Thèse présentée à la faculté des sciences pour l'obtention du grade de docteur ès sciences

Features Extraction for Low-Power Face Verification

Patrick Stadelmann

Acceptée sur proposition du jury :

Prof. Fausto Pellandini, directeur de thèse
PD Dr. Michael Ansorge, co-directeur de thèse
Prof. Pierre-André Farine, rapporteur
Dr. Nicolas Blanc, rapporteur

Soutenue le 23 mai 2008

Institut de Microtechnique
Université de Neuchâtel

Dissertation typeset by the author with LaTeX 2$_\varepsilon$ on Mac OS X.

Written using TeXShop, BibDesk and TeX Live 2008, part of the MacTeX distribution maintained by the MacTeX Technical Working Group. Additional editing made with TextWrangler by Bare Bones Software, Inc. Graphs created using Plot, written by M. Wesemann and B. J. Thijsse. Illustrations drawn in OmniGraffle, published by The Omni Group.

Université
de Neuchâtel

FACULTE DES SCIENCES
Secrétariat-Décanat de la faculté
- Rue Emile-Argand 11
- CP 158
- CH-2009 Neuchâtel

IMPRIMATUR POUR LA THESE

Features Extraction for Low-Power Face Verification

Patrick STADELMANN

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel,
sur le rapport des membres du jury

MM. F. Pellandini (directeur de thèse),
M. Ansorge (co-directeur de thèse, EPF Lausanne),
P.-A. Farine et N. Blanc (CSEM, Zürich)

autorise l'impression de la présente thèse.

Neuchâtel, le 15 septembre 2008

Le doyen :
F. Kessler

UNIVERSITE DE NEUCHATEL
FACULTE DES SCIENCES
Secrétariat - décanat de la faculté
Rue Emile-Argand 11 - CP 158
CH-2009 Neuchâtel

Felix Kessler

Abstract

Mobile communication devices now available on the market, such as so-called *smartphones*, are far more advanced than the first cellular phones that became very popular one decade ago. In addition to their historical purpose, namely enabling wireless vocal communications to be established nearly everywhere, they now provide most of the functionalities offered by computers. As such, they hold an ever-increasing amount of personal information and confidential data. However, the authentication method employed to prevent unauthorized access to the device is still based on the same PIN code mechanism, which is often set to an easy-to-guess combination of digits, or even altogether disabled. Stronger security can be achieved by resorting to biometrics, which verifies the identity of a person based on intrinsic physical or behavioral characteristics.

Since most mobile phones are now equipped with an image sensor to provide digital camera functionality, biometric authentication based on the face modality is very interesting as it does not require a dedicated sensor, unlike e.g. fingerprint verification. Its perceived intrusiveness is furthermore very low, and it is generally well accepted by users. The deployment of face verification on mobile devices however requires overcoming two major challenges, which are the main issues addressed in this PhD thesis.

Firstly, images acquired by a handheld device in an uncontrolled environment exhibit strong variations in illumination conditions. The extracted features on which biometric identification is based must therefore be robust to such perturbations. Secondly, the amount of energy available on battery-powered mobile devices is tightly constrained, calling for algorithms with low computational complexity, and for highly optimized implementations.

So as to reduce the dependency on the illumination conditions, a low-complexity normalization technique for features extraction based on mathematical morphology is introduced in this thesis, and evaluated in conjunction with the Elastic Graph Matching (EGM) algorithm. Robustness to other perturbations, such as occlusions or geometric transformations, is also assessed and several improvements are proposed. In order to minimize the power consumption, the hardware architecture of a coprocessor dedicated to features extraction is proposed and described in VHDL. This component is designed to be integrated into a System-on-Chip (SoC) implementing the complete face verification process, including image acquisition, thereby enabling biometric face authentication to be performed entirely on the mobile device. Comparison of the proposed solution with state-of-the-art academic results and recently disclosed commercial products shows that the chosen approach is indeed much more efficient energy-wise.

Keywords

Image processing, biometrics, face recognition, elastic graph matching, mathematical morphology, mobile devices, ASIC architectures, low-power VLSI design, VHDL description, FPGA implementation.

Résumé

Les appareils de communication portables actuellement sur le marché, comme par exemple les *smartphones*, sont bien plus évolués que les premiers téléphones cellulaires devenus populaires il y a une décennie. En plus de leur but premier, à savoir permettre l'établissement de communications vocales pratiquement partout, ils offrent maintenant la plupart des fonctionnalités d'un ordinateur et renferment ainsi de plus en plus d'informations personnelles et de données confidentielles. La vérification d'identité visant à empêcher l'accès aux personnes non autorisées est cependant toujours basée sur le code NIP qui est souvent facile à deviner, ou même complètement désactivé. La sécurité peut être accrue en utilisant la biométrie, qui identifie les personnes en se basant sur leurs caractéristiques physiques ou comportementales.

La majorité des téléphones portables actuels intégrant un capteur d'images pour permettre la prise de photographies, la reconnaissance biométrique du visage est très intéressante car elle ne requiert pas de capteur dédié, au contraire par exemple de la reconnaissance des empreintes digitales. De plus, elle n'est pas jugée invasive et est généralement bien acceptée des utilisateurs. Avant de pouvoir déployer la reconnaissance de visages sur un terminal portable, il faut cependant surmonter deux obstacles majeurs, qui sont les principaux sujets étudiés dans cette thèse.

Premièrement, les conditions d'illumination des images capturées par un appareil tenu en main dans un environnement non contrôlé peuvent grandement varier. Les caractéristiques extraites pour servir de base à la reconnaissance biométrique doivent donc être robustes en présence de telles perturbations. Deuxièmement, la quantité d'énergie disponible sur un appareil portable alimenté par batteries est très limitée, et nécessite des algorithmes à faible complexité et dont l'implantation est fortement optimisée.

Pour réduire l'influence de l'illumination, une technique de normalisation à faible complexité pour l'extraction de caractéristiques basée sur la morphologie mathématique est proposée dans cette thèse. La méthode est évaluée en combinaison avec l'algorithme de mise en correspondance de graphes élastiques, *Elastic Graph Matching (EGM)* en anglais. La robustesse à d'autres perturbations, telles que des occlusions ou des transformations géométriques, est également jaugée et plusieurs améliorations sont proposées. Afin de minimiser la puissance dissipée, une architecture matérielle pour un coprocesseur dédié à l'extraction de caractéristiques est présentée et décrite en VHDL. Ce composant est conçu pour s'intégrer dans un système sur puce réalisant la totalité du processus de vérification, y compris l'acquisition d'image, ce qui permet d'effectuer la reconnaissance biométrique du visage entièrement sur le terminal portable. La comparaison de cette solution avec l'état de l'art au niveau académique et avec des produits commerciaux récents indique que l'approche choisie est bien plus efficace en termes de consommation d'énergie.

Mots-clés

Traitement d'images, biométrie, reconnaissance de visages, mise en correspondance de graphes élastiques, morphologie mathématique, terminaux portables, architectures ASIC, conception VLSI à basse consommation, description VHDL, implantation FPGA.

Contents

Abbreviations

ALU	Arithmetic Logic Unit
AMD	Advanced Micro Devices
APS	Active Pixel Sensor
ARM	Advanced RISC Machine / Acorn RISC Machine
ASCII	American Standard Code for Information Interchange
ASI	Automatic Speaker Identification
ASIC	Application Specific Integrated Circuit
ASV	Automatic Speaker Verification
ATM	Automated Teller Machine
AVBPA	Audio- and Video-Based Biometric Person Authentication
BNF	Backus-Naur Formalism
C	Command (signal)
CAS	Chinese Academy of Science
CMOS	Complementary Metal-Oxide Semiconductor
COST	European Cooperation in the Field of Scientific and Technical Research
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
CPY	Copy (opcode)
CSEM	Centre Suisse d'Electronique et de Microtechnique
DCT	Discrete Cosine Transform
DE9	D9 – E9 (opcode)
DEC	Decrement (opcode)
DHCP	Dynamic Host Configuration Protocol

DLA	Dynamic Link Architecture
DLM	Dynamic Link Matching
$\mathrm{D_N}$	Dilation of Level N
DNA	Deoxyribonucleic Acid
DnE	Dilation Not Erosion (signal)
Dout	Dilation Output (register)
DR	Delta – Range (opcode)
dSE	Differential Structuring Element
$\mathrm{dSE_N}$	Differential Structuring Element of Level N
DSP	Digital Signal Processor
DWT	Discrete Wavelet Transform
EBGM	Elastic Bunch Graph Matching
EDIF	Electronic Design Interchange Format
EER	Equal Error Rate
EGM	Elastic Graph Matching
EMU	Elementary Morphology Unit
$\mathrm{E_N}$	Erosion of Level N
Eout	Erosion Output (register)
ESPLAB	Electronics and Signal Processing Laboratory, IMT
EXT ADDR	External Addressing Unit
FAR	False Acceptance Rate
FDA	Fisher Discriminant Analysis
FERET	Facial Recognition Technology
FFT	Fast Fourier Transform
FLD	Fisher's Linear Discriminant
FMR	False Match Rate
FNMR	False Non Match Rate
FOMA	Freedom of Mobile Multimedia Access
FPGA	Field-Programmable Gate Array
FRR	False Rejection Rate
FSE	Face Sensing Engine
GHT	Generalized Hough Transform
HIIDE	Handheld Interagency Identity Detection Equipment
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
ICA	Independent Component Analysis

ICB	International Conference on Biometrics
ICPR	International Conference on Pattern Recognition
ID	Incrementer-Decrementer
IDNORM	Incrementer-Decrementer for Normalization
ILOG	Inverse Logarithm Operation (opcode)
IMT	Institute of Microtechnology, Univ. of Neuchâtel
INC	Increment (opcode)
INVLOG	Inverse Logarithm
INVLOGMEM	Inverse Logarithm Memory
IP	Internet Protocol
JPEG	Joint Photographic Experts Group
JSR	Jump to Subroutine (opcode)
JTAG	Joint Test Action Group
KLT	Karhunen-Loève Transform
L	Level
LC	Load Column (register)
LD	Load (opcode)
LDA	Linear Discriminant Analysis
LE	Logic Element
LEM	Line Edge Map
LFA	Local Features Analysis
LMEM	Local Memory
LO	Load Offset (register)
LOG	Logarithm
LOG	Logarithm Operation (opcode)
LOGMEM	Logarithm Memory
LR	Load Row (register)
LSB	Least Significant Bit
LSU	Load Store Unit
LUT	Lookup Table
LWIN	Local Window
mA	Milliampere
MAC	Multiply-Accumulate
MAS	Modulo Adder-Subtracter
MB	Morphology Block
MC	Morphology operator C

MCU	Master Control Unit
MDF	Most Discriminant Features
MDLA	Morphological Dynamic Link Architecture
ME9	M − E9 (opcode)
MEF	Most Expressive Features
MEGM	Morphological Elastic Graph Matching
MHz	Megahertz
MIMD	Multiple Instructions Multiple Data
MISD	Multiple Instruction Single Data
mm	Millimeter
MM	Morphology operator M
MMU	Mathematical Morphology Unit
MO	Morphology Output (opcode)
MSB	Most Significant Bit
MSD	Morphological Signal Decomposition
mW	Milliwatt
NLI	Normalization Logarithm Increment
NLIC	Normalization Logarithm Increment Copy
NOP	No Operation
NORM	Normalization (register)
NTT	Nippon Telegraph and Telephone
OCR	Optical Character Recognition
PC	Personal Computer
PC	Program Counter
PCA	Primary Component Analysis
PDA	Personal Digital Assistant
$\mathrm{pD_N}$	Partial Dilation of Level N (register)
$\mathrm{pE_N}$	Partial Erosion of Level N (register)
PIN	Personal Identification Number
POS	Position (signal)
PSO	Previous Store Offset
RAM	Random Access Memory
RET	Return
RGB	Red Green Blue
RISC	Reduced Instruction Set Computing
RM	Reset Morphology (opcode)

ROC	Receiver Operating Characteristic
RSO	Reset Store Offset (opcode)
RTC	Return to Caller
RTL	Register Transfer Level
RTOS	Real-time Operating System
SAIF	Switching Activity Interchange Format
SC	Store Column
SDF	Standard Delay Format
SE	Structuring Element
SEIDX	Structuring Element Index (register)
SEMEM	Structuring Element Memory
SE$_{\mathrm{N}}$	Structuring Element of Level N
SEOUT	Structuring Element Output (register)
SIM	Subscriber Identity Module
SIMD	Single Instruction Multiple Data
SO	Store Offset
SR	Store Row
SRAM	Static Random Access Memory
ST	Store
TC	Temporary Column (register)
TCP	Transport Control Protocol
TR	Temporary Row (register)
TTF	Tabular Text File
UI	User Interface
UMC	United Microelectronics Corporation
UniNE	University of Neuchâtel
UNIS	University of Surrey
USB	Universal Serial Bus
V	Volt
VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuits
VLIW	Very Long Instruction Word
VLSI	Very Large Scale Integration
µA	Microampere
µm	Micrometer
µW	Microwatt

Chapter 1

Introduction

1.1 Motivations and objectives

In the last decade, mobile devices dedicated to communication, such as cellular phones, and those dedicated to information management, such as Personal Digital Assistants (PDAs), have become an integral part of our daily life. Technological innovations furthermore led to the convergence of these two kinds of devices, resulting for instance in the emergence of the so-called smartphones. Beside the original purpose of a mobile telephone, namely enabling wireless vocal communication between people, such devices offer many additional functionalities, so much that they can effectively be considered as miniaturized computers. As such, they hold an ever-increasing amount of personal — and possibly confidential — information. Moreover, they enable remote access to highly sensitive services, such as e-banking applications and paid-for online services. These devices are naturally meant to be carried along by their owner at all time, and are obviously designed to be very small and lightweight. These characteristics render them very prone to theft and loss, and therefore to various kinds of abuse that can potentially result in very damaging consequences. Whereas wireless communication networks resort to strong cryptographic techniques to guarantee the confidentiality of the transmitted data, the access to the mobile terminal itself is usually protected by a simple Personal Identification Number (PIN) code. Studies have shown that many users employ easily memorable digit combinations, such as their year of birth, or altogether deactivate this protection. There thus exists a strong need for alternate or complementary security measures, capable of providing enhanced access control to mobile devices with minimal impact on the user experience.

Biometric authentication methods constitute an interesting solution, as they verify the identity of a person based on some intrinsic characteristics that can neither be forgotten, nor stolen, nor lost. Since most mobile devices offer still and video camera functionalities, they are already equipped with a digital image sensor. Biometric authentication based on the face modality can therefore be implemented without requiring a dedicated sensor, as it would be the case e.g. for fingerprint recognition, which sensibly limits the impact on the manufacturing cost of the device.

Several major challenges incurred by the mobile nature of the foreseen applications nevertheless need to be overcome, which constitutes the main concerns of this PhD thesis. Firstly, in a scenario where the face images are acquired by a handheld device in an uncontrolled environment, strong variations in illumination conditions can be expected. Hence, the proposed solution must be robust to such perturbations. Secondly, mobile devices are powered by rechargeable or disposable batteries, and the amount of energy available is therefore severely constrained, so that methods with reasonable computational complexity are needed. Furthermore, the selected algorithms must be implemented very carefully, so as to minimize the energy they consume. In this context, dedicated hardware processing units offer the most efficient solutions, especially if they are all integrated into one single silicon circuit to form a System-on-Chip, thereby reducing the amount of external connections, and consequently also the power consumption and the manufacturing cost.

1.2 Context

A large part of the research activities presented here were carried out in the framework of several research projects financially supported by the Swiss Center for Electronics and Microtechnology (CSEM SA), which led to a first PhD thesis being published by Jean-Luc Nagel [1] in 2006. Even though the present report aims at providing all the information necessary to fully apprehend the discussed work and proposed solutions, on some occasions, the reader will be referred to the prior PhD thesis for additional details on specific points.

1.3 Structure of the report

Chapter 2 introduces the topic of biometric recognition, defines the related terminology and presents the existing modalities, which are then evaluated in light of the specific requirements of low-power mobile applications.

In Chapter 3, existing face authentication algorithms are discussed and the Morphological Elastic Graph Matching (MEGM) algorithm gets selected as the most interesting solution for mobile applications. In view of enhancing the robustness to variations in illumination conditions, a novel morphology features normalization step is then introduced and evaluated.

Chapter 4 reports the results of several experiments carried out to assess the performance and robustness of the MEGM algorithm, both under optimal and degraded image acquisition conditions.

In Chapter 5, a System-on-Chip structure enabling face verification to be performed on mobile devices is described, and the hardware architecture of a coprocessor dedicated to mathematical morphology features extraction is detailed.

In Chapter 6, an enhanced version of the coprocessor is introduced, that furthermore implements features normalization.

Chapter 7 presents an FPGA-based demonstrator enabling real-time validation of the architecture, and provides the power consumption and silicon die area obtained after ASIC synthesis.

Finally, Chapter 8 briefly discusses the existing state-of-the-art academic and commercial solutions for mobile face recognition, and concludes by outlining possible future research directions.

1.4 Contributions

The major contributions of this PhD work are:

- Introduction of a low-complexity normalization technique for mathematical morphology-based features, that reaches the same level of performance as Gabor-based ones.

- Description of a low-power architecture to efficiently extract mathematical morphology features.

- Description of an improved architecture implementing a features normalization step, and realization of the corresponding VLSI implementation.

Other contributions include:

- Evaluation of the robustness of the MEGM face verification algorithm under different kinds of degraded conditions.
- Introduction and evaluation of several heuristic approaches to enhance the robustness in presence of shadows.
- Realization of an FPGA-based demonstrator enabling real-time validation of the proposed VLSI architecture.

Chapter 2

Biometrics

2.1 Introduction

This chapter proposes an introduction to the topic of biometric recognition. In Section 2.2, the meaning of biometrics is defined and discussed. A short historical background follows in Section 2.3, which provides a chronological synopsis of selected major advances in this field and describes what can be considered to be the first true biometric system. The general concepts related to biometric recognition systems are then introduced in Section 2.4, including identification and definition of the main tasks needing to be performed. Some applications of biometric systems are then presented, and methods for evaluating and comparing their performance are discussed. Several biometric modalities are then listed and briefly described in Section 2.5. The specific requirements and constraints pertaining to the application of biometrics in the context of mobile environments are then exposed in Section 2.6. Finally, the formerly enumerated modalities are discussed again in light of the requirements of mobile applications, so as to identify the most suitable for implementation on mobile devices.

2.2 Definition

The term *biometrics* appears in the early 20th century, and is at first employed as a synonym of *biometry*, formed from the ancient Greek words *bios* (life) and *metrikos* (measure). Defined as "*the statistical analysis of biological observations and phenomena*" [2], biometry has many applications in diverse fields, such as biology, epidemiology or agriculture [3]. However, in the last decades of the 20th century, the term *biometrics* started to refer specifically to technologies aiming at identifying human beings based on unique traits such as fingerprint, voice pattern or hand geometry. The Biometric Consortium [4] proposes the following definition:

> *Biometrics are automated methods of recognizing a person based on a physiological or behavioral characteristic.*

Such a characteristic is called a *biometric* or a *biometric modality.* It can be pointed out that the terms *physical* or *anatomical* would actually be more appropriate than *physiological.* Indeed, the latter specifically relates to the functions and activities of living matter, and to the physical and chemical processes involved therein. Fingerprint or hand geometry for instance, two common biometric traits, are strictly speaking neither of behavioral nor of physiological nature; they are purely anatomical. Some modalities like voice are both behavioral and physical. To refer to purely behavioral characteristics, such as mouse gestures and computer keystroke, the tentative term *behaviometrics* has been recently proposed [5].

To qualify as a biometric modality, a characteristic must have certain properties [6]. It must be universal, which means that each person should possess it, and it must also be permanent. Furthermore, it also has to be collectible, implying that it can be quantitatively measured by some means. Finally, it must be distinctive, namely that the trait must be sufficiently different between any two persons.

2.3 History

Even though the subject of biometrics gained tremendous interest in the recent years, the desire or need to effectively assess the identity of a given person exists since ancient history. There are evidences for instance suggesting that the finger imprint discovered on a Chinese clay seal dating from 400 B.C. was used as an identification mean [7]. Fingerprints have also been found on archaeological artifacts documenting commercial transactions by Babylonians [8]. In the latter half of the 19th century, evolutions in occidental societies and in their criminal justice systems in particular, establish the need for a reliable method of identifying individuals. Indeed, criminal courts seek to treat first time offenders more clemently than repeat recidivists.

2.3.1 Bertillon's system

In 1881, Alphonse Bertillon, a French law enforcement officer, introduces a criminal identification system based on what he named *anthropometry*, designed to help identifying repeat offenders [9], as many of them were presenting themselves under a different identity each time they were arrested. Even though their picture was taken and filed, there was no coherent way of organizing the tens of thousands of collected photographs. Fingerprints, though already employed for forensic identification, were not helpful either, since the first classification methods were only established at the end of the century [7]. Thus, the difficulty was not in organizing the collection and archiving of information, but rather in having an efficient retrieval method available. *Bertillon's system* — commonly referred to as *bertillonage* — comprises both a measurement procedure and a sorting methodology designed to offer a "versatile means of classifying hundreds of thousands of cards" [10]. As such, it bears many similarities with modern automated identification systems, although it was operated completely manually.

In Bertillon's system, a paper card is established to add a record for a new individual. It holds the purported identity as well as two photographs of the person's face, one side-view and one frontal-view. The results of a series of body observations and measurements, such as the height, the color of the eyes and the length of the forearm are also listed on the card. Each card is then classified, firstly by attributing it to one out of three categories for the criteria "height", either short, medium, or tall. The same operation is then carried out for "size of the head", labeled as being either small, medium, or large, and the remaining characteristics are processed similarly. Thus, each card is eventually stored within a small set of records describing individuals with similar characteristics. As a result, a collection of 60'000 cards could ultimately be reduced into packets of approximately ten cards [11].

To determine whether a person has been previously arrested and filed, the officer performs a series of measurements on the subject, and uses them to quickly select one group of cards. He then goes through each record, looking for any matching unusual characteristic, such as scars or tattoos that are also described on the cards. Finally, the photographs on the few cards that remain after sorting are used for direct visual identification. Bertillon's system was quickly adopted in France as well as in several foreign countries, but was nevertheless abandoned early in the 20$^{\text{th}}$ century, once it was established that it could easily confound distinct individuals. It was quickly supplanted by fingerprints, shown to be far more discriminative [12]. Furthermore, whereas the latter can serve to identify perpetrators based on latent traces left on crime scenes, Bertillon's system is of no use in such cases.

2.3.2 Modern biometrics

In the middle of the 1960s, the first research efforts were initiated to use computer systems to automate the process of biometric identification. In 1963, a paper titled "Automatic Comparison of Finger-Ridge Patterns" [13] is published in *Nature* and probably constitutes the earliest reported attempt dedicated to such a task [14]. The following year, Bledsoe, Chan and Bisson began working on semi-automated face recognition [15], clearly inspired by the approach followed by Bertillon (see Subsection 3.2.1). In 1965, signature verification is the first behavioral characteristic considered, as reported in [16]. The list continues to grow in the 1970s with the apparition of systems based on hand geometry and speaker verification, followed by iris recognition in the 1980s, whereas vascular patterns and gait started to be considered in the 1990s. In the meantime, many systems based on fingerprint or hand geometry have been commercially deployed, sometimes at very large scale.

2.4 Biometric system

The purpose of biometric systems consists either in verifying or in determining the identity of people, based on their intrinsic physical or behavioral characteristics. The other non-biometric authentication methods can be grouped into two categories, each one being affected by drawbacks that do not exist with biometrics. In the first one, the operation is carried out using an object that the person is possessing, such as a key or a card. Obviously, what is really authenticated here is the object itself: a lock securing access to a building for instance verifies that the key being employed, and assumed original, is effectively authorized to open the door. The key can get lost, preventing access by a legitimate individual, or stolen by a third-party that would thus gain unauthorized access to the facility. The second category relies on a particular information, known only to legitimate users, such as a password or code. Unless the information gets written down, theft is of lesser concern, but a password can nevertheless be forgotten or unduly communicated to others, or cracked.

Biometric characteristics, on the other hand, can neither be lost, nor stolen, nor forgotten. Furthermore, they truly identify a person instead of a knowledge or an object. Like keys, some biometric traits can nevertheless be counterfeit. Procedures for creating fake gummy fingerprints for instance can easily be found on the Internet. Yet in most cases, counter measures can be implemented if such attacks are to be feared. In other situations, such protection can be deemed superfluous, for instance if the goal of a face authentication system is to protect a mobile device in case it gets stolen. It is indeed unlikely that the thief would possess a photograph of the legitimate owner anyway.

2.4.1 Overview

The basic block diagram of a biometric system is shown in Figure 2.1. It comprises a sensor, whose purpose is to acquire some kind of information from an individual (e.g. an image of the fingerprint pattern). Two main processing elements can be identified, namely features extraction and matching. The first operation consists in processing the acquired data to obtain a kind of numerical signature identifying the subject, which is stored in a database. The second operation compares a signature retrieved from the database with a newly acquired one, to decide whether the two belong to the same person.

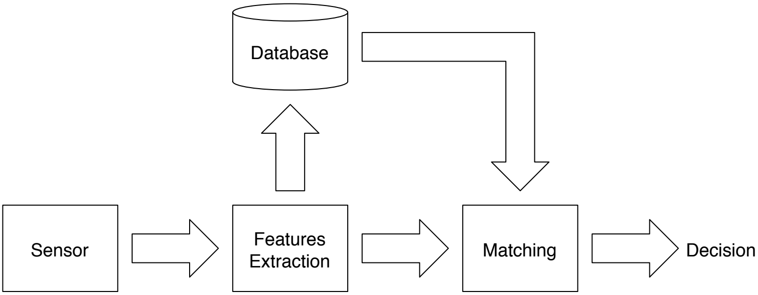


Figure 2.1 : Basic bloc diagram of a biometric system.

2.4.2 Operations

When considering the operation of a biometric system, two distinct phases can be identified: an *enrollment* procedure where the system learns about the identities of the users, and a recognition phase, where it actually attempts to determine said identities. For the second phase, two modes of operation can further be distinguished: *verification*, also called authentication, and *identification*. Depending on its expected applications, a biometric system can be either restricted to a single mode, or designed to operate in both. These three identified tasks, covering enrollment, verification and identification, are described below.

2.4.2.1 Enrollment

The enrollment is the procedure by which the identity of an individual, e.g. its name, is entered in a database, along with one or several templates, the latter consisting of biometric information collected from the person being enrolled. Depending on the application, this process can be performed partly manually, the user having for instance to help the system locate the eyes or the mouth in case the face modality is used, which assumes that the user is cooperating. In other situations, when the enrollment must be performed in a time interval of maximum one second e.g. due to the number of persons to process, the enrollment must be fully automated. This is the case for instance for the fingerprint system deployed at several major U.S. theme parks, that associates a ticket to its holder upon first entrance.

2.4.2.2 Verification

In verification mode, a biometric system matches an individual against a claimed identity. The corresponding template is retrieved from the database, and matched against the features extracted on-line. The latter are thus only compared against one single template, a process known as 1:1 matching. Consequently, the number of people enrolled in the database does not affect the performance of the system, neither in terms of computation time, nor in terms of accuracy. As already mentioned, this mode is also called authentication. The result of this process is usually a Boolean value, indicating whether the claimed identity is correct or not, possibly accompanied by a measure of confidence of the decision.

2.4.2.3 Identification

Conversely to the verification mode, no claimed identity is furnished to the system in identification mode. Here, the identity of the subject must be determined based on his biometric characteristics. To this effect, the extracted features are compared against those of all enrolled clients, a process known as 1:N matching, until a correspondence is found. The chances of misidentifying the individual obviously increase with the number of people being enrolled in the database, as does also the computation time. To accelerate the processing, an ordered database can be used, were individuals are grouped in subsets according to some easily measurable traits[1]. For instance, a system based on hand geometry could start by measuring the length of a given finger, and then in a first step, process only the templates associated to the corresponding finger having approximately the same size.

In this mode, the system can return the identity of the best match, implying that the comparison was performed against every enrolled people, or the system returns the first match to be sufficiently good. In the first situation, the operation executed effectively consists in classifying the test subject among N classes, each one corresponding to one identity. The situation where the tested person was not enrolled can be handled either by returning the best matched identity, or if the best match is considered as being not good enough, by returning a special code to indicate that the person was not recognized by the system.

Although the term *recognition* is sometimes used as a synonym of identification when referring to specific systems or algorithms, it is often used to mean either verification or identification, or both, especially when employed in a more general context. In this report, the term recognition is globally referring to both verification and identification methods, unless otherwise specified.

[1] Similarly to Bertillon's system, see Subsection 2.3.1.

2.4.3 Applications

Applications of biometrics are numerous and diverse, the most common one being of course access control. A biometric system can secure the entrance of a building, effectively restricting access to authorized persons only. It can also be employed to verify the identity of people entering a country, as part of a border control program. Conversely, it can be used to ensure that visitors exiting a facility such as a prison are really not inmates trying to escape. Instead of referring to physical places, access control can also concern confidential information, such as private data stored on a computer. Finally, it can also prevent unauthorized people from using a device, such as a car, for instance by requiring biometric authentication to trigger ignition.

Although many fields of application of biometrics, such as those enumerated above, are connected to security, other uses nevertheless exist. Among them, one can mention so-called smart environments or devices, which have the ability to automatically configure themselves for the identified user. In such a scenario, a car shared among several persons could for instance adapt the position of the seat and of the rear-view mirrors to each driver as he takes place behind the steering wheel. Additional settings could include selecting the radio station he listened to the last time he drove the car.

2.4.4 Performance evaluation

The biometric data extracted from a test subject and the reference template recorded at enrollment time are never exactly the same. Indeed, several perturbations affect the measures carried out by biometric sensors. Firstly, like all living creatures, human beings exhibit constantly evolving physical characteristics, some traits changing more deeply or more rapidly than others. Patterns formed by the ridges found on the fingertips, for instance, are stable even over a relatively long period of time. Conversely, the same face can appear sensibly different after only a few days, due to fatigue or stress affecting facial musculature. Moreover, biological phenomena like beard growing, or aging, also produce local changes in aspect. Finally, medical conditions such as cold or flu, as well as living habits (most notably smoking) strongly modify the sound of the voice.

Even if the physical traits do not vary, external influences can impair the quality of the measure. Some fingerprint sensors for instance produce images that can differ greatly, depending on the state of the finger, as shown in Figure 2.2. Environmental factors also influence the acquisition of biometric characteristics. In the case of face recognition, the intensity and spectral distribution of the incident light affects the pixel responses, as demonstrated in Section 4.2.1 in [1]. The direction of the illumination furthermore determine how the face is affected by cast shadows, potentially degrading the performance of the system, as it will be exposed in Section 4.4 and Section 4.5. Imperfections in the measure itself, caused by technological limitations, sensor noise or imprecise manipulations on the user's part, also result in different values being recorded with each new acquisition. Finally, in some situations, certain characteristics of the sensor used for recognition may differ from those of the sensor that was available when the database was established. This is for instance the case if the same reference templates, stored e.g. on an identity document such as a biometric passport, need to be matched against features acquired by different devices equipped with a variety of sensors.

A biometric system requiring a perfect match between the test and reference features would hardly work under practical conditions. Therefore, variations must be tolerated to a certain extent,

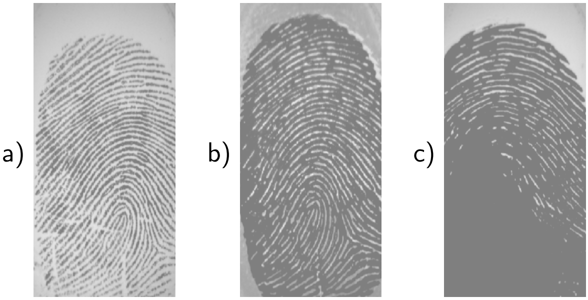


Figure 2.2 : Images of fingerprints in various conditions.
Images of the same finger, taken using the same optical sensor in a constant environment: a) normal finger, b) wet finger and c) greasy finger.

motivating the establishment of a distance threshold to be used to decide whether two distinct measures correspond to the same individual or not. Still, in certain cases, the distance for an enrolled user will be above the threshold, so that he will be denied access. Such an error is called *false rejection* or *false non-match*. Conversely, due to both the variability of the acquired features and the tolerance mentioned above, the distance of an impostor will sometimes fall below the threshold, causing an error named *false acceptance* or *false match*.

To evaluate the performance of a given biometric system, the following indicators have been introduced. The False Acceptance Rate (FAR) is defined as the ratio of false acceptances to the total number of impostors attempts. It expresses the a posteriori probability that an impostor be granted access. Similarly, the False Rejection Rate (FRR) is defined as the ratio of false rejections to the total number of attempts by enrolled users claiming their true identity. It represents the a posteriori probability that the system will reject an enrolled user. Obviously, the value of both the FAR and FRR are correlated, and depend on the chosen threshold. A low threshold corresponds to a high security

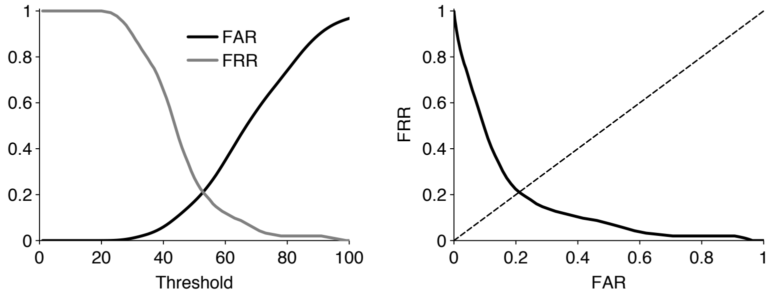


Figure 2.3 : Biometric system performance.

Two ways of reporting results. FAR and FRR curves as a function of the threshold index (left) and ROC curve of FRR as a function of FAR (right). The EER is given either by the crossing of the curves (left), or by the point where the ROC curve intercepts the diagonal (right).

operation mode, where one wants to minimize the FAR. The distance between reference and test features is thus required to be very small, otherwise the identity claim will be rejected. The FRR will therefore be large. The opposite holds e.g. for a lower security or convenience application, where the FRR must be low, resulting in a larger FAR.

To report the performance of a biometric system or algorithm independently of any predefined mode of operation, the number of false acceptances and false rejections is plotted as a function of the threshold. A typical result is presented in Figure 2.3 (left). Alternately, one can plot the FRR as a function of the FAR, as shown in Figure 2.3 (right), to obtain the Receiver Operating Characteristic (ROC) curve. In that case, each point in the graph corresponds to a different threshold value, and therefore to a different operating mode. The Equal Error Rate (EER) is defined as the point on the ROC where the FAR and the FRR are equal.

The chosen threshold determines the operating point of a deployed biometric system, and characterizes the level of security offered. It is usually chosen from statistical analysis of the results obtained using a set of evaluation clients and impostors, based on the hardly verifiable assumption that the same statistics will hold for test clients and impostors. Whereas the population to be enrolled can usually be characterized, it is much harder to do the same for the impostors. Evaluation data can also be used to normalize the computed distances, to enhance the discriminative properties of the system, as it will be discussed in Subsection 3.3.6.

2.4.5 Positive vs. negative recognition

It should be noted that expressing the performance of a biometric system using the FAR and the FRR is only unequivocal in the context of *positive recognition*, where one wants either to determine the identity of a person, or to establish whether a given individual is indeed who he claims to be. However, in some applications said to operate in *negative recognition* mode, the goal differs and consists instead in determining whether a person is part of a given group or not, the purpose being to prevent a single individual from assuming multiple identities. This could be enforced for instance in the framework of a welfare benefits distribution program where the same person could attempt to register several times under different names. To avoid this kind of abuse, the biometric system tries to match the person against a list of people already benefiting from the program. If the individual is erroneously found to match a record in the database, he will be denied access to the program, so that a false match will actually lead to a false rejection rather than to a false acceptance. Conversely, if the system is not successful in identifying an enrolled person, it will cause the latter to be mistakenly accepted again.

Each of the terms FAR and FRR can therefore have opposite meanings depending on the application. To avoid any confusion, performance figures can alternately be provided by specifying the False Match Rate (FMR) and the False Non-Match Rate (FNMR), whose meaning is not affected by the context. Nevertheless, the FAR and the FRR remain the most popular way of characterizing commercial biometric systems [7]. Indeed, most of them operate in positive recognition mode only, whereas negative recognition is mainly employed by government agencies.

2.5 Biometric modalities

In this section, a list of several modalities commonly employed for biometric identity verification or determination is established, and each one is briefly discussed.

2.5.1 DNA analysis

Deoxyribonucleic acid (DNA) is the biochemical molecule found in the nucleus of every living cell, that holds one's genetic patrimony. It can seem to be the ultimate biometric modality, and is increasingly used in connection with forensics, a field where it has proved to be extremely accurate. However, many challenges prevent its application to automated systems. Indeed, DNA matching currently involves complex chemical analysis processes and requires manipulations that can only be accomplished by skilled experts. Moreover, DNA does not only permit establishing the identity of a person, it also reveals a great deal of personal details such as one's genetic predispositions to some diseases. As such, it certainly strongly invades individual privacy. Conversely to most other biometrics, the information that the DNA carries does not change at all over time. It is determined at the instant of fecundation, and is neither affected by embryonic development, nor by the environmental factors influencing the individual after birth. As a consequence, it is incapable of differentiating identical twins, whereas most other biometric modalities can.

2.5.2 Fingerprints

Based on the structure of the ridges and valleys patterns found on fingertips, fingerprint matching is one of the most widespread biometric modality. Employed by forensic investigators for more then a century, it is to this day the only biometric, with DNA analysis, to be accepted by courts of law to convict or acquit a suspect. Fingerprints were originally taken using the well-known ink-and-roll technique, and stored on paper cards. This process can now be accomplished using electronic scanners that produce a digital image of the fingerprint pattern. Unlike many other modalities, a great variety of sensors can be employed to acquire fingerprints, such as optical, capacitive, ultrasonic or thermal ones [7]. The matching process itself was traditionally performed by comparing the type, location and orientation of special landmarks of the ridge pattern, called *minutiae*. Such specific traits include, among others: termination, bifurcation and crossover. Some automated systems follow the manual minutiae-based procedure, allowing their results to be validated by human experts if necessary. Other approaches rely on correlations, either of the direct fingerprint image, or of the extracted ridge pattern. Fingerprint matching has proved to be very accurate [17], yet its historical association with forensics caused concerns regarding the user acceptance in consumer applications. It seems however that the desire to prevent identity fraud is now prevailing [7].

2.5.3 Hand geometry

First patents related to a biometric system based on the geometry of the hand appeared in the early 1970s [18, 19]. Since then, solutions have been commercially deployed and installed in hundreds of different sites. Relatively inexpensive and easy to use, they are also mostly insensitive to environmental factors and individual skin conditions such as dryness. Although they necessitate the cooperation of the subject, usually requiring that the palm be flatly applied on a panel with outstretched fingers, they are very easy to use. However, the large size of the sensor needed to acquire the hand geometry restricts its deployment to stationary access control systems. Moreover, this modality is not very distinctive, and is therefore not very well adapted to identification applications among a large number of users. Finally, some medical conditions like arthritis, that diminishes the mobility of the hand, may prevent accurate measurements, potentially causing a large number of false rejections with affected individuals.

2.5.4 Face

This modality is an obvious one, since it is one of the most commonly used by people to recognize each other. Over the course of its evolution, the human brain has developed highly specialized areas dedicated to the analysis of facial images [20]. Indeed, in addition to the once vital necessity of being able to instantly determine whether the unexpectedly encountered individual is a friend or a foe, social interactions also strongly depend on the capability to interpret many subtle facial expressions conveying a great deal of information [21].

Automated face recognition is performed on facial images that can be acquired with minimal obtrusiveness, requiring only that the subject looks at a camera. Extracted characteristics are mainly internal, concentrating on the area between the forehead and the chin. It is however questioned whether the face itself is a sufficient basis for recognizing a person from a large population with great accuracy. Indeed, the human brain also relies on many contextual information. Under certain circumstances, the internal features of a face (such as mouth, nose and eyes) can be replaced by the ones from another person without affecting its ability to be correctly identified by test subjects [22], suggesting that the brain sometimes actually recognizes heads instead of faces.

The most problematic perturbation affecting the performance of face recognition systems are strong variations in pose and illumination. To alleviate these effects, 3D images can be used e.g. to reconstruct faces under reference illumination and pose [23, 24]. Alternatively, it is also possible to use the direct 3D information (e.g. reconstructed 3D meshes) to recognize faces [25], provided the resolution of the acquisition system is sufficient. When combining the 3D range data with 2D texture images, it is even possible to distinguish identical twins [26].

The existing algorithms pertaining to the face modality will be further discussed in Section 3.2.

2.5.5 Iris

The texture of the human iris is created by a chaotic process that starts during fetal development, and stabilizes during the first two years of life. As a result, it is assumed that each resulting pattern is unique. Given that they are very stable over time, and very difficult to tamper surgically, irises are certainly one of the most interesting biometric modalities. Image acquisition can be performed without requiring direct contact, so that the perceived intrusiveness is relatively low. Recognition is performed by firstly locating the iris using landmark features. A Gabor wavelet transform is then applied to extract spatial frequencies and orientations [27]. The resulting parameters are encoded into a 2'048-bit vector known a the IrisCode. Iris recognition was proved to be very accurate, and is capable of reaching virtually 0% FAR in real-world conditions [28]. This field was pioneered in the 1990s by John G. Daugman, who holds several key patents [29]. A a result, almost all commercially deployed iris recognition system are based on his method.

2.5.6 Retina

In addition to the iris, the human visual organ comprises another textured area supporting a pattern that is different for each eye. Constituted of a richly structured vascular network, the retina is considered to be one of the most accurate and safe biometric modality, particularly since it is located *inside* the body. This makes it very difficult to forge or replicate in a way that would confuse a biometric system. To be performed, retinal scans require the active cooperation of the subject. Practically, the user is asked to look into an eyepiece and to focus on a specific spot, the retina image being acquired using infrared illumination. This high level of intrusiveness as well as the common fear that the process actually involves projecting a laser beam into the eye, as seen in numerous movies, hinder public acceptance of this biometric modality. Moreover, it is believed that examination of the retina can reveal some medical conditions.

2.5.7 Voice

Together with face, voice is the most common modality used by humans to recognize each other. It is determined by several anatomical features, such as the shape and size of the laryngeal, buccal and nasal cavities as well as the characteristics of the vocal chords. The behavioral features of one's voice can be strongly affected by emotional state, smoking, aging, or diseases of the respiratory system. Automated biometric systems based on this modality are referred to as ASV (automatic speaker verification) or ASI (automatic speaker identification) systems. They should not be confused with speech recognition systems, which aim at transcribing the information conveyed by the voice into text. If the ASV or ASI system is text dependent, the user is required to speak a selected sequence of words, whereas any of kind of words or sentences can be uttered in the case of a text independent system.

2.5.8 Signature

Signatures have been used as a mean of authentication for many centuries. Moreover, it is one of the very few biometric modalities to be legally binding in most, if not all countries around the world, despite the fact that it is not difficult for skilled persons to create forgeries that can only be detected by graphology experts. However, it is much more difficult to imitate the movements executed by the signatory. Thus, instead of considering the resulting static pattern, automated signature-based authentication systems analyze the dynamics of the signature, such as the acceleration of the stylus, the angle it forms with the support on which the signature is drawn, and the pressure applied by the writer. Even though some people appear to generate a different graphic impression every time they sign, the dynamics of the movement is less variable. Being a behavioral biometric trait consciously executed, signatures can drastically change over time. Moreover, they are affected by emotional conditions such as stress or anger.

2.5.9 Gait

Defined as the distinctive way one walks, gait is one of the newest modality being investigated. It is not supposed to be very distinctive, nor invariant, due to fluctuations in body weight, injuries, weariness or inebriety. The analysis of the movement is usually performed on video sequences, making it rather unobtrusive. However, some approaches rely on wearable accelerometers [30].

2.5.10 Others

The following non-exhaustive list enumerates some of the many additional biometric modalities.

- Body odor, determined by analyzing the chemical compounds exuded by the skin.
- Thermogram, the thermal map of the hand or of the face.
- Keystroke, the specific characteristics of one person's way of typing on a computer keyboard.
- Mouse gestures, analyzing the way a computer mouse is displaced. Somewhat related to signature.
- Palmprint, similar to fingerprint but based on the pattern formed by the ridges of the palm.

- Ear geometry, based on the shape and the ear.
- Lips movement, analyzed during speech.
- Vascular patterns of the palm or of the back of the hand.

2.6 Biometrics for mobile applications

Mobile devices, such as Personal Digital Assistants (PDA) and cellular phones, are playing an ever-increasing role in people's lives, both professional and private. Moreover, they are becoming more sophisticated and hold data of raising sensitivity. Their small size and the fact that they are designed to be used almost anywhere make them more prone to theft and loss. Furthermore, the emergence of the so-called *mobile commerce*, amounting to an estimated US $200 billion in transactions by the end of 2005 [31], raises yet another concern for users with respect to security. Indeed, additionally to its intrinsic resale value, a mobile device that provides access to paid-for online services could potentially incur much larger costs in case of abuse.

In such conditions, the traditional security measures such as Personal Identification Number (PIN) codes are quickly becoming insufficient. Indeed, the multiple PIN codes and passwords one has to remember and use in its daily activities, for gaining access e.g. to mobile phone, banking ATM (Automated Teller Machine) or computer terminals and networks, incite some people to write them down since they cannot seem to remember all of them. Alternately, people use very simple and obvious passwords that can easily be guessed. Moreover, a PIN code can be observed when it is entered. As a consequence, equipping such devices with biometric access protection could greatly enhance the level of security offered.

Since mobile phones are obviously equipped with a microphone, speaker verification is a natural first choice modality for such devices. Even though automatic speech recognition is implemented in many phones, e.g. to provide vocal commands allowing quick access to frequently dialed phone numbers, speaker authentication is nevertheless not commonly found on these devices. Indeed, identity verification based on the voice is difficult to perform accurately in uncontrolled environments, due to perturbations such as the voices of the surrounding people or the multiple sources of background noise, such as vehicles or construction works in a city street. Moreover, since the voice signal is strongly compressed to limit the bandwidth used by each conversation in mobile telephony networks, there is little incentive to include high-quality microphones.

Fingerprint patterns can easily be acquired without being affected by the environment. Indeed, most fingerprint sensors, regardless of their type, only operate in a very limited range, usually a few millimeters. Consequently, their are immune to most external perturbations. Fingerprints nevertheless are not optimal in mainly two aspects. Firstly, they are associated to forensics, though this concern seems to be losing importance, as supported by the growing number of consumer devices that employ this modality, such as laptop computers and USB keys. The second drawback is however still very relevant and consists in the mandatory presence of a dedicated sensing device. Even when ordered in large quantities, fingerprint sensors cost around US $5 a piece, which is a significant price for a component that does not provide any additional functionality besides user authentication.

This economic concern is not present in the case of the face modality, since image sensors are found on an increasing number of devices, providing digital camera functionality to mobile phones or enabling Internet-based videoconferencing on PDAs and laptop computers. Moreover, the resolution and quality of the images they generate are already more than adequate for face verification. Coupled with the low invasiveness and high acceptability it is being associated with, the face modality certainly constitutes a very interesting choice. Furthermore, face is less subject to forgery attempts in the case of casual theft than fingerprints. Indeed, latent fingerprints are likely to be present on the surface of a mobile device, since the latter was designed to be handheld when operated. Latent fingerprints can be collected and used to create fake fingertips, that constitute a three-dimensional reproduction of the pattern of ridges and valleys found on the finger of the legitimate user. This technique has been demonstrated to be capable of fooling certain commercial fingerprint sensors, so that an impostor could use it to gain undue access to the device. In case the face modality is used, the stolen device would not provide any information that could help a potential intruder in constructing an object susceptible of deceiving the system. He would therefore need to obtain this information separately, e.g. by taking a picture of the legitimate owner's face, which is not an easy task in the considered context.

Due to the unconstrained nature of the acquisition conditions, robust algorithmic solutions are nevertheless necessary to handle environmental perturbations such as varying illumination if the face modality is used. A mobile device being usually handheld when operated, changes in scale, as well as rotations are also unavoidable. However, the flexibility offered by a handheld device can also be regarded as an advantage, since the user can, to a certain extend, alleviate some effects impacting the performance by simply displacing the device.

None of the other modalities reviewed in Section 2.5 appears suited to mobile applications. DNA analysis requires complex chemical manipulations and cannot currently be carried out automatically. Sensors capable of measuring hand geometry or palm imprints would be larger than the device itself. Signature authentication would be limited to PDAs equipped with handwriting recognition, ruling out mobile phones. Even though gait analysis has been demonstrated using mobile devices [32], it requires them to be equipped with accelerometers, a category of sensors not yet commonly found on mobile phones or PDAs[2].

2.7 Conclusion

In this chapter, the definition of biometrics has been provided, and the properties and tasks associated to biometric recognition systems have been introduced, firstly in a general context and later specifically in the realm of mobile applications. Among the many biometric modalities discussed, the face was determined to be a suitable choice for implementation on such devices as mobile phones or PDAs. The unconstrained acquisition conditions nevertheless require an algorithm that is robust to variations in pose and illumination. Additionally, tight constraints in computational resources and limited available energy source need to be taken into account and addressed. In the following chapter, we will therefore review various methods that have been proposed to perform biometric authentication based on facial images, in view of selecting the approach that can best fulfill these requirements.

[2] Some recently introduced mobile devices are nonetheless equipped with an accelerometer. The most famous examples are Apple's iPhone and some of the iPod models, where the accelerometer is used to detect when the user rotates the device, e.g. to change the display mode from portrait to landscape. Some games also exploit this capability and can be controlled by changing the orientation of the device.

Chapter 3

Face Verification

3.1 Introduction

Following the selection, in Section 2.6, of the face as a suitable modality for biometric authentication deployed on mobile devices, we will now review the methods proposed in the literature to perform such task, in view of determining the most adequate solution for the foreseen applications. Some algorithms will be detailed in Section 3.2, whereas for others, only references will be provided. At the end of this section, the motivations for having selected the Elastic Graph Matching algorithm as the best suiting face verification solution for mobile devices, are exposed. The implemented system and its behavior are then described in Section 3.3. Two kinds of features extraction techniques employed in conjunction with this algorithm will be discussed in Section 3.4, based on Gabor filters and on mathematical morphology, respectively. In view of increasing the robustness of the latter option with respect to changes in illumination, two normalization techniques are furthermore introduced, one of them being shown to offer performance very close to that of Gabor filters, while requiring only a fraction of the computational complexity.

3.2 Face recognition algorithms

In this section, three face recognition algorithms among the most commonly employed are discussed in Subsection 3.2.1 to Subsection 3.2.3. Additionally, references to some other approaches proposed in the literature, as well as references to the most recent surveys pertaining to this field, are provided in Subsection 3.2.4 and Subsection 3.2.5, respectively.

3.2.1 Feature-based methods

The early solutions developed to perform automatic face recognition seem to have been directly derived from the anthropometric approach followed by Bertillon (see Subsection 2.3.1). Indeed, they rely on comparing geometrical distances measured between a series of facial features, or landmarks, such as the nose, the chin or the eyes, as illustrated in Figure 3.1. The work by Bledsoe et al., which is recalled in [15], is an example of such methods. Here, the coordinates of several facial landmarks are manually extracted from photographs by human operators using a digitizing tablet, and they are then fed to a computer. From these coordinates, a set of 20 distances are computed and stored in a database alongside the corresponding identity. When asked to identify a face captured on-line, the computer compares the set of distances obtained from this face with those stored in the database, and returns either the closest match, or a short list of possible identities. To compensate for differences in pose, such as rotation and tilt, as well as variation is scale, geometry transformations need to be applied during the matching process.

