '08. Sixth Indian Conference on*, pages 583 –590, dec. 2008.

[48] Oliveira H. P. Matos H. Magalhães, F. and A. Campilho. Hgc2011 - hand geometric points detection competition database, "http://www.fe.up.pt/ hgc2011/". 2010.

[49] A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):62 –66, jan. 1979.

[50] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679 –698, nov. 1986.

[51] Xiangqian Wu, Kuanquan Wang, and D. Zhang. Fuzzy directional element energy feature (fdeef) based palmprint identification. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 95 – 98 vol.1, 2002.

[52] P.A. Recobos Rodriguez and J.D. Landa Silva. Biometric identification by dermatoglyphics. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 319 –322 vol.1, sep 1996.

[53] L. Liu and D. Zhang. Palm-line detection. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III269 – III272, sept. 2005.

[54] P.V.C. Hough. Machine analysis of bubble chamber pictures. In *Proc. Int. Conf. High Energy Accelerators and Instrumentation*, 1959.

[55] W.W. Boles and S.Y.T. Chu. Personal identification using images of the human palm. In *TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, volume 1, pages 295 –298 vol.1, dec. 1997.

[56] Wei Jia, Yi-Hai Zhu, Ling-Feng Liu, and De-Shuang Huang. Fast palmprint retrieval using principal lines. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 4118 –4123, oct. 2009.

[57] Junlin Hu, Yanxue Xue, Yongwei Zhao, and Jiwen Lu. Palmprint recognition based on multiple feature information fusion. In *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pages 1449 –1452, oct. 2008.

[58] Xiangqian Wu, D. Zhang, and Kuanquan Wang. Palm line extraction and matching for personal authentication. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):978 –987, 2006.

[59] Andrew Kingston and Imants Svalbe. Generalised finite radon transform for n×n images. *Image and Vision Computing*, 25(10):1620 – 1630, 2007. Discrete Geometry for Computer Imagery 2005.

[60] Xiangqian Wu, D. Zhang, and Kuanquan Wang. Palm line extraction and matching for personal authentication. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):978 –987, 2006.

[61] Alberto Martelli. An application of heuristic search methods to edge and contour detection. *Commun. ACM*, 19:73–83, February 1976.

[62] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

[63] Ana Rebelo, Artur Capela, Joaquim F. Pinto da Costa, Carlos Guedes, Eurico Carrapatoso, and Jaime S. Cardoso. A shortest path approach for staff line detection. In J. Delgado, K. Ng, P. Nesi, and P. Bellini, editors, *Third International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS 2007)*, pages 79–85, 2007.

[64] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.

[65] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100 –107, july 1968.

[66] Li Fang, M.K.H. Leung, T. Shikhare, V. Chan, and Kean Fatt Choon. Palmprint classification. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 4, pages 2965 –2969, oct. 2006.

[67] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, 3(6):610 –621, nov. 1973.

[68] Huan Yu, C. Caldwell, K. Mah, and D. Mozeg. Coregistered fdg pet/ct-based textural characterization of head and neck cancer for radiation treatment planning. *Medical Imaging, IEEE Transactions on*, 28(3):374 –383, march 2009.

[69] P. Gamba, M. Aldrighi, and M. Stasolla. Robust extraction of urban area extents in hr and vhr sar images. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 4(1):27 –34, march 2011.

[70] Manik Varma and Andrew Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62:61–81, 2005. 10.1023/B:VISI.0000046589.39864.ee.

[71] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 494–501, New York, NY, USA, 2007. ACM.