Several feature-based face identification methods are discussed in the survey by Samal et al. [34]. Although no performance figures are presented, it is nevertheless noted that many approaches necessitate strong assumptions to be verified in order for the fea-

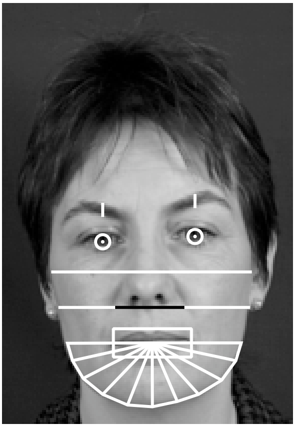


Figure 3.1 : Feature-based face recognition.

Based on the geometrical features employed by Brunelli and Poggio [33].

tures to be correctly detected. For instance, the face must be upright, devoid of tilt, and free of occlusions (e.g. by glasses). Effectiveness of the face detection is crucial because errors occurring at this stage will inevitably prevent successful identification. Several approaches are reported to solve this problem, such as using deformable templates to detect the eyes and the lips [35], or using the Hough transform [36] to locate the eyes (or more precisely, the irises). In their paper comparing face recognition methods based on features to those based on templates [37], Brunelli and Poggio use template matching by cross-correlation. Multiple templates are employed to account for variations e.g. in scale, and a normalization consisting in dividing the pixel values by the average intensity over a local neighborhood is applied beforehand. Once the eyes are located, their relative position is used to derive the scale and the angle of rotation of the face. Other features, such as the location and size of the mouth and eyebrows, are extracted using integral projections, as introduced by Kanade [38].

Other approaches proposed include methods based on the Discrete Cosine Transform (DCT) [39] or the Generalized Hough-Transform (GTH) [40]. Saber et al. [41] rely on the chromatic and geometrical properties of the face and exploit its symmetries. Graf et al. uses morphological operators to identify facial parts based on their expected shape [42].

3.2.2 Eigenfaces

The concept of eigenfaces is directly derived from the *eigenpictures* introduced by Sirovich and Kirby [43] as a mean of optimizing the representation of a set of images of human faces by expressing them in a low-dimensional space, using Principal Component Analysis (PCA), also known as Karhunen-Loève Transform (KLT). Their basis assumption was that what differentiates each picture $\vec{\varphi}_k$ in a set of M images is its departure, or deviation, from the *mean image*, defined as the average face $\vec{\psi}$:

$$\vec{\psi} \;=\; \frac{1}{M} \sum_{k=1}^{M} \vec{\varphi}_k \tag{3.1}$$

They thus search for a face space, spanned by a set of orthogonal eigenvectors, that maximizes the variance between the faces of a given set. They define the *caricature*, denoted $\vec{\phi}_k$ and expressing the difference between a face $\vec{\varphi}_k$ and the average face $\vec{\psi}$:

$$\vec{\phi}_k \;=\; \vec{\varphi}_k \;-\; \vec{\psi} \tag{3.2}$$

Figure 3.2 shows the average face obtained using the images of 37 individuals in the Extended Yale Face Database B [44], one of the original face and the corresponding caricature.

In the case of images of $N \times N$ pixels, $\vec{\phi}_k$ can be considered as a vector of dimension $1 \times N^2$. The basis of the system is

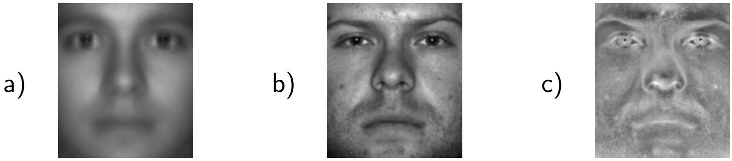


Figure 3.2 : Average face and caricature.
Images from the Extended Yale Face Database B. a) Average face, obtained from 37 images, b) Original face, c) Caricature.

constituted by the eigenvectors $\vec{u}_i$ of the $N^2 \times N^2$ covariance matrix $\mathbf{C}$, constructed as:

$$\mathbf{C} \;=\; \frac{1}{M} \sum_{k=1}^{M} \; \vec{\phi}_k \; \vec{\phi}_k^{\,T} \tag{3.3}$$

$$\mathbf{C} \; \vec{u}_i \;=\; \lambda_i \; \vec{u}_i \tag{3.4}$$

Obtaining the eigenvectors of an $N^2 \times N^2$ matrix is practically an intractable problem. However, since the subspace is constructed from M samples only, there can be at most $M-1$ meaningful eigenvectors and it has been shown [43, 45] that they can be derived from the eigenvectors of an $M \times M$ matrix. An image can then be reconstructed from the average face, and from a weighted sum of the eigenvectors $\vec{u}_i$ obtained above, that Sirovich and Kirby named *eigenpictures*:

$$\vec{\varphi}_k \;=\; \vec{\psi} \;+\; \sum_{i=1}^{M-1} \; \omega_i \; \vec{u}_i \tag{3.5}$$

The $M-1$ weight coefficients ω_i for a face $\vec{\varphi}_k$ are obtained by projecting the corresponding caricature $\vec{\phi}_k$ into the subspace, which is accomplished by computing the dot product between the caricature and each eigenpicture $\vec{u}_i$:

$$\omega_n \;= \vec{u}_i^{\,T} \cdot \vec{\phi}_k \;=\; \vec{u}_i^{\,T} \cdot (\vec{\varphi}_k \;-\; \vec{\psi}) \qquad i \in [1, \ldots, M-1] \tag{3.6}$$

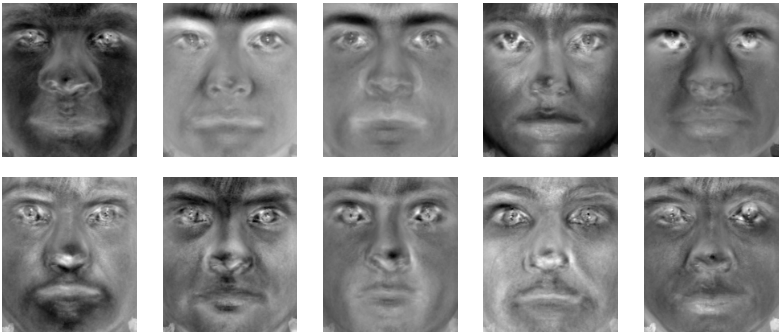


Figure 3.3 : Sample Eigenfaces.
First 10 eigenfaces obtained using $M = 37$ images from the Extended Yale Face Database B, from left to right and from top to bottom. To display better, the images have been normalized to the full 8-bit dynamic.

In order to reduce the dimension of the subspace, only a subset of M' eigenvectors having the largest eigenvalues, with $M' < M$, is retained so as to minimize the reconstruction error. Sirovich and Kirby showed that for a set of 115 face images of 128×128 pixels, only 40 coefficients in the face subspace are actually required to characterize a face with an error of 3%.

The results of Sirovich and Kirby inspired Turk and Pentland to use the vector formed by the set of the weight coefficients required to reconstruct a face as a mean of determining the identity of the individual [45]. They followed the same procedure as Sirovich and Kirby to obtain the eigenvectors, but refer to them as *eigenfaces* instead of eigenpictures. The first ten eigenfaces obtained using images from the Extended Yale Face Database B [44] are shown in Figure 3.3, while Figure 3.4 presents the reconstructed image using a varying number of eigenfaces. When all 36 eigenfaces are used, the reconstructed face is identical to the original image shown in Figure 3.2b.

Observing that the PCA not only maximizes between-class scatter, but within-class scatter as well, and that changes in illumination are retained in the eigenfaces [46], Belhumeur et al. conclude that while “PCA projections are optimal for reconstruction from a low dimensional basis, they may not be optimal from a discrimination standpoint” [47]. They thus propose to use the Linear Discriminant Analysis (LDA), also known as Fisher Linear Discriminant (FLD), after the PCA step, so as to obtain a new basis spanned by another set of eigenvectors that they named *fisherfaces*. Their results show that fisherfaces indeed are more robust in presence of variations in lighting and expression. In [48], Bartlett et al. propose to use Independent Component Analysis (ICA) instead of PCA, noting that PCA only separates the second order moments (i.e. the variance) of the inputs, whereas ICA additionally considers the higher order moments. The reported results indeed show ICA outperforming PCA, in particular when the images in the test set were acquired in the course of a different session than those in the enrollment set.

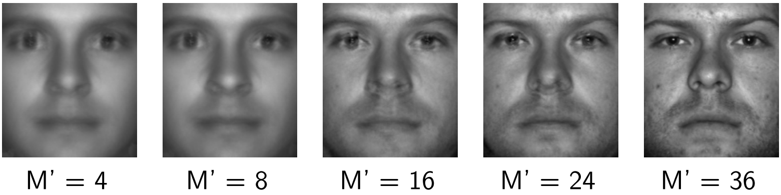


Figure 3.4 : Reconstruction using eigenfaces.

Reconstructed images using M' eigenvectors. The rightmost image, reconstructed with all eigenvectors, is identical to the original image.

3.2.3 Graph-based methods

The Dynamic Link Architecture (DLA) is a theory elaborated by von der Malsburg [49], as a possible explanation of how the brain processes and organizes information in order to represent complex data structures. According to the DLA, the latter have the form of graphs, composed of nodes (called units) connected by links. Units play the role of simple symbolic elements, whereas the links denote their mental associations. Both the units and the links are dynamic, and the resulting graphs constitute the "data format" by which the brain represents mental objects. Von der Malsburg reports applying the DLA to pattern recognition [50] by associating it to a neural description.

3.2.3.1 Gabor-based features

In 1993, Lades et al. introduced an algorithmic formulation of DLA, called *Elastic Graph Matching* (EGM), adapted to current digital hardware [51]. An implementation on a MIMD[1] architecture, composed of 23 transputers, was realized [52]. The demonstrated system recognizes office objects, as well as faces. The proposed algorithm consists in extracting a series of feature vectors at certain locations in the images. These specific points $\vec{x}_i$ are determined by the vertices, or nodes, of a regular orthogonal grid, arbitrarily laid out over the face. The feature vectors, also called *jets* and noted $\vec{J}(\vec{x}_i) = J_i$ are thus said to label the vertices. In the proposed implementation, the grid is composed of 7×10 nodes, and the features extraction is performed using a Gabor filter bank (see Subsection 3.4.1) comprising 5 frequencies and 8 orientations, resulting in feature vectors composed of 40 values each. Only the magnitude of the complex response of the Gabor filters is considered, so that the vectors J_i only contain elements whose value is both real and positive. The selection of Gabor filters was motivated by the fact that they appear to furnish a good approximation of the sensitivity profile of neurons in the visual cortex. The 70 reference jets — or model jets — denoted J_i^M constitute the signature of the face, and are stored in a database, alongside the identity of the individual.

[1] Multiple Instructions, Multiple Data.

In a first stage, the matching process is carried out by placing the same orthogonal grid over the test image, at an arbitrary location. Features extraction is then performed at each of the vertices, producing 70 new test jets — or image jets — denoted J_i^I and a new labeled graph. At each node $\vec{x}_i$, the jet similarity measure $S(i)$ is computed between the corresponding model and image jets, using a normed dot product:

$$S(i) = S\left(J_i^I, J_i^M\right) = \frac{J_i^I \cdot J_i^M}{\left\|J_i^I\right\| \, \left\|J_i^M\right\|} \qquad \forall i \mid \vec{x}_i \in \mathcal{V} \tag{3.7}$$

where $\mathcal{V}$ denotes the ensemble of all vertices. The cost function C_{jets} is defined as the opposite of the sum of the jet similarity measures $S(i)$ over the whole graph:

$$C_{jets} = -\sum_i S(i) \qquad i \mid \vec{x}_i \in \mathcal{V} \tag{3.8}$$

The orthogonal grid is then iteratively translated over the image, and at each new location, the process described above is repeated. The 70 feature vectors are extracted, and the corresponding cost function is computed, the goal being to find the location of the grid that minimizes C_{jets}.

In a second stage, the graph as a whole is not shifted anymore, but its vertices are sequentially displaced, in a random order and by a random vector below a preset maximum length, corresponding to half the original distance between vertices. A new total cost function, C_{total}, is introduced:

$$C_{total} = C_{jets} + \lambda \cdot C_{edges} \tag{3.9}$$

The non-negative term C_{edges} penalizes the deformation of the edges incurred during the second stage by the independent displacements of the vertices. Unless it is situated on the external border of the graph, each node $\vec{x}_i$ is connected to four neighbors

$\vec{x}_j$, and each connection forms an edge. The deformation penalty $D_e(i,j)$ for one edge corresponds to the square of the Euclidean norm of the displacement, or equivalently to the square of the Euclidean distance between the displaced and reference edges:

$$D_e(i,j) = \left\| \vec{\Delta}^I_{ij} - \vec{\Delta}^M_{ij} \right\|^2 \quad \forall (i,j) \mid \vec{x}_i \in \mathcal{V},\ \vec{x}_j \in \mathcal{N}(i) \tag{3.10}$$

$$\vec{\Delta}_{ij} = \vec{x}_i - \vec{x}_j \tag{3.11}$$

The symbol $\mathcal{N}(i)$ denotes the ensemble of the vertices connected to $\vec{x}_i$. For one node, the penalty D is obtained by considering all the edges it forms with its neighbors:

$$D(i) = \sum_j D_e(i,j) \qquad j \mid \vec{x}_j \in \mathcal{N}(i) \tag{3.12}$$

Finally, the deformation penalty C_{edges} for the whole graph is computed as:

$$C_{edges} = \sum_i D(i) \qquad i \mid \vec{x}_i \in \mathcal{V} \tag{3.13}$$

As seen in Equation 3.9, the contribution of C_{edges} is weighted by an elasticity factor λ, whose value is empirically determined.

The elastic deformation stage is carried out iteratively until C_{total} does not decrease anymore, or until a preset maximum number of displacements is reached. At this time, if C_{total} is below a given threshold, the matching is assumed successful and the face in the test image is considered to belong to the same individual as the reference. In their experiments, Lades et al. determined the threshold value based on the statistics of the final C_{total} values obtained on a set of training images.

In the algorithm described above, the vertices, and hence the extracted features, are not located on any specific facial landmarks.

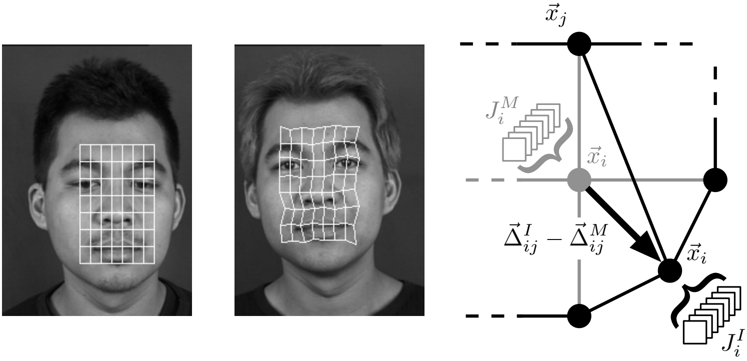


Figure 3.5 : Elastic Graph Matching.

Model graph placed over the reference face (left) and elastically displaced matched graph on the test image (middle). The drawing on the right shows the model and image graph, depicted in gray and black, respectively. The node $\vec{x}_i$ in the image graph has been displaced with respect to its position in the model graph, and the arrow illustrates the resulting distortion that affects the edge connecting the nodes $\vec{x}_j$ and $\vec{x}_i$. Each node is labeled with a feature vector, or jet, that is shown for the node $\vec{x}_i$ in the model and image graphs as J_i^M and J_i^I, respectively.

Consequently, the method can be categorized as a holistic one. Wiskott [53] follows a similar approach that he calls Dynamic Link Matching (DLM), using a 10 × 10 graph, but proposes to also consider the phase information of the Gabor features when computing the similarity. He also describes a variant of DLM, that he calls Elastic Graph Matching. This is an unfortunate choice since in his implementation, the nodes are placed on fiducial points, which "are assumed to be important and easy to find". Hence, the holistic nature of EGM as proposed by Lades et al. is lost, causing some authors [54, 55] to categorize it as a feature-based method instead. Wiskott et al. further apply EGM to gender determination [56] and introduce Elastic Bunch Graph Matching (EBGM) to better handle variations in pose and expressions [57]. In this system, the single model jet obtained at each node is replaced by a bunch of jets, corresponding to local variations of the extracted features. For instance, for a node located over an eye, the bunch may include jets from closed or opened eyes. During matching, for each node, the jet in the model bunch that maximizes the similarity with the test image jet is selected. Further improvements to EBGM are proposed by Okada et al. [58], who perform histogram normalization after locating facial landmarks, to adjust for differences in lighting. This requires a new set of features to be extracted from the corrected images, which means that Gabor filtering is actually performed twice.

Duc [59] follows the same approach as Lades et al. but uses an 8×8 grid and a Gabor filter bank consisting of 6 orientations and 3 frequency resolutions. He constructs the jets from the magnitude of the complex responses of the Gabor filters, and furthermore replaces the cost function C_{jets} with a sum of Euclidean distances:

$$C_{jets} = \sum_{i} \left\| J_i^I - J_i^M \right\| \qquad i \mid \vec{x}_i \in \mathcal{V} \tag{3.14}$$

A sum of Euclidean distances is also employed to compute the deformation penalty $D(i)$ associated to each node, used to obtain C_{edges} in Equation 3.13.

$$D(i) = \sum_{j} \left\| \vec{\Delta}_{ij}^I - \vec{\Delta}_{ij}^M \right\| \qquad j \mid \vec{x}_j \in \mathcal{N}(i) \tag{3.15}$$

Moreover, to avoid situations where the graph matching gets trapped in local minima, Duc devises a coarse-to-fine operating strategy for the rigid matching step. To this effect, he uses lower resolution images, obtained by Gaussian filtering and decimation. Moreover, observing that the matching error is not sufficient to discriminate impostors [60], he proposes to weight the contribution of each node when computing the cost functions, according to a factor statistically determined using Fisher Linear Discriminant (FLD) on training samples.

3.2.3.2 Morphology-based features

In [61], Kotropoulos et al. introduce a novel DLA variant, named Morphological Dynamic Link Architecture (MDLA), that substitutes Gabor features with multiscale mathematical morphology. Instead of information derived from the texture, the feature vectors now describe the shapes found in the image at various scales and are obtained by applying both erosions and dilations using 9 structuring elements (see Subsection 3.4.2). The value of the pixel in the original image is also included, leading to a total of 19 extracted features per graph node. They note that multiscale erosions and dilations capture important information for key facial features such as the eyebrows, eyes, nostrils and lips. The cost function is computed with Euclidean distances, as in [59], and the graph matching is carried out by following the same procedure as in [51]. However, the deformation penalty C_{edges} is discarded when computing the final distance between the test and reference images. Experimenting the proposed MDLA approach using the M2VTS database [62], Kotropoulos et al. conclude that MDLA outperforms the Gabor-based variant.

The graph matching procedure is later modified by Kotropoulos and Pitas [63] by applying Principal Component Analysis (PCA) on the extracted jets, replacing the 19 feature values by the 6 Most Expressive Features (MEF). Linear Discriminant Analysis is further performed [64] in view of obtaining the Most Discriminating Features (MDF) instead of the MEF, since the aim is not to minimize reconstruction errors but to maximize the ratio of between-class scatter to within-class scatter. The results achieved with PCA and LDA are however not improved. They also replace the graph matching with simulated annealing, claiming that it performs better than the approach proposed by Lades et al. [51]. Finally, they observe in [65] that the choice of the function used to construct the multiscale structuring elements employed for extracting the mathematical morphology features, does not affect the performance of the MDLA algorithm. It is therefore possible to select this function based on computational efficiency criteria.

While still using Gabor filters, Lades pursues his experiments by replacing the graph matching he introduced in [51] with a Monte Carlo-based optimization procedure [66]. In addition to translating the graph at each step, its scale and orientation are also modified in a process he calls “Global Move”. Further improvements are proposed by Kotropoulos et al. [67], consisting in associating a class-dependent discriminatory power to each graph node, used during matching to weight the contribution of each vertex to the total distance. Since one does not know a priori which nodes are the most discriminative ones, the weights are determined by statistical analysis performed on a training set of faces. Duc et al. apply the same idea to Gabor features in [68].

Since the extracted morphology features consist in finding local extrema among the pixel values, they are strongly affected by changes in illumination intensity. As a result, the achieved accuracy degrades significantly when the algorithm is evaluated under non-optimal conditions [69]. A normalization step is thus introduced [70, 71] before the features extraction, in view of reducing the illumination dependency. The proposed solution requires however a prior face detection, even though it is difficult to perform it in presence of perturbations such as non-uniform illumination or shadows.

3.2.4 Other methods

Several other approaches to automated face recognition were also proposed, based for instance on template matching [72, 35], Hidden Markov Models (HMM) [73, 74], neural networks [75], Line Edge Map (LEM) [76], Local Features Analysis (LFA) [77], and many more. All the solutions discussed so far rely on frontal images, however there exist methods that are based on profile views, attempting to reconstruct pseudo 3D images [78], by combining profile and frontal views. Other authors exploit only the profile curvature [79, 80], yet in both cases, the setup needed to acquire lateral face images restricts the deployment of such approaches to stationary applications. Two efficient ways of achieving robustness to changes in illumination consist either in using 3D models, and many novel approaches to face recognition are now involving range images [81], or in using thermal infrared images [82]. However, the size of these kinds of sensors, as well as the amount of energy they consume, make them currently incompatible with the requirements of mobile applications.

3.2.5 Recent surveys

Many studies and surveys on face recognition have been published, several of them being discussed in Section 3.5 in [1], where a classification of several algorithms is also provided. In a recent paper, Kong et al. [82] discuss the latest advances on this topic, considering methods based both on visual and on infrared information. They note that the latter is more robust in uncontrolled environments, and less easily deceived by disguised faces, and report that fusing both kinds of information results in improved recognition accuracy. Tan et al. [55] consider the specific problem of face recognition in scenarios where only one image per client can be collected at enrollment time. Since the variability of the samples inside the same class cannot be evaluated, this restriction negatively impacts the performance of the system, even when the environment, the pose and the expression are mostly constant. In the case of the Elastic Graph Matching algorithm, when up to four reference templates can be used, the value of the EER is half that of the single template variant, as it will be shown in Figure 3.11 in Subsection 3.4.1. Several methods alleviating this problem are discussed in [55], such as enlarging the training set by synthesizing virtual samples, e.g. by introducing artificial perturbations into the enrollment image.

Finally, a recent book by Wechsler [83] comprehensively covers the multidisciplinary nature of face recognition, discussing many theoretical elements such as face modeling and classification, drawing considerations both from neurosciences and from statistics and image processing. More practical aspects such as system evaluation, as well as privacy and security concerns, are also addressed.

3.2.6 Algorithm selection

Whereas many algorithms have been shown to perform satisfactorily in well-controlled environments, their face verification performance degrades sensibly when the tests are carried out under less constrained conditions. The application scenario considered in this work, namely face authentication on mobile devices, falls in the second category. The first kind of perturbations that can be expected are varying illumination conditions. Indeed, the user can find himself outdoor, in direct sunlight or under a cloudy sky, or indoor under a variety of artificial light sources. Moreover, since the device is handheld, changes in viewpoint, scale, and orientation of the face are likely to be experienced, even if the holder is fully cooperative. Such perturbations render face detection more difficult, and as a result, the localization will be less precise. As reported by Zhang et al. [84], this negatively impacts the performance of PCA-based methods such as eigenfaces. On the other hand, Elastic Graph Matching does not require the prior execution of a face detection step to extract features, since the matching procedure already embeds such functionality. The distance metrics can be rendered independent of changes in rotation and scale, as it will be discussed in Subsection 3.3.5. Furthermore, compensation for changes in illumination can be applied locally, thus handling non-uniform variations, independently of any facial landmark.

According to Tan et al. [55], EGM is quite robust to changes in expression, thanks to the elastic nature of the matching process, whereas PCA-based methods cannot handle them unless they are present in the training set. Since the foreseen application scenarios imply that the enrollment will have to be carried out on a very small number of samples per client, this condition is unlikely to be verified.

A crucial criterion to consider when targeting mobile devices is the amount of energy required to perform a given task. Contrary to other algorithms that perform well on smaller images, EGM requires higher resolution images [84] and, consequently, a larger number of computations. However, since both the extraction of mathematical morphology features and the iterative matching process are highly regular tasks, they are well suited for implementation on dedicated architectures. Such solutions offer the best trade-off between the computational power and the amount of energy consumed. Finally, both task are very much independent, so that they can be implemented as separate components. This results in a very modular system, where the features extraction technique chosen can be substituted with another, without impacting the rest of the system.

3.3 Elastic Graph Matching algorithm

This section describes the Elastic Graph Matching algorithm employed for the performance evaluations reported in Chapter 4, and for which the System-on-Chip architecture introduced in Chapter 5 has been designed. The starting point for our implementation is the approach proposed by Kotropoulos et al. in [61] — and further detailed in [85] — that was mentioned in Subsection 3.2.3.2.

3.3.1 Overview

The bloc diagram of the enrollment and verification procedures as executed by the EGM algorithm are depicted in Figure 3.6. During enrollment, an image of the face to enlist is acquired, from which features are extracted, as it will be further detailed in Section 3.4. If this process needs to be fully automated, a face detection step is necessary to correctly position the graph, e.g. using the method discussed in Section 5.1.2 in [1], that reuses the EGM algorithm. Several other approaches to face detection methods can be found in [86] and [87]. Otherwise, the system relies on the cooperation of the user, e.g. by asking him to precisely place his face inside of a box displayed on the device screen. Following this step, a feature vector is extracted at each node of the graph so as to create the face template, which is stored in a database, along with the provided identity. In situations where several image samples are available at enrollment, multiple templates can be obtained (see Subsection 3.3.4), or some statistical information can be extracted to enhance the robustness of the matching algorithm.

When a verification is requested, an image is again taken, and features extraction is performed. Face detection can be performed, however since the EGM algorithm intrinsically embeds

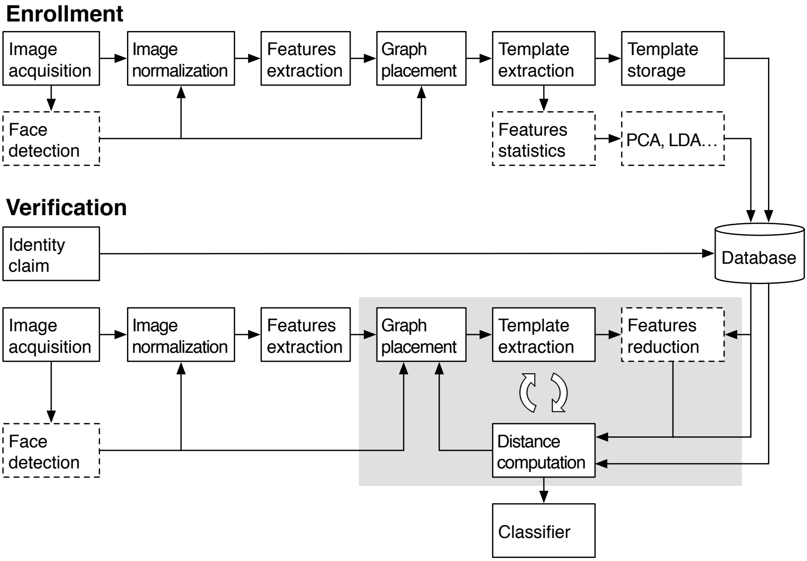


Figure 3.6 : Bloc diagram of the EGM algorithm.

this operation, it is optional. The face template corresponding to the claimed identity is then retrieved from the database, along with the statistical information if it is available. The face matching is then iteratively performed, by executing four sequential steps: 1) template extraction, consisting in getting the features associated with the current position of the graph nodes, 2) an optional features reduction step, 3) computation of the distance between the reference and the extracted template, and 4) generation of a new graph in view of minimizing said distance, as detailed in Subsection 3.3.5. The process stops once the distance cannot be minimized any further, or when a predefined maximum number of iterations is met. Once the process has been concluded, the achieved distance is compared to a threshold to decide upon validation or invalidation of the identity claimed by the user.

3.3.2 Image normalization

The image normalization step that follows the image acquisition is needed to perform photometric correction. Indeed, although the extracted features can be made illumination independent up to a certain point, they are only robust to changes in intensity. However, in real world situations, the mobile device will also be employed under various illuminants, i.e. light sources with different spectral distributions, which can affect the acquired image, as explained in Section 4.2 in [1]. In this work, we assume that the device will be able to apply such corrections, since they are required for regular operation of the digital camera, beyond the face authentication functionality.

3.3.3 Features space reduction

The features statistics bloc appearing in the enrollment, as well the corresponding features reduction bloc intervening in the verification process, are optional and are actually not used in the experiments discussed in this report. Indeed, as explained in Section 4.4 in [1], and in order to be effective, features space reduction techniques, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or node weighting, would require a larger number of samples of the faces to enroll than those available in the XM2VTS database.

3.3.4 Single and multiple templates

In the case where only a low number of images is available per identity for enrollment, each of these images can serve as the basis for extracting a different template, which get all associated to the same identity in the database. During verification, if the matching against the first template fails, the process is repeated with the remaining templates associated to the same identity. Note that all considered comparisons are carried out against the same test image, and that consequently, the features extraction step does not need to be performed again. In fact, only the iterative matching process indicated by the gray area in Figure 3.6 may need to be executed several times. Using multiple templates is an efficient way of improving the verification performance when the large number of samples required for statistical methods cannot be met. Indeed, as it will be shown in Subsection 3.4.3, the EER diminishes by up to 50% when multiple templates are available.

3.3.5 Matching distance

The iterative graph matching step aims at finding the best correspondence between the reference features retrieved from the database, and the features extracted at each node of the graph. Performing an exhaustive search, considering every possible combination of graph position, size, orientation and elastic deformation, would obviously require a huge amount of operations and is definitely intractable in the context of mobile applications. Consequently, a method capable of rapidly converging to the best graph configuration, by simultaneously optimizing the parameters enumerated above, is required. The discussed EGM implementation uses the simplex downhill[2] method [88] to optimize four graph parameters: two for the position, one for the scale factor, and one for the orientation, as explained in Section 4.5.1 in [1].

For mathematical morphology features, the distance is obtained

[2] Sometimes also called Nelder-Mead or Flexible Polyhedron Method.

by summing the Euclidean distances that are computed, at each node, between the two feature vectors, associated to the reference and test graph, respectively, as proposed by Duc [59]. For the coarse rigid and simplex downhill steps, the distance is thus identical to the cost function C_{jets} given in Equation 3.14. In the case of Gabor features, the dot product distance indicated in Equation 3.8 is used. Even though it does not significantly improve the results on the XM2VTS database, it performs much better in presence of non-frontal illumination, as it will be shown in Subsection 4.5.1.

The deformation penalty that gets added to the graph distance during the elastic graph matching phase, is the one based on physical models, discussed in Section 4.5.2 in [1].

$$D_e(i) = \sum_j \left[\left\| \vec{\Delta}^I_{ij} \right\| - \left\| \alpha \vec{\Delta}^M_{ij} \right\| \right]^2 \qquad j \mid \vec{x}_j \in \mathcal{N}(i) \tag{3.16}$$

The penalties in Equation 3.10 to Equation 3.13 and in Equation 3.15, proposed by Lades [51] and by Duc [59], respectively, are only invariant to translations, conversely to the one in Equation 3.16 that is additionally invariant to rotations, since the latter preserve the norm of vectors. Moreover, expansions or contractions of the graph (to account for different face sizes) are taken into account by the introduction of the scale factor α, which renders the penalty insensitive to changes of the graph size. Consequently, the modifications that can be induced by the simplex downhill step, which enables changes in scale and orientation, do not contribute to the deformation penalty.

The metric in Equation 3.16 is illustrated in Figure 3.7, depicting a node $\vec{x}_i$ and three of its neighbors, labeled A, B and C, respectively. The reference graph is displayed on the left part of the figure, whereas the test graph is shown on the right one. As a result of the change of scale and rotation applied by the simplex downhill step, the edge linking $\vec{x}_i$ to A in the image graph is not parallel to the same edge in the model graph. As a consequence, the norm of the vector $\vec{\Delta}_{ij}^{IM} = \vec{\Delta}_{ij}^{I} - \vec{\Delta}_{ij}^{M}$ is different from zero, and a deformation penalty would be incurred when applying Equation 3.12 or Equation 3.15. When considering Equation 3.16, however, the deformation penalty amounts to zero since the norms of the vectors $\vec{\Delta}_{ij}^{I}$ and $\alpha\vec{\Delta}_{ij}^{M}$ are equal. A penalty is nevertheless applied in the case of the edge connecting $\vec{x}_i$ to an elastically displaced node such as C. It can be argued that it is possible to displace the node C in a way that incurs no penalty. This condition is met when C gets moved onto a circle centered over $\vec{x}_i$, whose radius is equal to the length of the undistorted edge linking $\vec{x}_i$ to C. This would nevertheless distort other edges, such as those connecting C to D and to E. Therefore, a penalty would still be applied to account for the displacement of C, unless all displacements only result from the rotation of the entire graph, a geometric transformation that should not be penalized anyway.

The final distance, which gets communicated to the classifier, corresponds to the features distance — or cost function C_{jets} — obtained with the best graph, but without taking the deformation penalty C_{edges} into account.

3.3.6 Classification and distance normalization

The distance obtained at the end of the graph matching step is used to classify the tested individual either as the genuine client, or as an impostor, by comparing it to a threshold that is determined experimentally, as this section explains. Let us suppose that the system is used in authentication mode, and that only one individual is enrolled in the database. A so-called evaluation phase, composed of two steps, is carried out by firstly obtaining

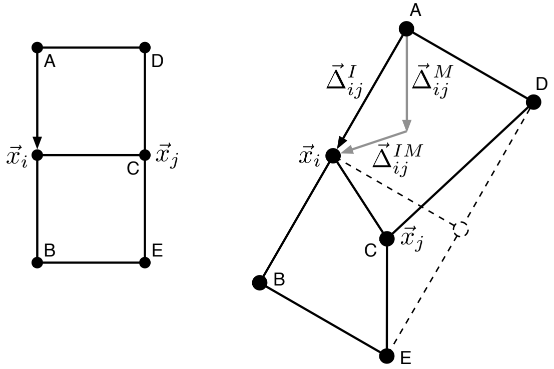


Figure 3.7 : Deformation penalty.
The reference graph is depicted on the left, and the scaled, rotated and elastically distorted test graph is represented on the right.

the matching distances corresponding to genuine tests, where faces of the enrolled person are being acquired and matched to the reference template. The second step consists in repeating the same procedure, but by presenting this time images of other people to the system, resulting in a set of matching distances corresponding to impostor tests. For the reasons discussed in Subsection 2.4.4, the two sets of distances are likely to overlap, which means that some values can be obtained by the genuine client as well as by impostors. Following Bayes decision rule [89], each value will be considered as belonging to the class, either genuine user or impostor, that has the highest a posteriori probability. The threshold can thus be selected for instance as the matching distance for which the probability is equal for both classes, so as to minimize the error probability. It should be however noted that by doing so, we consider that both kinds of errors — whether they result in accepting an impostor or in rejecting the genuine user — have the same severity, which is usually not the case in a real-world situation. Consequently, the actual threshold is selected according to the foreseen scenario. For instance, if the system protects highly sensitive information, it will be operated in a configuration where false acceptances are rendered much less likely than false rejections.

The situation is slightly different if several users are enrolled in the database, since the distributions of the genuine and impostor distances, denoted $d_i^{G,c}$ and $d_i^{I,c}$, respectively, will differ for each client, i.e. for each class c. In order to achieve a situation were a class-independent threshold can be defined, a normalization needs to be applied on the distances. The solution we retained is called Z-norm [89, 90] and consists in obtaining the so-called *standardized* impostor distances $\hat{d}_i^{I,c}$ for each class c. This is achieved by computing the Mahalanobis distances of the values $d_i^{I,c}$ to the mean value $\mu^{I,c}$ of the distribution:

$$\hat{d}_i^{I,c} = \frac{d_i^{I,c} - \mu^{I,c}}{\sigma^{I,c}} \tag{3.17}$$

where $\sigma^{I,c}$ is the standard deviation of the distribution. If the latter were normal, it would now have zero mean and unit standard deviation. The same operation is applied to the genuine distances, using the mean and standard deviation of the impostors:

$$\hat{d}_i^{G,c} = \frac{d_i^{G,c} - \mu^{I,c}}{\sigma^{I,c}} \tag{3.18}$$

The raw and normalized probability density functions (pdf) P and $\hat{P}$, respectively, corresponding to genuine and impostor distances[3] for two clients are plotted in Figure 3.8. The discussed normalization is interesting for evaluating the performance of the system, since it makes it possible to report a single FAR and FRR curve for all enrolled clients. In a practical application, it is also easier to employ a single, class-independent threshold instead of

[3] These distances are assumed to follow a Gaussian distribution, a property that is usually not verified in practice.

multiple, class-dependent ones. Indeed, the class-independent threshold for a new user cannot be determined without priorly obtaining the distances achieved both by the genuine user and by impostors claiming his identity. A class-independent threshold, derived from the results of experiments performed using evaluation clients and impostors, is therefore preferable.

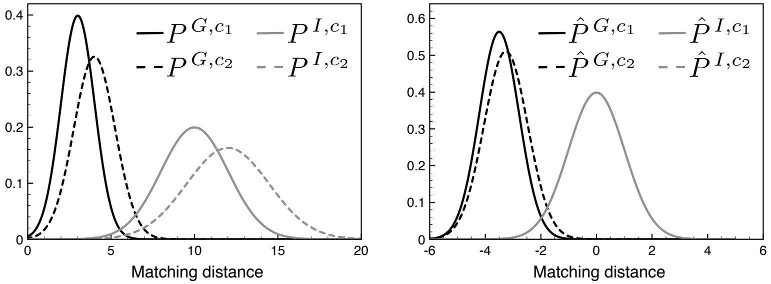


Figure 3.8 : Pdf of the matching distances.

The raw (left graph) and normalized (right graph) probability density functions (pdf) of the genuine P^G and impostor P^I matching distance distributions, plotted for two clients, c_1 and c_2.

3.4 Features extraction

The information attached to each graph node can be obtained using a variety of image processing techniques, which extract certain characteristics from the original image. Ideally, the resulting features must be very discriminative between faces belonging to different individuals. The features extracted must also be robust to external perturbations such as changes in illumination conditions. Finally, in the context of mobile applications, an efficient implementation on low-power devices must be feasible. In this section, two types of features extraction techniques are discussed, based firstly on Gabor filtering and secondly on mathematical morphology, respectively. Furthermore, a normalization is proposed, aiming at rendering the latter technique more robust with respect to varying illumination conditions.

3.4.1 Gabor filters

In the first implementation of Elastic Graph Matching, introduced by Lades et al. [51], Gabor wavelet filter banks were selected to perform the features extraction. This choice was motivated by the similarities between the Gabor transform and neuronal behavior observed in the visual cortex, as reported in [91]. This technique is also widely used in image processing — see for instance [92, 93, 94, 95] — since it offers an interesting compromise between spatial localization and frequency resolution.

This section provides a brief description of the Gabor-based features extraction algorithm used in conjunction with the EGM algorithm, whereas a more detailed discussion can be found in Section 4.3.1 in [1]. The Gabor filter bank we employ is based on the configuration described by Duc [68], and contains 2D filters characterized by the following complex impulse response:

$$g(x,y) \;=\; 2\pi\sigma_x\sigma_y \;\cdot\; \mathrm{e}^{i2\pi\omega_r\tilde{x}} \;\cdot\; \mathrm{e}^{-\pi^2\left(\sigma_x^2\tilde{x}^2 \;+\; \sigma_y^2\tilde{y}^2\right)} \tag{3.19}$$

where i is the imaginary unit that verifies $i^2 = -1$, and with:

$$\tilde{x} = \cos\phi + \sin\phi \quad ; \quad \tilde{y} = -\sin\phi + \cos\phi \tag{3.20}$$

In the spatial-frequency domain, the first exponential term in Equation 3.19 corresponds to a sinusoidal plane with frequency ω_r and orientation ϕ, the second one to an elliptic Gaussian envelope that has its major axis oriented along ϕ, and bandwidths[4] σ_x and σ_y. The filter bank is constituted of 18 Gabor wavelet filters, having 3 different frequency resolutions and 6 orientations,

[4] The bandwidth corresponds to the standard deviation of the Gaussian in the frequency domain, i.e. the points where its magnitude drops to $\frac{1}{\sqrt{\mathrm{e}}}$.

as shown in Figure 3.9. The selected orientations are multiples of $\frac{\pi}{6}$, and the radial bandwidths σ_x are distributed following an octave band structure, the bandwidth doubling at each step. The radial frequencies ω_r and the normal bandwidths σ_y are computed such that neighboring filters intersect along their principal axes at points where their magnitudes is equal to $\frac{1}{\sqrt{e}}$.

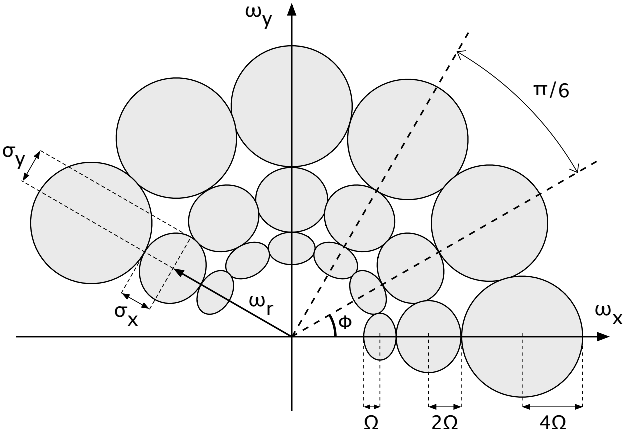


Figure 3.9 : Gabor filter bank.

Gabor filter bank comprising 18 filters, plotted in the 2D frequency domain.

The feature vector is constructed using the magnitude of the 18 complex resulting images. Figure 3.10 shows the response of two Gabor filters plotted in the spatial frequency domain, as well as the corresponding filtered images. The ROC curves (see Subsection 2.4.4) obtained when performing EGM with Gabor features are reported in Figure 3.11.

The achieved performance is very good, especially in the multiple templates case, where the FRR is only around 10% for FAR = 0%. In terms of computational complexity, Gabor fea-

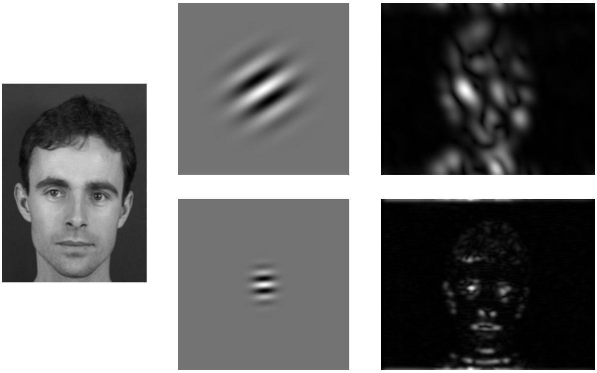


Figure 3.10 : Features extracted with 2 different Gabor filters. Original image (left), real part of the Gabor filters response in the frequency domain (center), and magnitude of the filtered images (right).

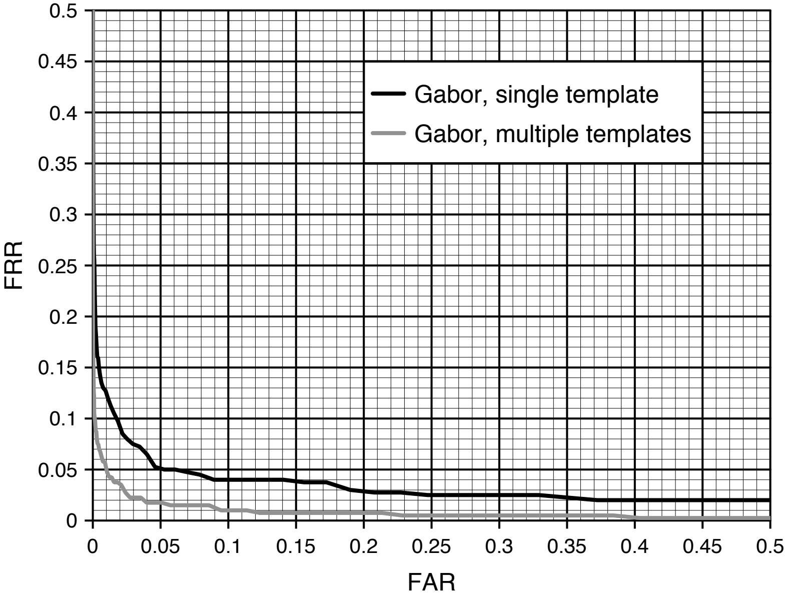


Figure 3.11 : ROC curves for EGM with Gabor features.

tures extraction is advantageously performed in the frequency domain, where the convolutions can be replaced with complex multiplications. Even though performing the filtering only for those pixels actually covered by a graph node might seem more interesting than processing the whole image, most of the pixels will actually be scanned when displacing the graph. The complexity thus amounts to computing the forward and reverse Fast Fourier Transform (FFT) of the image, and to the computation of the 18 spectral responses, each one requiring a complex multiplication, for each point in the transformed image. The required processing power is therefore prohibitively high for mobile applications, and it is necessary to employ a different features extraction method, such as mathematical morphology.

3.4.2 Multiscale mathematical morphology

Mathematical morphology is a non-linear image analysis technique, that was developed in 1964[5] at l'*École des Mines de Paris*, by Georges Matheron and Jean Serra [96]. It was originally employed in petrography and mineralogy, to characterize the mechanical properties of materials such as sections of rocks or polycrystalline ceramics, by analyzing their geometrical structures. One of its first applications was related to estimating the reserve of iron ores in Northern France. It has since then become a powerful image processing tool, discussed in most reference books (e.g. [97]), and used to perform tasks such as segmentation, shape analysis, or texture analysis. Mathematical morphology finds applications in fields as diverse as medical imaging, airborne image processing or optical character recognition (OCR).

[5] Coincidentally, research in automated face recognition started that very same year, see Subsection 2.3.2.

3.4.2.1 Minkowski algebra

Mathematical morphology is based on the Minkowski algebra [98] which defines two operations, the set addition and the set subtraction. Let us consider two objects A and B, that contain elements a and b, respectively, specified by their coordinates $[\ x,\ y\ ]$. The Minkowski addition is defined as:

$$A \oplus B = \{\ a + b \mid a \in A,\ b \in B\ \} \tag{3.21}$$

and the Minkowski subtraction as:

$$A \ominus B = \{\ h \mid b + h \in A \quad \forall\ b \in B\ \} \tag{3.22}$$

The + operator denotes the translation, defined as:

$$[\ x0,\ y0\ ] + [\ x1,\ y1\ ] = [\ x0 + x1,\ y0 + y1\ ] \tag{3.23}$$

As an example, we have:

$$\begin{aligned} A &= \{\ [0,1],\ [1,0],\ [1,1]\ \} && (3.24)\\ B &= \{\ [0,0],\ [0,1]\ \} && (3.25)\\ A \oplus B &= \{\ [0,1],\ [0,2],\ [1,0],\ [1,1], [1,2]\ \} && (3.26)\\ A \ominus B &= \{\ [1,0],\ [1,1]\ \} && (3.27) \end{aligned}$$

Note that in Equation 3.26, the element $[1,1]$ is actually obtained twice, once with $[1,0] + [0,1]$ and once with $[1,1] + [0,0]$, so that $A \oplus B$ contains only 5 elements instead of 6. Equation 3.24 through Equation 3.27 are illustrated in Figure 3.12, the elements contained in the objects being depicted by black squares.

3.4.2.2 Mathematical morphology

Mathematical morphology uses exactly the same operations, but they are now envisaged as *transforming* the object A into a new one, by the mean of a *structuring element*, the object SE. Therefore, the latter can be considered as the equivalent of the convolution kernel in linear filter theory. The two basic operations

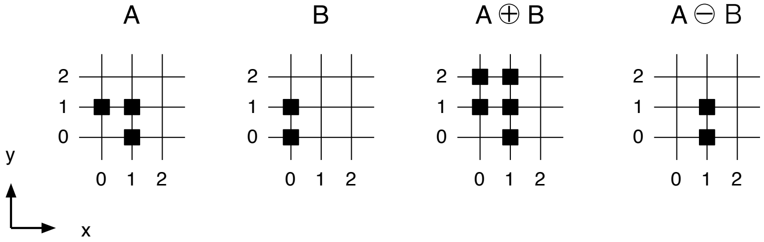


Figure 3.12 : Minkowski addition and subtraction.

are the dilation and erosion, corresponding to the Minkowski addition and subtraction, respectively. Referring to the object A, the *dilation by SE* is defined as:

$$D_{SE}(\ A\ ) = A \oplus SE \tag{3.28}$$

and similarly for the *erosion by SE*:

$$E_{SE}(\ A\ ) = A \ominus SE \tag{3.29}$$

Moreover, the dilation and erosion are now applied to binary images, by defining that the object consists of all the black pixels in the image. Mathematical morphology dilation and erosion applied to a binary image $I(x, y)$ are illustrated in Figure 3.13, where the 0 depicted over SE indicates the origin of the coordinate system. Therefore, SE can be described as:

$$SE = \{\ [0,0],\ [0,1],\ [1,0],\ [-1,0],\ [0,-1]\ \} \tag{3.30}$$

As can be seen in Figure 3.13, the dilated image $D_{SE}(\ I\ )$ consists of all the pixels that are covered by SE when its origin is placed over any pixel of I. Alternately, it can be defined as the ensemble of pixels where the SE can be centered so that it overlaps at least one black pixel. Conversely, the eroded image $E_{SE}(\ I\ )$ is the

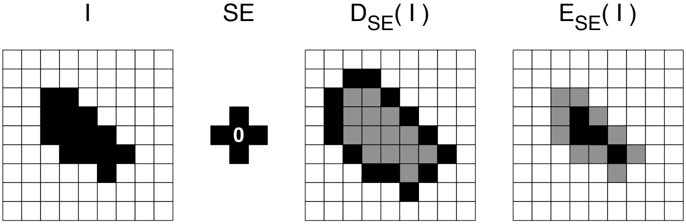


Figure 3.13 : Mathematical morphology dilation and erosion. For $D_{SE}(I)$, the gray pixels depict the pixels that are black in I, whereas the black pixels are those that are added consequently to the dilation. For $E_{SE}(I)$, the gray pixels indicate the black pixels in I that are removed by the erosion.

ensemble of all the pixels of I where SE can be centered, so that all the pixels it covers are black.

The dilation and erosion can be alternately described as follows. Let us assign the value 1 to black pixels and -1 to white pixels, respectively. Then, the value of each pixel in the dilated image can be obtained by centering the SE over the corresponding pixel in the original image, and taking the *maximal* value among the covered pixels. If the SE overlaps the object, this value will be 1, otherwise it will be -1. The erosion can be performed similarly, but now the *minimal* value is looked for. When the SE is placed entirely over the object, the obtained value will be 1, whereas it will be -1 when the SE covers at least one white pixel.

With this new definition, the dilation and erosion by a binary structuring element can be applied to conventional grayscale images as well. It should be nevertheless noted that in such images, white pixels have a higher value than black ones. Consequently, if the image I in Figure 3.13 were a grayscale image containing only white and black pixels as depicted, the illustrated results $D_{SE}(I)$ and $E_{SE}(I)$ would need to be interchanged.

3.4.2.3 Multiscale morphology

Multiscale mathematical morphology, or multiscale morphology was proposed by Jackway et al. [99] as an alternative to convolutions by Gaussian kernels originally used in space-scale analysis. Space-scale theory, formalized by Witkin [100], provides a framework for deriving and manipulating representations of signals at multiple scales. The motivation for such an approach is that real-world signals, conversely to mathematical functions, are naturally associated with physical dimensions. Different features therefore appear at different scales in such signals. For instance, in an image of a forest, the trees and the leaves are characteristics that are better observed at two different levels of details. A theoretical discussion of the general multiscale morphology and of its properties is available in [99]. Only the particular 2D case, consisting in dilating and eroding grayscale images with flat 2D structuring elements, will be discussed here.

Applying multiscale morphology to an image consists in obtaining a series of dilated and eroded versions of said image, using a set of SE sharing the same shape but differing by their size. One such example is the flat disk SE set, defined as follows:

$$SE_N = \left\{ \ [x, y] \ \mid \sqrt{x^2 + y^2} \ \leqslant \ N \right\} \tag{3.31}$$

where N is the scale parameter. We will therefore denote D_N and E_N the results of the dilation and erosion using SE_N, respectively, and refer to them as the dilation and erosion of level N.

It should be noted that the norm in Equation 3.31 is measured from the origin, either to a) the center of the pixel, or to b) the pixel corner being closest to the origin. The first 3 SEs obtained with both variants are depicted in Figure 3.14. It can be seen that in both cases, the width and height of SE_N correspond to $2N+1$. In this work, option b) was used for all the algorithmic evaluations reported in Chapter 3 and Chapter 4. However, since the

SEs depicted in a) cover less pixels, computing the corresponding dilations and erosions requires a smaller amount of operations. Moreover, experiments showed that the achieved results are almost identical in both cases. Consequently, we switched to option a) to obtain the performance figures for the hardware architectures discussed in Chapter 5 through Chapter 7.

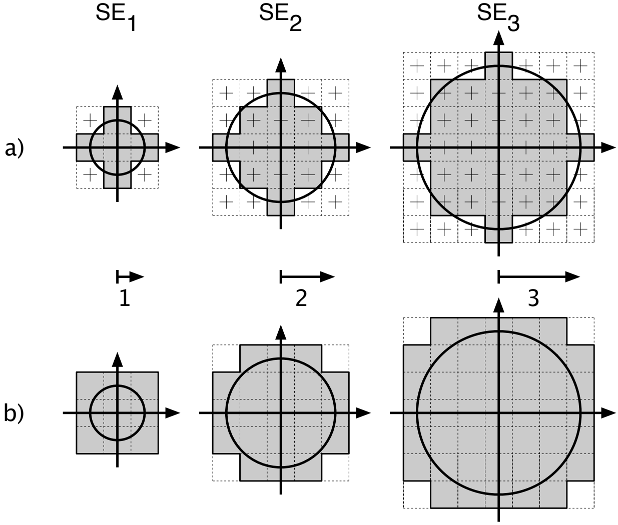


Figure 3.14 : Multiscale SE_N for $N = 1$, 2 and 3.
The distance is measured from the origin to either a) the center of the pixel, denoted by crosses, or to b) the corner being closest to the origin.

3.4.2.4 Mathematical morphology features

The motivations for employing mathematical morphology techniques to extract features to perform face recognition using EGM are exposed by Kotropoulos et al. in [85]. Firstly, dilation and erosion deal with local extrema in the image, a property associated with key facial features such as the eyes, the nostrils and the nose tip. Moreover, since they can be obtained using solely comparisons, the computational complexity is reduced with respect e.g. to Gabor features. Finally, fast recursive implementations are possible, as it will be seen in Section 5.3.

Following Kotropoulos et al. [61, 85], we construct the feature vectors at position (x, y) by taking the results of 9 levels of erosions and dilations, obtained using SE_1 through SE_9. The original value of the pixel $I(x, y)$ is included to these 18 results, leading to a feature vector $M(x, y)$ that contains 19 elements M_i obtained as follows:

$$M(x,y) = \{\ M_0(x,y),\ M_1(x,y),\ \dots\ ,\ M_{18}(x,y)\ \} \tag{3.32}$$

$$M_i(x,y) = \begin{cases} E_{9-i}(x,y) & \text{when } 0 \leqslant \text{i} \leqslant 8 \\ I(x,y) & \text{when i} = 9 \\ D_{i-9}(x,y) & \text{when } 10 \leqslant \text{i} \leqslant 18 \end{cases} \tag{3.33}$$

The images E_N and D_N obtained for the different levels N of erosion and dilation, respectively, are depicted in Figure 3.15, along with the corresponding SE_N, while the resulting ROC curves are reported in Figure 3.16. The achieved performance is substantially below that obtained with the Gabor features, both in single and multiple templates configurations. For instance, in the latter case, when the desired security level is high, e.g. corresponding to FAR below 1%, the FRR is only 5% with Gabor features, whereas the FRR reaches nearly 30% with morphology features. The same situation is observed for FRR below 1%, with FAR amounting to 10% for Gabor, versus 50% for morphology.

Since applying erosion and dilation over an image consists in finding local maxima and minima, the results of the morphological operations are strongly dependent on illumination. This is illustrated in Figure 3.17, which shows three extracted images for the same individual under different lighting. It is obvious that the

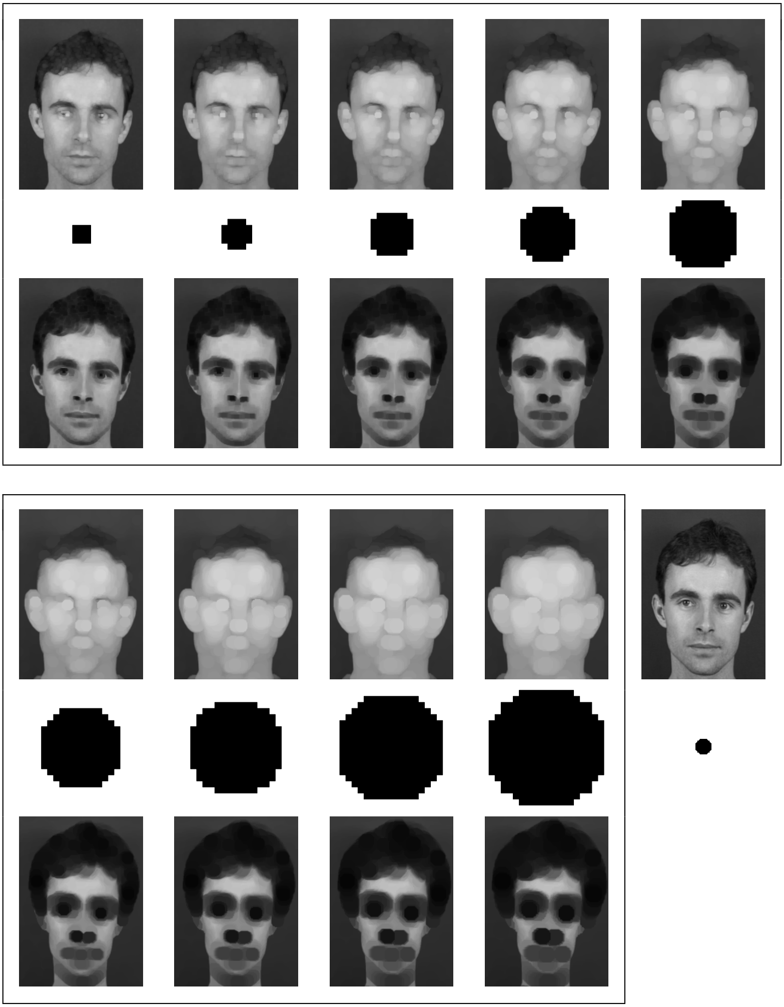


Figure 3.15 : Morphology Features.

Both boxes: eroded (top) and dilated (bottom) images, structuring elements SEs (middle). From left to right: Level 1 to 5 (top box), Level 6 to 9 (bottom box). The face images and the SEs appearing in the boxes are not represented at the same scale. The original image is depicted to the right of the bottom box, with a correctly scaled SE_9 underneath.

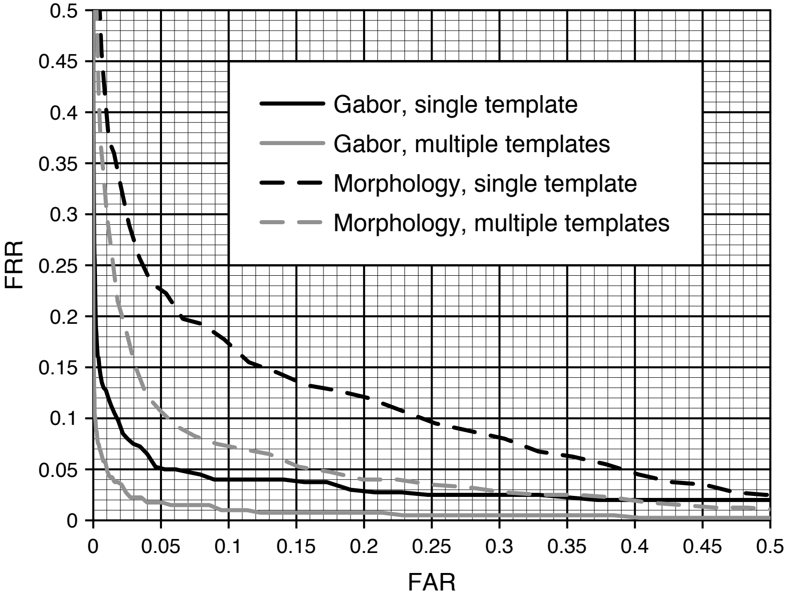


Figure 3.16 : ROC curves for Gabor and morphology features.

values obtained for nodes that fall on the shadowed part of the face will be lower than the ones obtained for the corresponding features in the image under frontal illumination. If the top and bottom rows correspond to the enrollment and test conditions, respectively, this leads to a large distance being computed between the two graphs. Even though at first sight, images in the XM2VTS database appear to have been taken under constant illumination, this is not exactly the case. Examination of the images of one same individual, taken during different sessions, reveals that the variations in pixel intensity in uniform areas, such as the forehead, reach more than 20. Consequently, it is necessary to normalize the results of the morphology operations, so as to alleviate the direct dependency between the illumination and the extracted features.

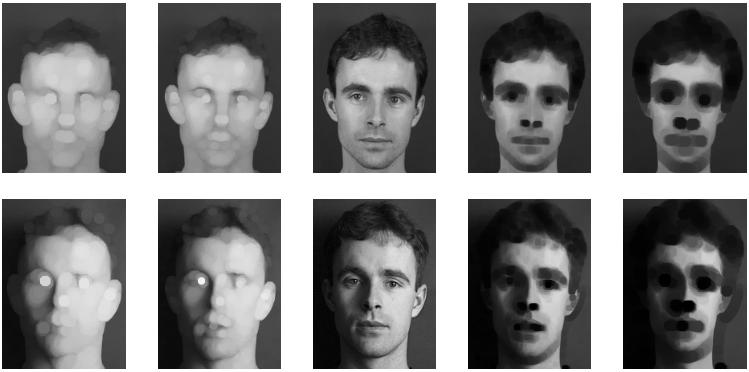


Figure 3.17 : Effect of illumination on morphology features. From left to right: images M_3, M_5, M_9, M_{12} and M_{15}, according to Equation 3.33, under frontal (top) and lateral (bottom) illumination. The two original images, M_9, come from the XM2VTS database, the top one from the regular set (Subsection 4.2.1), the bottom one from the darkened set (Subsection 4.4.1).

One solution proposed by Kotropoulos et al. [69, 70, 71] to handle lateral illumination consists in splitting the facial region into two half-ellipses, which approximate the shape of the face when combined. The values of the pixels in both ellipses are then corrected, so that their respective average intensity is rendered equal. For this procedure to be successful, an accurate face detection step is required beforehand. If the location is imprecise, the normalization can actually make matters worse, as demonstrated in Figure 4.6 in [1]. Consequently, a solution where the illumination can be corrected independently of face detection is preferred.

3.4.3 Normalized mathematical morphology

The proposed normalization must nevertheless be local, since in case of lateral illumination, the variations in luminance are not uniform over the face area, as seen in Figure 3.17. We thus considered firstly a straightforward approach, already employed e.g. in [37], that consists in dividing each pixel value by the average intensity computed over the neighboring pixels. However, instead of normalizing the pixels themselves, we perform this operation on the extracted features. The normalized features $\tilde{M}_i$ are thus obtained by dividing each element M_i of the feature vector by the average $\bar{M}$ of the 19 elements:

$$\bar{M}(x,y) = \frac{1}{19}\sum_{i=0}^{18} M_i(x,y) \tag{3.34}$$

$$\tilde{M}_i(x,y) = \frac{M_i(x,y)}{\bar{M}(x,y)} \tag{3.35}$$

The results obtained for the same two images as in Figure 3.17 are presented in Figure 3.18, and show that the impact of the changing illumination is indeed more limited. The corresponding ROC curves are provided in Figure 3.19, and compared to those obtained with Gabor features. Some amelioration can be noticed when considering low FRR, however the opposite effect is produced for low FAR. Moreover, in this case, the advantage previously incurred by the multiple template scenario is lost. One possible explanation is that the considered normalization tends to attenuate subtle variations found among the enrollment images of a single individual, a phenomenon that defeats the very purpose of extracting multiple templates.

An alternate normalization consists in considering the extrema found in the feature vector, instead of the average value. It is obvious that the minimum and maximum values are E_9 and D_9,

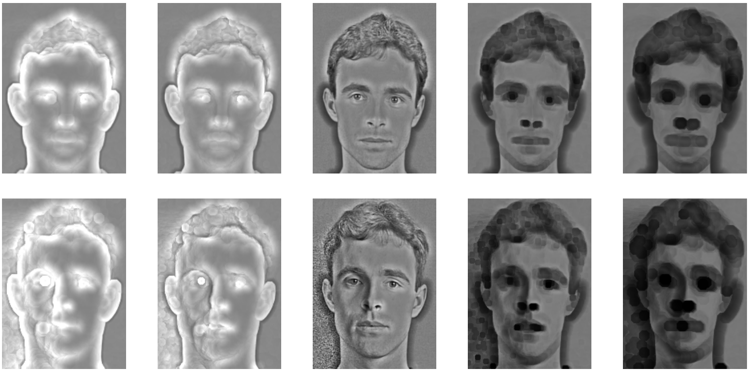


Figure 3.18 : Normalized features according to Equation 3.35.
From left to right: images $\tilde{M}_3$, $\tilde{M}_5$, $\tilde{M}_9$, $\tilde{M}_{12}$ and $\tilde{M}_{15}$, under frontal (top) and lateral (bottom) illumination.

respectively, so that the normalized features $\hat{M}_i$ are obtained as follows:

$$\hat{M}_i(x,y) = \frac{M_i(x,y) - E_9(x,y)}{D_9(x,y) - E_9(x,y)} = \frac{M_i(x,y) - M_0(x,y)}{M_{18}(x,y) - M_0(x,y)} \quad (3.36)$$

The resulting effect is that the feature vector is "stretched" over the entire available dynamic. Indeed, according to Equation 3.36, $\hat{M}_0(x,y)$ and $\hat{M}_{18}(x,y)$ will always be 0 and 1, respectively. These two elements can therefore be discarded from the feature vector, the latter being reduced to 17 elements. Examples of images obtained when normalizing according to Equation 3.36 are presented in Figure 3.20. The performance of the algorithm is strongly improved, as shown in Figure 3.19. In fact, the ROC curves are now almost the same as the ones obtained with Gabor features, especially when considering the multiple templates scenario. Experimentations carried out using a software-based demonstrator and a webcam confirmed that using the normalized features $\hat{M}_i$ does indeed increase the robustness of the system.

In terms of computational complexity, the application of the nor-

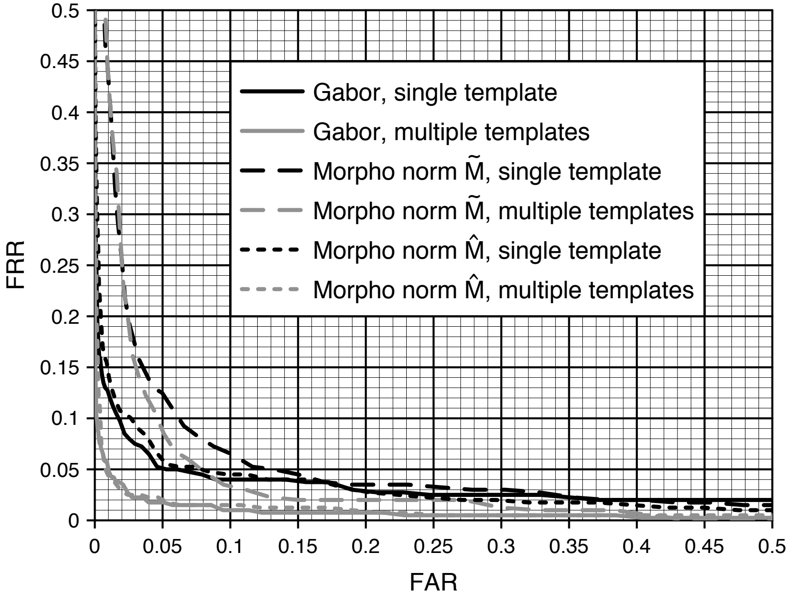


Figure 3.19 : ROC curves for normalized features $\tilde{M}$ and $\hat{M}$. The curves labeled "Morpho norm M̃" and "Morpho norm M̂" correspond to the multiscale morphology normalization performed according to Equation 3.35 and Equation 3.36, respectively.

malization proposed in Equation 3.36 requires 18 subtractions and 17 divisions per position. As it will be demonstrated in Chapter 6, since both the numerator and denominator are restricted to the range 0 to 255, the latter operation can be efficiently implemented using subtractions in the logarithmic space.

3.4.4 Other features

Besides the Gabor and morphology based methods discussed in the previous sections, EGM can be coupled with other features

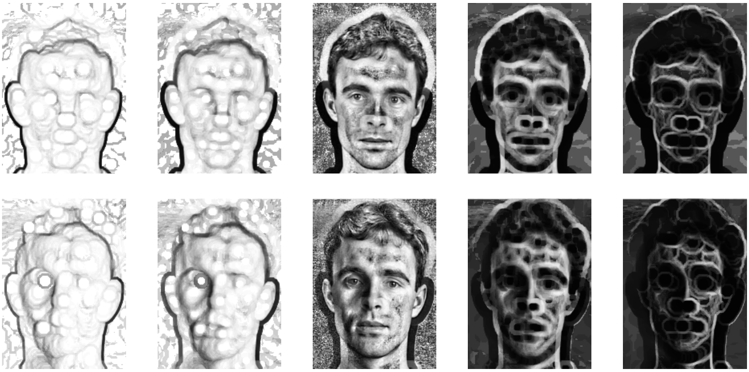


Figure 3.20 : Normalized features according to Equation 3.36. From left to right: images $\hat{M}_3$, $\hat{M}_5$, $\hat{M}_9$, $\hat{M}_{12}$ and $\hat{M}_{15}$, under frontal (top) and lateral (bottom) illumination.

extraction techniques as well. Tefas et al. [101, 102] proposed Morphological Signal Decomposition (MSD)[6], where the face image is modeled as a sum of components obtained with erosions and dilations, similarly to eigenfaces but relying on shape instead of texture information. Reported results achieved on the M2VTS database are similar to those obtained with Morphological Dynamic Link Architecture (MDLA).

The modified version of the Discrete Cosine Transform (DCT), introduced by Sanderson et al. under the name DCT-mod2, has been demonstrated to perform well in conjunction with Hidden Markov Models (HMM) [103, 104], and was also experimented with EGM in Section 4.3.2 in [1]. In this context, however, poor convergence of the algorithm was observed, and the results achieved on the XM2VTS database are not as good as normalized morphology. Although it is reported in Section 4.7.2 of [1] that the latter is outperformed by DCT-mod2 features in presence of variations of the illumination direction, we will show in

[6] Originally called Morphological Shape Decomposition [101].

Subsection 4.5.1 that this is actually not the case.

The DCT-phase, consisting in the sign of the transformed coefficients, has been recently demonstrated by Bracamonte et al. [105] to perform very well as a way of matching JPEG images in the compressed domain. It would therefore be interesting to see how it behaves when used as a features extraction technique for EGM. However, this last experimentation lies beyond the scope of the present PhD work.

3.5 Conclusion

In this chapter, we firstly discussed some of the numerous approaches that have been proposed to perform automatic face recognition. Taking into account the specific requirements of an implementation targeting low-power mobile applications, we selected the Elastic Graph Matching algorithm coupled with mathematical morphology features. In order to improve the robustness, a normalization of the extracted features was introduced, that strongly increases the performance of the system with respect to regular morphology features. We showed that the proposed solution, which will be referred to as *Morphological Elastic Graph Matching (MEGM)* for the remainder of this report, attains the same level of performance than Gabor features, but at a much lower computational complexity. In the following chapter, the performance of the proposed solution will be evaluated using the XM2VTS database, and compared to results reported for other methods. Its resilience to perturbations such as changes in scales and rotations, or presence of occlusions or shadows, will furthermore be assessed.

Chapter 4

Performance and Robustness Evaluation

4.1 Introduction

This chapter discusses several tests performed to evaluate the performance of the MEGM[1] algorithm introduced in Chapter 3. Firstly, Section 4.2 reports the results of experiments carried out using the XM2VTS face database, which was acquired under well-controlled conditions, resulting in mostly constant pose and illumination. Several algorithmic ameliorations are also introduced and evaluated, and our results are compared to those achieved by other methods in recent face recognition competitions. Section 4.3 then discusses additional tests performed using an artificially degraded version of the same database, to assess the robustness of the algorithm in presence of complex backgrounds, geometric transformations such as rotations and changes of scale, and occlusions. Experiments employing the darkened set of the XM2VTS database, where lateral illumina-

[1] Morphological Elastic Graph Matching.

tion of the test subjects causes the faces to be partially covered by shadows, are then discussed in Section 4.4. Finally, Section 4.5 reports the identification results achieved under more degraded illumination conditions, using the Yale Face Database B. Furthermore, several simple heuristic solutions aiming at alleviating the negative impact of the considered perturbations on the performance of the MEGM algorithm are introduced and evaluated in this chapter.

4.2 Frontal illumination

In this section, the proposed MEGM algorithm is evaluated on the standard XM2VTS face database, under optimal test conditions. Indeed, all the images were captured using the same setup, under frontal and mostly constant illumination, preventing the formation of shadows. The background is uniformly tinted, in blue in the original color images, which makes it easy to locate faces[2], and all the individuals are looking straight at the camera, with no rotations, no changes in scale, and no occlusions. Firstly, the XM2VTS database and the associated test protocols are presented. We then introduce three ameliorations that were incorporated in the algorithm, and compare the achieved results with those reported in the literature for different methods.

4.2.1 XM2VTS database

The Extended M2VTS database, or XM2VTS database [107] contains 2'390 images extracted from video sequences, acquired over a four-month period. For each of the 295 subjects, 4 recording sessions were carried out. Two images were extracted from

[2] These characteristics being exploited in [106] to artificially degrade the database by introducing occlusions, see Subsection 4.3.4.

each session, leading to a total of 8 faces per individual. The acquisition environment was very well controlled, the scene being frontally illuminated and the person looking straight into the camera. Some variations can however be observed on the faces themselves as they were recorded over the four sessions, such as a different haircut, glasses being worn at one time but not at another, and a beard getting either grown or shaved.

As provided, the images are in color RGB format, their size being 720 × 576 pixels. We converted them to grayscale, and scaled them down to 320 × 256 pixels using Adobe Photoshop to speed up processing. It should be noted that choosing to use the green component instead of computing the luminance — which could save some processing in an application scenario where the sensor provides color images — does not impact the performance of the algorithm. Two sample images corresponding each to a client from the XM2VTS database are shown in Figure 4.1.

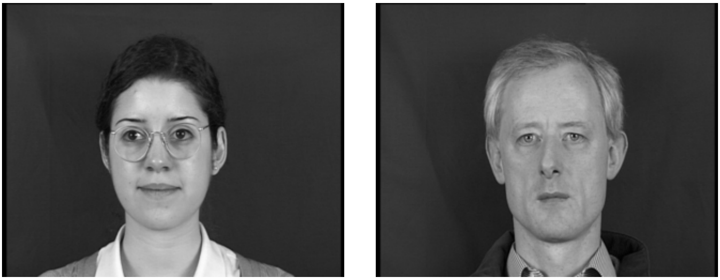


Figure 4.1 : Two sample images from the XM2VTS database.

To ensure that the results of experiences carried out by different teams using the XM2VTS database can be meaningfully compared, a standard methodology, known as the *Lausanne Protocol* [108], was established for reporting the performance of face recognition algorithms in verification mode. The said protocol subdivides the 295 subjects into three sets, constituted of 200 enrolled clients, 25 evaluation impostors, and 70 test impostors, respectively. For each client, the eight images available are furthermore subdivided into three sets, to be used for training, evaluation, and test, respectively. The training set contains images from which the reference features associated with each enrolled client are extracted. Since several training samples are furnished, the protocol makes it possible to consider application scenarios exploiting multiple reference templates per client, as discussed in Subsection 3.3.4. The images in the two evaluation sets enable the collection of series of genuine and impostors distances, from which the parameters needed e.g. for distance normalization (see Subsection 3.3.6) can be derived. Finally, the images in the two test sets are used to assess the performance of the algorithm, using impostors that were so far never seen by the system.

Two different repartition schemes, called *Configurations*, are defined in the Lausanne Protocol to specify how the client images are distributed among the training, evaluation and test sets. In Configuration I, three images are made available for training, namely shot number 1 of the recording sessions 1, 2 and 3. Shot number 2 in each of the same three sessions is part of the evaluation set, whereas the two shots in session 4 constitute the test set. In Configuration II, the training set contains 4 images, corresponding to the shots in sessions 1 and 2. The two images in session 3 are used for evaluation and, similarly to Configuration I, the two shots in session 4 are part of the test set.

In this report, we follow Configuration II unless specified otherwise, since it more closely resembles a real-world scenario where the enrollment will likely need to be executed over a very small number of sessions, possibly even only one. Indeed, only two sessions are used to provide enrollment images in the chosen configuration, whereas three session are used in Configuration I. Since Section 4.4 in [1] shows that features space reduction (with help from PCA, LDA or node weighting) is not adapted to the small number of training samples available in the XM2VTS database, this step was skipped in the tests discussed in this report.

4.2.2 Enhancements

In this subsection, three algorithmic modifications are described, that result in slightly enhanced authentication performance when evaluating the algorithm using the XM2VTS database. However, the effectiveness of the proposed enhancements was not qualitatively assessed under real conditions, contrarily to the regular version of the MEGM algorithm discussed in Chapter 3, which was implemented in a webcam-based demonstrator for this very purpose. As such, the impact of these ameliorations outside of the tests performed on the XM2VTS database, has not been established yet. It is therefore possible for instance that the enhanced inverse logarithm table discussed in Subsection 4.2.2.3 is only effective when used with the images of this specific database.

Each of the three modifications was firstly implemented and tested individually, in view of determining its proper impact and the most efficient parameters, as discussed in Subsection 4.2.2.1 to Subsection 4.2.2.3. For each individual experiment, slight improvements were observed, although only the results achieved once all three modifications were implemented will be provided, in Subsection 4.2.3.

4.2.2.1 Removal of largest node distances

The first enhancement intervenes during the computation of the graph distance, and consists in discarding the largest contributing nodes. This reduces the number of false rejections, by ignoring graph nodes that fall on very changeable areas, such as hair or beard. It can also ameliorate the performance of the system in presence of glasses, if they were not worn at enrollment time, or other perturbations like occlusions (see Subsection 4.3.4.1) and sideways illumination (see Subsection 4.4.2). However, if only the very few best matching nodes are retained, the number of false acceptances logically increases. Experimentally, the best results were obtained when discarding 32 nodes, or half the number of nodes in the graph. An alternate approach consisting in removing nodes whose contribution to the graph distance is above a predefined threshold was also explored, but this variant was abandoned since it was not improving the results.

4.2.2.2 Reduced number of features

The second amelioration consists in reducing the size of the features extracted at each node, by discarding the ones corresponding to the largest structuring elements. Indeed, we originally adopted the approach proposed by Kotropoulos et al. [64], who use 9 circular SE ranging in size from 3×3 to 19×19 pixels, but their choice was solely based on empirical observations. Note that the normalization introduced in Subsection 3.4.3 already led us to reducing the vector size from 19 down to 17 elements. Here, best results were obtained when ignoring SEs larger than 13×13, resulting in a feature vector containing only 13 elements. Similar experiments, but where the discarded features were those corresponding to the smallest SEs instead of the largest ones, did not result in any improvement.

4.2.2.3 Stronger inverse logarithm table

The third amelioration was inspired by an observation made when the divisions involved in features normalization were substituted — for hardware implementation reasons — with subtractions in logarithmic space. As it will be shown in Subsection 6.2.2, the performance of the system degrade sensibly in terms of accuracy, if the inverse logarithm operation is omitted at the end of the normalization. Since this operation consists in replacing the dashed line by the black curve as depicted in Figure 4.2, we tried to apply even stronger exponential functions, by setting $K = 256$ and by reformulating the inverse logarithm function as stored in the corresponding table as:

$$INVLOGMEM[\,N\,] \;=\; 256^{\frac{N}{256}} \;=\; \frac{256}{K}\;K^{\frac{N}{256}}$$

and then raising the value of the exponential base K. Best result are achieved for $K = 1536$. Even though both curves appear very similar at first sight when considered over the whole 256 values in the table (see subplot in grey in Figure 4.2), they nevertheless differ substantially as seen in Figure 4.2.

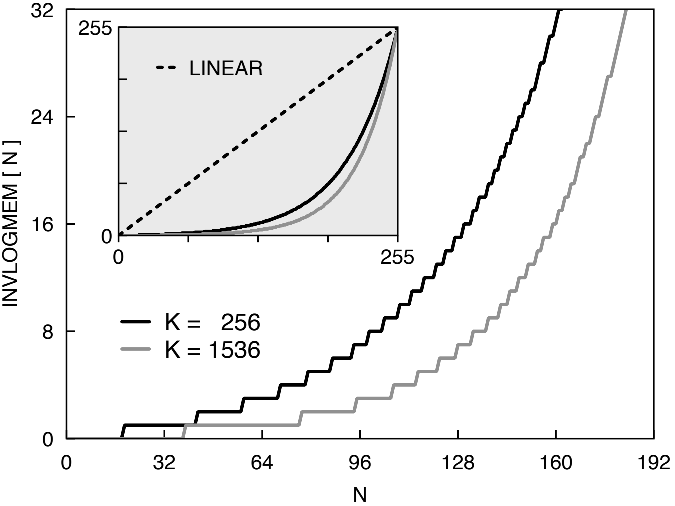


Figure 4.2 : Inverse logarithm.

Illustration of the influence of the base K on values in INVLOGMEM.

4.2.3 Results and computational cost

The ROC curves obtained when implementing all three enhancements discussed in this section are presented in Figure 4.3, both for the single and multiple templates scenarios. In the first case, the FRR diminishes sensibly, especially when considering the situation where the FAR is below 5%, whereas the gain is marginal in the second case. For instance, in the single template scenario, the value of the FRR that corresponds to a FAR of 1% is superior to 14% when the enhancements are not implemented, and inferior to 11% when they are. In the multiple templates scenario however, the FRR never decreases by more than 1%.

In terms of computational complexity, the first enhancement (see Subsection 4.2.2.1) necessitates additional operations to be executed in order to identify the smallest K node distances to retain. This can be accomplished with the help of a partitioning algorithm [109], which reorders a list by placing the smallest K values at the beginning, and the $64 - K$ largest ones at the end. Since the values inside both subsets need not be arranged in any specific order, this operation is less costly than fully sorting the list using an algorithm such as *quicksort* [110]. The average number of comparisons required to partition a series of N values can be approximated by $2N$ [109], the actual number depending on the preexisting order in the series. Experimentally, we indeed observed that, in average, 130 comparisons are performed to partition the 64 node distances. Since half the nodes are later discarded, we save 32 additions when computing the overall graph distance, so that the net increase in executed operations amounts to approximately 100 additions. When put in perspective with the 3'456 MAC operations already required to compute the graph distance[3], the relative overhead incurred is very low.

[3] See Section 6.6.1 in [1].

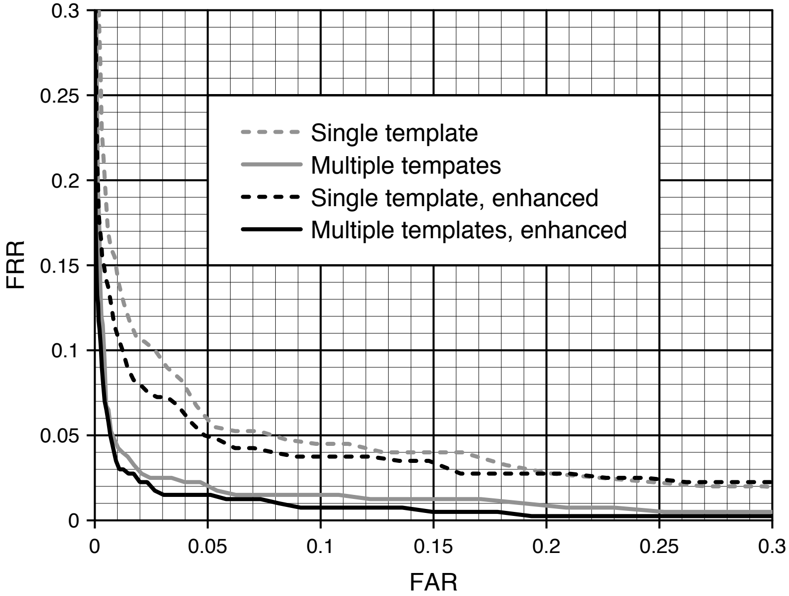


Figure 4.3 : Influence of the proposed enhancements.
ROC curves for both the single and multiple templates scenarios, with and without the three enhancements discussed in Subsection 4.2.2.1 to Subsection 4.2.2.3.

The second amelioration (see Subsection 4.2.2.2) obviously reduces the number of operations to execute to extract the features and to perform the graph matching, because the feature vectors comprise fewer elements. Finally, the third enhancement (see Subsection 4.2.2.3) has no effect on the computational complexity, since it simply consists in writing a different set of values in the inverse logarithm table when the latter gets initialized.

4.2.4 Comparisons with other algorithms

In the framework of the ICB 2006 conference, a competition was organized to assess the recent advances in face recognition [111]. It comprises two distinct parts, the first one being performed on the XM2VTS database, according to the Lausanne protocol, as it was the case for ICPR 2000 [112] and AVBPA 2003 [113], so that the results from these three competitions can be directly compared. The second part evaluates the performance in presence of non-frontal illumination, and will be discussed in Subsection 4.4.1. Compared to those that took place priorly, the competition organized for ICB 2006 [111] furnishes only one new result obtained using automatic face registration and Configuration II of the Lausanne protocol. Indeed, most of the submitted results were achieved using manual registration instead. This result is reported in Figure 4.4, labeled CAS, along with the results from AVBPA 2003 and the ROC curves for our algorithm. It is from the Chinese Academy of Sciences (CAS), whose method [114] consists in an ensemble learning classifier based on Gabor features (5 scales and 8 orientations), extracted from images that are previously photometrically normalized using region-based histograms equalization. The 40 Gabor features are then divided into multiple groups using Adaptive Boosting (AdaBoost), and one classifier is learned for each group through Fisher Discriminant Analysis (FDA). Finally, the classifiers are combined using a fusion strategy.

4.3 Degraded conditions

The images in the original XM2VTS database were acquired under a strictly controlled environment, which obviously corresponds to an ideal situation, but which provides a good initial comparison basis. Nonetheless, there are some limitations, e.g.

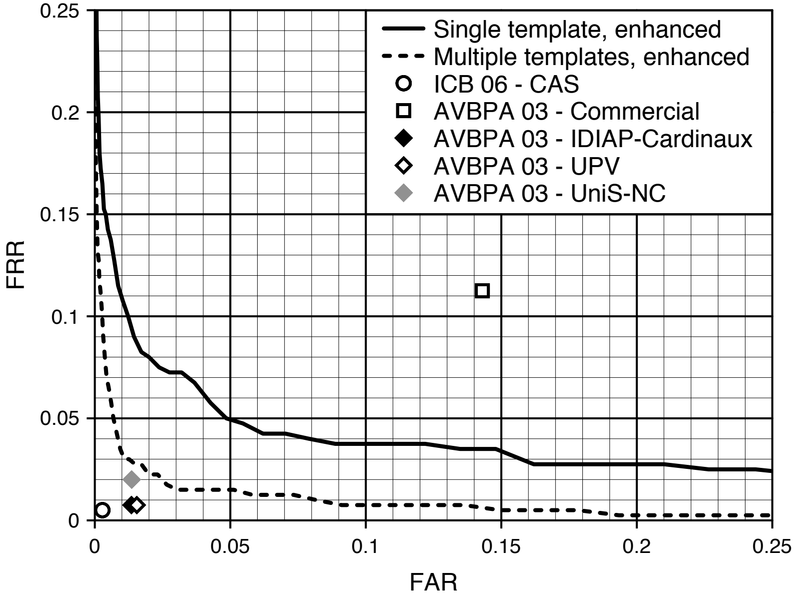


Figure 4.4 : ROC curves vs. ICB 2006.

Comparison between our results, illustrated by the ROC curves, and those that were submitted to ICB 2006 [111].

the faces are neither affected by changes in scale nor rotations, and the illumination is constant. The background is always the same as well, and is constituted of a uniform surface that facilitates the detection of the face within the image. All these aspects help in obtaining performance figures that truly reflect the discriminative capabilities of the evaluated algorithms, and the achieved results are thus very useful to compare different approaches to face verification.

4.3.1 XM2VTS degraded

In a real world setting, however, the situation is very different. Indeed, several perturbations can affect the images to be processed. Within the framework of the COST Action 275 entitled *Biometrics-Based Recognition of People over the Internet* [115], a program was developed to automatically apply several kinds of perturbations to the XM2VTS images, resulting in 11 degraded versions of the database [106]. Some of these simulated problems, such as packet loss during transmission over TCP/IP networks, or artefacts caused by strong JPEG compression, are unlikely to be experienced in our application scenario, namely identification carried out locally on a mobile device. Therefore, only three kinds of alterations were considered, namely occurrence of complex background, geometric transformations (rotation and scaling) and occlusions. In all cases, training and evaluation were performed using the images in the regular XM2VTS set, whereas degraded images were used during the test phase, with the exception of experiments exploiting multiscale references, where the training set contains images that have been scaled up and down, as it will be explained in Subsection 4.3.3.1. The achieved results in presence of each kind of degradations are presented and discussed in Subsection 4.3.2 to Subsection 4.3.4.

4.3.2 Complex background

In a first degraded version of the XM2VTS database, the images were processed to remove the uniform background and replace it with a randomly selected picture from a set of 18 photographs depicting real world environments, such as an office, a parking lot or the hall of a building. Face verification tests were carried out as in Subsection 4.2.1, but the degraded images were used for the test phase. As illustrated by the two images in Figure 4.5, the MEGM algorithm is still able to properly locate the faces.

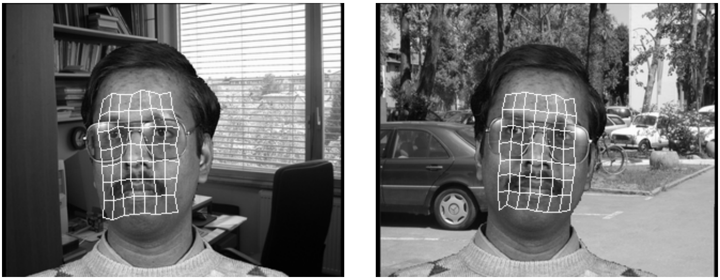


Figure 4.5 : Graph matching over complex backgrounds.

Since the face is correctly registered, i.e. located, despite the complex background, the performance of the algorithm should not vary much, as the faces themselves were left untouched. This is confirmed by the ROC curve reported in Figure 4.6. The small differences result from the best graph being slightly different in some cases. Indeed, since the simplex downhill algorithm does not exhaustively evaluates all possible graphs, the modified background can cause it to follow an alternative convergence path.

4.3.3 Scaling and rotation

A second degraded version of the XM2VTS database contains images to which one or several of the following geometric transformations were applied: change of scale, rotation and translation. Unfortunately, the combination of these 3 operations sometimes causes part of the face to end up outside of the image, which then results in the face being cropped. Since the location of the face is not known a priori in the test phase, translating it does not make much sense anyway. Therefore, we constructed our own databases, where the images are scaled and / or rotated, but not translated, so as to avoid cropping the faces. The procedure

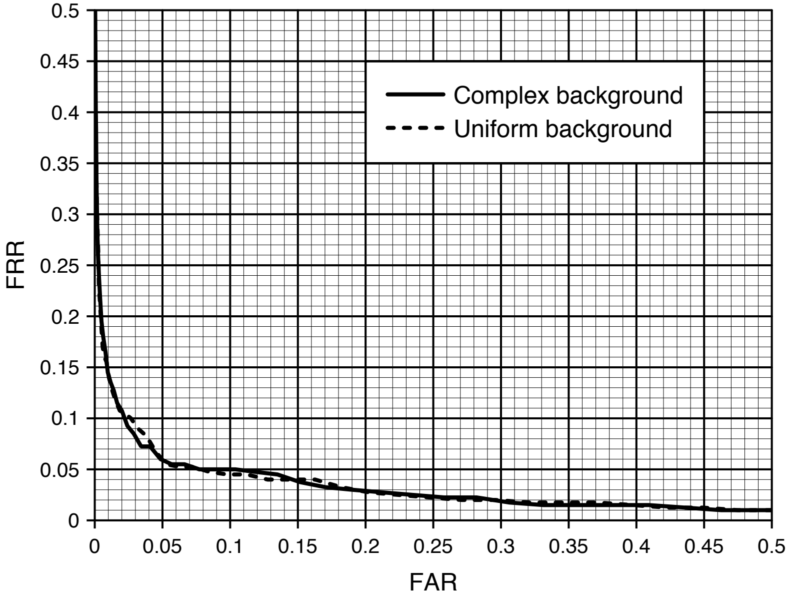


Figure 4.6 : ROC curves obtained with complex background.

we followed is detailed in Subsection 4.3.3.1, whereas the results achieved by the MEGM algorithm in presence of these geometric transformations will be discussed in Subsection 4.3.3.2.

4.3.3.1 Databases construction

The three databases discussed here were generated by applying either scaling, rotation, or both operations, to the images in the test sets of the original XM2VTS database. All manipulations where performed in Adobe Photoshop, using the highest quality interpolation settings available. To create the first database, only scaling operations were considered, using scale factors randomly selected among the four following values: 84%, 96%, 108% and

120%, arbitrarily chosen in the interval between 80% and 120%. Whereas in the degraded XM2VTS images, scale factors range from 70% to 130%, we did set the upper limit to 120% so that the enlarged faces would still fit entirely within the image. We furthermore observed that images scaled below 80%, resulting in face widths being reduced to ca 50 pixels, are problematic for the MEGM algorithm, even if the training and evaluation images are resized accordingly. Graph-based face recognition methods are indeed known to be less efficient in presence of small images, as noted in [54]. Consequently, we decided to set the inferior scale factor limit to 80%.

The second database we generated contains test images that were only rotated. The operation was applied center-wise, using an angle randomly selected among the following values: 0°, –12°, +12°, –24° and +24°, these numbers being arbitrarily chosen so as to cover what can be considered a reasonable worst case scenario for a handheld mobile device[4].

Finally, a third database was created, where the test images are both scaled and rotated, each parameter being randomly selected among the same values as those used to generate the first two databases. Two sample images from this last database are presented in Figure 4.7.

4.3.3.2 Experiments

In a first experiment, only scalings are considered. For the scale factors applied to create the first database, that range from 80% to 120%, the faces are correctly located by the regular variant of the MEGM algorithm, where the simplex downhill matching step adapts the dimension of the graph to account for the scaling of the face. Nevertheless, since the features extracted from the

[4] The rotations in the degraded XM2VTS database range from -30°to +30°.

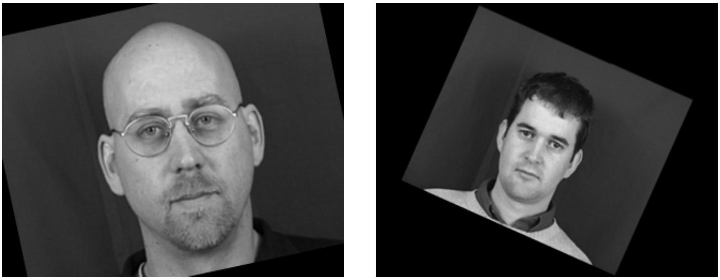


Figure 4.7 : Images from the scaled and rotated database.

scaled test images are matched against the templates obtained from the unscaled training faces, the achieved matching distance gets increased and the verification performance decreases. This is illustrated by the curve labeled "Scaling: single-scale reference" in Figure 4.8, where the EER exceeds 10% and amounts to almost twice the value obtained in the absence of scaling, the latter situation being pictured by the curve labeled "No scaling, no rotation". Indeed, the morphology features are not invariant to this kind of geometric transformations. An elegant and very simple method to compensate for the change of the face size, described in Subsection 4.5.1 in [1], consists in reindexing the structuring elements. However, this step must take place before features normalization since the latter requires the newly adapted values of D9 and E9 (see Subsection 3.4.3). As the change in scale is only estimated during the matching phase, this solution cannot be applied to systems where the features extraction and the matching constitute two independent steps that are performed sequentially. Moreover, this approach is not well suited to cases where the image might need to be enlarged, since the results of dilation and erosion with SEs larger than SE9 are not available.

A more flexible solution consists in extracting several templates from the reference image, each at a different scale. This is similar

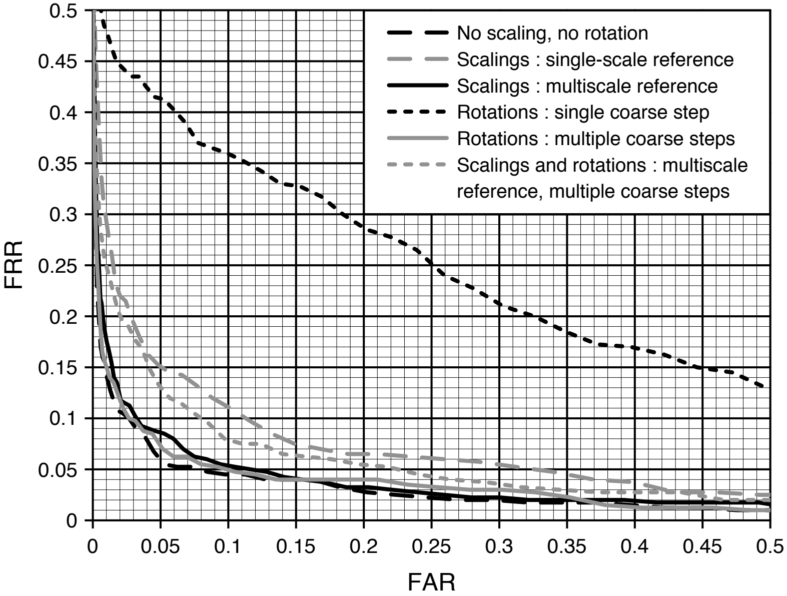


Figure 4.8 : ROC curves for scaled and / or rotated images.

to the multiple templates approach discussed in Subsection 3.3.4, except that the number of references per client is fixed. In our case, we employ four images, generated by applying the following scale factors to the first image in the training set of each client: 78%, 92%, 100% and 115%. These numbers were intentionally chosen so that they differ from those used to scale the images in the test sets (see Subsection 4.3.3.1). Indeed, in a practical application, any scale factor in the considered operating range could be encountered, so that in most cases, the scale factor in the test image will not exactly match any of those applied to the reference templates. Consequently, using the same set of scale factors would lead to a less realistic test case. When using multiscale reference templates, the results are indeed improved, as shown in Figure 4.8, where the corresponding curve, labeled "Scaling: multiscale reference", achieves an EER of less than 7%.

The next experiment involves images that were only rotated, and was carried out using the second database described in Subsection 4.3.3.1. Since the normalized mathematical morphology features are extracted using circular structuring elements, they are invariant to rotations. Furthermore, the orientation of the graph is one of the parameters that the simplex downhill matching step optimizes, as explained in Subsection 3.3.5, so that the presence of rotated faces is not expected to degrade the verification performance of the MEGM algorithm. The ROC curve obtained in this case, labeled "Rotations: single coarse step" in Figure 4.8, nevertheless indicates that the reality is different since the achieved ERR reaches 25%. This situation is due to the fact that the simplex downhill matching algorithm often fails to correctly optimize the orientation of the graph when the amplitude of the rotation is ± 24°, as illustrated by the left image in Figure 4.9. Since the face is not correctly registered, the number of false rejections gets very high.

To correct this convergence problem, multiple iterations of the coarse matching step can be performed, using the same rigid graph with different orientations, instead of relying solely on the simplex downhill algorithm to determine the amplitude of the rotation applied to the face. Indeed, the angle yielding the smallest coarse rigid matching distance can be considered as a rough estimate of the actual rotation angle, and it therefore constitutes a better starting point for the simplex downhill step than the default value, namely 0°. In the experiment discussed here, three coarse matching iterations are performed, with the rigid graph being rotated center-wise by 0°, –20° and +20°, respectively. With this technique, proper face registration can be reestablished, as shown by the right image in Figure 4.9. The corresponding curve in Figure 4.8, labeled “Rotations: multiple coarse steps” is very close to the one obtained in the absence of rotations. The verification performance of the MEGM algorithm can therefore be rendered almost insensitive to rotations, at least up to ± 24°.

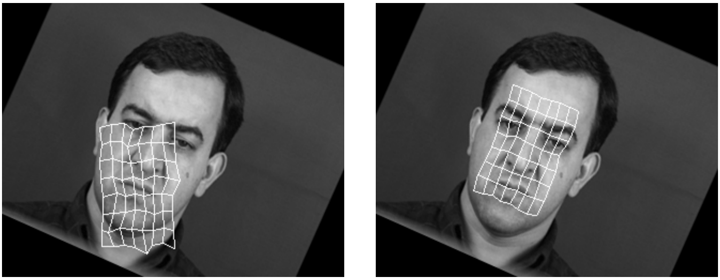


Figure 4.9 : Graph matching in presence of rotations. When the face is rotated by ± 24° or more, the simplex downhill matching algorithm often fails to correctly register the face (left). When additional coarse matching step iterations are performed, a better start angle can be determined and the simplex downhill is more likely to succeed (right).

Finally, a last experiment was conducted using the third database mentioned in Subsection 4.3.3.1, where images are both scaled and rotated. The graph matching is carried out as follows. Firstly, three iterations of the coarse rigid matching step are performed, using the same angles as in the second experiment described above, and the features extracted from the unscaled reference face. The best coarse graph then serves as the starting point for four iterations of the simplex downhill matching step. Each iteration uses a reference template extracted at a different scale, and the graphs are therefore scaled accordingly prior to starting the simplex downhill step. Each iteration of the latter optimizes the position, the size and the orientation of the considered graph. Finally, the graph that achieved the smallest distance is then furthermore optimized during the elastic matching step, using the reference template at the corresponding scale.

The resulting curve, labeled “Scaling and rotations” in Figure 4.8, achieves an EER of 9%.

4.3.4 Occlusions

To simulate occlusions, one of the set of the degraded XM2VTS database contains images where 30% of the face is painted with solid black. One out of four areas is affected, either the top, bottom, left or right part of the face. This situation clearly constitutes an ideal case, since occlusions occurring in a real-world scenario would not be as easy to detect. It is nevertheless interesting, in the sense that it will allow us to evaluate the robustness of the algorithm when only a part of the face is visible, using simple occlusions detectors in Subsection 4.3.4.1 and Subsection 4.3.4.2. A more realistic case, where occlusions exhibit complex patterns, will be briefly discussed in Subsection 4.3.4.3.

As it is obviously impossible to recover any information pertaining to the original image in the affected zones, the features extracted in occluded areas differ significantly from the expected ones. As a result, the distance contributed by the nodes that cover occluded parts of the face increases. Since the algorithm performs automated face registration by minimizing the graph distance, it tends to place the graph fully over the visible part of the face. This typically causes the graph to be scaled down or translated from its correct location, to avoid covering the occlusion, as shown in the left image in Figure 4.10. Unsurprisingly, the performance of the algorithm degrades drastically in such a situation. Indeed, since the face registration is not carried out correctly, the identification of the enrolled clients will hardly be successful. As shown by the corresponding curve, labeled “MEGM regular” in Figure 4.11, the EER in this case increases to more than 40%.

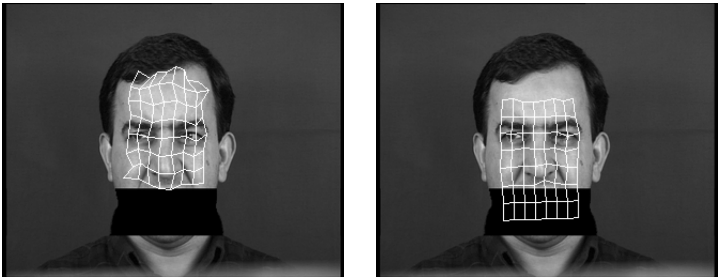


Figure 4.10 : Graph matching on occluded images.

Best graphs achieved when considering all 64 nodes (left) and only the 32 nodes with the smallest contributed distance (right).

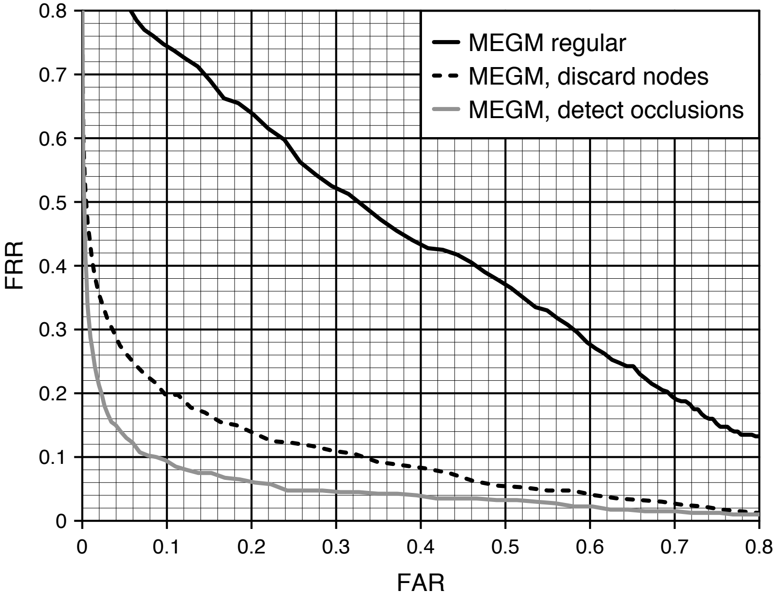


Figure 4.11 : ROC curves in presence of simple occlusions.

4.3.4.1 Discarding nodes with large distance

The achieved results are much better if the nodes with the largest contributed distances are ignored when computing the score, likewise to the procedure detailed in Subsection 4.2.2.1. Additionally, the discarded nodes are not displaced during the elastic graph matching phase. Indeed, since their contribution to the total graph distance is not taken into account anyway, it makes no sense to attempt to lower that value. Moreover, doing so often causes neighboring nodes outside of the occluded area to be consequentially displaced, in an attempt to reduce the deformation penalty incurred by the first displacement, which leads to graphs that are distorted near the occluded areas. Face registration is now carried out correctly in most cases, as illustrated by the right image in Figure 4.10. The improvement is also visible in the ROC curve, the achieved EER being now approximately 16% for the curve labeled “MEGM, discard nodes” in Figure 4.11.

4.3.4.2 Detecting occlusions

Another possible approach consists in attempting to detect the nodes that cover an occluded part of the face, and to ignore them when computing the graph distance. When using the occluded images set from the degraded XM2VTS database, this is easily achieved since occlusions are constituted of areas where the pixel values are all zero. To identify the covered area, the result of dilating the local neighborhood with SE4 is considered: if this value is zero, then the location is labeled as being part of an occluded area. Using a dilation result instead of the direct pixel value ensures that the selection does not include individual black pixels, or small black spots. Indeed, pure black pixels can be found in the non-occluded part of the image, e.g. in the nostrils. However, this restriction incurs the opposite effects since the pixel on the internal border of the occlusions are not selected anymore. Indeed, pixels lying just outside the occlusion will cause the value of SE4 for pixels just inside to be larger than zero. Moreover, since the normalization performed when extracting features consider the local neighborhood under the largest structuring element, nodes falling over the surroundings of a blackened area must also be discarded. To account for these two problems, the binary mask image indicating the detected occluded areas is further dilated using SE9.

During the matching phase, only nodes falling outside of the resulting binary mask are taken into account when computing the graph score S. To compensate for the reduced number of contributing nodes, the final score S_f is obtained by scaling S according to the ratio of the total number of nodes N_t to the number of retained nodes N_r, as indicated by Equation 4.1. The average distance contribution of the discarded nodes is thus considered to be equal to that of those retained.

$$S_f = S\,\frac{N_t}{N_r} \tag{4.1}$$

Identically to the previous solution, the elastic phase is modified to avoid displacing nodes located over occluded areas. Visual inspection of the registered graphs confirms that the face is now correctly located in most cases. The achieved results are better than those obtained when discarding nodes based on contributed distances, as shown by the curve labeled “MEGM, detect occlusions” in Figure 4.11. For genuine clients, the final graph contains an average of 16 discarded nodes, this number ranging from 12 to 24 in 90% of the tests.

4.3.4.3 Complex occlusions

As already noted, the case studied in the previous section is ideal, since the occlusion pattern found in the considered XM2VTS database set is uniform and known beforehand, whereas this would not be the case in a practical situation. Performing occlusion detection at the features extraction stage, as discussed in Subsection 4.3.4.2, is therefore likely to prove more difficult. During the matching stage however, the distances can be expected to remain larger for these nodes, since it was shown in Subsection 4.3.2 that face registration is correctly carried out over complex background. Consequently, the method consisting in discarding the nodes with the largest distances, exposed in Subsection 4.3.4.1, should also be effective in presence of complex occlusions.

To verify this hypothesis, the test images of the first eight clients in the database were manually modified, the occluded area being replaced by parts of the complex backgrounds taken from the images used in Subsection 4.3.2. Matching was then performed using the proposed approach of discarding the nodes with largest distances. Over the small test set used, the face registration is indeed successful, as illustrated in Figure 4.12. More refined approaches could consist in discarding nodes with large distance only when they appear in groups, instead of considering them independently. This would help ensuring that the high contributions are not due to local dissimilarities in the extracted features — in which case ignoring those nodes would result in increasing the number of false acceptances — but are found to cover a larger area, a situation more likely to denote an actual occlusion.

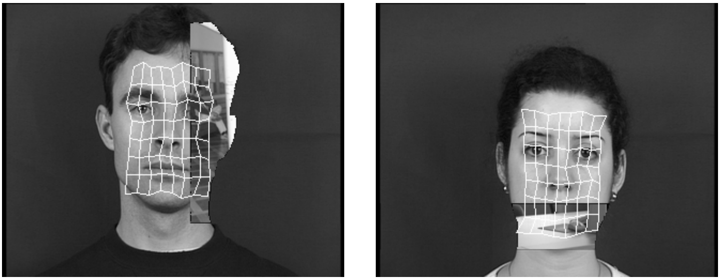


Figure 4.12 : Best graph on images with complex occlusions.

The proposed method therefore appears viable, but a larger scale experiment using a complete database of faces occluded with complex patterns would obviously need to be carried out to confirm or infirm these preliminary results.

4.4 Sideways illumination

In this section, the influence of varying illumination on the robustness of the algorithm is evaluated. The extracted normalized morphology features are designed to locally compensate for changes in intensity, so that the algorithm is mostly insensitive to changes in illumination intensity, even if the variation is not uniform over the face. One situation where such conditions can be encountered is the case where the light source is no more frontally located with respect to the subject, but comes from one of the sides. Since the face is not flat but has in rough approximation a cylindrical shape, lateral illumination coming from one side results in the opposite part of the face appearing darkened. Furthermore, salient features such as the nose obstruct the light path, causing cast shadows to appear.

Throughout this section, we use a different set of the XM2VTS database, containing images with sideways illumination, to assess the performance of our algorithm is such conditions. Discarding nodes with large contributed graph distance will be applied once again to enhance the robustness. Results achieved by other methods on the same data set in the frame of the ICB 2006 conference are also reported.

4.4.1 XM2VTS darkened

The second part of the ICB 2006 competition, mentioned in Subsection 4.2.4, employs the so-called *darkened set* of the XM2VTS database. In the regular set, all subjects are uniformly and frontally illuminated, so that virtually no shadow appears on the faces. The darkened set, on the other hand, contains images of the same individuals, but with the light source being displaced either to the left or to the right. There are four images available per subject, acquired during the last of the four recording sessions (see Subsection 4.2.1), two being lit from the left, and two from the right. Figure 4.13 shows two sample images from the darkened set, one for each illumination direction.

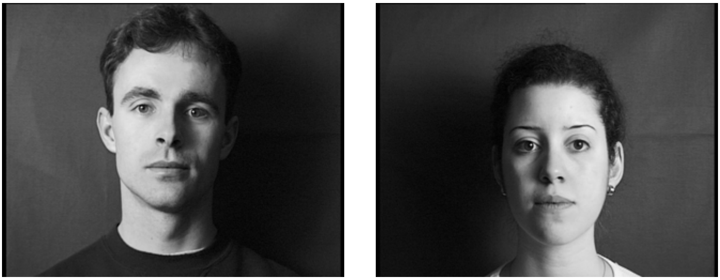


Figure 4.13 : XM2VTS darkened – sample images.

The competition guidelines specifiy that Configuration I of the Lausanne protocol be followed for the training and evaluation phases, using images from the regular XM2VTS database. For the test phase, images are taken from the darkened set, the protocol being slightly modified to account for the fact that there are now 4 images for both clients and impostors, instead of respectively 2 and 8 in the regular set. Two scenarios are proposed, one where the faces are manually located, and one where registration must be carried out automatically.

In Figure 4.14, we report the results published at ICB 2006 (see [111] that additionally provides ROC curves for some of the algorithms) for the manual face registration scenario, along with the ROC curves obtained with our algorithm. The unmodified version, labeled “MEGM regular”, achieves an EER of approximately 18.5%, which is much worse than the 5% reached on the regular XM2VTS database. An improved version is discussed in the following section.

4.4.2 Discarding nodes with large distance

Since the nature of the degradations observed in the darkened images is similar to those created by occlusions, we decided to follow the same approach as in Subsection 4.3.4.1. The situation improves sensibly, as shown by the curve labeled “MEGM discard” in Figure 4.14. Only two algorithms perform significantly better in this test, the first one being UNIS-Lda proposed by the University of Surrey. According to [111], they apply standard Linear Discriminant Analysis (LDA) on photometrically normalized images, a process carried out using filtering and histogram equalization. The second one is CAS, from the Chinese Academy of Science, described in Subsection 4.2.4. It is however noted in [111] that CAS did “relight the training and evaluation set data to simulate the illumination conditions of the test set”, which affects the comparability of the results. Indeed, in doing so they effectively tuned their system for authenticating faces with lateral illumination, which might impact the achieved performance when faces are uniformly illuminated. On the other hand, as seen in Subsection 4.2.2.1, the node discarding technique we employ was verified to avoid producing degraded results in presence of frontal lighting.

The performance of the MEGM algorithm is not as good when the face registration must be performed automatically. Indeed, in that case, the number of faces incorrectly located when testing genuine clients amounts to more than 30% with MEGM regular, and still reaches 14% when rejecting nodes with large contributed graph distances. Obviously, failure to correctly register the face sensibly increases the FRR, causing the ERR to raise above 25%, as illustrated by the gray curve in Figure 4.14. It can be never-

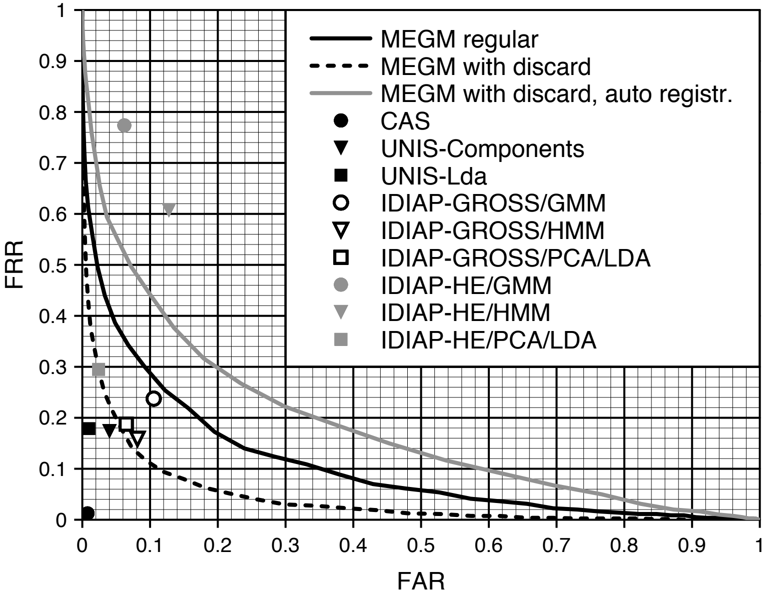


Figure 4.14 : ROC curves for the darkened XM2VTS set[5]. Comparison between our results, illustrated by the ROC curves, and those that were submitted to ICB 2006.

[5] Note that we permuted the values of FAR and FRR for UNIS-Lda, with respect to those reported in Table 5 in [111]. Indeed, the ROC curve in Figure 5 of the same reference makes it clear that they are inverted in said table. This is furthermore confirmed by the granularity of the FRR, which has to be 0.125% since 800 tests (200 clients × 4 images) are performed.

theless noted that our results in automatic face registration mode are superior to those achieved by two other methods when using manual registration.

4.5 Strongly shadowed faces

Even though the title of the paper reporting results discussed in the previous section evokes “severe illumination changes” [111], the situation can get much worse in uncontrolled environments. In this section, experiments using a database acquired under 64 different illuminations are reported. Another protocol is followed, where the performance is evaluated according to an identification scenario. New algorithmic enhancements are introduced, that aim at alleviating the sensitivity of the proposed method to strong perturbations incurred by severe illumination conditions.

4.5.1 Yale Face Database B

The Yale Face Database B [116] contains 5’760 single light source grayscale images of 10 subjects, each seen under 576 viewing conditions, namely 9 poses multiplied by 64 different illuminations. Acquisition was realized with the help of a spherical rig holding a series of spots providing the light sources. The size of the images is 640 × 480 pixels, and each one is labeled with a number identifying the individual, the pose, as well as the azimuth and elevation angles of the spotlight. For each image, the coordinates of the eyes are provided. The database is freely available and can be downloaded from [117].

Recently, a new version was made available, the Extended Yale Face Database B [118]. It includes 28 additional individuals, and can be freely obtained from [44]. The coordinates of the eyes are however not provided, the manual registration scenario relying instead on a second version of the database, containing manually cropped, scaled and rotated images.

In the experiments reported here, only the frontal pose was considered and the identification protocol proposed in [116] was followed. For each individual, only 45 images were retained and divided into four subsets according to the location of the illuminating spotlight, the azimuth and elevation of the light source increasing with the subset index. Two sample images from each subset are shown in Figure 4.15. The images in Subset 1, acquired with near frontal illumination, are used to create 7 reference templates for each individual. The faces in Subset 2, 3 and 4 are then matched against all 70 reference templates, using manual registration. The identification is successful if both the test face and the best matching reference belong to the same person. The percentage of errors in each subset for several algorithms is reported in Table 4.1.

	Error rate (%)		
Method	Subset 2	Subset 3	Subset 4
Cones-attached [116]	0.0	0.0	8.6
Cones-cast [116]	0.0	0.0	0.0
EGM Gabor (dot dist.)	0.0	0.0	12.1
EGM Gabor (Eucl. dist.)	0.8	5.8	63.6
EGM DCTmod2	0.0	5.0	67.1
EGM Morphology			
baseline	10.0	47.5	77.9
normalization 1	0.0	13.3	55.0
normalization 2	0.0	0.8	40.7

Table 4.1 : Identification results on the Yale B database.

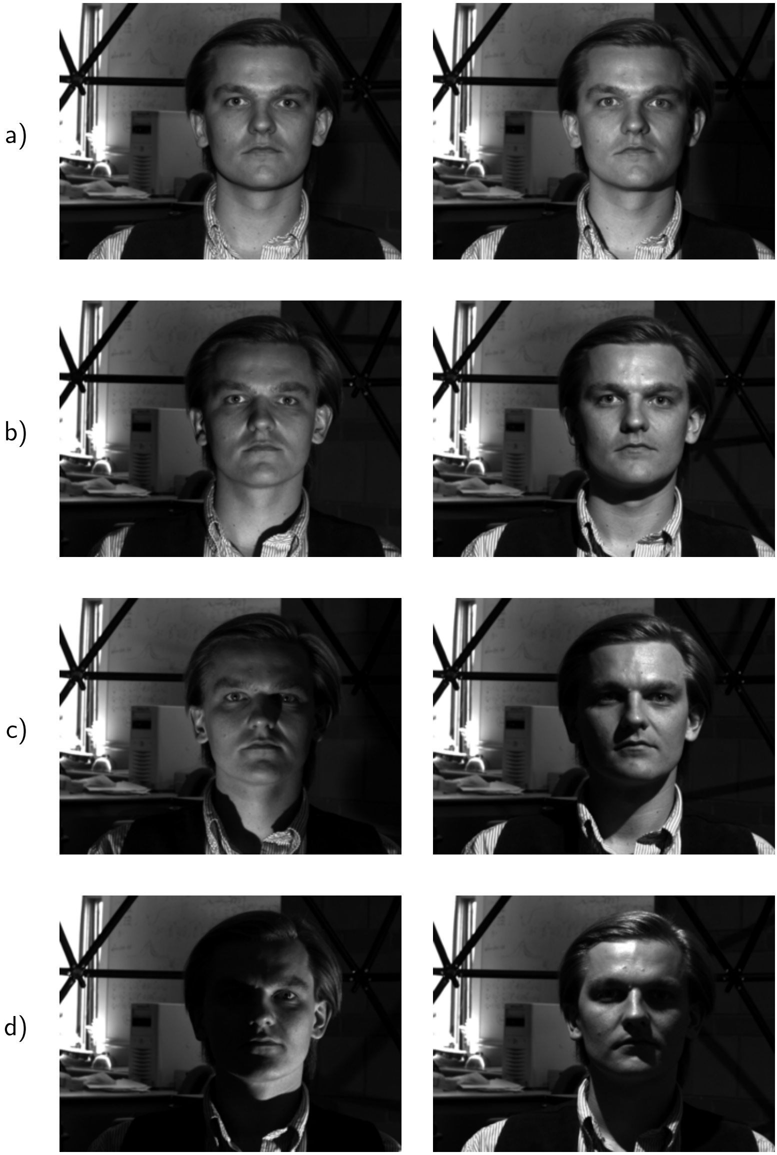


Figure 4.15 : Samples from the Yale B database.
The images belong to a) Subset 1, b) Subset 2, c) Subset 3, d) Subset 4.

For EGM using Gabor or DCT features, the results differ slightly from those reported in Section 4.7.2 in [1]. This is not surprising as both experiments did not use the exact same set of images. Indeed, The Yale B database as available from [117] contains images whose size is 640 × 480 pixels, which were downsized to 320 × 240 pixels. Depending on the software used to perform the resizing, and especially on the downsampling filter it implements, the resulting images will not be exactly identical. In the case of the normalized morphology features, however, the observed disparity is much larger and is definitely caused by another factor. Examination of the archived projects used to generate the results in [1] revealed that an elastic deformation step was performed after manual registration of the rigid graph. Since the expression of the subjects in the Yale B database does not vary much between the reference and test images[6], allowing elastic deformations of the graph will not yield much improved results. Indeed, when measured between two images of the same face, the score of the rigid and elastic graph is very close, as both graphs are almost identical.

However, if elastic deformations are not sufficiently constrained, that is if the elasticity coefficient λ in Equation 3.9 is too small, the number of identification errors increases greatly. Indeed, if the graph is allowed to get strongly distorted, the distance contributed by some nodes can get very low, even though the reference and test images do not represent the same face. This is illustrated in Figure 4.16, where the test image a) from Subset 3 is to be identified. As shown in b), with $\lambda = 0.005$, the best match found among the reference face images does not represent the same individual and the corresponding graph is indeed severely distorted. When the value of λ is increased to 0.5, the result is a graph that exhibits almost no elastic deformation, and the individual gets correctly identified.

[6] All images of the same subject were taken within a 2-second time frame.

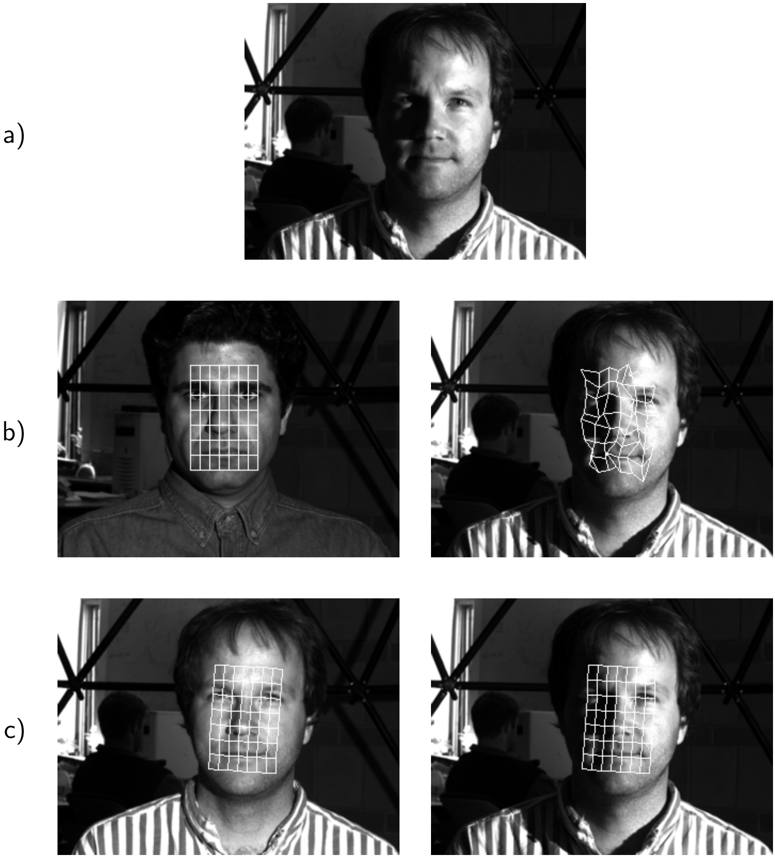


Figure 4.16 : a) Test image from Subset 3, b) Reference image identified as best match (left) and corresponding graph on test image (right) for $\lambda = 0.005$, c) same as b) but for $\lambda = 0.5$.

The error rate achieved by the MEGM algorithm when using the best empirically determined elasticity coefficient, $\lambda = 0.5$, is improved by approximately 2% on Subset 4 only, with respect to the case where elastic deformations are not executed, as seen in Table 4.2. For this value of λ, the weight of the deformation penalty on the graph score is of the same order as the contribution of the Euclidean distance. Only one identification error now occurs on Subset 3, however even though it is significantly better, the error rate for Subset 4 is still very high. The remainder of this chapter discusses several proposed modifications, aiming at improving the performance on this subset.

	Error rate (%)		
Method	Subset 2	Subset 3	Subset 4
EGM norm. morpho 2			
rigid graph	0.0	0.8	40.7
elastic graph, $\lambda = 0.5$	0.0	0.8	38.6
elastic graph, $\lambda = 0.005$	0.0	15.0	62.9

Table 4.2 : Influence of the elasticity coefficient λ.

4.5.2 Discarding nodes with large distance

The first approach is similar to the one discussed in Subsection 4.4.2, and consists in discarding the K nodes exhibiting the largest contributed graph distances. The results obtained for various values of K are reported in Table 4.3. The error rate improves somewhat on Subset 4 for $K = 20$, unfortunately at the cost of an increased error rate on Subset 3.

A variant was also tested, where discarded nodes are those whose contributed distance is above a given threshold T. Since the num-

	Error rate (%)		
K	Subset 2	Subset 3	Subset 4
10	0.0	1.7	37.9
20	0.0	1.7	37.1
30	0.0	2.5	39.3
40	0.0	2.5	42.9
50	0.0	5.8	42.9
60	1.7	9.2	61.4

Table 4.3 : Results on the Yale B database when discarding the K nodes with the largest distances.

ber of discarded nodes is not constant anymore, the graph score is scaled according to Equation 4.1. Several values of T were tried, starting with $T = 3.56$, that corresponds to the largest node distance observed. Some of the collected results are reported in Table 4.4. As it can be observed, the method has no effect for $T \geqslant 3.2$, whereas smaller values of T actually increase the error rate in both Subset 3 and 4.

Clearly, discarding nodes based on the sole criteria of their contri-

	Error rate (%)		
Threshold *T*	Subset 2	Subset 3	Subset 4
$\geqslant 3.2$	0.0	0.8	40.7
2.8	0.0	0.8	42.9
2.4	0.0	1.7	41.4
2.0	0.0	5.8	60.0

Table 4.4 : Results on the Yale B database when discarding nodes whose contributed distance is greater than T.

bution to the graph distance is not efficient, and other approaches will need to be implemented and tested.

4.5.3 Discarding shadowed nodes

Before looking for a method capable of detecting shadowed zones in the image to enable identification of those areas in which the nodes will have to be discarded, we will firstly evaluate the effectiveness of such a scheme. Hence, we need to determine the error rate that can be achieved when the affected areas are known a priori. To this effect, we exploit the fact that the position of the illumination source is specified for every image in the database. Considering only the azimuth, that is the horizontal angle between the light strobe and the frontal direction, we decompose the 8 × 8 graphs into a left and right half, composed of 4 × 8 nodes each. For each test image, the half graph located on the side of the face that is closer to the light source is labeled G (for *good*), whereas the other one is labeled B (for *bad*), as illustrated in Figure 4.17.

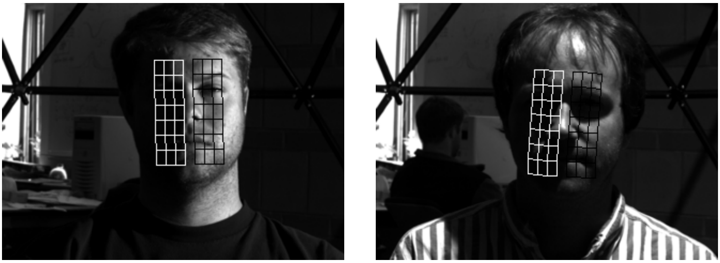


Figure 4.17 : Half graphs G (in black) and B (in white) for two light sources located on the right.

Note that for cases where the illumination source is also displaced vertically, as in the right image in Figure 4.17 for instance, strong shadows can also be found underneath the G graph. The identification scenario test was then performed twice. During the first execution, only the nodes belonging to the G graph were taken into account when computing the matching distance, whereas the second execution considered only the nodes of the B graph.

The error rate for both tests is reported in Table 4.5. It can be seen that the results are sensibly improved on Subset 4 when using the G graph, without affecting those achieved on Subset 2 and Subset 3. When considering the B graph, however, results are strongly degraded even in Subset 3.

Therefore, and provided we can find an effective way of determining which nodes are located inside a shadowed area without using a priori knowledge, we conclude that the proposed approach of discarding those nodes should indeed decrease the error rate.

	Error rate (%)		
Retained half	Subset 2	Subset 3	Subset 4
Good (G)	0.0	0.8	27.1
Bad (B)	0.0	19.2	72.9
Full graph (G + B)	0.0	0.8	38.6

Table 4.5 : Results on Yale B database when using *good* or *bad* half graph, according to a priori knowledge of the direction of illumination.

4.5.4 Shadow detection

In this section, several approaches are presented that were tested to select nodes to be rejected. For each method, results with the set of parameters giving the best results when applied in the frame of the considered identification scenario on the Yale B database are reported in Table 4.6. The corresponding detected shadowed areas for one test image in Subset 4 are presented in Figure 4.18. The detected area forms the *shadow mask*, depicted in white in the figures so as to render it more visible when displayed over the face images. During the matching process, the graph nodes falling over the shadow mask will be discarded when computing the graph score. The test image in Figure 4.18 was randomly selected among the ones that are incorrectly identified when considering all the nodes of the graph. Incidentally, the chosen image is not correctly labeled when only the nodes of the G graph of Subsection 4.5.3 are retained.

	Error rate (%)		
Shadow detection	Subset 2	Subset 3	Subset 4
Method 1	0.0	0.8	41.4
Method 2	0.0	0.8	36.4
Method 3	0.0	0.8	32.1

Table 4.6 : Results on Yale B database for Method 1, 2 and 3.

4.5.4.1 Method 1

The first approach tries to locate shadows in the images by identifying areas where the brightest pixel value is below a given threshold. It is based on the values of the dilations computed as part of the features extraction, before normalization is applied. Indeed, each dilation returns the highest value found among the pixels covered by the corresponding SE. This prevents the classification of individual dark pixels or small dark spots as shadowed area. Experimentally, we obtain the best results when considering D9, that is the dilation using the largest SE, and when defining as shadows those areas where $D9 < 32$. A variant with an additional condition, requiring $E9$ to be lower than a specified threshold, had no influence on the results. As shown in Table 4.6, the error rate actually increases when using this method.

4.5.4.2 Method 2

As can be seen in Figure 4.18b, many shaded parts of the image are not identified by Method 1. This is the case for instance for the shadow over the depression created by the orbital cavity on the right. Method 2 attempts to solve this problem by simply dilating the shadow mask of Method 1, using one of the SEs used during features extraction. Since the mask is a binary image, only 1-bit comparisons need to be performed. Best results are obtained when using the condition $D5 < 32$ for creating the mask, and by dilating the latter with SE9. The achieved error rate is slightly improved, however the test image in Figure 4.18a is still misidentified.

4.5.4.3 Method 3

The mask obtained with Method 2, shown in Figure 4.18c, does effectively cover the shadowed parts of the face. It does in fact cover most of the face, including shaded areas where the normalized morphology features should in fact be able to compensate the illumination variation. As such, a fair amount of information is lost, potentially limiting the ability of the algorithm to discriminate between individuals. By taking the difference between the mask from Method 2 and the one from Method 1, we can expect to obtain a more selective mask. Since some shadows are covered by both masks, the one from Method 1 is previously eroded using SE9, to avoid exclusion of these shadowed areas due to joint coverage by both masks. As seen in Figure 4.18d, the border of the shadows are still masked out, but the center of some shaded areas such as the cheek under the eye on the left are now preserved. Even though this part of the face appears completely

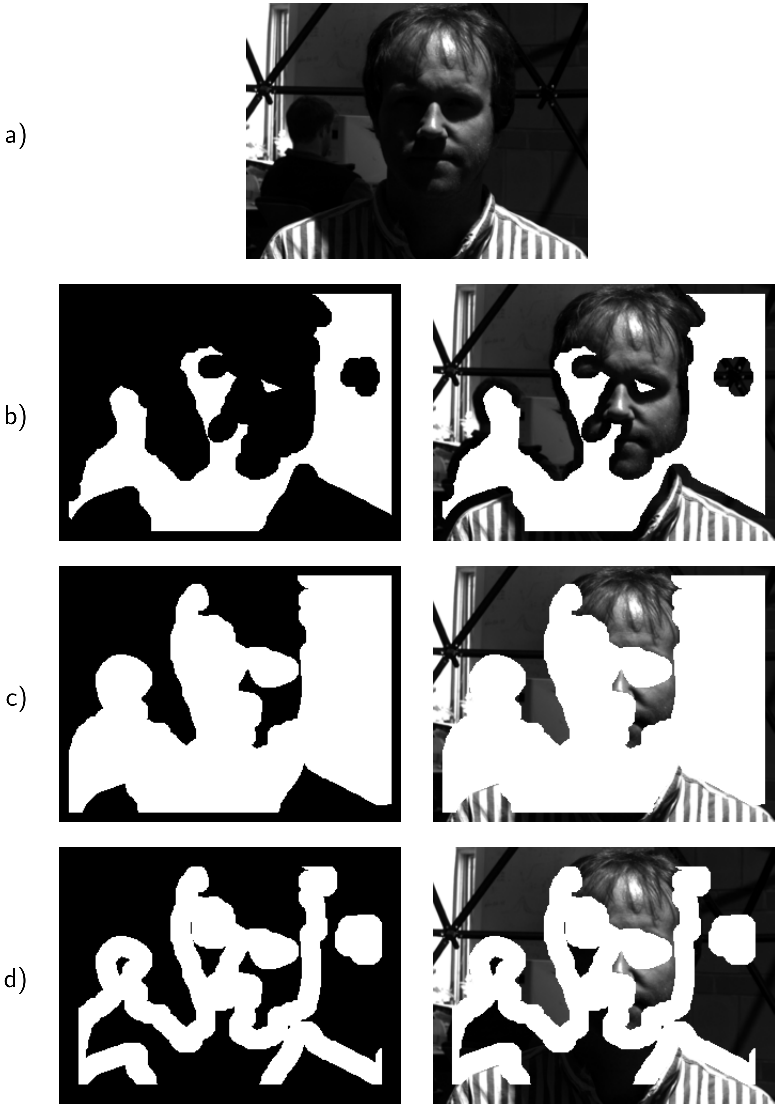


Figure 4.18 : Shadow detection.

a) Test image; b) shadow mask (left) and shadow mask displayed over the test image (right) using Method 1; c) same as b) but using Method 2; d) same as a) but using Method 3.

black in Figure 4.18a, it does in fact contain some recoverable information, as it can be seen in Figure 4.20a, depicting one of the extracted features image.

4.5.4.4 Method 4

Better results can be achieved by following the same approach as in Method 3, but using SE5 instead of SE9 to erode the initial mask. The four steps executed to construct this shadow mask are illustrated in Figure 4.19 and the experimental results are presented in Table 4.7. Quantitatively, the performance is now equal to that achieved in Subsection 4.5.3 when using a priori knowledge to select the best half graph. When considering Subset 4, the error rate now amounts to 27.1%, a 30% percent reduction compared to the 38.6% obtained when retaining all nodes (see Table 4.2). Qualitatively, as shown in Figure 4.20, the selectivity of the shadows is quite good. In the particular case of the test image in Figure 4.18a, the identification is now carried out successfully.

	Error rate (%)		
Shadow detection	Subset 2	Subset 3	Subset 4
Method 4	0.0	0.8	27.1

Table 4.7 : Results on Yale B database for Method 4.

Results obtained on Subset 2 of the Yale B database demonstrates that the proposed node selection approach does not impair the performance of the recognition algorithm for images where only light shadows are found. To further verify this observation and to assess its behavior in the absence of shadows, Method 4 was also incorporated in the XM2VTS tests. The resulting plot, presented in Figure 4.21, clearly shows that the

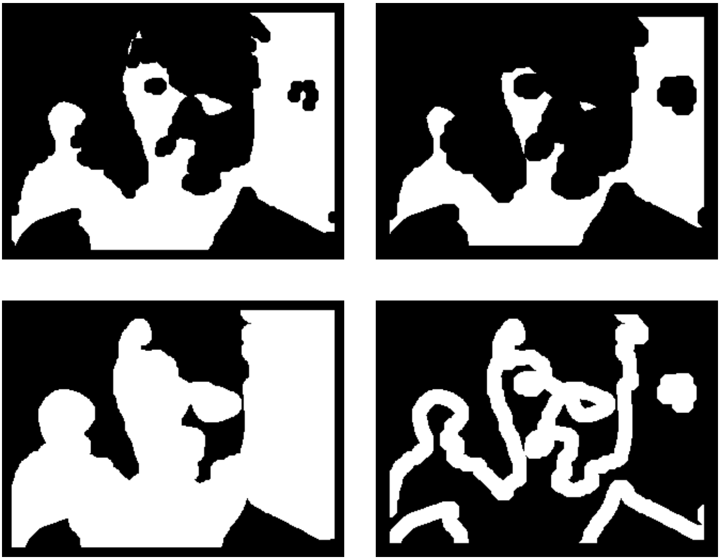


Figure 4.19 : Shadow mask construction for Method 4.

Initial mask (top left), eroded mask (top right), dilated mask (bottom left) and shadow mask (bottom right). The shadow mask corresponds to the difference between the dilated and eroded masks.

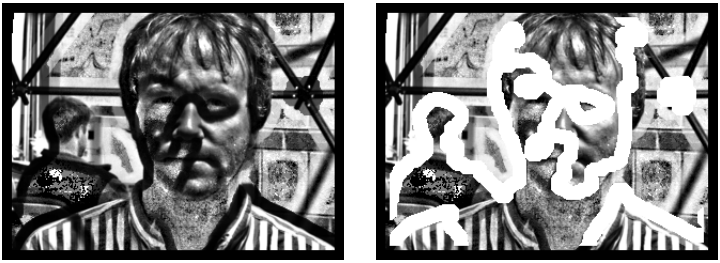


Figure 4.20 : Normalized feature image (left), same with shadow mask from Figure 4.19 (right).

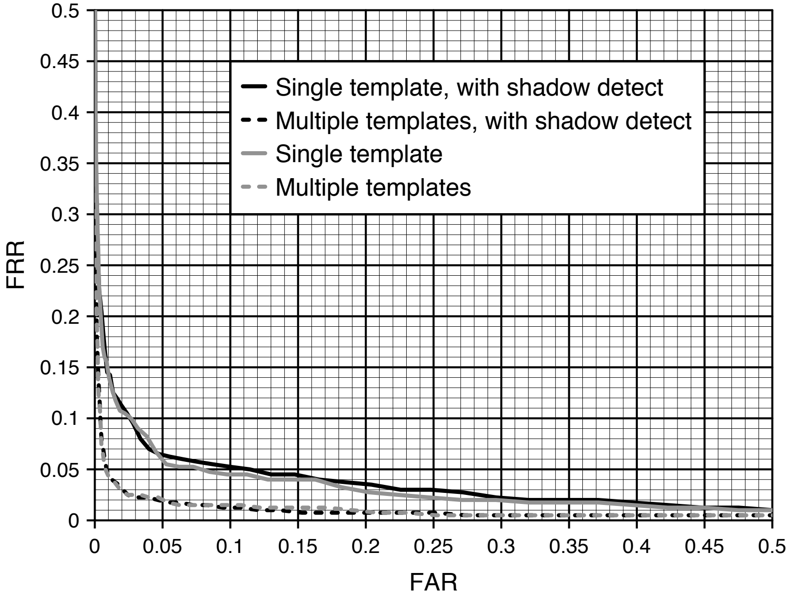


Figure 4.21 : Shadow detection on the standard XM2VTS.

performance is practically unaffected, especially in the multiple templates case[7]. This is not surprising when considering Table 4.8 showing that, in average, only one node per graph is discarded. The maximum number of discarded nodes is 18, and is reached during rigid matching, when part of the graph covers dark hair or somber clothing elements.

To determine the maximum and average number of discarded nodes when the graph is located precisely on the faces, shadow detection was applied during the enrollment of the 200 clients. During this operation, the graph is manually constructed and

[7] See Subsection 3.3.4.

	Discarded nodes		
Database	Min	Max	Average
YaleB			
Subset 1	0	6	0.7
Subset 2	0	12	2.0
Subset 3	0	25	9.9
Subset 4	8	35	16.2
XM2VTS			
Test	0	18	1.01
Enrollment	0	5	0.18

Table 4.8 : Number of discarded nodes for Method 4.
The reported values are the minimum, maximum and average numbers of nodes discarded per graph. Note that Subset 1 is the enrollment subset.

placed on the image according to the furnished eyes coordinates. In this case, the maximum number of discarded nodes amounts to only 5, while in average 0.18 nodes per graph are discarded.

In contrast, the number of discarded nodes is much higher when measured over the Yale B database. Recalling that manual face registration is used here, the reported numbers in Table 4.8 concern only graphs fully located over the faces. Up to one fourth of the nodes are discarded on average when considering Subset 4, and in extreme cases, the identification is based on less than half of the node graphs.

4.5.4.5 Method 5

The width of the black curves, observed at the borders of shadowed areas in Figure 4.20, depends on the size of SE9, the structuring element used to perform the normalization. Consequently, it can be reduced if a smaller SE is employed. The left image in

Figure 4.22 shows the results obtained when such normalization is applied using SE5, whose size is 11 × 11 pixels. The features obtained with larger SEs do not need to be extracted anymore, implying that the dimension of the feature vector can be reduced from 17 to 9 elements.

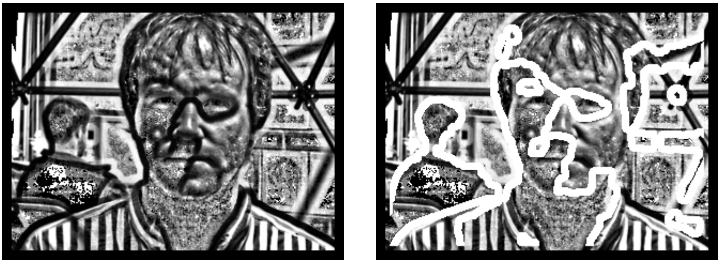


Figure 4.22 : Normalized feature image following Method 5 (left), same with corresponding shadow mask (right).

	Error rate (%)		
Shadow detection	Subset 2	Subset 3	Subset 4
Method 5	0.0	0.0	22.1

Table 4.9 : Results on Yale B database for Method 5.

The shadow mask is constructed as in Method 4, but without the erosion step. Instead, the initial mask is directly subtracted from the dilated mask, the latter being obtained using SE5. Nodes to discard are selected according to the same criteria as in Method 4. The achieved results on the Yale B database are presented in Table 4.9 and show a significative improvement: the error rate is now 0% for Subset 3, and decreases to 22.1% for Subset 4.

However, when applied on the XM2VTS database, Method 5 results in significantly degraded performance, as shown by the gray curve in Figure 4.23. This is not the consequence of discarding

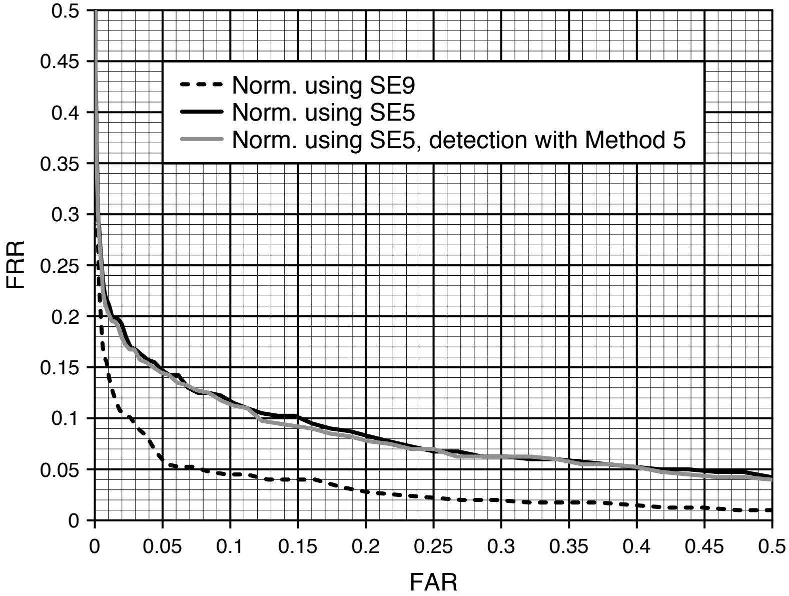


Figure 4.23 : Normalization with different SEs on XM2VTS. The performance degradation is due to the reduced size of the local neighborhood considered when normalizing extracted features, and not to the fact that some nodes are later discarded.

the nodes identified by the shadow detection step. As a matter of fact, there are less nodes discarded now than there were when using Method 4. The problem actually originates from using SE5 instead of SE9 to define the normalization neighborhood, as indicated by the solid black curve in Figure 4.23. The extracted features obtained with this approach are visibly less efficient. In presence of strong shadows, the benefit of being able to precisely identify the affected nodes, as it can be verified in Figure 4.22, nevertheless seems to offset said negative effect.

4.6 Conclusion

In this chapter, the performance of the Morphological Elastic Graph Matching algorithm has been evaluated and compared to other methods on various data sets. Several improvements have been proposed, that increase the robustness in some of the degraded conditions studied. Overall, the algorithm has been shown to perform very well in presence of complex backgrounds, whereas occlusions, as well as changes in scale and orientation, slightly degrade its performance. The results achieved with sideways illumination compare well with those reported by others, as seen in Section 4.4. In particular, one solution we proposed was evaluated using automated face registration and was nonetheless found to perform better than two other methods where this operation was carried out manually. Finally, identification tests were performed using a database acquired under severely degraded illumination conditions, inducing the presence of strong shadows on the faces. A mechanism for selecting and discarding the graph nodes located inside the affected areas was proposed. It was furthermore demonstrated to reduce the number of errors by 30% on the most degraded images, without impacting the performance on uniformly illuminated faces.

All the solutions proposed in this chapter to alleviate the effects of degraded test conditions, intervene during the graph matching stage, and have no impact on the features extraction step. As the following chapter, that introduces a System-on-Chip (SoC) structure for a mobile face verification system, mainly focuses on the VLSI architecture of a coprocessor dedicated to morphology features extraction, the implementation of said solutions in the SoC will not be covered in this report.

Chapter 5

Low-Power Architecture

5.1 Introduction

This chapter describes the hardware architecture elaborated to implement the morphology features extraction algorithm presented in Chapter 4. It only discusses the initial coprocessor, designed before the need for normalizing the extracted features was established. The architecture modified to also handle this task will be covered in Chapter 6. In Section 5.2, an overview of the complete face verification system envisioned in this project is presented, followed in Section 5.3 by a description of how the algorithm was optimized to reduce its computational complexity and the number of required memory transfers. The remainder of the chapter focuses on the proposed architecture, which is initially described in Section 5.4 using a simplified coprocessor with only one processing unit. The actual implementation comprising four processing units working in parallel is then detailed in Section 5.5. Finally, Section 5.6 discusses the architecture and behavior of the complete coprocessor, and also provides performance figures obtained through simulations.

5.2 System overview

In this section the complete face verification system is presented. As depicted in Figure 5.1, it comprises an image sensor, an external memory (RAM), a Master Control Unit (MCU) and two coprocessors, which are dedicated to morphology features extraction and to elastic graph matching, respectively. A shared bus connects these last three elements to the RAM. To prevent conflicting accesses to resources, the MCU ensures that only one component is controlling the bus at any given time. The MCU can be any appropriate low-power micro-controller programmed using a high-level language, such as an Intel 8051 compatible processor [119]. Indeed, its duties mainly consist in handling the system setup and control, these two tasks requiring neither the transfer of large amounts of data, nor the execution of computationally intensive operations[1].

In a practical application, the face verification system will most probably be a subsystem of a particular device. For instance, it could provide biometric facilities to a mobile phone or to a Personal Digital Assistant (PDA). In such a scenario, it would be turned off most of the time, to be powered up by the device's main processor, only when user authentication is required, for instance when access to sensitive information stored locally in the device, or when connection to paid-for on-line services are requested. The MCU would then initialize the face verification subsystem and, when instructed to do so, start up the verification process consisting of the steps described in Subsection 5.2.1. Note that depending on the application scenario, the tasks of the MCU could also very well be handled by the device's main processor. Finally, the User Interface (UI) will most probably be provided by the device's own facilities dedicated to this task.

[1] Unless the MCU has also to handle other demanding tasks unrelated to face authentication.

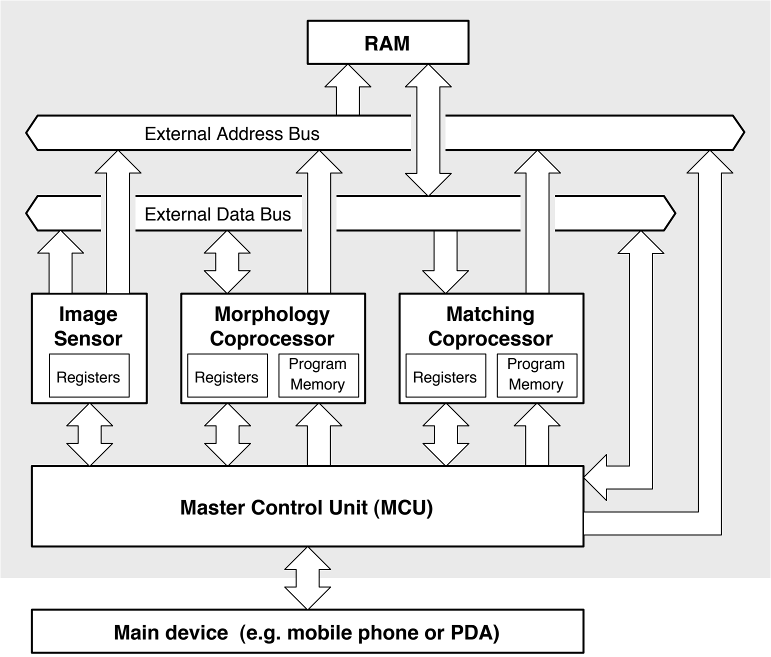


Figure 5.1 : Face verification system.

5.2.1 Face verification process

At start of the process, the MCU performs any necessary initialization operations (e.g. uploading program code to the coprocessors) before delegating the control of the shared bus to the image sensor controller and instructing it to acquire an image to be stored at a specified address in RAM. If implemented, optional image pre-processing steps such as face detection (see Subsection 3.3.1) or image normalization (see Subsection 3.3.2) are then performed. Nowadays, most portable devices are equipped with a color image sensor, yet the considered features extraction algorithm operates on grayscale information only. Even though it is possible to compute the luminance image from the three RGB components, this conversion can be avoided and the green channel can be used directly in place of the true luminance as it is done for instance for the demonstrator that will be discussed in Chapter 7. Experiments carried out on the XM2VTS database resulted in almost identical face verification performance in both cases. Once notified that the image has been successfully transferred, the MCU hands over the control of the bus to the morphology coprocessor that reads the image from the RAM, processes it and stores the extracted features back to the RAM, as detailed later. The MCU then retrieves the features corresponding to the claimed identity from the database containing the data of known users, and uploads them to the matching coprocessor, in order to compare them against the features extracted from the acquired image. The matching coprocessor will read morphology features from the RAM, and then perform the various steps of the elastic graph matching algorithm (EGM) [1]. Highly regular and computationally intensive operations (e.g. features space reduction and distance computations) are performed by the coprocessor whereas less frequent, less regular calculations (e.g. graph rotations) are executed by the MCU. As these interactions between the two components require graphs to be transferred, a fast and low-latency communication channel is desirable. The matching coprocessor then returns the resulting score — the lowest distance obtained between the reference and extracted features — to the MCU. If the score is lower than a predefined threshold (see Subsection 3.3.6), the device is notified that the verification was successful. If the authentication fails and a multiple templates system is being employed, the matching procedure gets repeated against one or several additional reference features associated with the same identity, as explained in Subsection 3.3.4. Finally, if none of the matches succeeds, the identity claim gets rejected.

5.2.2 System partitioning

Whereas the implementation of the image sensor, RAM and MCU using separate components is obvious due to the clearly different nature of their function and tasks, it is necessary to explain more in detail the motivation for designing two separate coprocessors for the morphology features extraction and for the elastic graph matching. Indeed, in case a single coprocessor executes both algorithms, the number of memory accesses may be much lower, because the extracted features could be directly used by the matching algorithm instead of transiting through the RAM. Moreover, the features extraction could be limited to the image locations effectively covered by a node of the graph at least once during the whole graph matching process. This would however require the features extraction to be performed on demand every time the graph is displaced, for every one of its nodes, and this would necessitate loading all the pixels covered by the structuring elements located at each of the 64 nodes of the graph[2]. On the other hand, when performing the features extraction comprehensively for every pixel by scanning the whole image sequentially, so as to consider every possible graph position, most of the loaded pixels can be reused between neighboring positions. As a result, fewer memory transfers are required compared to the previously described “on demand” approach. As most pixels will be covered at least once — and many of them actually several times — in the course of the graph displacement, computing morphology for every location in the image does not actually incur any overhead.

Another important reason in favor of splitting the tasks over two separate coprocessors is the fact that they are very different in nature: morphology features extraction uses mostly comparisons,

[2] This represents for instance 253 pixels per node for a circular 19 × 19 SE set constructed as indicated in Figure 3.14a in Subsection 3.4.2.3.

whereas features reduction and graph matching consists mainly in multiply-and-accumulate (MAC) operations [1]. Therefore, very few hardware resources could be shared between the two coprocessors anyway. As a matter of fact, only the instruction fetch — but not the decoding, since this operation is specific to each coprocessor — and the external memory addressing unit could actually be shared. The savings thus would amount to less than 5% of the total area of the synthesized coprocessor discussed in Chapter 7. Finally, as the EGM algorithm can be applied on different kinds of features [120], using a dedicated coprocessor for features extraction adds flexibility to the system. Indeed, if an alternate features extraction method would be preferable in a particular context, the morphology coprocessor can simply be replaced by a different one, while keeping the EGM coprocessor unmodified.

5.3 Algorithm implementation

This section describes the various modifications applied to the multiscale morphology algorithm performing features extraction that was discussed in Subsection 3.4.2, so as to reduce the computational complexity as well as the number of memory transfers.

5.3.1 Differential multiscale morphology

As explained in Subsection 3.4.2.2, dilating and eroding an image with a given binary structuring element (SE) consists in finding the maximal and minimal pixel value under the given SE, respectively. If $\{SE\}$ denotes the set of the values of the pixels located under the SE, then the dilation D and the erosion E can be expressed as:

$$D = \max(\{SE\}) \tag{5.1}$$
$$E = \min(\{SE\}) \tag{5.2}$$

SE can be divided into two arbitrary parts, SE' and SE'', as long as the latter verify:

$$\{SE'\} \cup \{SE''\} \equiv \{SE\} \tag{5.3}$$

Equation 5.1 and Equation 5.2 can therefore be written as:

$$D = \max(\ \max(\{SE'\}), \max(\{SE''\})\ ) \tag{5.4}$$
$$E = \min(\ \min(\{SE'\}), \min(\{SE''\})\ ) \tag{5.5}$$

In the case of multiscale morphology (see Subsection 3.4.2.3), we denote by SE_N the SE of level N, whereas D_N and E_N indicate the result of the image dilation and erosion operations performed using SE_N, so that:

$$D_N = \max(\{SE_N\}) = \max(\ \max(\{dSE_N\})\ ,\ D_{N-1}\ ) \tag{5.6}$$
$$E_N = \min(\{SE_N\}) = \min(\ \min(\{dSE_N\})\ ,\ E_{N-1}\ ) \tag{5.7}$$

where the differential structuring element dSE_N verifies:

$$\{dSE_N\} \cup \{SE_{N-1}\} \equiv \{SE_N\} \tag{5.8}$$
$$\{dSE_N\} \cap \{SE_{N-1}\} \equiv \varnothing \tag{5.9}$$

Figure 5.2 illustrates the conditions expressed in Equation 5.8 and in Equation 5.9 for $N = 2$. Multiscale morphology can thus be implemented by computing erosions and dilations using differential SEs. D_N and E_N are recursively obtained with increasing level index $N \in [1, ..., 9]$, whereas D_0 and E_0 are both equal to the value of the original pixel lying under the SE center. The total number of comparisons required to compute all dilation and

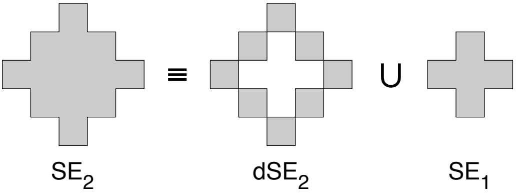


Figure 5.2 : Differential SE.

erosion levels from 0 to N thus amounts to the number of comparisons that would be needed to compute directly D_N and E_N only, augmented by the $2N$ comparisons permitting the recursive computation of the other levels.

However, this recursive approach suffers from one major drawback in the way the pixels have to be addressed, as it will be explained in the following section.

5.3.2 Partial differential morphology

When pixels are accessed in the order required to compute erosions and dilations of increasing levels, the differential erosions and dilations are computed sequentially. However, pixels must be read following a predefined sequence that requires a complex addressing scheme.

Let us consider the 5×5 pixels image and the three structuring elements shown in Figure 5.3. Here, $P_{L,i}$ represents the ith pixel of level L, whereas $P_{X,i}$ denotes pixels that are not covered by the SE set. To compute the three erosion and dilation levels when the center of the SE set is positioned over the pixel in the middle of the image, the following operations are to be performed. Firstly,

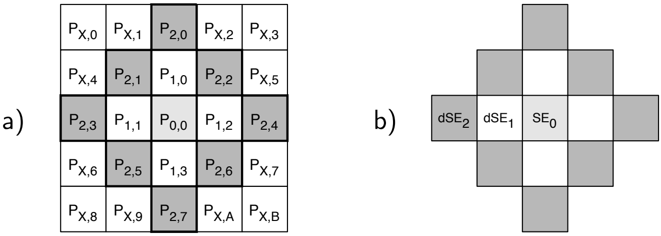


Figure 5.3 : a) 5 × 5 pixels image, b) differential SEs.

D_0 and E_0 are trivially obtained:

$$\mathbf{D_0} \leftarrow \mathbf{P_{0,0}} \quad ; \quad \mathbf{E_0} \leftarrow \mathbf{P_{0,0}}$$

Two registers, pD and pE, are used to hold partial results for the dilation and erosion processes, respectively. dD_1 and dE_1 can then be calculated as follows:

$$\begin{aligned} pD &\leftarrow 0 &;\quad pE &\leftarrow 255 \\ pD &\leftarrow \max(pD, P_{1,0}) &;\quad pE &\leftarrow \min(pE, P_{1,0}) \\ pD &\leftarrow \max(pD, P_{1,1}) &;\quad pE &\leftarrow \min(pE, P_{1,1}) \\ pD &\leftarrow \max(pD, P_{1,2}) &;\quad pE &\leftarrow \min(pE, P_{1,2}) \\ pD &\leftarrow \max(pD, P_{1,3}) &;\quad pE &\leftarrow \min(pE, P_{1,3}) \end{aligned}$$

At this point, $dD1 = pD$ and $dE1 = pE$ so that:

$$\mathbf{D_1} \leftarrow \mathbf{max}(\mathbf{pD}, \mathbf{D_0}) \quad ; \quad \mathbf{E_1} \leftarrow \mathbf{min}(\mathbf{pE}, \mathbf{E_0})$$

The process then continues to compute dD_2 and dE_2:

$$\begin{aligned} pD &\leftarrow 0 &;\quad pE &\leftarrow 255 \\ pD &\leftarrow \max(pD, P_{2,1}) &;\quad pE &\leftarrow \min(pE, P_{2,3}) \\ &\ldots &;\quad &\ldots \\ pD &\leftarrow \max(pD, P_{2,7}) &;\quad pE &\leftarrow \min(pE, P_{2,7}) \end{aligned}$$

at which point we have $dD_2 = pD$ and $dE_2 = pE$, which gives:

$$\mathbf{D_2} \leftarrow \mathbf{max}\,(\,\mathbf{pD}, \mathbf{D_1}\,) \quad ; \quad \mathbf{E_2} \leftarrow \mathbf{min}\,(\,\mathbf{pE}, \mathbf{E_1}\,)$$

To compute dD_1, dE_1, dD_2 and dE_2, the following pixels were successively read: $P_{1,0}$, $P_{1,1}$, $P_{1,2}$, $P_{1,3}$, $P_{2,0}$, $P_{2,1}$, $\ldots$, $P_{2,7}$. Accessing pixels in this order however requires random[3] addressing. To move from one pixel to the next, the modification to apply to the memory address differs every time. Moreover, the addressing sequence depends on the chosen SE set, and a new sequence needs to be used if the SE set change.

5.3.3 Sequential morphology and SE table

It is however possible to compute erosions and dilations while reading pixels sequentially, as demonstrated below:

$$\begin{aligned} pD1 \leftarrow 0 \quad &; \quad pE1 \leftarrow 255 \\ pD2 \leftarrow 0 \quad &; \quad pE2 \leftarrow 255 \end{aligned}$$

$$\begin{aligned} pD2 \leftarrow \max(\,pD2, P_{2,0}\,) \quad &; \quad pE2 \leftarrow \min(\,pE2, P_{2,0}\,) \\ pD2 \leftarrow \max(\,pD2, P_{2,1}\,) \quad &; \quad pE2 \leftarrow \min(\,pE2, P_{2,1}\,) \end{aligned}$$

$$pD1 \leftarrow \max(\,pD1, P_{1,0}\,) \quad ; \quad pE1 \leftarrow \min(\,pE1, P_{1,0}\,)$$

$$\begin{aligned} pD2 \leftarrow \max(\,pD2, P_{2,2}\,) \quad &; \quad pE2 \leftarrow \min(\,pE2, P_{2,2}\,) \\ pD2 \leftarrow \max(\,pD2, P_{2,3}\,) \quad &; \quad pE2 \leftarrow \min(\,pE2, P_{2,3}\,) \end{aligned}$$

$$pD1 \leftarrow \max(\,pD1, P_{1,1}\,) \quad ; \quad pE1 \leftarrow \min(\,pE1, P_{1,1}\,)$$

$$Dout \leftarrow P_{0,0} \quad ; \quad Eout \leftarrow P_{0,0}$$

At this point, $\mathbf{D_0} = \mathbf{Dout}$ and $\mathbf{E_0} = \mathbf{Eout}$.

$$pD1 \leftarrow \max(\,pD1, P_{1,2}\,) \quad ; \quad pE1 \leftarrow \min(\,pE1, P_{1,2}\,)$$

[3] i.e. non sequential.

$$pD2 \leftarrow \max(pD2, P_{2,4}) \quad ; \quad pE2 \leftarrow \min(pE2, P_{2,4})$$
$$pD2 \leftarrow \max(pD2, P_{2,5}) \quad ; \quad pE2 \leftarrow \min(pE2, P_{2,5})$$

$$pD1 \leftarrow \max(pD1, P_{1,3}) \quad ; \quad pE1 \leftarrow \min(pE1, P_{1,3})$$

$$Dout \leftarrow \max(pD1, Dout) \quad ; \quad Eout \leftarrow \min(pE1, Eout)$$

At this point, $\mathbf{D_1} = \mathbf{Dout}$ and $\mathbf{E_1} = \mathbf{Eout}$.

$$pD2 \leftarrow \max(pD2, P_{2,6}) \quad ; \quad pE2 \leftarrow \min(pE2, P_{2,6})$$
$$pD2 \leftarrow \max(pD2, P_{2,7}) \quad ; \quad pE2 \leftarrow \min(pE2, P_{2,7})$$

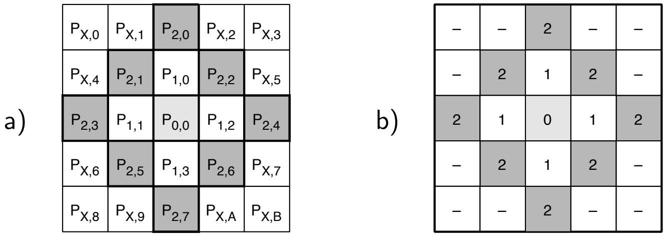
$$Dout \leftarrow \max(pD2, Dout) \quad ; \quad Eout \leftarrow \min(pE2, Eout)$$

At this point, $\mathbf{D_2} = \mathbf{Dout}$ and $\mathbf{E_2} = \mathbf{Eout}$.

Instead of the two registers holding the partial results pD and pE, the new scheme uses two sets of registers, $pD1$ and $pD2$, as well as $pE1$ and $pD2$, to store the partial results for each dilation and erosion level. It is also required that, for every pixel being processed, the corresponding dSE shall be known so that the correct registers pDL and pEL, with $L \in [1, 2]$ can be selected. It is easy to see that the value L corresponding to each pixel is directly related to the dSE covering the said pixel, with $L = 0$ being used to indicate the position of the central pixel. Let us introduce a special value for L, $L = IGNORE$, denoted by a dash in the formulas below, indicating that the comparison operation is disabled, so that the level index of the dSE set can be stored in a table, as depicted in Figure 5.4b. The dash localizes the pixels $P_{X,i}$ that are not lying under any dSE and that must therefore be discarded.

Let us now define the following abbreviated notations, the letters R, M and C standing for Reset, Morphology and Combine, respectively:

$$R \quad \equiv \quad pDL \leftarrow 0 \; ; \; pEL \leftarrow 255 \qquad \forall L \in [1, 2]$$



a)

$P_{X,0}$	$P_{X,1}$	$P_{2,0}$	$P_{X,2}$	$P_{X,3}$
$P_{X,4}$	$P_{2,1}$	$P_{1,0}$	$P_{2,2}$	$P_{X,5}$
$P_{2,3}$	$P_{1,1}$	$P_{0,0}$	$P_{1,2}$	$P_{2,4}$
$P_{X,6}$	$P_{2,5}$	$P_{1,3}$	$P_{2,6}$	$P_{X,7}$
$P_{X,8}$	$P_{X,9}$	$P_{2,7}$	$P_{X,A}$	$P_{X,B}$

b)

–	–	2	–	–
–	2	1	2	–
2	1	0	1	2
–	2	1	2	–
–	–	2	–	–

Figure 5.4 : a) Image, b) SE table describing the SE set.

$$\begin{aligned} M(L,P) &\equiv pDL \leftarrow \max(pDL, P)\,;\; pEL \leftarrow \min(pEL, P) \\ C(L) &\equiv Dout \leftarrow \max(pDL, Dout)\,;\; Eout \leftarrow \min(pEL, Eout) \end{aligned}$$

with the following special cases:

$$\begin{aligned} M(0,P) &\equiv Dout \leftarrow P\ ;\ Eout \leftarrow P \\ M(-,P) &\equiv \textit{ignore pixel} \end{aligned}$$

The sequence of operations given above can thus be rewritten as:

reset partial registers

R

process first image row

$M(-, P_{X,0})$

$M(-, P_{X,1})$

$M(2, P_{2,0})$

$M(-, P_{X,2})$

$M(-, P_{X,3})$

process second image row

$M(-, P_{X,4})$

$M(2, P_{2,1})$

$M(1, P_{1,0})$

$M(2, P_{2,2})$

$M(-, P_{X,5})$

process third image row
$M(2, P_{2,3})$
$M(1, P_{1,1})$
$M(0, P_{0,0})$

At this point, $\mathbf{Dout} = \mathbf{D_0}$ and $\mathbf{Eout} = \mathbf{E_0}$

$M(1, P_{1,2})$
$M(2, P_{2,4})$

process fourth image row
$M(-, P_{X,6})$
$M(2, P_{2,5})$
$M(1, P_{1,3})$

$C(1)$

At this point, $\mathbf{Dout} = \mathbf{D_1}$ and $\mathbf{Eout} = \mathbf{E_1}$

$M(2, P_{2,6})$
$M(-, P_{X,7})$

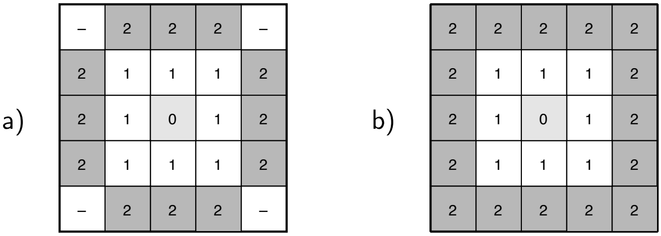
process fifth image row
$M(-, P_{X,8})$
$M(-, P_{X,9})$
$M(2, P_{2,7})$

$C(2)$

At this point, $\mathbf{Dout} = \mathbf{D_2}$ and $\mathbf{Eout} = \mathbf{E_2}$

$M(-, P_{X,A})$
$M(-, P_{X,B})$

It can be seen that the parameters of the M operations — the level L and the pixel P — can now be read sequentially from the image and from the SE table, respectively. The SE set used to perform morphology features extraction can moreover be changed by simply modifying the SE table accordingly, as illustrated in Figure 5.5.



a)

–	2	2	2	–
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
–	2	2	2	–

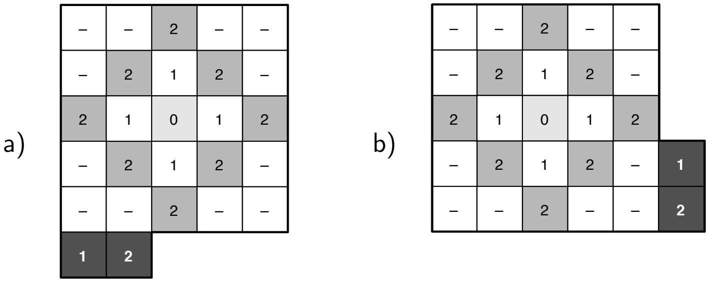
b)

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Figure 5.5 : Alternative SE tables.

As the C operations also require a parameter L, the size of the SE table can be augmented so as to incorporate theses parameters. When executing a C operation, the L value would then be read from the SE table, but for this operation no pixel would be read from the image memory. These operations appear once all pixels belonging to a given level have been processed, which corresponds to the last occurrence of this level in the SE table. As seen when comparing the three SE tables in Figure 5.4b, Figure 5.5a and Figure 5.5b, the position where theses operations occur depends on the SE set in use. It is however possible to arbitrarily delay these operations, the only requirement being that the order in which they are executed gets preserved. For instance, in the sequence above, $pD1$ and $pE1$ are not modified anymore once pixel $P_{1,3}$ has been processed. Consequently, all C operations could be moved to the end of the process, while the values of their L parameters would be appended at the end of the SE table.

This option is depicted in Figure 5.6a, where the cells holding



a)

–	–	2	–	–
–	2	1	2	–
2	1	0	1	2
–	2	1	2	–
–	–	2	–	–
1	2			

b)

–	–	2	–	–	
–	2	1	2	–	
2	1	0	1	2	
–	2	1	2	–	1
–	–	2	–	–	2

Figure 5.6 : Location of the C operations in the SE table. SE tables with morphology results being generated a) in the end, b) regularly over the whole process, in case of row-wise ordered execution.

parameters for the C operations are denoted by white text over dark background. A more interesting solution however consists in distributing these operations over the whole process, as illustrated in Figure 5.6b. Indeed, in that case the results can be stored back to the external memory while features extraction continues, whereas with the first solution, additional cycles would need to be inserted at the end of the process. Note that the SE table is read sequentially, so that both tables shown in Figure 5.6 could be stored in 27 consecutive words in a memory. Their representation as bi-dimensional arrays is simply designed to clarify the layout of the SE sets inside the memory.

5.3.4 Local image memory

Computing all morphological levels at one position in the image requires accessing a lot of pixels. For instance, using the SE set illustrated in Figure 3.14a in Subsection 3.4.2.3, a total of 253 pixels are needed. It is obvious that most of these pixels will also be employed when performing the same operation for the neighboring positions, which means that many external memory accesses can be avoided if a local memory is used to store these pixels.

To extract morphological features using 9 levels of dilation and erosion, a window of 19 × 19 pixels is required. When the process starts, the local memory is loaded with the 19 × 19 monochrome pixels window located on the top left corner of the original image, as shown in Figure 5.7. The window then moves by increments of 1 pixel to the left, so that a new column of 19 pixels has to be loaded before starting working on the new position. These new pixels are written over the 19 leftmost pixels which will not be needed anymore while processing the current row. It should be noted however that some of these pixels will have to be reloaded later on, for instance when the second row of the 19 × 19 pixels window has been completely scanned, so that the window reaches again the leftmost column. Indeed, storing every pixel locally from the moment it is read for the first time and up to the point where it is not needed anymore would not be practical, as this would require a much larger local memory.

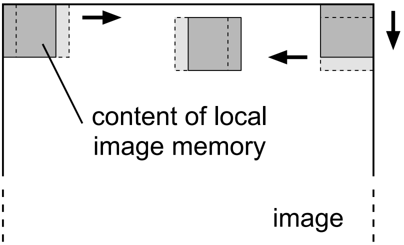


Figure 5.7 : Local image memory updating process.

Each time the rightmost image column is reached, the window is moved down by one pixel, and the window displacement continues on the new line, from right to left. To simplify the addressing scheme used to dispatch pixels to the morphology processing units, new data are written to the local memory so that the whole window can be read sequentially by simply incrementing the address pointer, as exposed in Subsection 5.5.6.

5.4 Single-unit coprocessor architecture

A basic morphology coprocessor — whose architecture derives from the algorithm implementation discussed in Section 5.3 — will be presented in Subsection 5.4.2. It comprises one single processing unit, the Elementary Morphology Unit, which is described in Subsection 5.4.1.

5.4.1 Elementary morphology unit

The Elementary Morphology Unit (EMU) is the core element of the coprocessor, and is depicted in Figure 5.8. Its main purpose is to implement the operations required to execute the steps enumerated in Subsection 5.3.3. The EMU has one data input, the pixel P to process, and two control inputs: the value L furnished by the SE table and a command signal C provided by the instruction decoder. The purpose of this command signal is to distinguish between the M(L, P) and the C(L) operators defined in Subsection 5.3.3. To perform the dilation and erosion operations in parallel, the unit contains two 8-bit comparators, each one being connected to a register bank. The first comparator, implementing the MAX operation, is used to compute dilations and is connected to the pD register bank, while the second one implements the MIN operation necessary to compute erosions and is connected to the pE register bank. As the SE set used comprises 9 structuring elements, each bank contains nine registers, holding the partial dilation or erosion results, respectively. Two output registers, each one connected to one of the two comparators, can be read from outside the unit. The width of all registers is 8-bit.

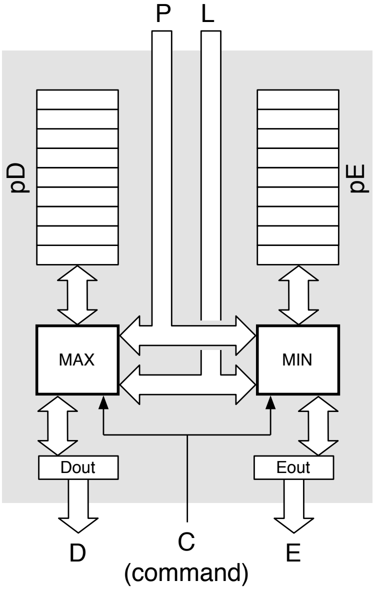


Figure 5.8 : Elementary Morphology Unit (EMU).

To be able to perform the algorithm discussed in Subsection 5.3.3, the EMU must implement the following operations, derived from the operators introduced in said section:

RM, for **R**eset **M**orphology : Before features extraction can begin, the two register banks must be reset to neutral values, respectively 0 for dilation and 255 for erosion.

MM, for **M**orphology operator **M**(L, P) : According to the value of the input L, the EMU must either ignore the incoming pixel P (when $L = -$), or store it in both output registers (when $L = 0$), or perform one step of partial differential erosion and dilation (when $L > 0$) by comparing P with pDL and pEL. The result of each comparison is then written back to pDL and pEL, respectively.

MC, for **M**orphology operator **C**(L) : The morphology unit needs to combine the dilation and erosion results for the differential level L, stored in pDL and pEL, by comparing them to Dout and Eout, respectively. The result of each comparison gets written to Dout and Eout, respectively.

NOP : No operation instruction. Sent to the EMU when it does not need to perform any task. This is the case for instance when updating the content of the local image memory.

Note that the control unit actually schedules operations MM and MC by sending the same instruction code to the EMU, the latter determining which operator to consider based on the separate command signal C, also furnished by the control unit. Defining two separate instruction codes therefore seems to constitute a more logical option, which is indeed true for the single-unit architecture discussed here. However, in the case of the four-way parallel coprocessor, to be introduced in Subsection 5.5.2, the signal C will need to be delayed for some EMUs, and it will consequently need to be handled separately from the opcode.

5.4.2 Coprocessor architecture

The basic structure of a morphology coprocessor containing only one processing unit is depicted in Figure 5.9. It is connected to an external bus that provides access to the RAM that contains the image and where the extracted morphology features will be stored. It incorporates two local memories and one EMU. A multiplexer can be used to determine from which one of the dilation and erosion output registers the value to write to the external memory must be taken. The control signal EnD — for *erosion not dilation* — is provided by the external addressing unit. Data transfers from the local memories are made to a register, so that the read and write operations can span over the entire clock cycle for their completion. This allows the use of slower, less power consuming components. Pixels read from the external memory are nevertheless sent to the local image memory directly, without transiting through a register, since it is assumed that the local memories are synchronous[4], meaning that they behave like a register, namely that they sample the input value on the active clock edge. Additionally, an instruction fetch and decoding unit, constituting the control unit, that is not depicted in Figure 5.9, is responsible for reading instructions from a dedicated program memory —that is not depicted either — and for generating the control signals steering the various components, including the signal C furnished to the EMU, as shown in Figure 5.8.

The features extraction as performed by the morphology coprocessor can de decomposed into two tasks executed alternately. The first one consists in updating the local image memory so that it contains all the pixels needed to extract the morphology features at the position currently considered, as described in Subsection 5.3.4. At this time, neither the EMU nor the SE Memory is used. When the second task is executed, where the morphology features are computed and written to the external memory as they become available, all three components of the coprocessor, namely the EMU, the SE Memory and the Local Image Memory are active.

[4] This is indeed the case both for the FPGA and for the ASIC implementations discussed in Chapter 7.

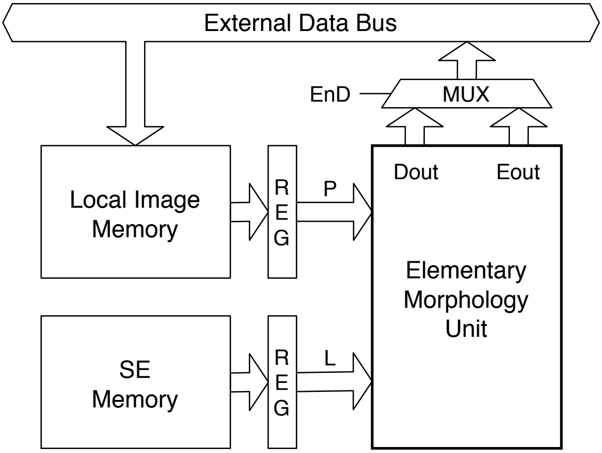


Figure 5.9 : Structure of a morphology coprocessor with a single processing unit.

5.5 Four-way parallel coprocessor

To diminish the number of cycles required to process one image, it is possible to parallelize the task by using multiple independent elementary morphology units, each one performing features extraction at a different location in the image. This can lead either to faster processing, or to reduced power consumption, the latter case being achieved as follows. The operating frequency can be decreased since more processing is now performed during a given time unit, which causes the slack margin[5] to increase. Consequently, propagation delays can augment without impairing the functionality of the circuit, which enables the supply voltage to be lowered. Since the power consumption is proportional to the square of the supply voltage, the former is also reduced.

[5] The slack margin is the difference between the duration of the clock cycle and the delay of the critical path.

To maximize reuse of pixels, the considered window locations must obviously be contiguous. The architecture presented here makes use of four identical elementary morphology units, or EMUs, extracting features at each of four neighboring horizontal window locations. This number was selected to ensure that a complete face verification could be carried out in less than one second, assuming an operating frequency of 10 MHz. As reported in [121], the elastic graph matching needs 680 ms to execute, leaving at most 320 ms for features extraction. It was estimated that a four-unit morphology coprocessor would require approximately 200 ms to process one image and would thus be the most adequate solution. Simulations results presented in Subsection 5.6.3 later confirmed this estimation.

5.5.1 Single instruction, multiple data architecture

The straightforward way of implementing parallelism when the exact same operations need to be executed on different data would be to use a vector or SIMD (single instruction, multiple data) approach [122]. In our case, that translates to sending the same control information to all four EMUs, while feeding each one with a different pixel. This can be achieved by inserting a pipeline at the output of the local image memory, and by connecting each EMU to a different pipeline stage, as illustrated in Figure 5.10.

However, this approach is not optimal. Indeed, there are pixels that are to be taken into account by some of the EMUs, but must be ignored by the others. This is the case for instance for pixels on the leftmost column of the local window, that are to be processed only by the EMU extracting features on the leftmost location, or for most pixels under the top row of a circular SE set, as seen in Figure 5.33. In a SIMD scheme, all units execute

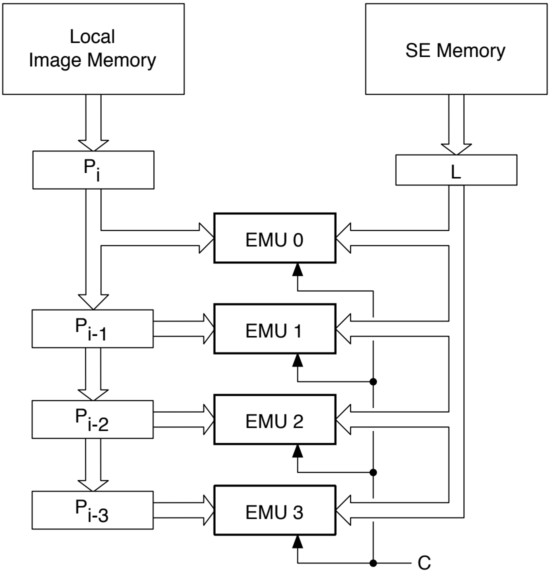


Figure 5.10 : Data pipeline in the SIMD approach.

the same operation and as such, a ”neutral” operand should be fed to EMUs that are to ignore the current pixel. Since pixels must have a value of 0 to be neutral to the dilation, and a value of 255 to be neutral to the erosion, there exists no pixel value that is neutral to both operations.

5.5.2 Delayed instructions, single data architecture

A more advantageous approach consists in delaying the control signals instead of the data, in a MISD (multiple instructions, single data) fashion [122]. The same pixel is now fed to all four EMUs, accompanied by different control signals. Figure 5.11 illustrates the time organization of four EMUs extracting features at locations P3, P4, P5 and P6 in such a scheme. The values read from the SE table — and displayed above each EMU — indicate the control value L corresponding to each pixel. It is easy to see that values of L for EMU 1, 2 and 3 are simply value of L for EMU 0 but delayed by 1, 2 and 3 cycles, respectively. For instance, L4 is used by EMU 0 when processing pixel P4, but EMU 1 only needs it during the following cycle, when it processes pixel P5.

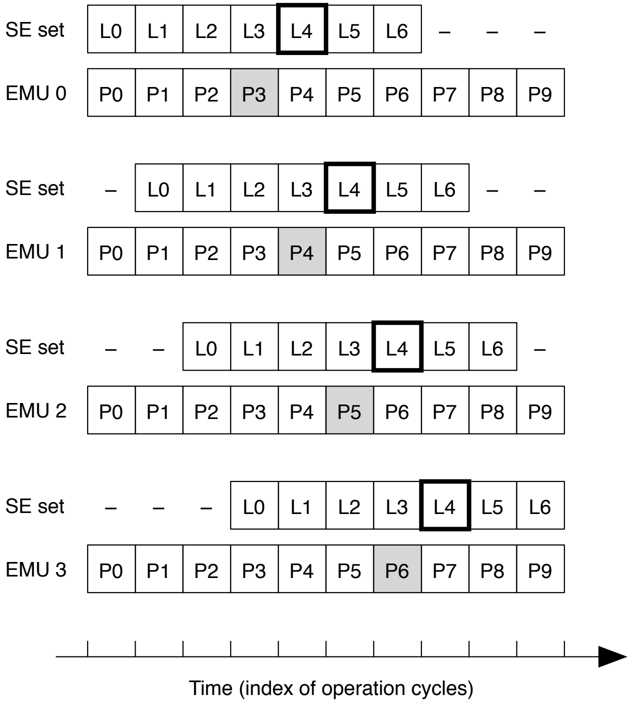

SE set L0 L1 L2 L3 L4 L5 L6 – – –
EMU 0 P0 P1 P2 P3 P4 P5 P6 P7 P8 P9
SE set – L0 L1 L2 L3 L4 L5 L6 – –
EMU 1 P0 P1 P2 P3 P4 P5 P6 P7 P8 P9
SE set – – L0 L1 L2 L3 L4 L5 L6 –
EMU 2 P0 P1 P2 P3 P4 P5 P6 P7 P8 P9
SE set – – – L0 L1 L2 L3 L4 L5 L6
EMU 3 P0 P1 P2 P3 P4 P5 P6 P7 P8 P9
Time (index of operation cycles)

Figure 5.11 : SE pipeline following the "delayed instructions" approach.

When the current pixel is to be ignored by one EMU, the value L = IGNORE — denoted by a dash — is sent. As seen in Figure 5.11, the four EMUs must ignore the pixel values preceding L0 and those coming after L6. To this effect, the pipeline is initialized with IGNORE values, and a multiplexer enables the insertion of IGNORE values once L6 has been read from the SE memory, as detailed in Subsection 5.5.5.

Finally, considering the size of the registers, it can be seen that this approach only requires storing 5 bits at every stage of the pipeline: the 4-bit level L provided by the SE Block (see Subsection 5.5.5) and the 1-bit control flag C distinguishing the combine operation MC from the regular partial morphology operation MM[6]. This compares favorably to the 8 bits that would be required in the SIMD case where the pipeline has to store pixels.

The overall architecture of the four-way parallel morphology coprocessor implemented using this “delayed instructions” approach is shown in Figure 5.12. The single EMU has been replaced by a new component, the Morphology Block, discussed in Subsection 5.5.3.

5.5.3 Morphology block

The Morphology Block (MB), depicted in Figure 5.13, replaces the single EMU of the morphology coprocessor shown in Figure 5.9. It encompasses four independent EMUs, whose outputs are connected to multiplexers controlled by two signals — POS and EnD — furnished by the Incrementer-Decrementer Block (ID Block, see Subsection 5.5.7), so that the MB has only one output M. It also contains three sets of two registers, responsible for delaying the control signals L and C from one EMU to the next one, as exposed in Subsection 5.5.2. The inputs P of all

[6] The other control signals — conveying the instruction code that triggers a RM, a MM/MC, or a NOP (see Subsection 5.4.1) — are not delayed.

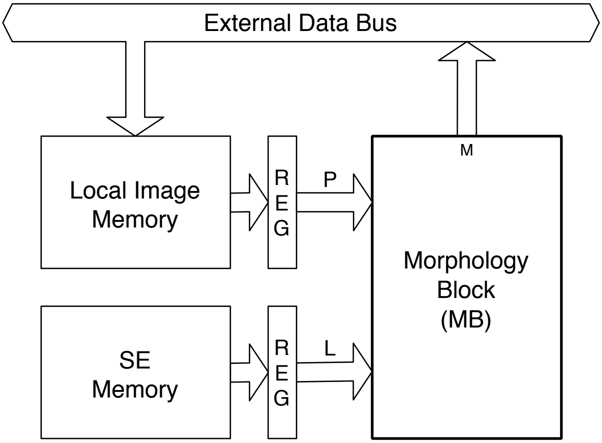


Figure 5.12 : Four-way parallel morphology coprocessor.

four EMUs are conversely directly connected to one single bus, the P input of the MB.

5.5.4 Local memory block

Loaded pixels are stored in the local image memory (see Subsection 5.3.4 and Figure 5.12) so that they can be read multiple times without requiring additional accesses to the larger external memory. As the local window moves up or down, each pixel will be read 19 times. The size of the local memory was increased to 19×22 monochrome pixels, globally requiring 418 bytes, so as to cover the aggregated area of the four memory windows needed by the EMUs. Originally, the way the window was displaced over the image was directly derived from Figure 5.7, except that four columns of 19 pixels were read at a time instead of only one, when moving toward the left or the right, because of the existence of the four EMUs. However, this approach was later modified to

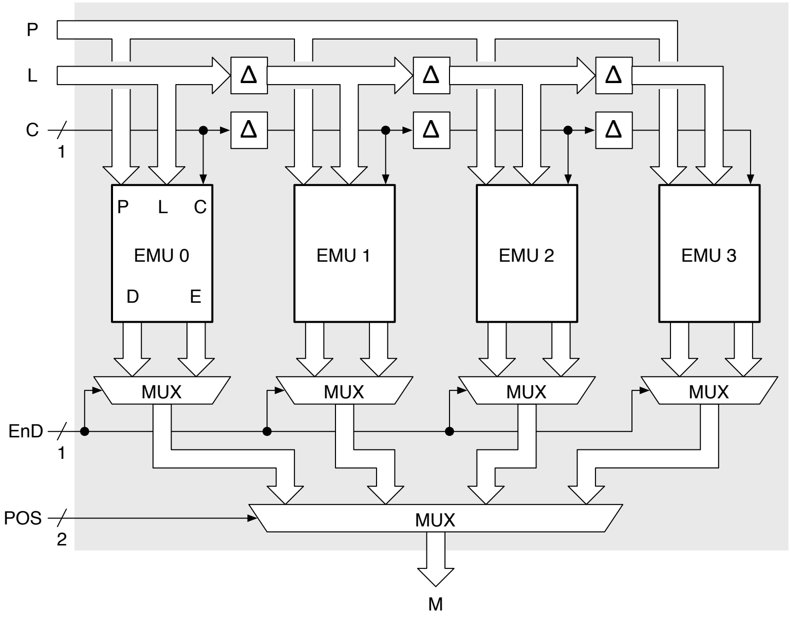


Figure 5.13 : Morphology Block (MB) containing four instances of the EMU shown in Figure 5.8.

move over the image column-wise instead of row-wise, as shown in Figure 5.14.

While the number of steps remains the same, the former approach depicted in Figure 5.7 would have involved loading 4×19 pixels most of the time, whereas with the new approach it is only the case when the local memory window reaches the top or bottom rows of the image. Otherwise, it suffices to load a single 22-pixel row. The total number of external memory read operations can thus be significantly lowered. In the case of an image of 146×146 pixels, the number of memory loads reported in [123] is reduced by 70%, as can be seen in Table 5.2.

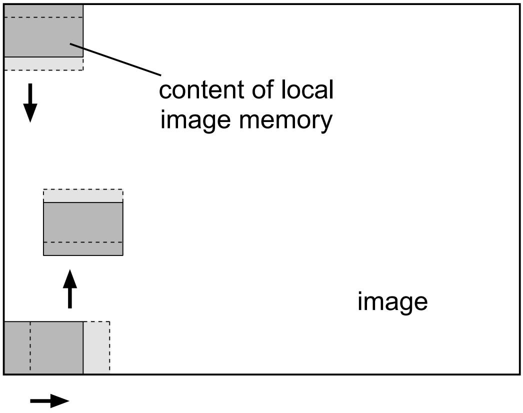


Figure 5.14 : New local memory update scheme.

Figure 5.15 shows the Local Memory Block (LMEM Block), which comprises the local image memory itself, LMEM, and an output register, LREG. Every value read from LMEM is stored in this register before being sent to the morphology block, to relax timing constraints on the memory. The data input D of LMEM is connected to the external data bus, cf Figure 5.12, and receives the pixels read from the external memory. The address provided to the A input comes from one out of two registers, either LMW or LMR, located inside the MAS Block that will be described in Subsection 5.5.6.

5.5.5 Structuring elements block

The Structuring Element Block (SE Block) is depicted in Figure 5.16 and comprises a memory (SEMEM) as well as the associated addressing logic. SEMEM holds the description of the SE set, corresponding to the content of the SE table introduced

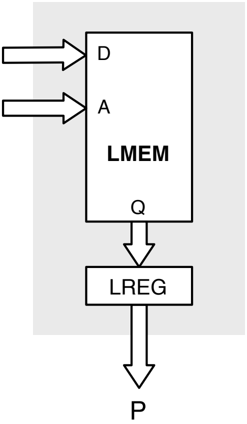


Figure 5.15 : Local Memory Block (LMEM Block).

in Subsection 5.3.3. It is used to communicate to the EMUs which partial result registers need to be considered for either the partial morphology M(L, P) or the combine C(L) operation, by supplying the value of the L parameter, which ranges from 1 to 9. Additionally, L can take the value 15, corresponding to the dash employed previously to denote the IGNORE symbol, to indicate that the current pixel shall be discarded, which happens when the latter falls outside of the surface covered by the structuring element set associated to the EMU. Finally, L will take the value 0 to notify that the current pixel corresponds to the center of the SE set, and that it must be directly copied to both output registers Dout and Eout (see Figure 5.8).

Since L can take a total of 11 different values, namely 0, 1 to 9 and 15, SEMEM needs to store words that are 4-bit wide. Moreover, the memory must be able to hold the 19×19 SE set, as well as the 9 parameters used to combine the differential morphology results (see Subsection 5.3.3), hence requiring a total of 370 words, which corresponds to a total size of 1'480 bits. The actual content of one SE table can be seen in Figure 5.33. As discussed in Subsection 5.3.3, the SE table is read sequentially, so that the SEMEM is addressed using a simple counter, constituted of one register and one incrementer, labeled SEIDX and SEINC, respectively. Since the memory contains 370 words, the address needs to be 9-bit wide. An output register, SEREG, is inserted on the data path between SEMEM and the EMUs, again to relax the timing constraints on the memory.

As the three leftmost columns of the memory window stored in LMEM are not used by all four EMUs, the L pipeline in the Morphology Block is initialized with IGNORE values when the processing starts for a new set of positions. Similarly, when feeding each of the three leftmost pixels, a multiplexer (see Figure 5.16) enables the insertion of an IGNORE value, instructing EMU 0 to discard the current pixel. No value is read from SEMEM in this case, and its address register SEIDX is not incremented. Like all the L parameters fed to the Morphology Block, the inserted IGNORE values will then propagate to the remaining EMUs, as explained in Subsection 5.5.3, causing each EMU to ignore the pixels that are not relevant to the image position it processes.

Note that in Figure 5.16, no data input is shown for SEMEM. This is due to the fact that the coprocessor never needs to directly write to this memory, as its initialization is handled by the MCU.

5.5.6 Addressing logic

With the exception of the registers LREG and SEREG inserted on the data path to relax timing constraints of memories, all registers lying outsides of the MB are part of the addressing logic. In addition to SEINC (see Subsection 5.5.3), the morphology coprocessor contains two adders responsible for updating these

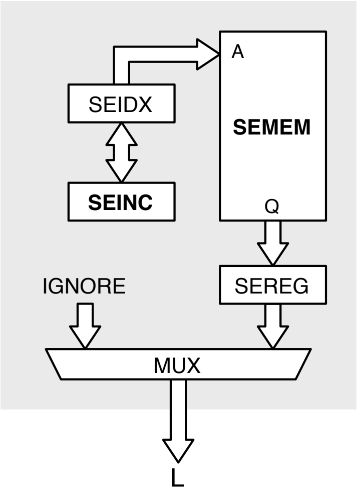


Figure 5.16 : Structuring Element Block (SE Block).

registers, listed in Table 5.1.

5.5.6.1 Modulo adder-subtracter

The Modulo Adder-Subtracter (MAS) is employed to update the read and write address pointers of the local memory, as well as the row and column offsets of the local window accessed in external memory (see Subsection 5.5.4). It can be demonstrated that it is sufficient for the MAS to be able to increment and decrement the registers by three different constants, namely 1, 4 and 18, to perform this task. Hence, the MAS is restricted to perform these six operations, and takes a single operand, indicating the index of the register to modify. The opcode specifies which operation to perform (see Subsection 5.5.7), so that the values of the constants enumerated above do not need to be encoded into the instruction word.

Name	Width [bits]	Adder	Description
SEIDX	9	SEINC	address used when reading SEMEM
LMR	9	MAS	address used when reading LMEM
LMW	9	MAS	address used when writing LMEM
LR	8	MAS	load row index in external memory
LC	8	MAS	load col. index in external memory
TR	8	ID	temporary copy of LR
TC	8	ID	temporary copy of LC
LO	2	ID	load offset
SR	7	ID	store row index in external memory
SC	5	ID	store col. index in external memory
SO	7	ID	store offset
RPAGE	3	—	offset to image in external memory

Table 5.1 : Control registers and associated adders.

In order for the sliding window mechanism presented in Subsection 5.5.4 to work, it is necessary that the local memory acts like a circular buffer, which means that incrementing a pointer beyond the highest address must transparently roll-over to the beginning of the memory. Conversely, decrementing an address below zero must result in addressing the upper end of the memory. This is accomplished by implementing the increment operations — denoted A1, A4 and A18, respectively — as:

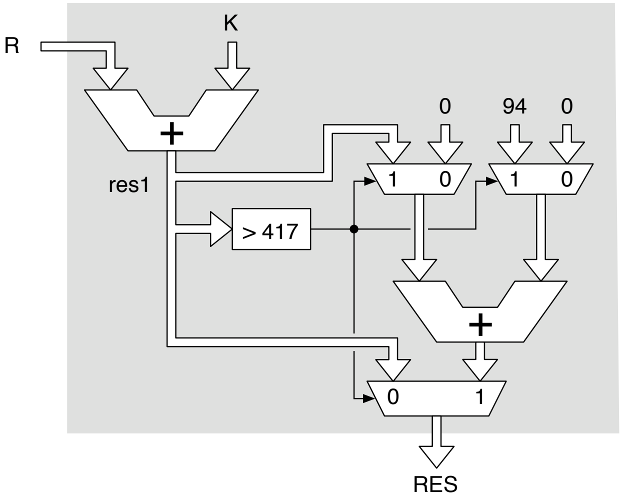
$$\begin{aligned} (R+1) \bmod 418 & \qquad \mathbf{A1} \\ (R+4) \bmod 418 & \qquad \mathbf{A4} \\ (R+18) \bmod 418 & \qquad \mathbf{A18} \end{aligned}$$

and the decrement operations — denoted S1, S4 and S18, respectively — as:

$$\begin{aligned} (R-1) \bmod 418 &\equiv (R+417) \bmod 418 \qquad \mathbf{S1} \\ (R-4) \bmod 418 &\equiv (R+414) \bmod 418 \qquad \mathbf{S4} \end{aligned}$$

$$(R - 18) \bmod 418 \quad \equiv \quad (R + 400) \bmod 418 \qquad \textbf{S18}$$

The straightforward implementation of the MAS is depicted in Figure 5.17 and consists in resorting to a 9-bit adder to perform the addition, to check whether the operation needs to perform a roll-over, in which case it executes it by adding 94 to the result, this value being the difference between 418 and 512, the natural modulo operation of the 9-bit adder. This approach is interesting in the sense that the second addition can be bypassed when the modulo operation does not need to perform a roll-over. The average amount of energy consumed by the second adder is therefore very low. However, since the result of the first adder constitutes one of the operands of the second adder, the critical path of the MAS is twice that of the other units in the coprocessor, leading to an unbalanced design.

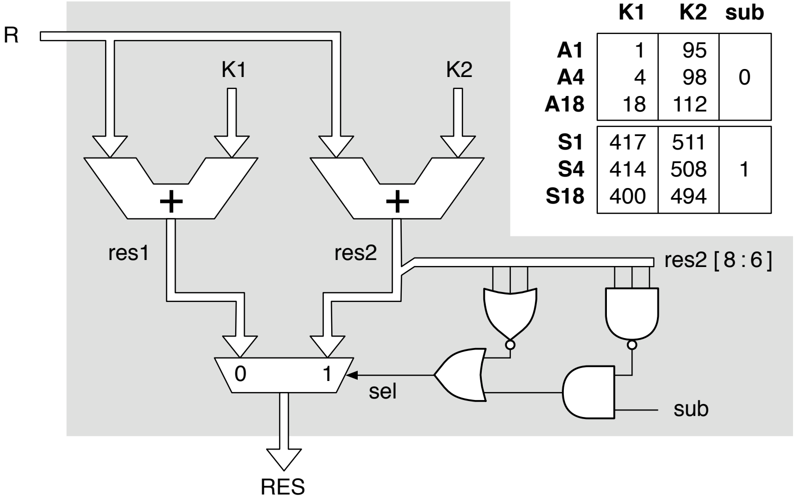


	K
A1	1
A4	4
A18	18
S1	417
S4	414
S18	400

Figure 5.17 : Sequential implementation of the MAS.
The table on the left indicates the value of the constant K supplied to the first adder for each MAS operation.

An alternative implementation consists in performing two additions in parallel. In this scheme, illustrated in Figure 5.18, the second adder uses a constant that already incorporates the offset of 94 for the computation of the modulo operation. The selection between the two results res1 and res2, cf Figure 5.18, depends upon on the three MSB of res2. Indeed, if the modulo was necessary, in the sense that it applied a roll-over to its argument, then res1 must lie within the range [418–435] if the operation was A1, A4 or A18, and outside the range [400–417] for the operations S1, S4 and S18. Thus, res2 must be inside the interval [0–17] for An operations, and outside [494–511] for Sn operations, respectively, for $n = 1, 4, 18$. Moreover, since the maximum value of the operand is 417, res2 cannot fall in the intervals [18–94] for An operations, and [417–493] for Sn operations. Consequently, the condition can be simplified by considering the intervals [0–63] and [448–511] instead. The modulo operation is thus performing a roll-over when res2 is less than 64 for An operations, and less than 448 for Sn operations. This requires testing only bits of res2 with index 8 down to 6, as well as the opcode bit *sub* indicating whether the unit is performing an Sn or an An operation. The resulting simplified selection logic is shown in Figure 5.18.

The structure of the MAS Block, encompassing the MAS and the four registers LR, LC, LMR and LMW, is presented in Figure 5.19. It should be noted that additions with modulo operations are only needed when dealing with registers that address LMEM, namely LMR and LMW, and not with registers LR and LC that address the external memory. However, the latter case represents less than 0.3% of the operations performed by the MAS, which does not justify the addition of a dedicated instruction. Since the considered image width and height are 256 pixels at most, LR and LC contain values in the interval [0–255] and are not affected by the modulo operation anyway. The MAS therefore automatically behaves like a regular adder-subtracter



	K1	K2	sub
A1	1	95	0
A4	4	98	
A18	18	112	
S1	417	511	1
S4	414	508	
S18	400	494	

Figure 5.18 : Parallel implementation of the MAS.
The table on the right indicates the values of the constants K1 and K2 supplied to the adders for each MAS operation, as well as the value of the signal “sub” furnished to the logic that selects the result.

when operating on these two registers.

5.5.6.2 Incrementer-decrementer

Figure 5.20 presents the structure of the ID Block, which incorporates the so-called Increment-Decrementer (ID) controlling the registers used to construct external read (TR, TC and LO) as well as write (SR, SC and SO) addresses, as explained in Subsection 5.5.6.3. While all six registers can be decremented, this operation is actually only necessary either for SR or for SC, depending on the direction in which the local window in the external memory is being displaced. The other four registers need only be incremented.

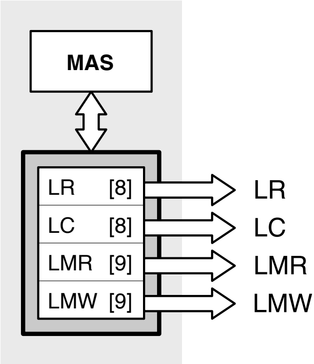


Figure 5.19 : MAS Block.

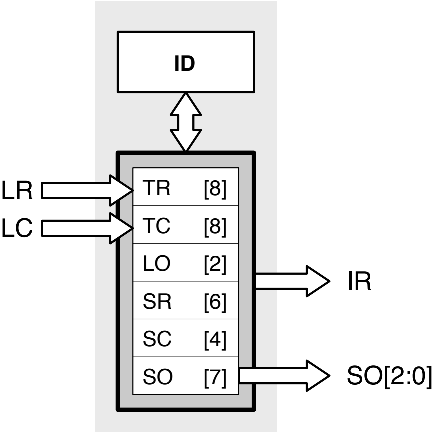


Figure 5.20 : ID Block.

5.5.6.3 External memory addressing unit

The morphology coprocessor contains three addressing units, two for to the internal memories (LMEM and SEMEM), and one dedicated to the external memory. Both internal memories are addressed with register indirect mode, meaning that the address used is simply a register value: LMR or LMW for LMEM (see Subsection 5.5.4) and SEIDX for SEMEM (see Subsection 5.5.5). This section details the external addressing unit (EXT ADDR) whose structure is presented in Figure 5.21.

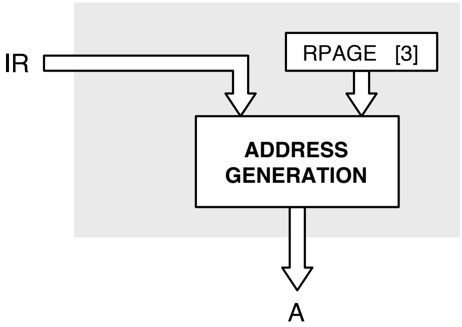


Figure 5.21 : External Addressing Unit (EXT ADDR).

The full memory map of the external memory is given in Figure 5.22. When storing results, each page is seen as containing 128×128 pixels, or one complete morphology level. The original image is stored at hexadecimal address 0x50000, corresponding to the beginning of memory page 20. Although the dimensions of the original image are 146×146 pixels, each row occupies 256 pixels. Indeed, this makes address generation possible by simple concatenation of the row and column address register. Thus, as the memory area storing the image is 146×256 pixels in size, it covers a little more than two pages. However, when reading pixels from the original image, the memory is seen as being structured in pages of 256 × 256 pixels. The area occupied by the original image is thus located in page 5, ranging from hexadecimal addresses 0x50000 to 0x5FFFF.

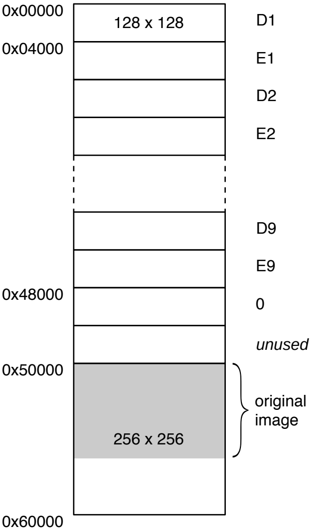


Figure 5.22 : External memory layout.
Layout of the original image and extracted features stored in the external memory.

5.5.6.4 Load address generation

The detailed layout of the original image as stored in the external memory is presented in Figure 5.23. The external memory load address is constructed from four registers, as shown in Figure 5.24. A 3-bit register RPAGE — standing for *read page* — that gets initialized by the MCU holds the constant value 5 used to locate the beginning of the memory page. On each row, pixels occupy addresses 2 to 147 instead of 0 to 145, as explained below.

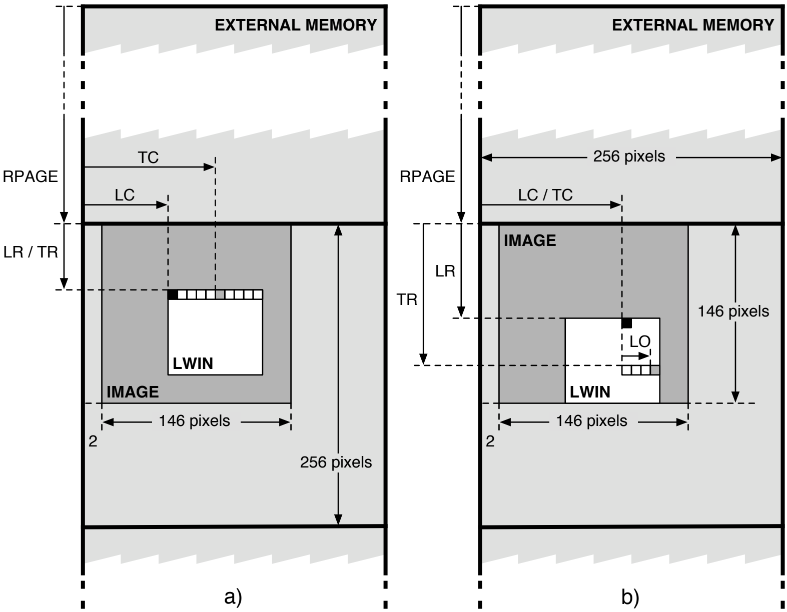


Figure 5.23 : External memory load addresses.

Situation when a) loading a new row, b) loading 4 new columns.

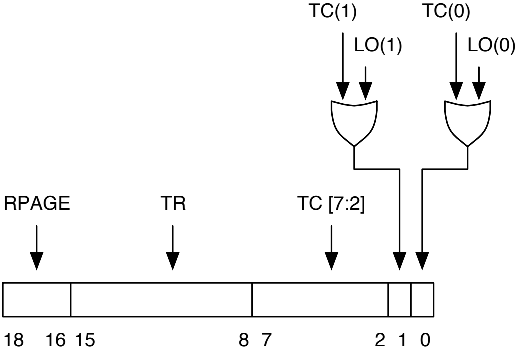


Figure 5.24 : External memory load address generation.

LR and LC always point to the top-left pixel of the area of the image that must be loaded, this pixel being pictured in black in Figure 5.23, and are only updated when the window LWIN is being translated over the memory space. TR and TC are temporary copies of LR and LC, produced when the CPY instruction (see Subsection 5.6.2) is encountered, and modified in the course of updating the local image memory. TR always locates the row index of the pixel to read, depicted in gray in Figure 5.23, whereas the role of TC depends on the action being performed. When the LWIN window was displaced vertically, meaning that a new row of 22 pixels must be loaded, TC indicates the column index of the pixel to read. This situation is illustrated in Figure 5.23a, where LWIN is moving upward, so that the new row to load is located on top of the window. However, when LWIN was displaced horizontally after it reached the top or the bottom of the image — which means that 4 new columns of 19 pixels must be loaded — TC indicates the index of the leftmost of the four columns to be read, as shown in Figure 5.23b. In this case, the 2-bit register LO is incremented after each load and used to specify the horizontal offset from TC to the pixel to read, as detailed below.

Once the window reaches the bottom or the top of the image, four new columns of 19 pixels must be loaded as explained in Subsection 5.5.4. The horizontal offset to the first column to load is 22 from the first pixel of the row. Since the horizontal offset of this pixel is 2 as noted above, the actual address of the first column is 24. It is thus possible to obtain the offset of all four columns — namely 24, 25, 26 and 27 — by simply performing an OR operation between the last two bits of LC and LO, as seen in Figure 5.24. As the horizontal displacement of the window only consists in moving by 4 columns to the right, this will be always be true. When LWIN is displaced vertically and rows are being loaded, the value of LO is maintained at zero, so that the bit-wise applied OR operation in Figure 5.24 has no effect.

It is worth noting that this scheme was designed when the image was originally scanned row-wise, to avoid losing one cycle after loading a pixel in the fourth column. Indeed, in this situation, both the row and the column indexes need to be modified: TR must be incremented, whereas TC must be decremented by 3. With the current column-wise processing, loading 4 columns only happens a total of 31 times for the considered image width of 146 pixels. Each time, 18 additional cycles would be required to update both TR and TC when moving from a pixel in the fourth column to the pixel in the first column on the next row, in case a *decrement by 3* operation is implemented. Otherwise, if the operation is realized by successively applying operators S4 and A1, 36 additional cycles would be needed. Even in the latter case, only 1'116 additional cycles would be required over the whole image, which is negligible. Consequently, the scheme involving the use of register LO as explained above, could be abandoned. When considering the original row-wise scanning method however, 141'732 additional cycles would have been needed, corresponding to almost half the number of cycles that were then devoted to loading pixels, as reported in [123].

5.5.6.5 Store address generation

The detailed layout of the area occupied by the resulting data stored in the external memory is presented in Figure 5.25. The external store address is computed as shown in Figure 5.26. SO[6:2] indicates the erosion and dilation level, while SO[1:0] is used to identify the position of the pixel. SO[2:0] also controls the output multiplexers of the morphology block, selecting the output register containing the result to store. More precisely, SO[2] provides the EnD signal sent to MB in Figure 5.13 while SO[1:0] generates the POS signal. With this scheme, incrementing SO by 1 after each store operation in the external memory suffices to handle the updating of both the offset for each of the four positions and the dilation and erosion level.

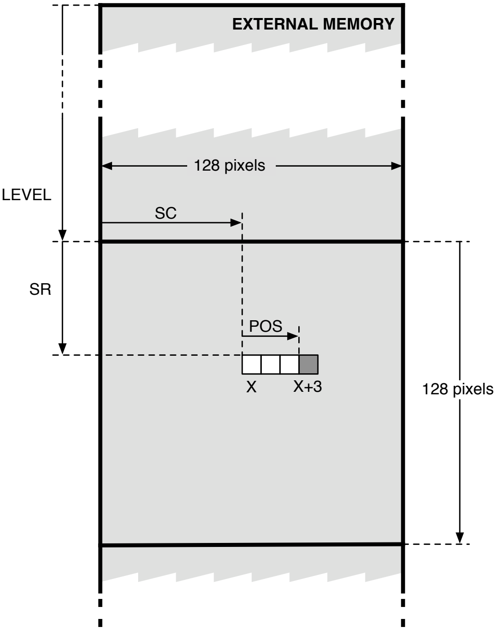


Figure 5.25 : External memory store addresses.

For one given level L, the eight results for the memory locations X to X+3 as depicted in Figure 5.25 are generated and transferred to memory in the following order:

$$E_X\ ,\ E_{X+1}\ ,\ E_{X+2}\ ,\ E_{X+3}\ ,\ D_X\ ,\ D_{X+1}\ ,\ D_{X+2}\quad D_{X+3}$$

Levels L are ordered in memory in the chronological order in which they are generated, from Level 1 to 9 with the exception of Level 0 that consists of only four values — namely those of

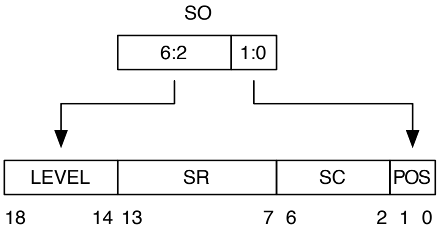


Figure 5.26 : External memory store address generation.

the original pixels[7] — and which will be located after Level 9 (see Figure 5.22), even tough these are the first results to be available. This allows the memory addresses of images that were eroded and dilated with the same Level L to differ by only one bit: addresses corresponding to dilated images have bit 14 set to 0, while it is set to 1 for eroded images. To achieve this behavior, only SR and SC are updated after all results for the current four positions have been stored. The register SO on the other hand is not reset until after the four results for Level 0 of the new locations have been stored.

5.5.7 Control unit and instruction set

The control unit is responsible for fetching the instructions from the program memory and for generating the control signals sent to the other units. This first task is linked to the architecture of the morphology coprocessor. The control unit also maintains the PC stack and the loop counters, this second twofold task depending only on the behavior and implementation of the PC

[7] Even though the original pixels are already available in memory as part of the original image, they are stored again so that the matching coprocessor can use a single scheme to construct memory addresses.

unit. For this second task, the morphology coprocessor uses the same implementation as the graph matching coprocessor, which is thoroughly described in Section 6.5.1 in [1], so that only a quick description of the control unit is given here.

The coprocessor instruction word is 14-bit wide, and is divided into 5 fields, as shown in Figure 5.27a. The ID and MAS fields control the Incrementer-Decrementer module described in Subsection 5.5.6.2 and the Modulo Adder-Subtracter described in Subsection 5.5.6.1, respectively. The MMU field (for Mathematical Morphology Units) holds commands targeting the EMUs (see Subsection 5.4.1) and the SE Block (see Subsection 5.5.5), the opcode MN being used to schedule a MM operation together with the insertion of an IGNORE value, as explained in Subsection 5.5.5. The LSU field (for Load and Store Unit) contains instructions for the External Memory Addressing Unit (see Subsection 5.5.6.3), whereas the role of the JMP bit is to indicate the end of a loop, as explained below.

Loops are declared using the dedicated opcode LOOP in the ID field. In this case, part of the bits in the instruction word hold the number of times the loop must be iterated, minus one[8], while the remaining bits are unused (see Figure 5.27b). The end of the loop is indicated by the JMP bit in the instruction word, cf Figure 5.27a. When this bit is set and the loop counter is nonzero, the latter is decremented and the execution continues at the start of the loop. Otherwise, the loop exits and the following instruction is executed. As the address of the next instruction to fetch must be known before the current instruction is executed, the assembler will set the JMP flag (signaling the end of the loop) on the second to last instruction in the loop. The loop counter is thus evaluated while the last loop instruction is fetched. On the

[8] As the first iteration is always performed, a value of e.g. 10 would cause the loop to be executed 11 times. The assembler takes care of this conversion.

following cycle, the fetch unit knows whether it needs to retrieve the instruction at the top of the loop or the instruction following the loop. Consequently, no instruction is ever fetched and then discarded due to changes in the execution flow, with saves both energy and time.

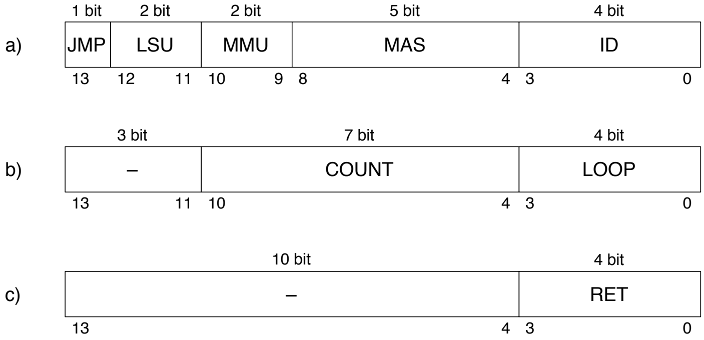


Figure 5.27 : Morphology coprocessor instruction word.
Format of the instruction word when holding a) a regular instruction, b) a loop declaration, and c) the RET instruction.

At the end of the program, a RET (return) instruction is mandatory. When it is encountered, the MCU is notified that the coprocessor has completed its task and that it has now entered an idle state. In the current implementation, the handshaking is performed through a bit in a control register periodically polled by the MCU, which was found to be the easiest solution for the FPGA-based demonstrator (see Section 7.3). A more efficient solution would consist in letting the coprocessor trigger an interrupt signals on the MCU, so as to avoid unnecessary data transfers. As shown in Figure 5.27c, the opcode signaling the RET instruction, located in the ID field, disables the remainder of the instruction word, so that no other operation can be executed at this time.

The opcodes of each field, along with their binary form representation are enumerated in Figure 5.28. Note that to minimize the size of the instruction word, the RM operation was made part of the MAS field, even though it would have been more logical to place in the MMU field.

5.6 Coprocessor characteristics

This section discusses and characterizes the architecture of the complete morphology coprocessor. Its behavior is illustrated using a few typical code snippets taken from the actual program it executes to perform features extraction. Finally, some performance figures are given in Subsection 5.6.3, that were obtained through simulations using the tools introduced in Section 7.2.

5.6.1 Architecture overview

The architecture of the complete morphology coprocessor, constituted of the components described previously, is presented in Figure 5.29. To reduce clutter, the control block, encompassing the fetch and decode unit as well as the program memory, is not shown here. Furthermore, the communication interface with the MCU, the clock and reset signals as well as the buses allowing the MCU to access any register and memory in the coprocessor are not depicted either. The coprocessor uses a Harvard-style architecture, where the instructions and the data use separate buses and memories, and corresponds to a RISC (reduced instruction set computer) machine. Indeed, the instructions share all the same size, they all execute in one cycle and the processing units only read their operands from registers, and write results to registers. Additionally to the parallelism implemented in the Morphology Block, the coprocessor can dispatch up to four in-

JMP

—	0	Regular instruction
JUMP	1	Indicates the second to last instruction in a loop

LSU

NOP	1 1	No operation
CPY	1 0	Copy LC to TC and LR to TR
LD	0 1	Load pixel from external memory
ST	0 0	Store one result to external memory

MMU

NOP	1 1	No operation
MM	1 0	Partial differential morphology, L ←SEREG
MC	0 1	Combine differential morphology results
MN	0 0	Partial differential morphology, L ←IGNORE

MAS

NOP	1 1 1	–	No operation
RM	1 1 0	–	Reset MORPHO BLOCK
S1	1 0 1	*R*	Subtract 1 from *R*, mod 418
A1	1 0 0	*R*	Add 1 to *R*, mod 418
S4	0 1 1	*R*	Subtract 4 from *R*, mod 418
A4	0 1 0	*R*	Add 4 to *R*, mod 418
S18	0 0 1	*R*	Subtract 18 from *R*, mod 418
A18	0 0 0	*R*	Add 18 to *R*, mod 418

R

0 0	LR
0 1	LC
1 0	LMW
1 1	LMR

ID

NOP	1 1 1	1	No operation
RSO	1 1 1	0	Reset SO
RET	1 1 0	1	Terminate execution
LOOP	1 1 0	0	Loop declaration
DEC	*S*	1	Decrement register *S*
INC	*S*	0	Increment register *S*
opcode	binary re- presentation		description

S

0 0 0	SO
0 0 1	LO
0 1 0	TC
0 1 1	TR
1 0 0	SC
1 0 1	SR

Figure 5.28 : Opcodes of the morphology coprocessor.

dependent operations per cycle. The architecture does however not follow the superscalar approach, where the unit selection and scheduling is handled at runtime by the control unit. Indeed, it is more interesting to perform the unit selection and scheduling at compile time where plenty of computational power is available[9], rather than letting the processor analyze and detect parallelism on the fly, as it executes the code. The coprocessor can thus be categorized as a VLIW (Very Long Instruction Word) machine, even though its instruction word size amounts to 14 bits only. Indeed, the determining characteristic is the presence of four independent fields, each scheduling an instruction to a specific functional unit.

5.6.2 Sequencing

This section illustrates the features extraction as performed by the morphology coprocessor, using three typical segments of the executed program.

Firstly, Figure 5.30 shows a snippet of code used to update the local memory window when traveling the image downward. Both LMR and LR get incremented to point to the next row in LMEM and in the external memory, respectively. Before starting the loop that performs the actual loading of the 22 new pixels in LMEM, the CPY instruction is issued to replicate LC to TC and LR to TR. Finally, the RM instruction is issued to reset the L pipeline inside the Morphology Block, the register banks inside the EMUs as well as SEIDX and SEOUT in the SE Block.

The segment of code in Figure 5.31 performs morphology features extraction on rows 2 to 10 of the SE set. For each row,

[9] Since no dedicated compiler was developed in this project, the scheduling and the resource allocation effort must currently be furnished by the human being that writes the assembly code.

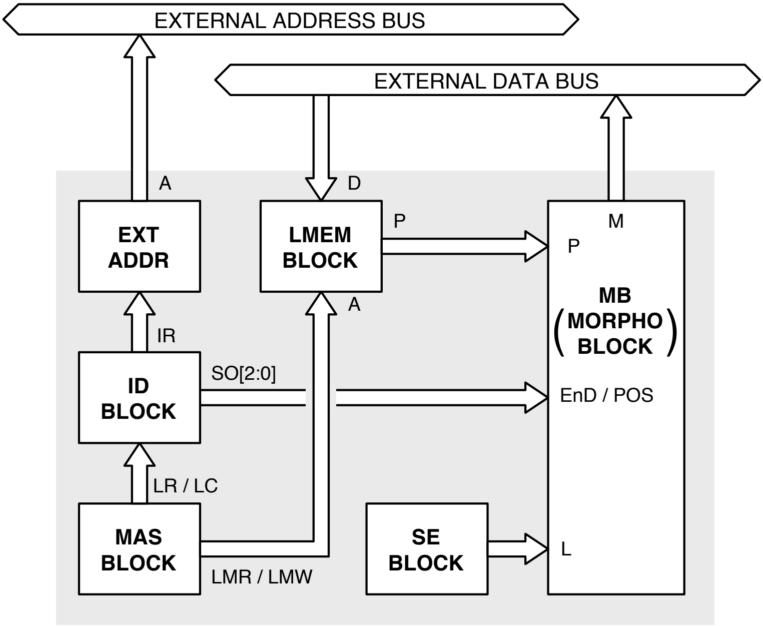


Figure 5.29 : Architecture of the morphology coprocessor.

```
;  LSU       MMU        MAS           ID

   nop   :   nop   :   a18 lmr   :   nop
   nop   :   nop   :    a4 lmr   :   nop
   cpy   :   nop   :   inc lr    :   nop

   LOOP 11
     ld  :   nop   :   inc lmw   :   inc tc
     ld  :   nop   :   inc lmw   :   inc tc
   JUMP

   nop   :   nop   :     rm      :   nop
```

Figure 5.30 : Snippet of code for updating the local memory.

19 MM followed by 3 MN instructions are issued, while LMR gets incremented during each cycle. Processing one row of the SE set requires 22 cycles, totaling 23 cycles when adding the one consumed by the declaration of the inner loop.

```
;     LSU      MMU        MAS        ID

   _se_loop_row_2_10_
   LOOP 9

     LOOP 9
       nop :   mm   :   inc lmr  :   nop
       nop :   mm   :   inc lmr  :   nop
     JUMP

     nop   :   mm   :   inc lmr  :   nop
     nop   :   mn   :   inc lmr  :   nop
     nop   :   mn   :   inc lmr  :   nop
     nop   :   mn   :   inc lmr  :   nop

   JUMP
```

Figure 5.31 : Snippet of code for extracting morphology features.

The segment of code in Figure 5.32 performs the morphology features extraction on rows 12 to 18 of the SE set. It is similar to the previous code segment, except that during the first 8 instructions of each of the row, results are stored back to the external memory. This is accomplished using the ST instruction while incrementing SO. At this time, all processing units are active. Towards the end of each of the row, an MC instruction is issued to generate the final results.

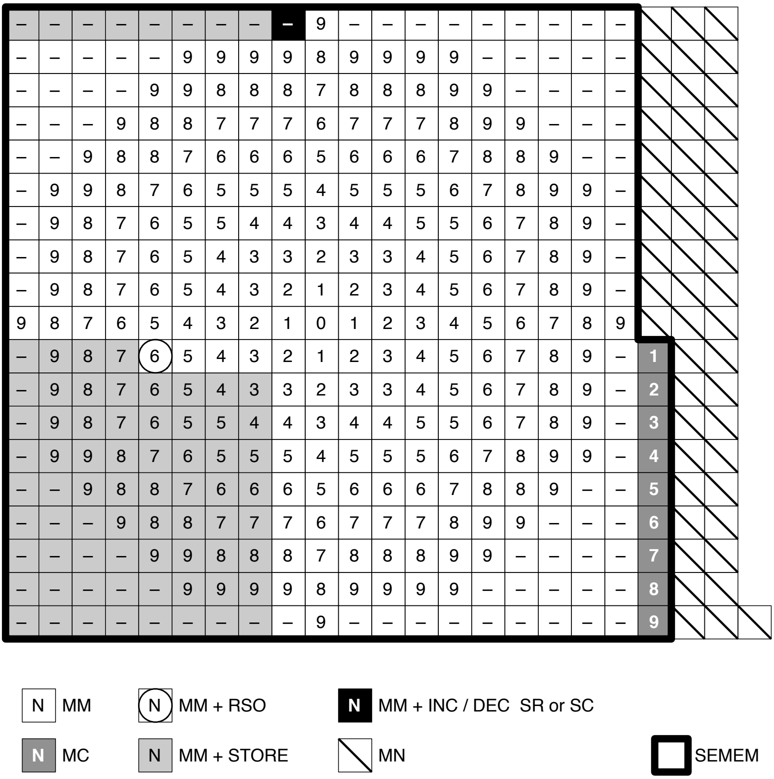
The sequencing of the morphology features extraction is illustrated in Figure 5.33. Every cell corresponds to one cycle, and indicates the value read from SEMEM — unless the IGNORE symbol is inserted by the MN instruction — as well as the operations being executed. The first store operations transfer the last 8 results (erosion and dilation Level 9) from the previous set of

```
;     LSU       MMU        MAS         ID

   _se_loop_row_12_18_
   LOOP 7

     LOOP 4
       st   :   mm   :   inc lmr  :  inc so
       st   :   mm   :   inc lmr  :  inc so
     JUMP

     LOOP 5
       nop  :   mm   :   inc lmr  :   nop
       nop  :   mm   :   inc lmr  :   nop
     JUMP

     nop    :   mm   :   inc lmr  :   nop

     nop    :   mc   :   inc lmr  :   nop
     nop    :   mn   :   inc lmr  :   nop
     nop    :   mn   :   inc lmr  :   nop

   JUMP
```

Figure 5.32 : Snippet of code for extracting the morphology features and for storing results to external memory.

4 locations, before one of the external store address registers gets updated according to the direction of displacement. The first results to be output for the new locations are the original central pixels. At this time, the value of SO[6:2] is 18, corresponding to the 128 × 128 memory page where Level 0 will be stored (see Figure 5.22). The RSO instruction is issued afterwards to reset SO to 0, so that it now points to the first memory page, where results will be stored for the dilation Level 1.

Replacing the circular SE set used here with a different one would simply require changing the values stored in SEMEM. No modification to the architecture of the coprocessor or to the code of the program would be required.



–	–	–	–	–	–	–	–	–	–	9	–	–	–	–	–	–	–	–	
–	–	–	–	–	9	9	9	9	8	9	9	9	9	–	–	–	–	–	
–	–	–	–	9	9	8	8	8	7	8	8	8	9	9	–	–	–	–	
–	–	–	9	8	8	7	7	7	6	7	7	7	8	9	9	–	–	–	
–	–	9	8	8	7	6	6	6	5	6	6	6	7	8	8	9	–	–	
–	9	9	8	7	6	5	5	5	4	5	5	5	6	7	8	9	9	–	
–	9	8	7	6	5	5	4	4	3	4	4	5	5	6	7	8	9	–	
–	9	8	7	6	5	4	3	3	2	3	3	4	5	6	7	8	9	–	
–	9	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	9	–	
9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	
–	9	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	9	–	1
–	9	8	7	6	5	4	3	3	2	3	3	4	5	6	7	8	9	–	2
–	9	8	7	6	5	5	4	4	3	4	4	5	5	6	7	8	9	–	3
–	9	9	8	7	6	5	5	5	4	5	5	5	6	7	8	9	9	–	4
–	–	9	8	8	7	6	6	6	5	6	6	6	7	8	8	9	–	–	5
–	–	–	9	8	8	7	7	7	6	7	7	7	8	9	9	–	–	–	6
–	–	–	–	9	9	8	8	8	7	8	8	8	9	9	–	–	–	–	7
–	–	–	–	–	9	9	9	9	8	9	9	9	9	–	–	–	–	–	8
–	–	–	–	–	–	–	–	–	–	9	–	–	–	–	–	–	–	–	9

Figure 5.33 : Graphical view of the sequencing of the features extraction algorithm executed on the morphology coprocessor. The number 15, denoting the IGNORE value, has been replaced by a dash for clarity.

5.6.3 Performance

Table 5.2 reports the simulation results for the morphology coprocessor. The program consists of 273 instructions of 14 bits, requiring a memory size of 3’822 bits. Thanks to the use of the local memory, pixels in the original image are only read 4.3 times on average during the processing.

Type of operation	**Nb of occurrences**
External memory read	92’182
External memory write	311’304
Local image memory read	1’720’320
Local image memory write	92’182
SE memory read	1’519’616
EMU comparisons (circular SE set)	8’552’448
EMU comparisons (maximum)	12’091’392
SEINC increments	1’519’616
MAS additions / subtractions	1’825’537
ID increments / decrements	412’286
Loop declarations	127’030
NOP cycles (all units being idle)	19
Total nb of executed cycles	**1’966’882**
Static nb of instructions (program size)	**273**

Table 5.2 : Performance of the morphology coprocessor.

The total number of comparisons executed to extract morphology features from one image depends on the shape of the structuring elements in the SE set. It amounts to approximately 8.5 million comparisons when using the circular SE set shown in Figure 5.33, but it can increase to more than 12 million when SEMEM does not contain any IGNORE value[10]. The number of cycles to execute to process one image is approximately 2 million, which means that a very low operating frequency would still enable features to be extracted in much less than one second.

Note that these results differ from the ones reported in [123] on two accounts. Firstly, the static number of instructions in the program increased from 217 to 273. This was caused by modifications to the implementation of the loop mechanism, so as to avoid wasting cycles when jumping. As a consequence, loops are now restricted to contain at least two instructions, as it was also explained in Section 6.5.1 in [1]. This necessitated the application of loop unrolling in some parts of the program, which led to substantially increasing the number of instructions it contains. Secondly, the change in the way the local memory window is displaced over the image, as explained in Subsection 5.5.4, resulted in reducing both the number of external memory read operations and the number of cycles dedicated to manipulating the corresponding address registers. Globally, the number of executed cycles now amounts to less than 2 million, representing a reduction of 12% with respect to the figures previously reported in [123].

Task	**% of executed cycles**
Loop declarations	6.4 %
Control and local memory refresh	6.1 %
Features extraction and results stores	87.5 %

Table 5.3 : Percentage of execution cycles devoted to loop declarations and to the two main tasks performed by the morphology coprocessor.

Table 5.3 shows the relative importance, in terms of the number

[10] This would be the case for instance when using a square SE set.

of consumed execution cycles, of the two main tasks performed by the coprocessor. The first one, consisting in control operations and refreshing the local memory, represents only about 6% of the execution time while the features extraction, even though it is performed using 8 comparators working in parallel, requires almost 15 times more cycles.

Finally, Table 5.4 reports the duty cycle of the adders and comparators of the morphology coprocessor. Depending on the chosen SE set, the 8 comparators inside the EMUs are working during 54% to 77% of the number of executed cycles. The MAS is particularly active, since it is being employed during every phase of the features extraction program, as seen in Figure 5.30 to Figure 5.32. Unlike the MAS, the ID unit is only used when refreshing the local memory or when storing results to the external memory, and as such it is only active for approximately one fifth of the duration of the complete program.

Component	**Duty cycle**
Comparators in EMUs (circular SE set)	54.4 %
Comparators in EMUs (maximum)	76.8 %
Modulo Adder-Subtracter (MAS)	92.8 %
Incrementer-Decrementer (ID)	21.0 %
SE Block incrementer (SEINC)	77.3 %

Table 5.4 : Duty cycle of the components of the morphology coprocessor.

5.7 Conclusion

In this chapter, a coprocessor dedicated to low-power morphology features extraction was presented. It was shown that by using a specifically crafted architecture, it is possible to minimize the number of comparisons, as well as to drastically reduce the number of necessary external memory accesses. Less than 2 million cycles were found to be sufficient for extracting features out of one face image. As a result, such a device could operate at a very low frequency yet still performing operations fast enough to be acceptable from the user's point of view. Even though it is highly optimized, the proposed solution is not tied to a predefined set of structuring elements. Indeed, the latter can be easily replaced since this operation simply consists in uploading the description of the new SE set to SEMEM, whereas both the architecture and the software program remain identical. To a certain degree, more extensive changes, such as those concerning the image size or to the number of considered morphology levels, can be implemented by modifying the software program.

In addition to its validation carried out using the custom tool discussed in Section 7.2, the morphology coprocessor was described in VHDL in the framework of a master's thesis project [124], in view of realizing an FPGA-based demonstrator. Whereas the VHDL implementation was completed and simulations could be performed to validate the architecture, the development of the demonstrator was not finalized beyond closure of the MSc project, and this for the following reason. Indeed, in the meantime, the need for normalizing the extracted features according to Subsection 3.4.3, was established, so that no further work was furnished on the FPGA implementation of the solution discussed in this chapter. Instead, the architecture was modified to include the normalization step, which led to a second coprocessor version being developed. The latter, which will be introduced in Chapter 6, was then described in VHDL and implemented on an FPGA-based demonstrator, as it will be discussed in Chapter 7.

Chapter 6

Architecture including Features Normalization

6.1 Introduction

This chapter describes the enhanced coprocessor, which normalizes the morphology features after they have been extracted, as described in Subsection 3.4.3. The algorithm implementation is laid out in Section 6.2, which explains how the normalization task was incorporated into the existing architecture so as to minimize both the additional hardware resources required as well as the impact on the execution time. Section 6.3 then details the modifications brought to the individual components and discusses the newly introduced support for subroutines. Finally, Section 6.4 illustrates the behavior of the coprocessor using code samples extracted from the actual features extraction and normalization code, and provides some performance figures obtained through simulations and later validated using the FPGA-based demonstrator presented in Chapter 7.

6.2 Implementation of the normalization

According to Equation 3.36 and as discussed in Subsection 3.4.3, the normalized morphology features $\hat{M_i}$ are obtained as follows:

$$\hat{M_i} = \frac{M_i - E9}{D9 - E9} \quad i \in [1, \ldots, 17] \ , \ E9 = M_0 \ , \ D9 = M_{18} \quad (6.1)$$

Normalization of any extracted feature thus requires knowledge of the values of E9 and D9 for the corresponding position. As E9 and D9 are the last values computed when performing morphology features extraction, normalization cannot start before all results have been obtained for the four current locations. To avoid power-consuming accesses to external memory, these 19 values must be stored locally until normalization can be carried out. Each EMU already comprises two banks of 9 registers each (see Subsection 5.4.1), which, ideally, would have to be reused to store the extracted features. A supplementary register should still be added in one of the register banks depicted in Figure 5.8, to store the value of the central pixel when it is fed to the EMU, whereas the nine erosion and dilation results would be stored to the register banks when they are recursively generated, in addition to being written to the two output registers.

In order to minimize the number of additional cycles induced by the normalization task, the latter should preferably be performed in parallel with features extraction. However, as already mentioned, it cannot start before all features have been extracted. Neither can it be executed while the local memory is updated, as both tasks would interfere given the fact that they would both need to access the external memory. Moreover, refreshing the local memory usually involves loading a new row of 22 pixels (see Subsection 5.5.4), whereas the number of features to normalize amounts to 68. Indeed, there are 4 locations processed in parallel, each producing 17 values as explained in Subsection 3.4.3. Hence, the ideal solution would consist in performing normalization while extracting features for the next four locations. Yet the register banks holding the features to normalize will start getting overwritten with the partial results of the ongoing morphology operations. This conflicting use of resources can however be lifted, provided that we can impose one minor constraint to the shape of the structuring elements.

6.2.1 Sequencing of the normalization operations

It can be observed that when an MC instruction is executed inside one EMU, it frees up one register in each bank. Indeed, once the differential erosion and dilation for level L have been combined with the previous results E_{L-1} and D_{L-1}, respectively, the level won't appear again for the current location, as no additional pixels belonging to $dSEk$, with $k \leqslant L$, needs to be processed. Considering the first half of the features extraction algorithm, covered by the upper half of the structuring element set (see Figure 5.33), we observe a symmetrically reversed situation: registers pD_k and pE_k, with $k \leqslant L$, are not needed before encountering the first pixel underneath dSE_k. If we now assume the upper half of the SE set to be symmetrical to the lower half[1], it becomes possible to keep the morphology results in pD_k and pE_k until the first pixel underneath dSE_k is processed when working on the next four positions, as illustrated in Figure 6.1. Consequently, 22 cycles (see Subsection 5.6.2) would be available to normalize all eight morphology features of one single level. As an additional cycle will be needed anyway to retrieve the values to normalize from the register banks of the EMUs, a total of 23 cycles will effectively be available.

Once all morphology features are available in the EMUs, namely that the features extraction was completed for the current posi-

[1] This is already required to obtain features invariant to rotations.

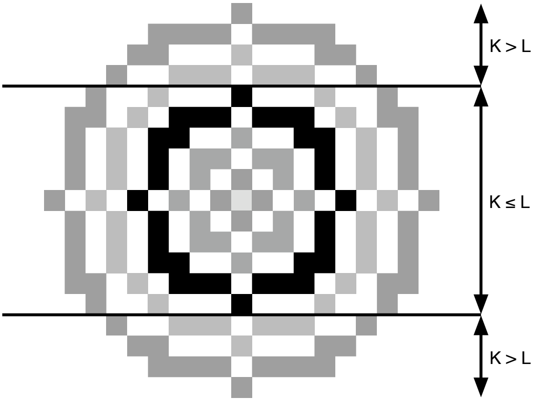


Figure 6.1 : Symmetrical SE set. dSE_L is drawn in black.

tions, the normalization task itself is a two-stage process. The first step, described in Subsection 6.2.3, aims at obtaining the constants that will be required when normalizing the extracted features, $E9$ and $D9 - E9$. At this time, both $D9$ and $E9$ are available in the output registers *Dout* and *Eout* of the morphology units. The four values $D9 - E9$ can be computed in parallel by their respective EMUs, provided that we upgrade one of the comparators to a complete subtracter. At this time, $pD9$ and $pE9$ are reset, so that features extraction can start for the following four positions. The second step consists in performing the actual normalization of each of the 4×17 extracted features, which is also a two-stage process: first compute $M_i - E9$, and then execute the division, as explained in Subsection 6.2.4.

Since a subtracter is needed to perform these operations, the normalization unit will be implemented as a part of the ID BLOCK, which was therefore renamed IDNORM. One single adder will thus be used both for the subtractions necessary to the normalization process and for the original tasks of the ID BLOCK, namely updating the address registers. Once they have been normalized, the features will finally be written to the external memory.

6.2.2 Implementation of the division

The required divisions are implemented as subtractions in logarithmic space, with the help of two lookup tables (LUT). The first one is used to convert both operands to logarithms, while the second performs the inverse operation. The values $\hat{M}_i$ in Equation 6.1 are thus replaced by $\hat{M}_i''$ according to Equation 6.3:

$$\hat{M}_i' = \log_a \hat{M}_i = \log_a (M_i - E9) - \log_a (D9 - E9) \tag{6.2}$$

$$\hat{M}_i'' = a^{\hat{M}_i'} = \text{invlog}_a\, \hat{M}_i' = \hat{M}_i \tag{6.3}$$

Extracted morphology features are integers ranging from 0 to 255, which means that LOGMEM, the memory holding the logarithm LUT, needs to contain 256 words. The logarithm values are computed as follows:

$$LOGMEM[\,N\,] \;=\; \begin{cases} 0 & \text{when N} = 0 \\ 256 \cdot \log_{256}(N) & \text{when } 1 \leqslant \text{N} \leqslant 255 \end{cases}$$

They are then truncated to the nearest integer, so that they can be stored as 8-bit values. The inverse logarithm operation is implemented using another LUT, stored in a second memory, INVLOGMEM. The values it contains are obtained as follows:

$$INVLOGMEM[\,N\,] = 256^{\frac{N}{256}} \qquad N \in [0, \ldots, 255]$$

Likewise to LOGMEM, the computed values are then truncated to the nearest integer and stored as 8-bit values. Implementing the proposed normalization thus requires the inclusion of two

256 × 8 bits memories, as well as the replacement of the incrementer / decrementer found in the ID Block by a true subtracter.

Simulations were performed on the XM2VTS database to evaluate the influence of replacing the division on the performance of the algorithm. The results, reported for both the single and multiple templates scenarios when using 8-bit precision, are presented in Figure 6.2 and clearly show that the impact is very limited in both cases.

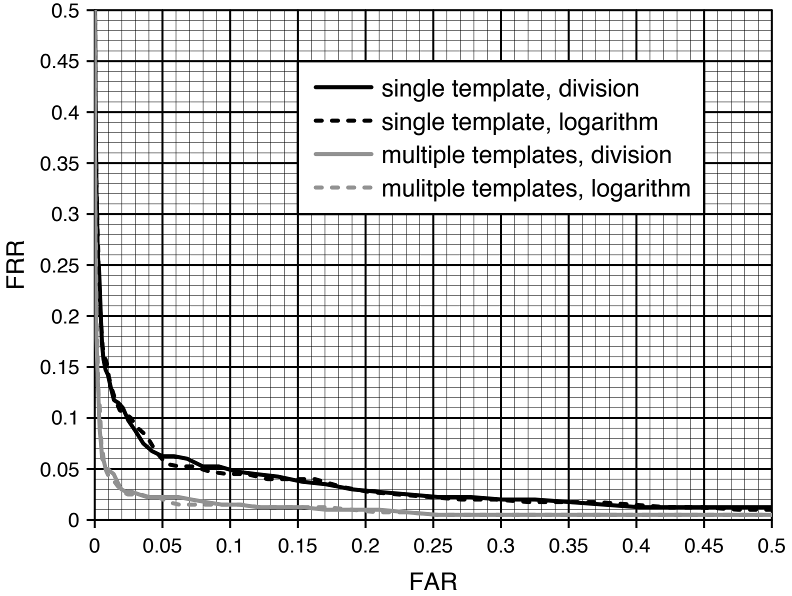


Figure 6.2 : ROC with subtractions in logarithmic space.
ROC curves for normalized morphology features, using either full floating-point divisions, or 8-bit subtractions in the logarithmic domain, the face matching being performed using $\hat{M}''_i$ in Equation 6.3 in the latter case.

Since both $\hat{M}'_i$ and $\hat{M}''_i$ are monotonous functions, further tests were carried out on the XM2VTS database to determine whether the step in Equation 6.3 could be suppressed by considering directly $\hat{M}_i'$ for the matching of the faces. This would remove the need for the second lookup table and for the memory that contains it. However, the results obtained reveal a strong degradation of the system accuracy when graph matching is performed using features expressed in the logarithmic domain. Indeed, when considering low FAR, the achieved FRR is much worse when the inverse logarithm operation is skipped, as shown in Figure 6.3, both for the single and multiple templates scenarios. Consequently, the inverse logarithm operation was retained.

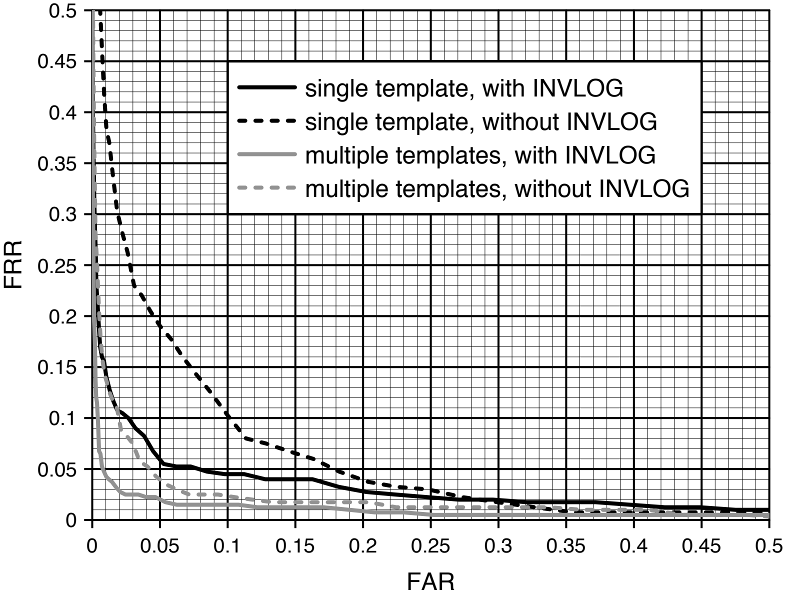


Figure 6.3 : ROC without inverse logarithm operation.

ROC curves for normalized morphology features, with and without the inverse logarithm operations.

6.2.3 Initialization of the normalization

Before starting normalization for all extracted features at one given position, the following initialization operations must be executed:

$$\begin{aligned} rE9 &\leftarrow E9 \\ RANGE &\leftarrow D9 - E9 \\ LOGRNG &\leftarrow LOGMEM[\, RANGE\,] \end{aligned}$$

At this time, registers $pD9$ and $pE9$ are also reset to 0 and 255, respectively, so that features extraction can proceed for the next positions alongside the features normalization for the current position. At the first occurrence of the MC instruction, registers *Dout* and *Eout* will be overwritten. As all four $E9$ values must be kept available until the last feature has been normalized, they have to be preserved in some way. To this effect, a register $rE9$ is added next to each EMU as seen Figure 6.7.

Since D9 and E9 are still present in the output registers at this time, the three operations can be expressed as:

$$\begin{aligned} rE9 &\leftarrow Eout \\ Dout &\leftarrow Dout - Eout \\ LOGRNG &\leftarrow LOGMEM[\, Dout\,] \end{aligned}$$

The first two operations above can be executed in parallel, and at the same time in the four morphology units, in response to a new opcode, DE9 (short form standing for D9 – E9), belonging to the MMU field of the instruction word (see Subsection 6.3.7). As the third operation requires access to LOGMEM, only one value can be computed per cycle. Consequently, the normalization unit will need four cycles to obtain the four LOGRNG values. A new opcode, NLI, is introduced in the IDNORM field to trigger the execution of the logarithm operation that generates LOGRNG.

The signal POS, constituted by the lower two bits of SO, cf Figure 5.26, is used to select between the four EMUs, similarly to the way this was performed when results to be stored to external memory were selected in the original architecture, cf Figure 5.13. In addition to performing the logarithm operation, the NLI instruction also increments SO, hence its name which stands for *Normalization Logarithm Increment.* As only the four Dout registers are accessed at this time, the signal EnD is set to 0. The sequence of operations needed to initialize the normalization for the four current locations thus becomes:

$$\left.\begin{array}{rcl} rE9[\,0\,] & \leftarrow & \mathit{Eout}[\,0\,] \\ \mathit{Dout}[\,0\,] & \leftarrow & \mathit{Dout}[\,0\,] - \mathit{Eout}[\,0\,] \\ rE9[\,1\,] & \leftarrow & \mathit{Eout}[\,1\,] \\ \mathit{Dout}[\,1\,] & \leftarrow & \mathit{Dout}[\,1\,] - \mathit{Eout}[\,1\,] \\ rE9[\,2\,] & \leftarrow & \mathit{Eout}[\,2\,] \\ \mathit{Dout}[\,2\,] & \leftarrow & \mathit{Dout}[\,2\,] - \mathit{Eout}[\,2\,] \\ rE9[\,3\,] & \leftarrow & \mathit{Eout}[\,3\,] \\ \mathit{Dout}[\,3\,] & \leftarrow & \mathit{Dout}[\,3\,] - \mathit{Eout}[\,3\,] \end{array} \quad \mathbf{DE9} \;\right\} \text{ 1 cycle}$$

$$\left.\begin{array}{rcll} \mathit{LOGRNG}[\,0\,] & \leftarrow & \mathit{LOGMEM}[\,\mathit{Dout}[\,0\,]\,] & \mathbf{NLI} \\ \mathit{LOGRNG}[\,1\,] & \leftarrow & \mathit{LOGMEM}[\,\mathit{Dout}[\,1\,]\,] & \mathbf{NLI} \\ \mathit{LOGRNG}[\,2\,] & \leftarrow & \mathit{LOGMEM}[\,\mathit{Dout}[\,2\,]\,] & \mathbf{NLI} \\ \mathit{LOGRNG}[\,3\,] & \leftarrow & \mathit{LOGMEM}[\,\mathit{Dout}[\,3\,]\,] & \mathbf{NLI} \end{array} \right\} \text{ 4 cycles}$$

One occurrence of the DE9 opcode is sufficient to obtain, in one cycle, the values $RANGE$ (available in $Dout$) and $rE9$ for all four EMUs, whereas four instances of the NLI opcode are needed to generate the four $LOGRNG$, so that five cycles are required for initializing the normalization procedure for the four current positions. These five cycles can be scheduled in parallel with the local memory update process, as it will be shown in Subsection 6.4.2. Consequently, the total number of executed cycles does not increase.

6.2.4 Normalization pipeline

When the initialization stage is completed, each morphology feature $M_i[n]$ — computed by EMU n, with $n \in [0, \ldots, 3]$ — can be normalized and stored to the external memory by executing the following sequence of five operations:

$$\begin{array}{rcll}
DELTA & \leftarrow & M_i[\,n\,] - rE9[\,n\,] & \textbf{ME9} \\
LOGDELTA & \leftarrow & LOGMEM[\,DELTA\,] & \textbf{LOG} \\
LOGNORM & \leftarrow & LOGRNG[\,n\,] - LOGDELTA & \textbf{DR} \\
NORM & \leftarrow & INVLOGMEM[LOGNORM] & \textbf{ILOG} \\
SRAM[\,...,n,i\,] & \leftarrow & NORM\ ;\ SO \leftarrow SO + 1 & \textbf{ST}\,;\textbf{INC}
\end{array}$$

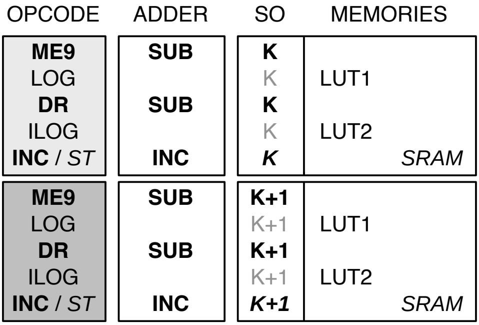
The aim of this sequence is to obtain the value of the normalized feature $\hat{M}_i''$ according to Equation 6.3. Firstly, the opcode ME9 (short form standing for $\mathrm{M_i}$ − E9) computes the numerator $DELTA = M_i - E9$, the logarithm of $DELTA$ being then retrieved from the first LUT by the LOG opcode. The third step, denoted by the opcode DR (short form standing for DELTA − RANGE), performs the subtraction in the logarithmic space. However, instead of computing $LOGDELTA - LOGRNG[\,n\,]$, the two operands are interchanged, so that the result $LOGNORM$ can never be negative[2]. The following operation, ILOG, consists in obtaining the inverse logarithm of $LOGNORM$. To compensate for the exchange of operands that occurs during the third step, the values inside the second LUT are stored in reverse order. At this point, the normalized feature $\hat{M}_i''$ is available in the NORM register, and can be stored to the external memory using the ST opcode of the LSU. During this last step, the opcode INC furthermore instructs the IDNORM unit to increment the SO register to prepare for the next iteration.

[2] Indeed, since $M_i - E9 \leqslant D9 - E9\ \forall i$, then $LOGDELTA \leqslant LOGRNG$.

Before each new dilation and erosion level is processed, a new instruction MO, for *morphology output*, is issued to the EMUs. In response, all four units copy the result of the corresponding dilation and erosion level, from the register banks pD and pE to their output registers Dout and Eout, so that they can be later read by the ME9 operation. Similarly to the other EMU instructions, the register banks are addressed using the control signal L, originating from the SE Block. To this effect, the size of SEMEM will need to be increased (see Subsection 6.3.4), so that it can store 9 additional parameters, corresponding to the parameters of MO for Level 0 to 8.

The sequence of operations discussed at the beginning of this subsection must be executed eight times to normalize one entire morphology level (four erosion and four dilation results to process), which would require a total of $8 \times 5 = 40$ cycles, much more than the 23 cycles found to be available to perform normalization in parallel with features extraction (see Subsection 6.2.1). It is nevertheless possible to pipeline the normalization sequence, in order to output a new result every three cycles instead of five, as shown in Figure 6.4, that illustrates two iterations of the considered sequence. Indeed, since the two operations LOG and ILOG do not make use of the adder in the IDNORM unit, each one can be scheduled in parallel with another operation.

The resulting pipelined sequencing of the normalization sequence is shown in Figure 6.5. During the second cycle, namely when the LOG operation is executed, the register SO gets incremented as denoted by the INC opcode, taking advantage of the fact that the adder is otherwise idle. This makes it possible to start normalizing the second feature during the fourth cycle, by scheduling a ME9 operation on the adder, which is not used by the ILOG operation pertaining to the normalization of the first feature. As a result, the adder is now constantly in use, and a new normal-



OPCODE	ADDER	SO	MEMORIES
ME9	**SUB**	**K**	
LOG		K	LUT1
DR	**SUB**	**K**	
ILOG		K	LUT2
INC / *ST*	**INC**	***K***	*SRAM*
ME9	**SUB**	**K+1**	
LOG		K+1	LUT1
DR	**SUB**	**K+1**	
ILOG		K+1	LUT2
INC / *ST*	**INC**	***K+1***	*SRAM*

Figure 6.4 : Sequence of the normalization operation.
The sequence of operations needed to normalize 2 features is depicted. The background color of the OPCODE columns differentiates between the 5 cycles concerning the first (light gray) and second feature (dark gray). Boldfaced text indicates that the opcode requires the adder (OPCODE column), and also which operation is performed (ADDER column), and the value read from the SO register (SO column). For the ST operation, italicized text indicates the value read from SO. The MEMORIES column indicates whether the SRAM, or the logarithm or inverse logarithm LUT, denoted LUT1 and LUT2, respectively, are accessed.

ized feature can be output every three cycles. It is nevertheless mandatory to keep a copy of the value held by SO before it is incremented, for use by the DR and ST operations for the first feature, since they occur after the second step, i.e. after SO gets incremented. To this effect, a new register, PSO — for *previous SO* — gets added to the IDNORM unit.

As the opcode ILOG always occurs together with the opcode ME9, see Figure 6.5, the former can be implicitly scheduled by the latter, so that the ILOG opcode can be discarded. Similarly, the LOG opcode is removed and replaced with NLIC, standing for *Normalization Logarithm Increment Copy.* In response to this new opcode, the IDNORM unit performs the logarithm operation that computes LOGDELTA, copies the value of SO to

PSO, and increments SO. The INC opcode is however not removed from the instruction set, since it is also needed outside of the normalization sequence, to update the address registers as explained in Subsection 5.5.6.2.

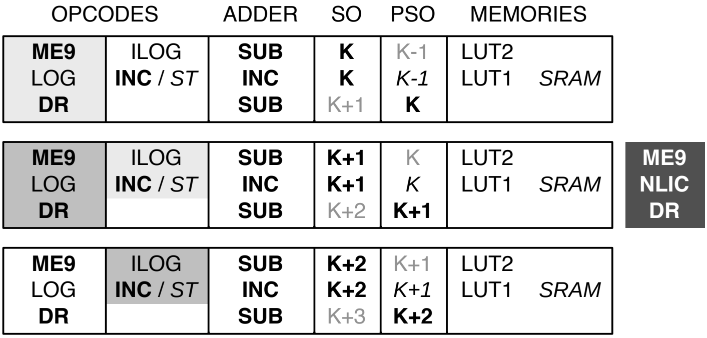
Since the pipeline now outputs one normalized value every three cycles, normalizing one morphology level for four locations in the image requires 24 cycles. Given that we previously determined in Section 3.2 that 23 cycles were available, only one extra cycle has to be added, instead of $40 - 23 = 17$ for the non-pipelined solution. Compared to the 22 cycles that are needed to extract the morphology features without normalization, as reported in Subsection 5.6.2, the net increase corresponds to 2 cycles per level, or 16 cycles for all 8 levels. Finally, when considering the whole image, the increase amounts to 65'536 cycles, representing less than 4% of the total number of cycles reported in Table 5.2, so that the impact of the normalization on the execution time of the algorithm has been rendered almost negligible.

6.3 Modified architecture

This section details the modifications brought to the architecture of the coprocessor, to its different components and to its instruction set, in order to implement features normalization following the scheme outlined in Section 6.2.

6.3.1 Elementary morphology unit

The following modifications were made to the design of the EMU. Firstly, the MC instruction was modified to store its results back



OPCODES		ADDER	SO	PSO	MEMORIES		
ME9	ILOG	**SUB**	**K**	K-1	LUT2		
LOG	**INC** / *ST*	**INC**	**K**	*K-1*	LUT1	*SRAM*	
DR		**SUB**	K+1	**K**			
ME9	ILOG	**SUB**	**K+1**	K	LUT2		**ME9**
LOG	**INC** / *ST*	**INC**	**K+1**	*K*	LUT1	*SRAM*	**NLIC**
DR		**SUB**	K+2	**K+1**			**DR**
ME9	ILOG	**SUB**	**K+2**	K+1	LUT2		
LOG	**INC** / *ST*	**INC**	**K+2**	*K+1*	LUT1	*SRAM*	
DR		**SUB**	K+3	**K+2**			

Figure 6.5 : Pipelined execution of the normalization step.
The same sequence of operation as in Figure 6.4 is depicted, using a pipelined implementation. For each cycle, represented by one row in the table, boldfaced characters indicate which operation uses the adder, as well as the register furnishing the control signals (either SO or PSO). Similarly, italicized text is used to denote the register used to construct the SRAM address needed by the ST operation. The three opcodes typeset in white over dark gray on the right replace the ones in the two OPCODES columns, except for ST since this opcode does not belong to the IDNORM unit.

to the partial results register banks:

$$\begin{array}{rcl} pD[\ L\ ]\ ,\ Dout & \leftarrow & MAX(\ pD[\ L\ ],\ Dout\ ) \\ pE[\ L\ ]\ ,\ Eout & \leftarrow & MIN(\ pE[\ L\ ],\ Eout\ ) \end{array}$$

The added complexity is marginal, as the comparators outputs were already connected to the register banks, so that partial results could be updated when an MM instruction is executed.

Once D9 and E9 are available in the output registers, a new instruction, DE9 is issued to instruct the dilation comparator to compute the normalization range, D9 - E9. At the same time, the four E9 values are preserved in registers inside the Morphology Block (see Subsection 6.3.2) as they are needed when normalizing (see Equation 6.2). Finally, this instruction also resets pD[9]

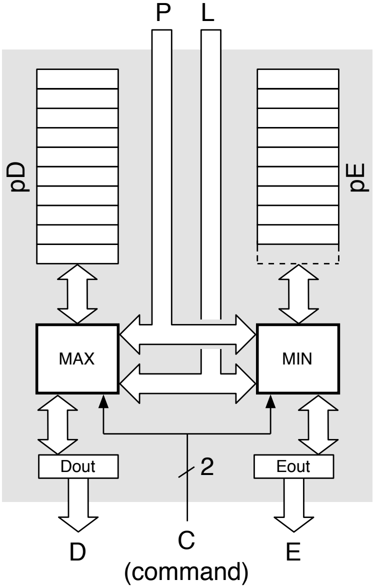


Figure 6.6 : Modified Elementary Morphology Unit.
Since for Level 0, the result — namely, the value of the original pixel — is identical for both the dilation and the erosion, register pE0 is superfluous.

and pE[9] to 0 and 255, respectively:

$$\begin{aligned} Dout &\leftarrow Dout - Eout \\ pD[\ 9\ ] &\leftarrow 0 \\ pE[\ 9\ ] &\leftarrow 255 \end{aligned}$$

An additional register was inserted in the dilation bank to store the value of the central pixel, as shown in Figure 6.6. This operation occurs when the unit performs partial differential morphology with L = 0, in addition to copying the pixel value to both output registers, to serve as the basis for the recursive computation of all dilation and erosion levels:

$$\begin{aligned} Dout &\leftarrow P \\ Eout &\leftarrow P \\ pD[\,0\,] &\leftarrow P \end{aligned}$$

Another new instruction MO — for *morphology output* — is introduced to copy the full erosion and dilation results from the registers banks to the output registers, so that they can be fed to the normalization pipeline. It also resets the two registers it reads to 0 or 255, so that partial erosion and dilation computations can begin for the specified level:

$$\begin{aligned} Dout &\leftarrow pD[\,L\,] \\ Eout &\leftarrow pE[\,L\,] \\ pD[\,L\,] &\leftarrow 0 \\ pE[\,L\,] &\leftarrow 255 \end{aligned}$$

The RM instruction still resets the L pipeline inside the Morphology Block and SEIDX, while the registers in the partial results are now reset by the MO instruction, one level at a time as previously exposed. Moreover, it also resets register SO — previously reset by the discarded RSO instruction — as well as register PSO in the IDNORM block.

As there are now three operations to delay — partial morphology, combine differential results and output features — the width of the signal C increased from 1 to 2 bits.

6.3.2 Morphology block

The modified Morphology Block, presented in Figure 6.7, includes four new registers to preserve the E9 values until normalization is complete. A multiplexer was also added so that the normalization unit (see Subsection 6.3.5) can select and read the register rE9 corresponding to the position being processed. The other output, previously connected to the external data bus and used to send extracted features to the external memory, is now connected to the normalization unit. Additionally, this connection is also used to retrieve the RANGE value from the Dout registers, after the DE9 operation has been performed by the EMUs. As noted in Subsection 6.3.1, the width of C increased and thus registers in the pipeline used to generate the delayed control signals for the EMUs are now also 2-bit wide.

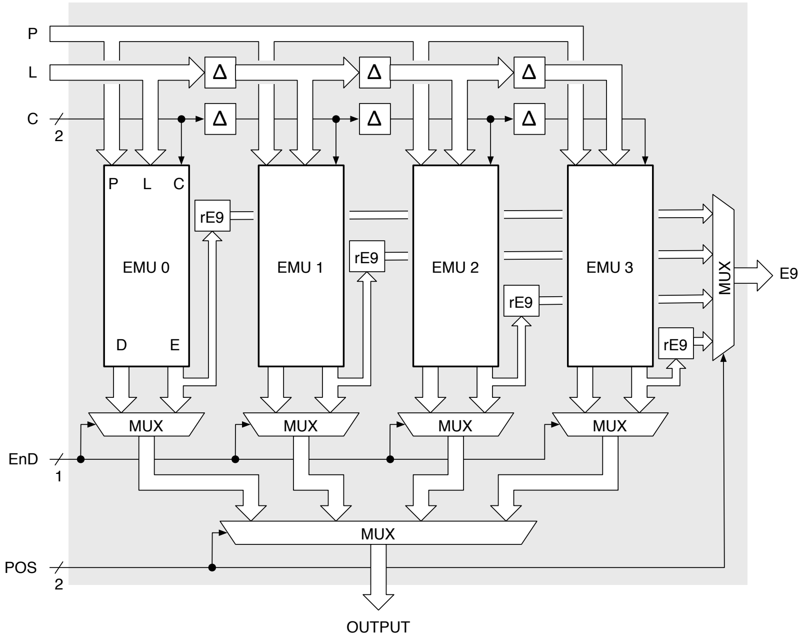


Figure 6.7 : Modified Morphology Block.

6.3.3 Local memory block

No changes were made to this block, with respect to the description given in Subsection 5.5.4.

6.3.4 Structuring elements block

The architecture of the SE Block is identical to the one described in Subsection 5.5.5. However, to accommodate the parameters for the new MO instruction that was introduced in Subsection 6.2.4, SEMEM now stores 380 4-bit words.

6.3.5 Normalization unit and addressing logic

While the MAS Block could be used as it is without modifications, the ID Block required extensive changes. In addition to being responsible for incrementing and decrementing the addresses used when loading data from — and storing results to — the external memory, it now also handles most operations required to normalize the extracted features. As already mentioned, its name was consequently changed to IDNORM.

6.3.5.1 Normalization unit

The structure of the Normalization Unit (IDNORM) is shown in Figure 6.8. In addition to the registers already found in ID Block, IDNORM contains PSO, the LOGRNG register bank, the registers belonging to the normalization pipeline as well as the LOGMEM and INVLOGMEM memories (see Section 6.2.). The address A furnished to LOGMEM is provided by a multiplexer steered by the control unit, which selects the values RANGE for NLI instructions and DELTA for NLIC instructions, respectively.

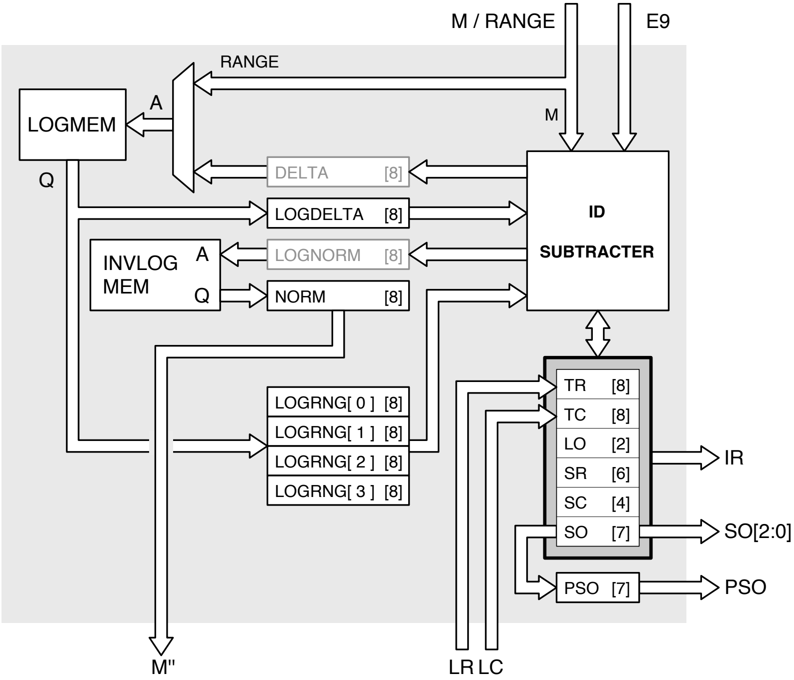


Figure 6.8 : Normalization Unit (IDNORM).

Registers DELTA and LOGNORM are drawn in gray to indicate that they are not actually needed. Indeed, both LOGMEM and INVLOGMEM are synchronous memories and as such, they latch the provided address internally. They can thus be directly connected to the output of the subtracter. However, both registers are implemented in the current design so that their value can be read for debugging purpose.

The first normalization operation is performed in response to the NLI instruction, and consists in computing LOGMEM[RANGE] and storing the result in one of the four registers in the bank LOGRNG. As explained in Subsection 6.2.3, this is executed for the four current positions, by sequentially reading the Dout register of each EMU. Priorly, the opcode DE9 was issued to the EMUs to request that the value RANGE = D9 – E9 be computed and placed in Dout.

NLI

$$LOGRNG[\ SO[1:0]\ ] \quad \leftarrow \quad LOGMEM[\ Dout[\ SO[1:0]\ ]\ ]$$

The three other new instructions, ME9, NLIC and DR, handle the normalization process, as explained in Subsection 6.2.4.

ME9

$$\begin{aligned} DELTA &\leftarrow OUTPUT[\ SO[2:0]\ ] - E9[\ SO[1:0]\ ] \\ NORM &\leftarrow INVLOGMEM[\ LOGNORM\ ] \end{aligned}$$

NLIC

$$\begin{aligned} LOGDELTA &\leftarrow LOGMEM[\ DELTA\ ] \\ PSO &\leftarrow SO \\ SO &\leftarrow SO + 1 \end{aligned}$$

DR

$$LOGNORM \quad \leftarrow \quad LOGRNG[\ PSO[1:0]\ ] - LOGDELTA$$

The signal OUTPUT used by the ME9 instruction originates from the Morphology Block (MB), see Subsection 6.3.2, and is used to read the results available in the EMUs, according to the value of the register SO. The two LSB, SO[1:0] are connected to the POS signal in Figure 6.7 and they determine which EMU is accessed, whereas S[2] is connected to the signal EnD and selects between the two output registers Dout and Eout. To retrieve the values stored in the Dout registers, the opcode NLI also sets POS to SO[1:0], but forces the signal EnD to 0.

6.3.5.2 External memory addressing unit

The external addressing unit, EXT ADDR, depicted in Figure 6.9, was not modified with respect to the one presented in Figure 5.21. However, it now relies on register PSO instead of SO for deriving the LEVEL and POS part of the store address, as explained in Subsection 6.2.4. Except for this difference, the load and store addresses are constructed as before (see Subsection 5.5.6.4 and Subsection 5.5.6.5).

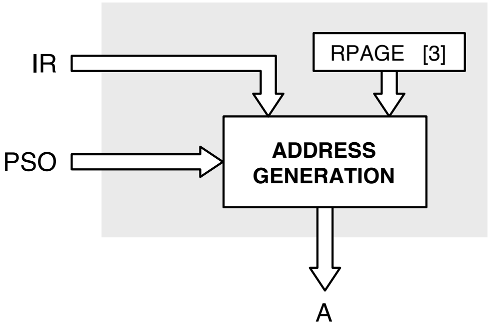


Figure 6.9 : Modified external addressing unit (EXT ADDR).

Whereas results were previously generated in the ascending order of dilation and erosion levels, starting with the original image, the sequence is now reversed. The new layout of the external memory, shown in Figure 6.10 follows the order in which results are normalized: starting with Level 8, descending down to Level 1 and finishing with the original values, the latter corresponding to Level 0.

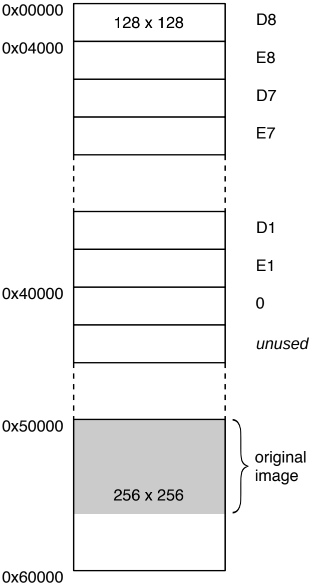


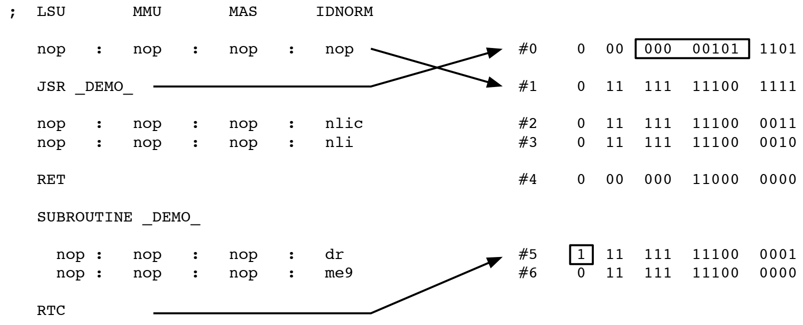
Figure 6.10 : Layout of normalized features in memory.

6.3.6 Subroutines support

In the baseline version of the program, the last results obtained were written to the external memory only after starting work on the next four positions, as explain in Subsection 5.5.6.5. In consequence, some control instructions responsible for updating registers used to compute external memory addresses had to be executed while morphology features were being extracted. As these control operations would depend on the direction of the displacement, it was not possible to replace the four morphology processing segments with a single subroutine. This is no longer the case with the normalized version: all external memory write operations occurring during features extraction deal with the same set of four positions.

Subroutine support in the coprocessor can be built on top of the loop mechanism with almost no additional hardware. Indeed, a call to a subroutine (denoted by the *jump to subroutine*, or JSR instruction) requires that the decoder be able to memorize the address of the following instruction, for use when reaching the *return to caller* instruction, or RTC, indicating the end of the subroutine. It must also be able to continue execution at an absolute memory address specified in the opcode. The first action is very similar to the result of a loop declaration, where the address of the current instruction is to be memorized in the stack. In loop declarations, the information retrieved from the opcode is used to initialize the loop counter, while it will specify the next program memory address to fetch in the case of a JSR instruction. The second action is identical to the behavior exhibited when a JUMP instruction is encountered at the end of a of a loop, except that the jump is always taken. The same stack of PC can be used to store addresses linked to loops declarations (address of the first instruction in the loop) and subroutines calls (address of the first instruction following the call), each register containing now a new one-bit flag employed by the control unit to determine whether a JMP corresponds to the end of a loop or to the RTC of a subroutine. Indeed, both are encoded using the same bit in the instruction word, as seen in Figure 6.14 and cannot be differentiated in the object code. As it is the case with loops, the JUMP flag is placed on the second to last instruction, so that the address of the next instruction to execute is known early enough to avoid fetching an instruction at an incorrect address. This requires that, similarly to loops, subroutines contain at least two real instructions (excluding RTC). Finally, jump instructions (whether they denote the end of loop or the end of subroutine) must always be separated by two regular instructions.

Figure 6.11 shows a sample call to subroutine. The source code — on the left — gets translated into binary object code by the assembler, which permutes the JSR with the preceding instruction. It also sets the JMP flags on the second to last instruction in the subroutine. Once the whole source code as been parsed a first time, a second pass is executed to resolve references to subroutine labels. At this time, the absolute address of the first instruction in the subroutine is placed into the dedicated field of the instruction word.



```
; LSU         MMU         MAS        IDNORM

  nop    :    nop    :    nop    :    nop                #0    0  00  000  00101  1101

  JSR _DEMO_                                             #1    0  11  111  11100  1111

  nop    :    nop    :    nop    :    nlic               #2    0  11  111  11100  0011
  nop    :    nop    :    nop    :    nli                #3    0  11  111  11100  0010

  RET                                                    #4    0  00  000  11000  0000

  SUBROUTINE _DEMO_

    nop :     nop    :    nop    :    dr                 #5    1  11  111  11100  0001
    nop :     nop    :    nop    :    me9                #6    0  11  111  11100  0000

  RTC
```

Figure 6.11 : Sample assembly code for calling a subroutine.

The execution of the sample code of Figure 6.11 is detailed in Figure 6.12. The PC indicates the address of the instruction being fetched, which will be executed during the following cycle. When instruction #0 is executed, the next value of PC as computed by the adder is stored in the first available register *jmp_addr* in the PC stack, while the associated flag *jmp_is_jsr* is set to indicate that *jmp_addr* is the return address of a subroutine, and not the top address of a loop. The actual value that the PC takes during the next cycle is *jsr_addr*, extracted from the instruction word. When instruction #5 is executed, the *is_jmp* signal is asserted. As *jmp_is_jsr* is true, the jump is treated as an RTC, and the PC is simply set to *jmp_addr*. In the other case where the jump denotes the end of a loop, it would have been handled as described in [1].

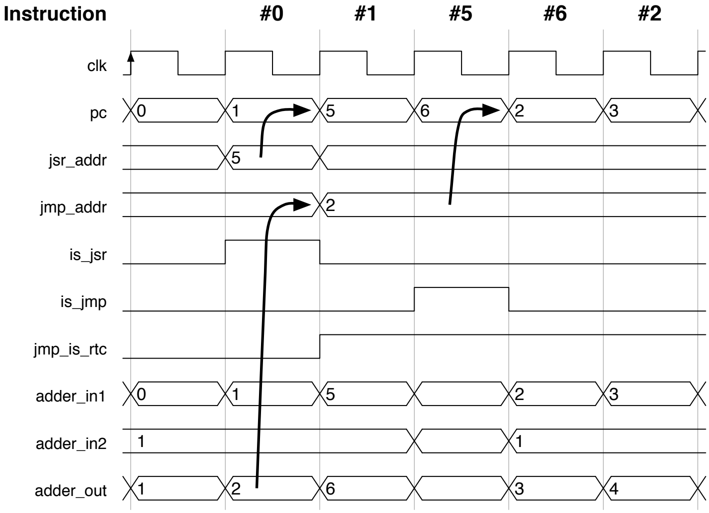


Figure 6.12 : Temporal execution of a call to a subroutine.

6.3.7 Instruction set

The new coprocessor instruction word, whose size is increased from 14 to 15 bits due to the additional opcodes introduced in the MMU field, is shown in Figure 6.13. The JSR calls are denoted by a dedicated opcode in the IDNORM field, while some of the remaining bits in the instruction word hold the absolute address of the subroutine. As the current program memory holds a maximum of 256 words, only 8 bits are reserved for storing this address. The same number of bits is used to store the COUNT field for loops declarations, allowing up to 256 iterations to be executed. The maximum number of iterations currently used amounts to 128 and corresponds to the size of the resulting images.

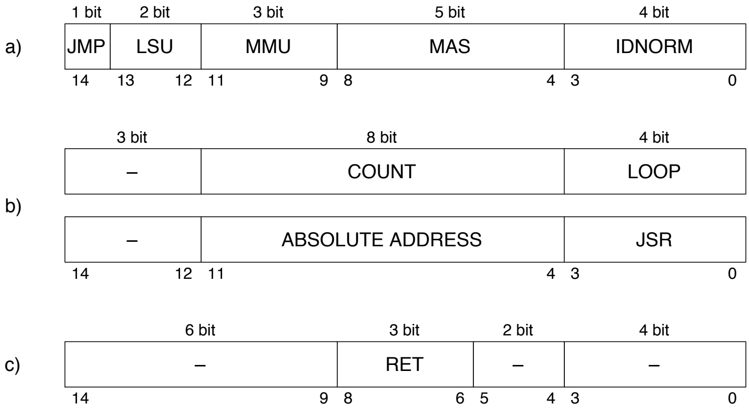


Figure 6.13 : Format of the normalized morphology coprocessor instruction word for a) regular instructions; b) loop declarations and subroutine calls; c) RET instruction.

The corresponding opcodes for each field are given in Figure 6.14 and Figure 6.15. It can be noted that RSO does no longer exist. Indeed, the task of resetting SO (and PSO) is now handled by the RM instruction, additionally to resetting the L pipeline in the Morphology Block and resetting SEIDX and SEOUT, as these two operations can now be executed simultaneously. As both CPY and RM schedule operations for this unit — copy LC and LR to TC and TR and reset SO and PSO, respectively — the IDNORM field must be set to NOP when either of theses opcodes appear in the instruction word. The simulator and the assembler would report an error if this condition is not respected. To accommodate the new normalization instructions without increasing the size of the field IDNORM, the DEC operation can no longer access all the registers in the unit, but is restricted to registers SC and SR only. As noted in Subsection 5.5.6.2, it was never necessary to decrement the other four registers in the first place, so that this limitation does not have any practical impact.

opcode	binary representation	description
JMP		
—	0	Regular instruction
JUMP	1	Indicates the second to last instruction in a loop or in a subroutine
LSU		
NOP	1 1	No operation
CPY	1 0	Copy LC to TC and LR to TR
LD	0 1	Load pixel from external memory
ST	0 0	Store one result to external memory
MMU		
NOP	1 1 1	No operation
—	1 1 0	*unused*
RM	1 0 1	Reset MORPHO BLOCK, SEIDX and SEOUT
DE9	1 0 0	Compute RANGE and reset pD9 and pE9
MO	0 1 1	Output results for normalization
MC	0 1 0	Combine differential morphology results
MM	0 0 1	Partial differential morphology, L ← SEREG
MN	0 0 0	Partial differential morphology, L ← IGNORE

Figure 6.14 : Opcodes of the normalized morphology coprocessor.

6.4 Coprocessor characteristics

In this section, the complete architecture of the normalized coprocessor is presented. The execution of some typical parts of the program code is detailed. Finally, some performance figures are provided, that were obtained through simulations using the

MAS

NOP	1 1 1	–	No operation
–	1 1 0	–	*unused*
S1	1 0 1	*R*	Subtract 1 from *R*, mod 418
A1	1 0 0	*R*	Add 1 to *R*, mod 418
S4	0 1 1	*R*	Subtract 4 from *R*, mod 418
A4	0 1 0	*R*	Add 4 to *R*, mod 418
S18	0 0 1	*R*	Subtract 18 from *R*, mod 418
A18	0 0 0	*R*	Add 18 to *R*, mod 418

R

0 0	LR
0 1	LC
1 0	LMW
1 1	LMR

IDNORM

NOP	1 1 1	1	No operation
RET	1 1 1	0	Terminate execution
JSR *A*	1 1 0	1	Jump to absolute address *A*
LOOP	1 1 0	0	Loop declaration
DEC	1 0 1	*S*	Decrement register *S*
INC	1 0 0	*S*	Increment register *S*
INC	0 1 1	*T*	Increment register *T*
INC	0 1 0	*U*	Increment register *U*
NLIC	0 0 1	1	Compute LOGDELTA
NLI	0 0 1	0	Compute LOGRANGE
DR	0 0 0	1	Compute LOGNORM
ME9	0 0 0	0	Compute DELTA
opcode	binary representation		description

S

0	SC
1	SR

T

0	TC
1	TR

U

0	SO
1	LO

Figure 6.15 : Opcodes of the normalized morphology coprocessor (cont'd).

tools discussed in Section 7.2.

6.4.1 Architecture overview

The complete architecture of the morphology coprocessor implementing the features normalization is presented in Figure 6.16. The control block, encompassing the fetch and decode unit as well as the program memory, is not shown here. Also missing are the communication interface with the MCU, the clock and reset signals as well as the buses allowing the MCU to access any register and memory in the coprocessor. Compared to the original architecture, shown in Figure 5.29, the major difference concerns the data path followed by the extracted features. Whereas the latter were directly transferred from the EMUs to the external memory, they are now fed to the IDNORM unit beforehand.

6.4.2 Sequencing

Similarly to Subsection 5.6.2, three portions of typical code executed by the normalized coprocessor are discussed. The first code snippet, shown in Figure 6.17 is similar to the code snippet of Figure 5.30. It also updates the content of the local memory, but additionally performs the initialization step of the normalization process, using instructions DE9 and NLI, as described in Subsection 6.2.3. Finally, the RM instruction is issued to reset the L and C pipelines in the Morphology Block, and SEIDX and SEOUT in the SE Block as explained in Subsection 6.3.1.

Figure 6.18 shows as snippet of code that performs morphology features extraction on row 1 to 8 of the SE set. Moreover, as explained in Subsection 6.2.1, previously extracted features residing in the register banks of the EMUs are normalized and stored to external memory. At the beginning of each row of the SE set,

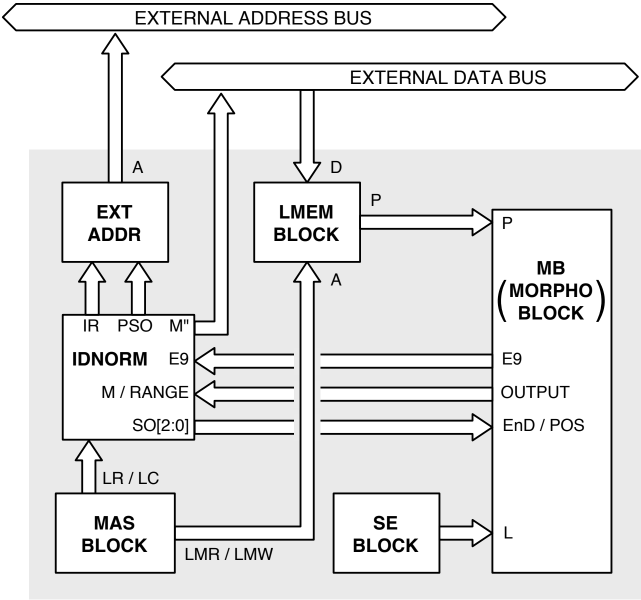


Figure 6.16 : Complete normalized morphology coprocessor.

```
;  LSU       MMU         MAS      IDNORM

   cpy   :   de9   :   nop       :  nop
   nop   :   nop   :   nop       :  nli
   nop   :   nop   :   a18 lmr   :  nli
   nop   :   nop   :   a4 lmr    :  nli
   nop   :   nop   :   inc lr    :  nli

   LOOP 11
     ld  :   nop   :   inc lmw   : inc tc
     ld  :   nop   :   inc lmw   : inc tc
   JUMP _11_

   nop   :   rm    :     nop     :  nop
```

Figure 6.17 : Snippet of code for updating the local memory and initializing the normalization.

the MO operation is issued to the EMUs so that the extracted features of the current level being normalized get transferred to the output registers. The total number of cycles to process one row of the SE set (including the cycle taken by the loop declaration) has risen to 25, since normalization requires 24 operation cycles for each row as explained in Subsection 6.2.4.

```
;     LSU       MMU        MAS      IDNORM

   _se_loop_row_1_8_
   LOOP 8

     nop    :   mo   :    nop     :  dr

     LOOP 6

       nop  :   mm   :  inc lmr   :  me9
        st  :   mm   :  inc lmr   :  nlic
       nop  :   mm   :  inc lmr   :  dr

     JUMP _6_

     nop    :   mm   :  inc lmr   :  me9
     st     :   mn   :  inc lmr   :  nlic
     nop    :   mn   :  inc lmr   :  dr
     nop    :   mn   :  inc lmr   :  me9
     st     :   nop  :    nop     :  nlic

   JUMP
```

Figure 6.18 : Snippet of code for extracting, normalizing and storing morphology features.

Finally, Figure 6.19 shows the code handling the bottom half of the SE set. It is basically the segment presented in Figure 5.32, however without the instructions handling the storage of the morphology features to the external memory.

The sequence of instructions is represented in graphic form in Figure 6.20. As was the case with the previous architecture, the SE set can still be changed by simply replacing the values stored inside SEMEM.

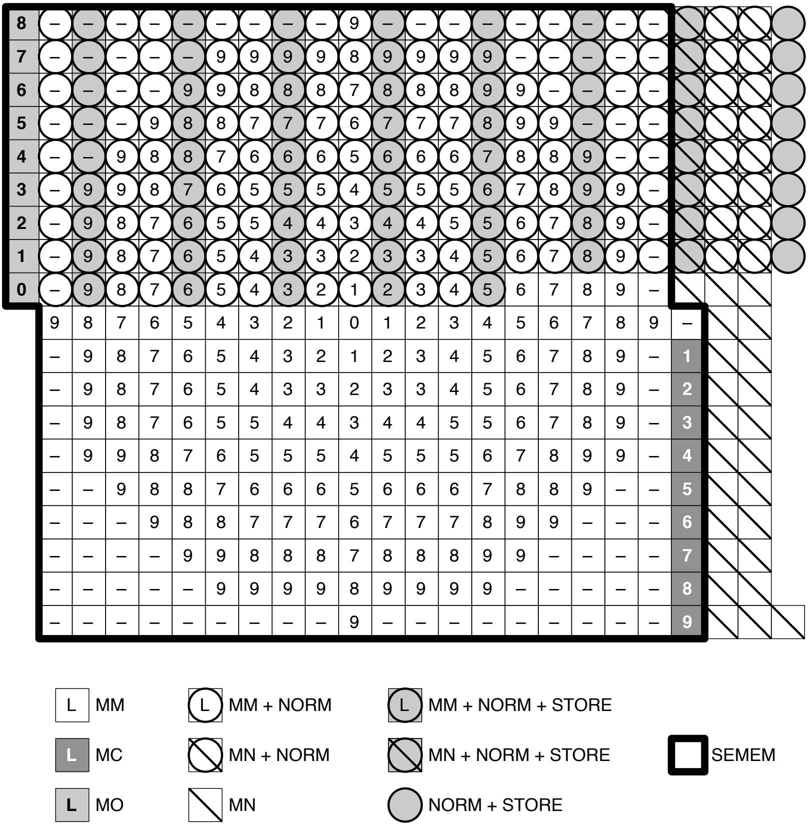
```
;     LSU      MMU         MAS       IDNORM

   _se_loop_row_10_19_
   LOOP 10

     LOOP 9
        nop :  mm    :  inc lmr   :  nop
        nop :  mm    :  inc lmr   :  nop
     JUMP

     nop    :  mm    :  inc lmr   :  nop

     nop    :  mc    :  inc lmr   :  nop
     nop    :  mn    :  inc lmr   :  nop
     nop    :  mn    :  inc lmr   :  nop

   JUMP
```

Figure 6.19 : Snippet of code to extract morphology features.

6.4.3 Performance

Table 6.1 presents the simulation results for the normalized coprocessor. The architecture functionality as well as the required number of cycles were furthermore validated based on the VHDL description of the hardware, running both on the FPGA demonstrator and using post-synthesis simulations of the ASIC design, as described in Section 7.3 and Section 7.4.

Comparing the results to the ones given in Table 5.2, it can be noted that the number of write instructions to the external memory has decreased. This is due to the reduced set of features containing only 17 values per position instead of 19. While the number of comparisons performed by the EMUs did not change, the latter now also perform one subtraction per position being processed, in response to the DE9 instruction. The simplifications in the program code that led to the handling of subroutines also caused the number of loop declarations to decrease. Whereas the number of instructions in the code is reduced by almost 40%,



8	–	–	–	–	–	–	–	–	–	9	–	–	–	–	–	–	–	–	–	
7	–	–	–	–	–	9	9	9	9	8	9	9	9	9	–	–	–	–	–	
6	–	–	–	–	9	9	8	8	8	7	8	8	8	9	9	–	–	–	–	
5	–	–	–	9	8	8	7	7	7	6	7	7	7	8	9	9	–	–	–	
4	–	–	9	8	8	7	6	6	6	5	6	6	6	7	8	8	9	–	–	
3	–	9	9	8	7	6	5	5	5	4	5	5	5	6	7	8	9	9	–	
2	–	9	8	7	6	5	5	4	4	3	4	4	5	5	6	7	8	9	–	
1	–	9	8	7	6	5	4	3	3	2	3	3	4	5	6	7	8	9	–	
0	–	9	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	9	–	
	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	–
	–	9	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	9	–	1
	–	9	8	7	6	5	4	3	3	2	3	3	4	5	6	7	8	9	–	2
	–	9	8	7	6	5	5	4	4	3	4	4	5	5	6	7	8	9	–	3
	–	9	9	8	7	6	5	5	5	4	5	5	5	6	7	8	9	9	–	4
	–	–	9	8	8	7	6	6	6	5	6	6	6	7	8	8	9	–	–	5
	–	–	–	9	8	8	7	7	7	6	7	7	7	8	9	9	–	–	–	6
	–	–	–	–	9	9	8	8	8	7	8	8	8	9	9	–	–	–	–	7
	–	–	–	–	–	9	9	9	9	8	9	9	9	9	–	–	–	–	–	8
	–	–	–	–	–	–	–	–	–	9	–	–	–	–	–	–	–	–	–	9

Figure 6.20 : Graphical sequencing of the features extraction algorithm executed on the morphology coprocessor. The actual value of the IGNORE symbol, denoted here by a dash, is 15.

Type of operation		**Nb of occurrences**
External memory read		92’182
External memory write		282’693
Local image memory read		1’720’511
Local image memory write		92’182
SE memory read		1’560’752
LOG memory read		294’984
INVLOG memory read		282’693
EMU comparisons (circular SE set)		8’552’448
EMU comparisons (maximum)		12’091’392
EMU subtractions		16’388
SEINC increments		1’560’752
MAS additions / subtractions		1’833’919
ID increments / decrements		957’257
Loop declarations		94’271
JSR calls		4’096
NOP cycles (all units idle)		8’211
NOP cycles (no JSR)		19
Total nb of executed cycles		**2’022’425**
”	*w/o JSR*	**2’010’137**
Static nb of instructions (program size)		**142**
”	*w/o JSR*	**235**

Table 6.1 : Performance of the normalized coprocessor.

the number of cycles increases by less than 1% when using subroutines. The total cost of each JSR call is three instructions: in addition to the cycle consumed by the instruction itself, the subroutine contains two NOP operations, so as to fulfill the constraints imposed on the location of the RTC symbol.

The repartition of the number of executed cycles among the various tasks performed by the coprocessor is given in Table 6.2. The numbers do not differ much from the previous architecture and reflect the slight increase of instructions needed to accommodate the normalization pipeline.

Task	**% of executed cycles**
Loop declarations	4.7 %
Calls to subroutines	0.2 %
Control and local memory refresh	5.0 %
Features extraction and results stores	90.1 %

Table 6.2 : Percentage of execution cycles devoted to loop declarations, calls to subroutine and to the two main tasks performed by the coprocessor.

As seen in Table 6.3, while it remains stable for the other components, the duty cycle of the adder in the IDNORM (formerly ID Block) unit did more than double. Unlike before where it was only employed when accessing the external memory, it is now also employed while normalizing the features, which corresponds to approximately half the processing time devoted to morphology features extraction.

Component	Duty cycle
Comparators in EMUs (circular SE set)	53.1 %
Comparators in EMUs (maximum)	74.9 %
Modulo Adder-Subtracter (MAS)	91.1 %
IDNORM adder (ID)	47.6 %
SE Block incrementer (SEINC)	77.2 %

Table 6.3 : Duty cycle of the components of the coprocessor.

6.5 Conclusion

This chapter introduced a modified coprocessor that expands the architecture that was presented in Chapter 5. In addition to extracting features, the new coprocessor also performs the normalization step proposed in Subsection 3.4.3. By reusing existing hardware, it limits the additional resources that need to be implemented. It also preserves the flexibility that makes the SE set easily modifiable. Moreover, by executing the additional operations in parallel with the existing processing, it can process one complete image with less than 3% additional cycles with respect to the original solution. Although 4'096 bit of additional memory is required to implement the division, the total increase is reduced by 40% once the reduced memory footprint of the software program is taken into account.

In addition to being validated using the tools discussed in Section 7.2, this coprocessor was furthermore described in VHDL and synthesized both for FPGA and ASIC, as discussed in Chapter 7. The FPGA version, which also incorporates the EGM coprocessor detailed in [1], was used to produce a real-time demonstrator for the face verification system presented in Section 5.2.

Chapter 7

Architecture Validation and Demonstrator

7.1 Introduction

In Chapter 5, a coprocessor dedicated to low-power mathematical morphology features extraction was presented, that was subsequently enhanced in Chapter 6 to additionally handle features normalization. The present chapter describes the design process that was followed to develop and validate these two hardware architectures. It starts by discussing the high-level implementation of the algorithm, in Section 7.2, that furthermore covers the selection of the architecture as well as the tools developed to simulate the coprocessors and to elaborate their software programs. An FPGA-based demonstrator, whose twofold purpose is firstly to validate the VHDL description of the coprocessor incorporating features normalization and, secondly, to enable real-time execution of the complete face verification process, is then presented in Section 7.3. Finally, the resulting silicon die area occupied by the morphology coprocessor in a 0.18 µm ASIC technology is reported, along with the power consumption obtained through simulations carried out on the synthesized design.

7.2 Hardware and software co-design

This section reports how the two coprocessors, detailed in Chapter 5 and Chapter 6, respectively, were derived from the morphology features extraction algorithms introduced in Chapter 3, and how their respective software programs were written and tested.

7.2.1 High-level software implementation in C

In the course of their development, algorithms are usually implemented firstly in environments based on high-level languages, such as MATLAB or C, that typically offer a great deal of flexibility and provide facilities to optimize, debug and analyze various implementation variants. At this stage, all computations are performed using high-precision arithmetic, using for instance the *double* floating-point type in case the C language is being employed. Indeed, processors found in modern workstations are equipped with specialized units capable of performing operations on high-precision floating-point numbers at the same speed as their integer unit counterparts. When ultimately the goal is to target an embedded platform that is more limited performance-wise, such as a fixed-point Digital Signal Processor (DSP) or a dedicated custom processor implemented in an ASIC (Application-Specific Integrated Circuit), it is necessary to ensure that the algorithm still performs satisfactorily with reduced precision. In the case of the morphology features extraction algorithm considered in this work, this step was not necessary since all operations are carried out on 8-bit integer numbers. Regarding the version of the algorithm that incorporates features normalization, simulations were performed to verify that the implementation of the division by means of a subtraction in 8-bit logarithmic space does not degrade the accuracy of the face authentication procedure, as reported in Subsection 6.2.2.

7.2.2 Architecture selection

Both the description of the algorithms given in Chapter 4 and the corresponding implementations in the environment discussed in Subsection 7.2.1 are purely sequential, i.e. they consist in a series of operations executed one at a time in a predetermined order. The major benefit of a hardware architecture on the other hand is its ability to perform several operations at the same time, thanks to multiple processing units working in parallel. To help selecting the most adequate architecture to implement a given algorithm, it is therefore necessary to analyze beforehand the considered sequence of operations, in view of determining where and how parallelism can be introduced. The first step consists in identifying sections of the algorithm that either can, or cannot be performed concurrently, either because one section relies on results generated by another, or because they both require access to the same hardware resources. Indeed, processing units such as arithmetic logic units (ALU) or comparators can usually be replicated without much impact if such need is established, whereas other components such as shared memories, used for instance to exchange data between processing units, are harder to duplicate without affecting the data flow.

Moreover, operations must be carefully scheduled to avoid situations where multiple accesses are attempted simultaneously to a shared resource. Whereas in a fixed logic architecture, such scheduling can be achieved, it is not the case with programmable systems, where the sequence of operations is software-based and therefore subject to change. Consequently, it is often easier to serialize the tasks that could lead to conflicting resource usage. This is the case for instance with the two tasks that access the memory LMEM, either to update its content or to retrieve the values of the pixels to process, as explained in Subsection 5.4.2.

The exploration of the various possible approaches leading to a parallel architecture was carried out using the C language, and was based on a framework developed in the course of a previous project [125]. Here, a clock cycle is decomposed into two successive stages. Firstly, actual operations to execute in parallel are performed sequentially, since the C language does not support parallelism, but all the variables that emulate storage resources, such as registers or memories, are restricted to read operations and cannot be modified. Instead, all write operations are directed towards temporary storage variables, to accurately mimic the behavior of a synchronous architecture in which all registers are updated simultaneously at the end of the clock cycle. Moreover, variables that represent storage resources cannot be accessed directly, but only through so-called *accessors*, namely functions that, in addition to implementing the read or write operation, also perform several verifications. For instance, accessors make sure that the source and destination registers are indeed connected to the units that try to access them, and that the values of the supplied parameters — e.g. an immediate constant — are valid. All results generated during one clock cycle are transferred from the temporary to the actual storage resources in the course of the second stage, which emulates the clock edge triggering the update of the registers in the actual circuit.

Performing this exploratory process in the C language offers much more flexibility than in VHDL for instance. Indeed, some parts of the code can be directly reused from the high-level implementation, giving the opportunity to rework the algorithm "piece by piece" while preserving the ability to execute it in full at any point in time during the development cycle. This is very useful to ensure that the functionality is maintained throughout the whole optimization process. For both variants of the morphology coprocessor, this step resulted in a detailed description of the hardware architecture to implement being laid out, including the size of all required registers and memories, the kinds of processing units needed, as well as the exact number of cycles consumed to perform the complete algorithm. Even though it would have been possible to use this description as the basis for the VHDL implementation, the architectures were firstly ported to the C++ simulator discussed in the next section, to enable both a more formal validation and the finalization of their respective assembly software programs.

7.2.3 Software development

The C++ simulator was adapted from the tool originally elaborated during the development of the EGM coprocessor, discussed in Section 6.2.5 in [121], to add support for the operators that are specific to the morphology features extraction algorithm. In addition to providing a true register transfer level (RTL) description of the architecture, the simulator was also employed to develop and debug the software program. The provided facilities include step by step execution, display and manipulation of the values in any register or memory, as well as collection of statistics concerning the activity of each unit. The software program that the coprocessor has to execute is written in assembly language, using the instruction set and syntax presented in Subsection 5.5.7 and in Subsection 6.3.7. The text file containing the program is firstly fed to a scanner, or tokenizer, implemented using the popular tool Flex [126], which recognizes lexical patterns described by a set of regular expressions (regexps). The output is then sent to GNU Bison [127], a language parser that generates executable C or C++ code according to a provided description of the language syntax. The latter, called a grammar, is written in Backus-Naur formalism, or BNF [128], a widely used method of describing the syntax of computer languages. The C++ simulator furthermore serves as an assembler that generates the binary object code to be uploaded to the program memory of the coprocessor.

7.2.4 VHDL description

Following the steps discussed so far, the morphology coprocessor incorporating features normalization as introduced in Chapter 6, was described in VHDL and synthesized to be implemented on the FPGA demonstrator that will be presented in Section 7.3. Once the VHDL was fully tested and debugged, the ASIC synthesis of the design was carried out, targeting the UMC 0.18 µm VLSI process. The achieved figures for silicon die area and power consumption will be reported in Section 7.4.

7.3 FPGA demonstrator

The goal of the FPGA[1] demonstrator is twofold. Firstly, it enables real-time validation of the proposed architecture, which cannot be achieved when running VHDL simulations. Secondly, it can be used to perform the complete face verification process, from the image acquisition up to the obtainment of the matching distance. To this effect, the demonstrator, shown in Figure 7.1, is constituted of several components that are briefly described in Subsection 7.3.1 to Subsection 7.3.5.

[1] Field-programmable gate array.

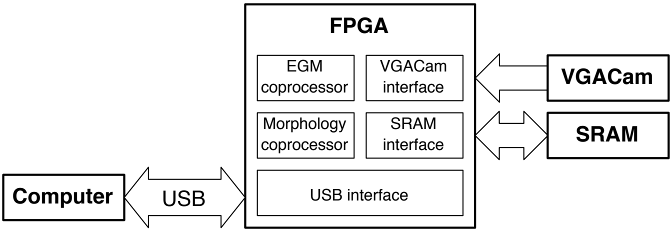


Figure 7.1 : FPGA-based demonstrator.

7.3.1 VGACam image sensor

VGACam is a low-power CMOS color image sensor [129], which features a resolution of 648 × 488 pixels at 10 bits per pixel[2]. The sensor is covered by a layer of colored filters, each pixel being either red, green or blue, following an arrangement known as the Bayer pattern [130] in which there are twice as much green pixels as there are red and blue ones, respectively. Since the morphology features extraction algorithm operates on monochrome images, only one pixel out of four is considered. Indeed, green pixels are retrieved and used directly, in place of the true luminance, avoiding the need for computing the latter out of the red, green and blue channels. The resulting image resolution therefore corresponds to 324×244 pixels, from which an array of 146 rows of 256 pixels gets extracted to be stored in the SRAM, according to the layout depicted in Figure 5.22.

7.3.2 FPGA and SRAM boards

The Constellation board [131] manufactured by Nova Engineering supports an APEX 20K600-E3 FPGA from Altera Corpo-

[2] However, only 8 bits per pixel are used in the demonstrator.

ration, which incorporates the morphology and matching coprocessors, as well as the interfaces for the SRAM, the VGACam image sensor and the USB communication. Additionally, the board also includes an USB port and controller, which can be used to interact with the FPGA from an application running on a computer. The synthesized design is uploaded to the FPGA through the USB connection, to be stored either temporarily or permanently. In the first case, the design is only retained while power is supplied to the device. This mode is used for debugging and testing purpose, but once a stable version is available, the second mode becomes useful, where the design gets written to a dedicated embedded Flash memory. Subsequently, the FPGA will automatically configure itself when powered up. Naturally, the design can be replaced at any time by simply reprogramming the Flash memory. The FPGA primary clock signal runs at 40 MHz and is furnished by an on-board oscillator. Since the amount of memory embedded in the FPGA is too limited to accommodate a complete image, an expansion board supporting four static random-access memories (SRAM) is connected to the Nova board.

7.3.3 Computer

The computer, or PC, plays a dual role in the demonstrator. Firstly, it provides the user interface, in the form of a Windows application that communicates with the FPGA board through the USB connection, and secondly, it emulates the functionality of the master control unit (MCU) discussed in Section 5.2. The Windows application manages a simple database, where the reference features extracted from the clients at enroll time are stored. Thereafter, acquired images can be matched against templates in the database in order to be authenticated. Instead of relying on VGACam sensor for acquiring images, the latter can alternately be read from files, which is useful to formally verify that the results furnished by the demonstrator are correct.

7.3.4 USB interface

The Windows application on the PC and the FPGA communicate over a USB connection. The USB controller on the Nova board exchanges data with the USB interface in the VHDL design through 64 8-bit registers [131], that are accessed using four elementary functions provided by Nova, namely: read register, write register, burst read, and burst write. The burst mode speeds up the transfer of large blocks of data, by transmitting only the base address, that gets subsequently incremented after each read or write operation in the sequence. To enable access to the entire set of registers in the design, as well as to the local and external memories, wider addresses are needed. To this effect, the PC application decomposes the address and data to be transmitted into several parts, each one being written in a different 8-bit register in the USB interface. A write operation therefore consists of up to 7 parts, namely: three 8-bit write operations to transmit a 24-bit address, three 8-bit write operations to send the 24-bit data, and one 8-bit write operation to trigger the execution of the requested action. In the FPGA, the 24-bit addresses and data are reconstructed by simple concatenation. Since the latency of the USB communication channel is rather high, a cache mechanism is implemented on the PC to maintain a copy of the content of the USB interface registers. Consequently, the only write operations that actually need to be performed are those that cause the values in the interface registers to change.

7.3.5 SRAM interface

The SRAM memories available on the expansion board are asynchronous and therefore require that the data and address buses be stable for the whole duration of the write pulse. Indeed, if the address is not settled at the start of the write cycle, data might get written at random memory locations, potentially overwriting useful information. Since the coprocessor is fully synchronous, the address and data buses, as well as the write pulse, are guaranteed to be stable on the rising clock edge only. Indeed, in case a synchronous memory was used, the latter would sample the buses and the signal at this instant. To work around this problem, the address and data buses are latched for the duration of two cycles, whereas the write pulse gets delayed by half a cycle. As a consequence, write operations targeting the external memory can only occur every two cycles. This is an issue neither for the EGM coprocessor, which does not write to the external memory, nor for the morphology coprocessor, since the latter only generates results every three cycles, as seen in Subsection 6.4.2. No latching or delaying needs to be applied for read operations, which is fortunate since that would prevent the coprocessors from loading one pixel every cycle. Indeed, glitches appearing on the address bus, even after the start of the read pulse, are not problematic since the results of the read operations are always stored in a register, which means that only the value that is available on the data bus when the clock edge occurs is taken into consideration.

7.3.6 Results of the FPGA synthesis

Synthesis for the FPGA is performed using firstly Physical Compiler from Synopsys, that converts the VHDL description of the design into an Electronic Design Interchange Format (EDIF) netlist of standard FPGA cells. The netlist is then loaded into Quartus II from Altera for the final compilation stages, which involves resources allocation (consisting in physically allocating logic elements, or LEs, and embedded memories), routing (interconnecting the LEs, the FPGA ports and the embedded RAM cells) and finally, timing analysis. During this last step, the tool examines the propagation delays of the signals and verifies that all timing requirements are met. The output of the compilation consists in a Tabular Text File (TTF) that contains the design to upload to the FPGA. The resources required by each component are reported, in Table 7.1 for the number of LEs and in Table 7.2 for the number of memory bits. Note that the provided figures are those of the coprocessor already encompassing the Ethernet interface that will be introduced in Subsection 7.3.8. Moreover, numbers for all embedded memories are reported separately in Table 7.2, even those for LMEM and SEMEM, although the latter are actually part of LMEM Block and SE Block, respectively, as shown in Figure 5.15 and Figure 5.16. Finally, PROGMEM denotes the program memory, which holds the instructions that the coprocessor has to execute.

The complete design, including both the morphology and the EGM coprocessors, occupies less than half the total number of logic cells, and one quarter of the available memory. It would therefore be possible to incorporate additional functionalities to the demonstrator, such as a micro-controller core for handling part of the processing currently taking place on the PC. The critical path delay amounts to 66.1 ns, which translates to a slack margin of 33.9 ns since the demonstrator runs at 10 MHz, a clock that is derived from the 40 MHz signal furnished by the Nova board. The maximal operating frequency is therefore 15.1 MHz, which could be achieved, provided a more complex clock division logic gets implemented.

Components	**Flip-flops**	**Area in LEs**	**Percentage of FPGA**
Clock division	2	2	0.01
USB / Ethernet interface	188	609	2.5
VGACam interface	138	257	1.05
SRAM interface	2	155	0.64
EGM coprocessor	2'503	6'740	27.7
Morphology coprocessor	**1'052**	**3'304**	**13.6**
MCU interface	32	245	1.01
Control unit	132	408	1.68
EXT ADDR	3	25	0.1
LMEM Block	8	55	0.23
IDNORM	108	455	1.88
MAS Block	34	136	0.56
SE Block	13	58	0.24
Morpho Block	722	1'922	7.9
Control	*18*	*207*	*0.85*
EMU 0	*176*	*419*	*1.72*
EMU 1	*176*	*433*	*1.78*
EMU 2	*176*	*441*	*1.81*
EMU 3	*176*	*422*	*1.74*
Total	**3'885**	**11'067**	**45.5**

Table 7.1 : Occupation of the FPGA logic elements (LEs).

Components	Size in bits	Percentage of FPGA
EGM coprocessor	64’768	20.8
Morphology coprocessor	**12’816**	**4.1**
PROGMEM	3’840	1.23
LMEM	3’344	1.07
LOGMEM	2’048	0.66
INVLOGMEM	2’048	0.66
SEMEM	1’536	0.49
Total	**77’584**	**24.9**

Table 7.2 : Occupation of the FPGA memory bits.

7.3.7 USB demonstrator performance

The USB interface available on the Nova Constellation board, discussed in Subsection 7.3.4, incurs a high latency affecting the communication with the computer. When a face verification is performed, the matching coprocessor needs to interact with the MCU, which is emulated by the PC running the Windows application. The configuration of the graphs used by the simplex downhill matching algorithm are for instance computed on the MCU, as explained in Section 6.3.3 in [1]. Consequently, during this stage, data need to be exchanged quite frequently between the PC and FPGA. Whereas the actual processing time required by the matching coprocessor to handle one face verification is approximately 0.6 second, the overhead of the communication over the USB bus increases this value to approximately 8 seconds, in spite of the caching mechanism mentioned in Subsection 7.3.4. It must be stressed that this problem will not affect the actual System-on-Chip system, where the communication between the MCU and the coprocessors is carried out over dedicated data and address buses (see Figure 5.1), incurring virtually no latency and enabling one face verification to be processed in less than one second. Nevertheless, the demonstrator was enhanced, to try and offer a user experience that more closely resembles that of the actual system.

To lower the communication overhead, two actions can be taken. The first one would consist in reducing the time needed to exchange data, which requires a different, faster interface. Indeed, both the USB driver on the computer and the USB controller on the FPGA board are closed components that cannot be modified. Among the possible replacement interfaces, the following were considered: USB 2.0, Bluetooth and Ethernet. The original idea was to add an external board supporting the new connector or transmitter, as well as a hardware module dedicated to handling the physical layer of the chosen protocol, whereas the higher level of the communication stack would be implemented on the FPGA. However, all these protocols are rather complex, so that replacing the communication channel would represent a substantial development effort. The second approach would consist in seeking to reduce the amount of data being exchanged between the FPGA and the computer, by performing some of the tasks currently handled by the Windows application, directly on the demonstrator board instead. To this effect, a micro-controller core would need to be implemented on the FPGA.

7.3.8 Ethernet interface

Eventually, we settled for a third option, consisting in using an external board called *Ethernut*, that combines the advantages of the two solutions discussed above. Indeed, as it will be seen in the next subsection, it both replaces the USB connection with Ethernet and provides a CPU capable of performing certain operations without requiring data to be exchanged with the PC. The enhanced demonstrator is depicted in Figure 7.2.

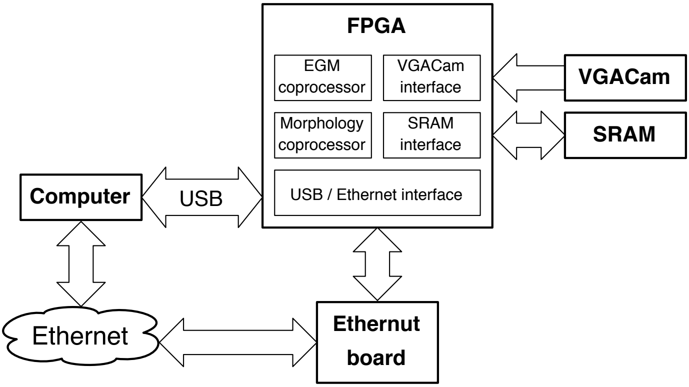


Figure 7.2 : FPGA-based demonstrator with Ethernut board.

7.3.8.1 Ethernut board

Ethernut [132] is an open source hardware and software project, aiming at providing Ethernet connectivity to embedded devices. The hardware consists in a family of standalone boards[3] equipped with a 8-bit micro-controller, namely an Atmel ATMega 128 with embedded SRAM and Flash memory, running at 14.7 MHz, as well as a full duplex Ethernet controller. The board also supports a serial interface, a JTAG connector and an expansion port. On the software side, Ethernut runs the NutOS, a very simple real-time operating system (RTOS) targeted at embedded systems, that offers libraries for TCP/IP network software development, in addition to a fairly complete implementation of the C standard libraries. Provided demonstration applications comprise a DHCP client, a simple telnet server, as well as a basic HTTP

[3] The board used in our demonstrator is Version 2.0 Revision A.

server. Starting with Version 3.0, Ethernut is now built around a much faster and more powerful CPU, namely a 32-bit ARM7 TDMI processor running at 73.7 MHz.

The expansion port of the Ethernut board enables direct access to the ATMega address and data buses, as well as to the associated memory read and write control signals. As a result, it was possible to implement communication with the FPGA by simply mapping the existing interface registers into an otherwise unused part of the ATMega address space. To this effect, the existing USB interface on the FPGA was modified to detect and respond to read or write operations performed by the ATMega that target addresses falling into the mapped memory space. The addition of this new functionality was carried out in a way that did not affect the existing support for theses operations when performed over the USB interface. As a matter of fact, it is even possible to use the Ethernet and the USB connections concurrently, provided that no temporal overlap between two operations initiated on a different interface occurs[4].

The ATMega on the Ethernut board operates at 14.7 MHz, and the pulses on the read and write signals are only guaranteed to last as least 67 ns. However, the processor can be configured to increase the duration of the pulses when accessing some specified memory ranges. We make use of this facility to obtain pulses lasting 134 ns when targeting the FPGA memory space, so that the read and write signals can be sampled using the 10 MHz clock. Furthermore, a simple electrical interface is required to connect the FPGA to the Ethernut board. Indeed, the former uses 3.3 V I/O signals, whereas the latter mostly employs 5 V, the standard I/O level of the ATMega CPU[5]. Unidirectional ad-

[4] Currently, operating both interfaces at the same time furthermore requires disabling the cache mechanism discussed in Subsection 7.3.4, although it would be easy to modify the PC application to remove this limitation.

[5] Some address bits are however latched by a CPLD and use 3.3 V.

dress signals output by the ATMega can be directly connected to the FPGA, since the latter can be configured to accept 5 V inputs [133]. The ATMega however does not tolerate 3.3 V inputs, so conversion needs to be applied on the data bus. To this effect, a bi-directional level-shifter is employed. By default, it converts the 5 V outputs of the ATMega to 3.3 V inputs suitable for the FPGA. When the Ethernut interface on the latter has determined that the ATMega is attempting to perform a read operation from the FPGA address space, it reverses the direction of the level shifter, so that it now converts the 3.3 V signals driven by the FPGA to the 5 V level that the ATMega requires.

7.3.8.2 Communication protocol

The communication between the PC and the Ethernut board is based on the telnet protocol [134]. A simple telnet client was included in the Windows application, so that the latter can open a TCP/IP connection to the Ethernut board when starting up. Currently, the Ethernut board uses a static TCP/IP address that must be coherent with the subnet it is connected to, and known to the Windows application. If the board gets moved to a different subnet, its firmware will therefore need to be updated to become aware of the new address. Furthermore, the latter must also be communicated to the Windows application. This could be avoided by configuring the board to rely on its built-in DHCP client to dynamically obtain an IP address from the network when booting. A simple broadcast-based discovery scheme could then be included in the Windows application, to enable it to automatically find out the address attributed to the board.

Using the overloading mechanism offered by the C++ language, the four elementary USB functions described in Subsection 7.3.4 can be transparently replaced with functions that perform the same tasks by using the telnet protocol instead. Each command consists of a short text string, starting with a single 8-bit ASCII character, followed by the address — and data in the case of a write operation — written in decimal. For burst operations, the number of bytes that need to be transmitted is included in the text command, whereas the data block itself gets transmitted in binary form. On the Ethernut board, the standard functions of the C library are used to parse and interpret the command string, which is then translated into a read or write operation that targets the memory space mapped to the FPGA.

At this point, the demonstrator can be fully operated over Ethernet, without requiring that the USB connection be available. The gain is however marginal, the observed latency of the Ethernet connection being almost as high as that of the USB[6]. To effectively improve the performance of the demonstrator, the number of transactions that take place between the FPGA and the PC must still be reduced. This was achieved by introducing 10 higher-level commands, which are interpreted by the ATMega on the Ethernut board, to generate sequences of elementary commands to be sent to the FPGA. For instance, a single command replaces the seven operations that were required to write a 24-bit value, as explained in Subsection 7.3.4. Following this step, the time required to perform one face verification is 3 seconds, representing one third of the duration measured with USB.

Finally, we implemented one supplementary high-level command, that triggers and manages the execution of the following sequence of operations, that is part of the simplex downhill matching algorithm mentioned in Subsection 3.3.5:

1. Upload a new graph configuration from the PC to the FPGA.

[6] Note however that no caching mechanism was implemented for Ethernet.

2. Set the program counter (PC) of the EGM coprocessor to a known, constant address.
3. Start the EGM coprocessor by sending it a *run* command.
4. Poll the coprocessor until it indicates that it has completed the execution of the requested task.
5. Retrieve the resulting graph score from the FPGA and transmit it to the PC.

The new command simply uploads the graph generated by the PC to the Ethernut board, the latter taking care of interacting with the FPGA to execute the four remaining operations in the list above. Therefore, only one Ethernet transaction suffices for the whole sequence. After this last command was introduced, the duration of the face verification process was reduced to 1.6 second. The communication overhead was therefore reduced by a factor of 10, dropping from 8 seconds to 0.8 second, approximately. The 15 commands available when using the Ethernet interface are summarized in Figure 7.3.

7.4 ASIC Synthesis

Following the validation of the hardware architecture carried out on the FPGA demonstrator, the design was synthesized in a low-power ASIC technology. Indeed, to verify that the proposed solution is effectively suitable for mobile, battery-powered devices, it is necessary to accurately assess the energy required for operating the coprocessor. This section discusses the ASIC synthesis flow, and reports the achieved results in terms of both silicon area and power consumption. Synthesis was carried out using memories and standard cells from a library supplied by Virtual

code	*address*	*data*	*count*	
r	8			Read interface register at *address*
w	8	8		Write interface register at *address*
R	8		16	Burst read *count* bytes at *address*
W	8		16	Burst write *count* bytes at *address*
s	24			Read 8 bits at *address*
t	24			Read 16 bits at *address*
u	24			Read 24 bits at *address*
x	24	8		Write 8 bits at *address*
y	24	16		Write 16 bits at *address*
z	24	24		Write 24 bits at *address*
S	24		16	Burst read *count* bytes [8-bit data]
T	24		16	Burst read *count* bytes [16-bit data]
X	24		16	Burst write *count* bytes [8-bit data]
Y	24		16	Burst write *count* bytes [16-bit data]
A				Perform simplex downhill step

Figure 7.3 : Commands available with the Ethernet interface. For each command, the table indicates the size (in bits) of the operands. Burst write operations are followed by *count* bytes of binary data, containing either 8-bit or 16-bit words. The command **A** is followed by 128 bytes, corresponding to the coordinates of the 64 nodes of the graph to upload.

Silicon Technologies. The target technology was the 0.18 µm, 6 metal layers process from UMC [135].

7.4.1 Synthesis flow

During the initial ASIC synthesis stage, the VHDL description is translated into a netlist of standard cells using Physical Compiler from Synopsys, which is instructed to optimize the design so as to minimize the total area. The netlist is output in VHDL format, alongside a Standard Delay Format (SDF) file containing the complete timing specifications of the circuit, such as setup and hold constraints, as well as the propagation delays over the nets and through the standard cells. Both files are then loaded into ModelSim from Mentor Graphics, where simulations are performed using a temporal resolution of 1 ps. This very low number ensures that all transitions incurred by glitches are taken into account when determining the activity — or toggle rate — of each net in the coprocessor. To verify the functionality of the circuit, the test bench employed consists in the actual morphology features extraction software program, executed over an entire face image[7]. The activity gets written to a Standard Activity Interchange Format (SAIF) file, which gets loaded into Physical Compiler so as to annotate each net in the design with the toggle rate determined by ModelSim. Further optimizations are then carried out, aiming now at minimizing the dynamic power consumption of the circuit. The resulting updated VHDL netlist and SDF files are again fed to ModelSim, and a second simulation is carried out, using the same test bench as before. This leads to the generation of a new SAIF file, that Physical Compiler uses to update the design annotation. Finally, the power analysis and reporting tool of Synopsys is invoked to compute the power consumption achieved by the final design.

7.4.2 Area of the synthesized ASIC

The area of the resulting ASIC for the morphology coprocessor incorporating features normalization is detailed in Table 7.3. The total surface of the circuit amounts to 0.33 mm^2, i.e. more than half of the area being occupied by the five embedded memories. The four Elementary Morphology Units (EMUs), that implement

[7] This operation takes approximately seven hours when carried out on a SunBlade 2000 Ultra-Sparc III workstation running at 1.2 GHz.

the core processing functionality, account for nearly 60% of the area devoted to standard cells. One can note that, even though they were synthesized from multiple instances of the same VHDL entity description, the four EMUs are not implemented using the exact same set of standard cells. Indeed, the compiler was not restricted in replicating the same synthesized design for all four instances, but was on the contrary given the opportunity to independently optimize each one. For instance, it can select among multiple variants of the same standard cell, differing only by the size of their output buffer, depending on the capacitive load that must be driven by each particular instance.

7.4.3 Estimation of the power consumption

As described in Subsection 7.4.1, simulations carried out on the synthesized design consisted in performing features extraction over a entire face image, to precisely and realistically determine the activity of each node in the netlist. Using this information, together with the values of the node capacitances that were obtained during synthesis, the power analysis tool can accurately derive the average power dissipated by the circuit when processing an image. The achieved results for the dynamic power consumption are reported in Table 7.4. A distinction is made between the internal power, consisting in the power consumed by the node transitions that occur inside the standard cells, and the switching power, representing the power dissipated by the transitions of the nets that interconnect the cells. The overall static power consumption, or leakage power, is much smaller than the dynamic one and amounts to 2.3 µW only.

Components	**Area (in µm^2)**			**Percentage**	
	combinational	non-combinational	**Total**		w/o mem
MCU interface	3’167	2’984	6’151	1.86	4.31
Control unit	8’902	9’731	18’633	5.65	13.0
EXT ADDR	564	223	787	0.24	0.55
LMEM Block	1’052	593	1’645	0.5	1.15
IDNORM	9’438	7’664	1’7102	5.19	12.0
MAS Block	3’868	2’348	6’216	1.88	4.36
SE Block	1’597	777	2’374	0.72	1.66
PROGMEM		48’430	48’430	14.7	
LMEM		46’431	46’431	14.1	
LOGMEM		32’641	32’641	9.9	
INVLOGMEM		32’641	32’641	9.9	
SEMEM		26’827	26’827	8.14	
Morpho Block	37’414	52’493	89’907	27.2	62.9
Control	4’829	1’297	6’126	1.86	4.29
EMU 0	8’154	12’799	20’953	6.35	14.7
EMU 1	8’148	12’799	20’947	6.35	14.7
EMU 2	8’145	12’799	20’944	6.35	14.7
EMU 3	8’138	12’799	20’937	6.35	14.7
Glue logic	32		32	0.01	0.02
Total	**66’034**	**263’783**	**329’817**		
Memories			**186’970**	**56.7**	
Cells			**142’847**	**43.3**	

Table 7.3 : Area of the synthesized ASIC.

7.5 ASIC Synthesis with clock gating

In a synchronous design, the clock tree is usually directly connected to every flip-flop. Due to the switching capacitance of the gates located behind the clock pin, power is consumed each time a clock edge occurs, even when the flip-flop value does not change. As seen in Table 7.4, the contribution of the clock tree amounts to approximately one third of the total switching power. If unnecessary transitions could be avoided in the flip-flops, this number would be reduced. Let us consider the case of the erosion

Components	**Power (in mW)**			**Percentage**	
	internal power	switching power	**Total**		w/o mem
MCU interface	0.024	0.000	0.024	0.68	2.03
Control unit	0.112	0.043	0.155	4.4	13.1
EXT ADDR	0.002	0.001	0.003	0.09	0.25
LMEM Block	0.009	0.014	0.023	0.65	1.95
IDNORM	0.080	0.019	0.099	2.81	8.38
MAS Block	0.031	0.009	0.040	1.14	3.39
SE Block	0.015	0.004	0.019	0.55	1.61
PROGMEM	0.546		0.546	15.5	
LMEM	0.423		0.423	12.0	
LOGMEM	0.504		0.504	14.3	
INVLOGMEM	0.504		0.504	14.3	
SEMEM	0.363		0.363	10.3	
Morpho Block	0.563	0.138	0.701	19.9	59.4
Control	0.026	0.013	0.039	1.11	3.30
EMU 0	0.134	0.032	0.166	4.71	14.1
EMU 1	0.134	0.029	0.163	4.63	13.8
EMU 2	0.134	0.032	0.166	4.71	14.1
EMU 3	0.135	0.032	0.167	4.74	14.1
Clock tree		0.117	0.117	3.32	9.91
Total	**3.176**	**0.345**	**3.521**		
Memories			**2.340**	**66.4**	
Cells			**1.181**	**33.6**	

Table 7.4 : Dynamic power consumption.

and dilation register banks in the EMUs. At any time, at most one 8-bit register per bank can get modified, which amounts to a total of 8 registers, or 64 flip-flops, for the four EMUs. Since there are a total of 4 dilation and 4 erosion register banks in the EMUs, counting 10 and 9 registers each, respectively, the number of flip-flops globally amounts to 608. Consequently, at least 554 flip-flops — corresponding to more than half the total number in the entire circuit — are clocked during every cycle, even though their content is guaranteed not to change.

To avoid this situation, a technique called *clock gating* can be employed. The flip-flop update logic is usually implemented by using an enable condition to command a multiplexer connected to the flip-flop input. In the case where the flip-flop does not need to be updated, its current data output is selected to be the next data input value. With clock gating, the multiplexer is removed and the signal carrying the new data value goes directly to the flip-flop input. The clock pin of the flip-flop is connected to the output of an AND gate, whose inputs are the clock signal and the latched enable signal that was formerly steering the multiplexer. The latch, which is transparent when the clock signal is low, is inserted to protect from glitches that can occur on the enable signal. If the flip-flop needs no updating, the clock edge will be canceled by the AND gate, so that no transition occurs on the clock pin. When applied to a single flip-flop, this technique does not make much difference. Indeed, the transitions on the clock input gate of the flip-flop now occurs on the AND gate instead. The impact can however become significant when considering a register, since the latter needs only one latch and one AND gate for all its flip-flops.

7.5.1 Clock gating insertion

Physical Compiler is in theory capable of automatically inserting clock gating during the synthesis process. Registers that share a common enable condition are indeed correctly identified, and all their clock pins get connected to the output of one single AND gate. However, the library cell chosen to implement the gate has an input capacitance that is actually almost seven times higher than the one on the clock pin of a flip-flop. In the considered design, most registers are constituted of 8 flip-flops, so that even if none of the latter were ever updated, the gain would be marginal. Moreover, when the value held by a register needs to be modified, the energy consumed almost doubles with respect to the situation without clock gating, since transitions now occur both on the AND gate and on the clock pin of the flip-flops. As a matter of fact, the reported power consumption of the circuit is effectively higher with clock-gating. Fortunately, the library provides dedicated standard cells for the clock gating logic, that Physical Compiler can be instructed to use instead. Since the input capacitance of these cells is slightly inferior to that of the clock pin on the flip-flops, the power consumption gets indeed reduced in this case, as reported in the following section.

7.5.2 Power consumption

As seen in Table 7.5, the total dynamic power consumption of the cells decreases by 32% when clock gating is applied. The most significant reductions are observed in the elements that contain many registers that are rarely updated, such as the MCU interface[8], the IDNORM and the EMUs. Conversely, the situation slightly worsens for the SE BLOCK, whose two registers are modified during most of the executed cycles. Finally, the power dissipated by the clock tree is reduced by 45%, and the static power consumption now amounts to 2.2 µW.

7.5.3 Area of the circuit

Enabling clock gating during synthesis also reduces the area occupied by the library cells by more than 7.6%, as it can be seen in Table 7.6. As explained in Subsection 7.5.1, the modifications made by the synthesizer consisted in replacing the input multiplexers of the flip-flops in the clock-gated registers with a

[8] Used to manage read and write requests made by the MCU, so that the registers it contains are never modified when the coprocessor is running.

Components	**Power (in mW)**			**Percentage**	
	internal power	switching power	**Total**		w/o mem
MCU interface	0.003	0.000	0.003	0.1	0.38
Control unit	0.088	0.044	0.132	4.21	16.6
EXT ADDR	0.002	0.001	0.003	0.1	0.38
LMEM Block	0.009	0.010	0.019	0.61	2.39
IDNORM	0.047	0.018	0.065	2.07	8.19
MAS Block	0.024	0.009	0.033	1.05	4.16
SE Block	0.015	0.006	0.021	0.67	2.64
PROGMEM	0.546		0.546	17.4	
LMEM	0.423		0.423	13.5	
LOGMEM	0.504		0.504	16.1	
INVLOGMEM	0.504		0.504	16.1	
SEMEM	0.363		0.363	11.6	
Morpho Block	0.311	0.143	0.454	14.5	57.2
Control	0.023	0.012	0.035	1.12	4.41
EMU 0	0.071	0.036	0.107	3.41	13.5
EMU 1	0.073	0.032	0.105	3.35	13.2
EMU 2	0.072	0.033	0.105	3.35	13.2
EMU 3	0.072	0.030	0.102	3.25	12.9
Clock tree		0.064	0.064	2.04	8.06
Total	**2.839**	**0.295**	**3.134**		
Memories			**2.340**	**74.7**	
Cells			**0.794**	**25.3**	

Table 7.5 : Dynamic power consumption with clock-gating.

single dedicated cell. Since the latter drives the clock pin of all the flip-flops in the register, it must be equipped with a large output buffer, and it is expected to be rather large. Based on the cell sizes provided in the library datasheet, applying clock gating should result in a larger circuit. Indeed, for one EMU for instance, the inserted clock gating cells increase the size by 6%, whereas the multiplexers that were removed represent less than 1% of the area. The explanation for the size reduction is actually furnished by the fact that different flip-flops are used. Indeed, whereas flip-flops in the original circuit were equipped with an

Components	**Area (in µm^2)**			**Difference**
	combi-national	non-combi-national	**Total**	compared to non-gated (%)
MCU interface	3’135	2’345	5’480	-10.9
Control unit	9’357	8’506	17’863	-4.1
EXT ADDR	561	212	773	-1.8
LMEM	1’058	487	1’545	-6.1
IDNORM	9’806	6’541	16’347	-4.4
MAS Block	3’890	2’009	5’899	-5.1
SE Block	1’597	761	2’358	-0.7
PROGMEM		48’430	48’430	
LOCALMEM		46’431	46’431	
LOGMEM		32’641	32’641	
INVLOGMEM		32’641	32’641	
SEMEM		26’827	26’827	
Morpho Block	37’578	44’109	81’687	-9.1
Control	4’838	1’249	6’087	-0.6
EMU 0	8’183	10’715	18’898	-9.8
EMU 1	8’180	10’715	18’895	-9.8
EMU 2	8’187	10’715	18’902	-9.7
EMU 3	8’190	10’715	18’905	-9.7
Glue logic	32		32	0
Total	**67’014**	**251’940**	**318’954**	**-3.3**
Memories			**186’970**	
Cells			**131’984**	**-7.6**

Table 7.6 : Area of the ASIC with clock-gating.

enable pin, the ones used in conjunction with clock-gating lack such input, and are on average 25% smaller. Since it was verified that the enable pins were not used anyway in the former case, it is surprising that Physical Compiler chose to use such cells.

7.5.4 Power consumption at lower supply voltage

The reported slack margin of the synthesized circuit is 87 ns when operating at 10 MHz at the nominal 1.8 V supply voltage.

Therefore, the coprocessor would still function correctly when clocked at higher frequencies, up to 75 MHz. In such a case, the power consumption would increase linearly with the operating frequency, but it would be compensated by the fact that the time interval required to complete a task would get smaller. Consequently, the amount of energy needed to process one face would not change. The slack margin can also be exploited to reduce the power consumption, by operating the circuit at a lower supply voltage, as detailed below.

According to the *alpha-power law* [136, 137], reducing the supply voltage V_{dd} increases the propagation delay τ, which reduces the slack margin:

$$\tau \;=\; k\,\frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{7.1}$$

V_{th} denotes the threshold voltage and α represents the velocity saturation of the charge carriers, which depends on the technology. k is a constant for a particular design in a given technology. In the case of the UMC 0.18 μm process, the values are as follows:

$$V_{dd} \;=\; 1.8\text{ V} \qquad V_{th} \;=\; 0.5\text{ V} \qquad \alpha \;\cong\; 1.5$$

Since the slack margin is 87 ns, the largest cumulated gate delay in our design can be estimated to 13 ns, so that we can rewrite Equation 7.1 to obtain the value of k:

$$k \;=\; \tau\,\frac{(V_{dd} - V_{th})^{\alpha}}{V_{dd}} \;=\; 10.7 \tag{7.2}$$

In order to determine the lowest supply voltage that can be applied, we then let $\tau = 100$ ns in Equation 7.1 and solve it numerically. The achieved result is 0.67 V, which corresponds to a reduction factor of 2.69 with respect to the nominal V_{dd} value. It is well known that the dynamic power consumption is proportional to the square of the supply voltage, which means that the power would then be reduced by a factor of 7.23. Theoretically, if the circuit was running at 10 MHz with a supply voltage of 0.67 V, the power consumption of the cells would therefore amount to 0.11 mW only.

The memories provided by Virtual Silicon for the target process are designed for high-speed applications running at several hundreds megahertz. Indeed, their access time is inferior to 3 ns, which means that when they operate at 10 MHz, the slack margin is higher than 97 ns. To estimate the power consumption that could be achieved with more suitable components, we rely on one SRAM, reported in [138] and whose characteristics are recalled in Table 7.7. Since our architecture was designed so that the memories are directly connected to registers, the specified access time is sufficiently fast.

Technology	**Memory size**	**Supply voltage**	**Access time**	**Current**
0.25 µm	4096 × 16 bits	0.8 V	65 ns	9 µA/MHz

Table 7.7 : Characteristics of a low-power SRAM.

Using the alpha-power law, we can determine that the standard cells in the morphology coprocessor would consume 0.16 mW with a 0.8 V supply voltage and a 10 MHz operating frequency. Under these conditions, the power consumption of the 64-kbit SRAM above would amount to 72 µW. Since the total size of all five memories in the circuit represents less than 13 kbits, we can conservatively estimate that their combined power consumption would be less than 40 µW when implemented using the same technique as the SRAM in Table 7.7. Consequently, with appropriately designed memories, the power dissipated by the complete coprocessor and all the embedded memories at 10 MHz can be estimated to be less than 0.2 mW when the supply voltage gets reduced to 0.8 V.

If the supply voltage is lowered to 0.8 V, the power consumption of the standard cells in the EGM coprocessor would amount to 0.63 mW. Since the total size of the embedded memories is five times higher in the EGM coprocessor than in the morphology coprocessor, the power consumption incurred by the memories in the later can be roughly estimated to represent 200 µW. The total power consumed by the EGM coprocessor, including the memories, therefore amounts to ca 0.83 mW. Considering the execution time needed to carry out features extraction and graph matching, namely 0.2 sec and 0.66 sec, respectively, we can calculate the energy required to perform each operation, namely 0.04 mJ and 0.55 mJ, respectively. The two coprocessors therefore consume less than 0.6 mJ to perform one face verification.

7.6 Conclusion

In this chapter, the various steps related to the design and the validation of the hardware architecture presented in Chapter 5 and Chapter 6 were discussed. Two versions of an FPGA-based demonstrator system, employed to validate one of the proposed solutions, were described. Finally, the results obtained after the morphology coprocessor supporting the features normalization step was synthesized in a 0.18 µm ASIC technology are reported, in terms of silicon die area and power consumption. Since the slack margin is very high at the considered operating frequency, an estimation of the power consumption that could be achieved by reducing the supply voltage and using low-power memories, is provided. In such conditions, the coprocessor would offer the same computational power and would therefore still need 0.2 second to process one image, yet its power consumption would decrease to approximately 0.2 mW.

It can be noted that further savings in power consumption could probably be achieved by using advanced optimization techniques. For instance, it is sometimes interesting to recode the signals transmitted by the buses, to reduce the activity of the latter by minimizing the average number of transitions occurring between two consecutive states. The resulting gains are however typically much smaller than those that can be achieved at the algorithmic and system levels, which were the levels considered in this project.

Chapter 8

Conclusion

8.1 Summary and recall of contributions

Following the apparition in the course of the last few years of new generations of mobile devices, that provide many additional functionalities beside vocal communication, the need for enhanced access control security grew stronger. In this context, biometric authentication based on the face modality provides an interesting solution, combining low intrusiveness with the faculty of reusing the image sensor already providing digital camera functionality. However, the deployment of face verification on mobile devices is confronted with several challenges in order to fulfill the constraints of such an environment. The research activities that were presented in this PhD report mainly focused on two of these aspects. Firstly, uncontrolled image acquisition conditions require algorithms that are robust to perturbations such as varying illumination or pose. Secondly, the computational complexity must be sufficiently low, so as to remain compatible with battery-powered applications, calling for dedicated hardware architectures that are highly optimized for specific tasks.

The following contributions were reported in this PhD thesis. Firstly, a normalization technique for mathematical morphology-based features extraction was proposed in Chapter 3, bringing the face authentication performance of the Elastic Graph Matching algorithm to the same level that gets achieved with Gabor-based features, while incurring only a small increase in computational complexity. Another contribution was given in Chapter 4, where further algorithmic enhancements were introduced, that yield slightly improved results. Other contributions in Chapter 4 concerned the assessment of the robustness of the Morphological Elastic Graph Matching (MEGM) algorithm under various degraded conditions, as well as the evaluation of several heuristic approaches to enhance its robustness in such cases. In Chapter 5, another contribution was made, namely the description of the hardware architecture of a coprocessor dedicated to mathematical morphology features extraction for a low-power face authentication System-on-Chip. An enhanced version implementing an additional features normalization step was then introduced in Chapter 6. Finally, a twofold contribution can be identified in Chapter 7, in the form of an FPGA-based demonstrator on one hand, enabling real-time validation of the proposed solution, and of the synthesis of the corresponding ASIC on the other, furnishing accurate silicon die area and power consumption figures.

8.2 Other state-of-the-art solutions

In this section, the most significant academic research results and commercial products recently disclosed in the field of face recognition targeting low-power mobile applications are presented. Only the cases where the biometric algorithms are completely executed on the mobile device itself are considered. No solution equivalent to the hardware-based approach discussed in this report could be identified, and so the discussed systems are mostly software-based. In many cases, the algorithm is executed on a variant of the fifth generation of the ARM 32-bit processor architecture, known as ARMv5 [139] and commonly found in current PDAs and high-end mobile phones such as smartphones.

8.2.1 Academic work

In the framework of the SecurePhone project [140], a prototype platform called the *SecurePhone PDA* was designed, by supplementing a commercially available mobile phone[1] with specific software modules and a customized SIM card. One of the goal of the project is to enable multi-modal biometric authentication, based on voice, face and signature to be carried out on the device. The approach followed for the face modality is described in [141]. It is based on the Discrete Wavelet Transform (DWT), and the discussed implementation can process 10 images per second on the embarked XScale[2] PXA263 processor that runs at 400 MHz and consumes 500 mW. The energy required to perform one face verification thus amounts to 50 mJ. The system is not very accurate, the achieved EER — measured over a custom face database acquired with the phone built-in camera — reaching 28%.

A solution based on eigenfaces (see Subsection 3.2.2) is reported in [142]. The highly optimized software implementation needs 5.2 seconds to perform one authentication when running on a 400 MHz XScale PXA255 processor, which consumes 400 mW. Hence, the amount of energy required for one authentication represents 2 J. The achieved EER, estimated from the provided ROC curve obtained with the FERET database [143], lies between 7%

[1] The chosen model bears the model name Qtek 2020 in European markets.

[2] XScale is the name of the Intel implementation of the ARMv5 architecture, that was sold to Marvell Technology Group in 2006.

and 8%. Using similar optimization techniques, an alternate solution running on the same processor but based on Elastic Bunch Graph Matching (EBGM) is proposed in [144]. The approach followed is that of Wiskott et al. [57], where the graph nodes are located over facial landmarks and the features extraction is performed with Gabor filters. To reduce the computational complexity, lookup-tables are used, however the required memory size is not specified. One face authentication can be performed in 1.3 second, which corresponds to 520 mJ. The reported EER ranges from 5% to 20%, depending on the database set employed.

Finally, a hardware approach is proposed in [145], where the biometric algorithm is based on eigenfaces (see Subsection 3.2.2) and executed on the Xtensa reconfigurable processor architecture [146] running at 100 MHz. A dedicated coprocessor was added to the system, to accelerate the image enhancement stage, since it was found that the latter represents more than 85% of the total processing time[3]. The optimized image enhancement step is reported to consume 28.3 mJ per image when the coprocessor is implemented in a 0.18 µm technology, but no figures are provided for the complete system. Performing one face verification takes 1.4 second, whereas an EER of 6.3% is achieved when evaluating the solution using sets of the FERET database where the faces are frontally illuminated.

8.2.2 Commercial devices and solutions

In 2006, two mobile phones with face verification capabilities were commercially released in Japan. The user's manual of each device stresses that great care should be taken to acquire images where the face is evenly and brightly lit to avoid false rejections.

[3] This stage is in particular responsible for precisely scaling and aligning the faces, as required by eigenfaces-based methods.

Moreover, both devices require the user to setup an alternate identification method, such as a secret code, to enable access to the phone when the biometric authentication does not succeed.

The first device, the Vodafone 904SH [147] manufactured by Sharp Electronics, was introduced in April 2006 by Vodafone Japan[4]. It uses a face recognition technology developed by the Japanese company OKI and called FSE (Face Sensing Engine), that can perform one authentication in 200 ms on an ARM9 processor running at 200 MHz. Since the latter consumes 160 mW when integrated in a 0.18 µm process, the corresponding amount of energy can be estimated to 32 mJ for one face authentication.

No information is available regarding the solution employed by the second mobile phone, the FOMA P903i [148] released in November 2006 by NTT DoCoMo. The device is manufactured by Panasonic and needs one second to process one face, albeit the processor type and operating frequency are unknown.

Beside customer products such as cellular phones, face verification was also implemented on devices employed by law enforcement agencies. However, in most mobile solutions, the device merely handles the acquisition of the face images, and transmits them to a remote server where the actual biometric operations are performed. Nonetheless, the HIIDE (Handheld Interagency Identity Detection Equipment) Series 4 commercialized by SecuriMetrics [149] constitutes a complete standalone solution. The device, which is rugged and weighs over 1.3 kg, is equipped with an AMD Geode 533 processor running at 400 MHz and consuming 1.1 W, which enables biometric face verification to be carried out onboard. However, no indication is given regarding the duration of the operation, so that it is not possible to estimate the amount of energy needed to authenticate one individual.

[4] The company was subsequently sold and became Softbank Mobile.

Finally, Omron, another Japanese company, also announced and demonstrated a face authentication solution targeted at mobile phones in 2005. According to the published information, processing one image requires one second when running on an ARM9 processor operating at 266 MHz, which corresponds to more than 200 mJ being consumed per face verification if a 0.18 µm technology is assumed.

When considering solely the energy consumed by the processor that executes the biometric authentication algorithm — ignoring the energy needed for related operations such as acquiring the image or storing it — the best result is achieved by OKI, whose solution requires 32 mJ to verify one face on a processor integrated in 0.18 µm. In comparison, post-synthesis simulations showed that the hardware solution discussed in Chapter 7 only needs ca 6 mJ in a similar technology when operating at the nominal 1.8 V supply voltage, and less than 0.6 mJ if the latter is reduced to 0.8 V. Moreover, the System-on-Chip structure enables further power consumption savings by strongly reducing the number of signals that must be driven off-chip. Dedicated hardware solutions such as the one proposed in this report therefore remain significantly more efficient energy-wise then software-based approaches, even when the latter are implemented on processors specifically designed for mobile embedded systems.

8.3 Perspectives

At the time of writing this report, the major challenge that remains to be surmounted before face recognition can become an efficient biometric modality on mobile devices is robustness to changes in image acquisition conditions, particularly in the case of varying illumination. To alleviate theses problems, many approaches now rely on 3D information, exploited in addition to regular 2D images, in view of modeling and compensating nonlinear effects such as shadows. Even though newer generations of 3D cameras are smaller, cheaper and less power hungry, they are still far from satisfying the constraints of mobile devices such as cellular phones. Therefore, it still makes sense to pursue research on methods based purely on 2D images, either by looking for features extraction techniques that are more invariant to environmental conditions, or by applying supplementary preprocessing steps to enhance the robustness of the algorithm. By resorting to highly optimized hardware architectures dedicated to specific tasks, huge savings in energy consumption can be achieved, thereby enabling additional operations to be implemented and executed on the device itself while still fulfilling the power consumption constraints of mobile applications.

With respect to the contributions presented in this report, several future research activities can be envisioned. Firstly, as mentioned in Subsection 3.4.4, other kinds of features could be experimented in conjunction with the Elastic Graph Matching algorithm, such as the DCT-phase that has proved to be very effective for matching images. Further experiments should also be carried out to assess the performance of the MEGM algorithm under degraded conditions, such as those discussed in Chapter 4, considering for instance the combined influence of several kinds of perturbations jointly affecting the acquired face images. Finally, the morphology and the graph matching coprocessors, discussed in this report and in Jean-Luc Nagel's thesis [1], respectively, could be integrated into one single design, possibly including a microcontroller, a memory and an image sensor, so as to realize a complete low-power System-on-Chip solution for providing face authentication on mobile devices.

References

[1] J.-L. Nagel, “Algorithms and VLSI Architectures for Low-Power Mobile Face Verification,” *PhD thesis No. 1850*, University of Neuchâtel, Switzerland, 2006.

[2] http://www.merriam-webster.com/dictionary/biometry

[3] http://www.tibs.org/interior.aspx?id=290

[4] http://www.biometrics.org/intro.htm

[5] M. Nisenson, I. Yariv, R. El-Yaniv, and R. Meir, “Towards Behaviometric Security Systems: Learning to Identify a Typist,” in *Proc. 7th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003)*, Cavtat-Dubrovnik, Croatia, Sep. 2003, pp. 363–374.

[6] A. Jain, A. Ross, and S. Prabhakar, “An Introduction to Biometric Recognition,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, 2004.

[7] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer, 2003.

[8] A. C. Weaver, “Biometric Authentication,” *IEEE Computer*, vol. 39, no. 2, pp. 96–97, 2006.

[9] A. Bertillon, “Une Application Pratique de L’Anthropométrie sur un Procédé D’Identification,” *Annales de Démographie Internationale*, vol. 5, pp. 330–350, 1881.

[10] C. Champod, C. Lennard, P. Margot, and M. Stoilovic, *Fingerprints and Other Ridge Skin Impressions*, International Forensic Science and Investigation Series, CRC Press, 2004.

[11] F. J. Mouat, "Notes on M. Bertillon's Discourse on the Anthropometric Measurement of Criminals," *Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 20, pp. 182–198, 1891.

[12] R. Fosdick, "The Passing of the Bertillon System of Identification," *Journal of the American Institute of Criminal Law and Criminology*, vol. 6, no. 3, pp. 363–369, 1915.

[13] M. Trauring, "Automatic Comparison of Finger-Ridge Patterns," *Nature*, vol. 197, no. 4871, pp. 938–940, 1963.

[14] A. Roddy and J. Stosz, "Fingerprint Features-Statistical Analysis and System Performance Estimates," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1390–1421, 1997.

[15] M. Ballantyne, R. S. Boyer, and L. Hines, "Woody Bledsoe: His Life and Legacy," *AI Magazine*, vol. 17, no. 1, pp. 7–20, 1996.

[16] N. Herbst and C. Liu, "Automatic Signature Verification Based on Accelerometry," *IBM Journal of Research and Development*, vol. 21, pp. 245–253, 1977.

[17] D. Maio, D. Maltoni, R. Cappelli, J. Wayman, and A. Jain, "FVC2004: Third Fingerprint Verification Competition," in *Proc. 1st Int'l Conf. on Biometric Authentication (ICBA '04)*, Honk Kong, July 2004, pp. 1–7.

[18] R. H. Ernst, "Hand ID System," *U.S Patent No. 3'576'537*, 1971.

[19] R. P. Miller, "Finger Dimension Comparison Identification System," *U.S Patent No. 3'576'538*, 1971.

[20] N. Kanwisher, J. McDermott, and M. Chun, "The Fusiform Face Area: A Module in Human Extrastriate Cortex Specialized for Face Perception," *Journal of Neuroscience*, vol. 17, no. 11, pp. 4302–4311, 1997.

[21] J. Haxby, E. Hoffman, and M. Gobbini, "Human Neural Systems for Face Recognition and Social Communication," *Biological Psychiatry*, vol. 51, no. 1, pp. 59–67, 2002.

[22] P. Sinah and T. Poggio, "I Think I Know that Face...," *Nature*, vol. 384, no. 6608, p. 404, 1996.

[23] F. Tsalakanidou, S. Malassiotis, and M. G. Strintzis, "Exploitation of 3D Images for Face Authentication Under Pose and Illumination Variations," in *Proc. 2nd Int'l Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'04)*, Chapel Hill, NC, USA, 2004, pp. 50–57.

[24] V. Blanz, S. Romdhani, and T. Vetter, "Face Identification across Different Poses and Illuminations with a 3D Morphable Model," in *Proc. 5th IEEE Int'l Conf. on Automatic Face and Gesture Recognition (FG '02)*, 2002, pp. 192–197.

[25] C. Beumier, "Identity Authentication Through 3D Face Analysis," *PhD thesis*, Ecole Nationale Supérieure des Télécommunications de Paris, France, 2003.

[26] A. Bronstein, M. Bronstein, and R. Kimmel, "Expression-Invariant 3D Face Recognition," in *Proc. 4th Int'l Conf. on Audio- and Video-based Biometric Person Authentication (AVBPA '03)*, Guildford, UK, June 2003, pp. 62–69.

[27] J. Daugman, "How Iris Recognition Works," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 1, p. 21, 2004.

[28] J. Daugman, "Results from 200 Billion Iris Cross-Comparisons," *Technical Report No. 635*, Computer Laboratory, University of Cambridge, UK, 2005.

[29] J. G. Daugman, "Biometric Personal Identification System Based on Iris Analysis," *U.S Patent No. 5'291'560*, 1994.

[30] J. Mäntyjärvi, M. Lindholm, E. Vildjiounaite, S.-M. Mäkelä, and H. Ailisto, "Identifying Users of Portable Devices from Gait Pattern with Accelerometers," in *Proc. 30th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, Philadelphia, USA, Mar. 2005, vol. 2, pp. 973–976.

[31] L. Bai, D. Chou, D. Yen, and B. Lin, "Mobile Commerce: Its Market Analyses," *International Journal of Mobile Communications*, vol. 3, no. 1, pp. 66–81, 2005.

[32] T. Iso and K. Yamazaki, "Gait Analyzer Based on a Cell Phone with a Single Three-Axis Accelerometer," in *Proc. 8th Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '06)*, Helsinki, Finland, Sep. 2006, pp. 141–144.

[33] R. Brunelli and T. Poggio, "Face Recognition Through Geometrical Features," in *Proc. of the 2nd European Conference on Computer Vision (ECCV 92)*, Santa Margherita Ligure, Italy, May 1992, pp. 792–800.

[34] A. Samal and P. A. Iyengar, "Automatic Recognition and Analysis of Human Faces and Facial Expressions: A Survey," *Pattern Recognition*, vol. 25, no. 1, pp. 65–77, 1992.

[35] A. L. Yuille, "Deformable Templates for Face Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 59–70, 1991.

[36] M. Nixon, "Eye Spacing Measurement for Facial Recognition," *SPIE Proceedings*, vol. 575, pp. 279–285, 1985.

[37] R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, 1993.

[38] T. Kanade, "Picture Processing System by Computer Complex and Recognition of Human Faces," *PhD thesis*, Department of Information Science, Kyoto University, Japan, 1973.

[39] M. Zobel, A. Gebhard, D. Paulus, J. Denzler, and H. Niemann, "Robust Facial Feature Localization by Coupled Features," in *Proc. 4th IEEE Int'l Conf. on Automatic Face and Gesture Recognition (FG '00)*, Grenoble, France, Mar. 2000, pp. 2–7.

[40] A. Schubert, "Detection and Tracking of Facial Features in Real Time Using a Synergistic Approach of Spatio-Temporal Models and Generalized Hough-Transform Techniques," in *Proc. 4th IEEE Int'l Conf. on Automatic Face and Gesture Recognition (FG '00)*, Grenoble, France, Mar. 2000, pp. 116–121.

[41] E. Sabert and A. M. Tekalp, "Frontal-View Face Detection and Facial Feature Extraction Using Color, Shape and Symmetry Based Cost Functions," *Pattern Recognition*, vol. 19, no. 8, pp. 669–680, 1998.

[42] H. P. Graf, T. Chen, E. Petajan, and E. Cosatto, "Locating Faces and Facial Parts," in *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, Zurich, Switzerland, June 1995, pp. 41–46.

[43] L. Sirovich and M. Kirby, "Low-Dimensional Procedure for the Characterization of Human Faces," *Journal of the Optical Society of America A*, vol. 4, no. 3, pp. 519–524, 1987.

[44] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html

[45] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[46] Y. Moses, Y. Adini, and S. Ullman, "Face Recognition: The Problem of Compensating for Changes in Illumination Direction," in *Proc. 3rd European Conf. Computer Vision*, Stockholm, Sweden, May 1994, pp. 721–732.

[47] P. N. Belhumeur, J. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[48] M. S. Bartlett, M. H. Lades, and T. J. Sejnowski, "Independent Component Representations for Face Recognition," *Proc. SPIE*, vol. 3299, pp. 528–539, 1998.

[49] C. von der Malsburg, "The Dynamic Link Architecture," in *The Handbook of Brain Theory and Neural Networks* (M. A. Arbib, ed.), pp. 365–368, MIT Press, Cambridge, MA, USA, 2nd ed., 2002.

[50] C. von der Malsburg, "Pattern Recognition by Labeled Graph Matching," *Neural Networks*, vol. 1, no. 2, pp. 141–148, 1988.

[51] M. Lades, J. C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Würtz, and W. Konen, "Distortion Invariant Object Recognition in the Dynamic Link Architecture," *IEEE Trans. Computers*, vol. 42, no. 3, pp. 300–311, 1993.

[52] M. Lades, J. C. Vorbrüggen, and R. P. Würtz, "Recognising Faces with a Transputer Farm," in *Proc. Third Int'l Conference on Applications of Transputers*, Glasgow, U.K., Aug. 1991, pp. 148–153.

[53] L. Wiskott, "Labeled Graphs and Dynamic Link Matching for Face Recognition and Scene Analysis," *PhD thesis*, Ruhr-Universität, Bochum, Germany, 1995.

[54] W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld, "Face Recognition: A Literature Survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, 2003.

[55] X. Tan, S. Chen, Z. Zhou, and F. Zhang, "Face Recognition from a Single Image per Person: A Survey," *Pattern Recognition*, vol. 39, no. 9, pp. 1725–1745, 2006.

[56] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face Recognition and Gender Determination," in *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, Zurich, Switzerland, June 1995, pp. 92–97.

[57] L. Wiskott, J. Fellous, N. Kuiger, and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, 1997.

[58] K. Okada, J. Steffens, T. Maurer, H. Hong, E. Elagin, H. Neven, and C. von der Malsburg, "The Bochum/USC Face Recognition System and How it Fared in the FERET Phase III Test," in *Face Recognition: From Theory to Applications* (H. Wechsler, P. J. Phillips, V. Bruce, F. F. Soulié, and T. S. Huang, eds.), pp. 186–205, Springer-Verlag, 1998.

[59] B. Duc, "Feature Design: Applications to Motion Analysis and Identity Verification," *PhD thesis No. 1741*, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 1997.

[60] B. Duc, S. Fischer, and J. Bigun, "Face Authentication with Sparse Grid Gabor Information," in *Proc. of the 1997 IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP '97)*, Munich, Germany, Apr. 1997, vol. 4, pp. 3053–3056.

[61] C. Kotropoulos, I. Pitas, S. Fischer, and B. Duc, "Face Authentication Using Morphological Dynamic Link Architecture," in *Proc. 1st Int'l Conf. on Audio- and Video-Based Biometric Person Authentication (AVBPA '97)*, Crans-Montana, Switzerland, Mar. 1997, pp. 169–176.

[62] S. Pigeon and L. Vandendorpe, "The M2VTS Multimodal Face Database (Release 1.00)," in *Proc. 1st Int'l Conf. on Audio- and Video-Based Biometric Person Authentication (AVBPA '97)*, Crans-Montana, Switzerland, Mar. 1997, pp. 403–409.

[63] C. Kotropoulos and I. Pitas, "Face Verification Based on Morphological Dynamic Link Architecture," in *Proc. 3rd IEEE Workshop on Nonlinear Signal and Image Processing (NSIP '97)*, Mackinac Island, Michigan USA, Sep. 1997.

[64] C. Kotropoulos, A. Tefas, and I. Pitas, "Frontal Face Authentication Using Morphological Elastic Graph Matching," *IEEE Trans. Image Processing*, vol. 9, no. 4, pp. 555–560, 2000.

[65] C. Kotropoulos and I. Pitas, "Face Authentication Based on Morphological Grid Matching," in *Proc. Int'l Conf. on Image Processing (ICIP 97)*, Santa Barbara, CA, USA, Oct. 1997, vol. 1, pp. 105–108.

[66] M. Lades, "Face Recognition Technology," in *Handbook of Pattern Recognition and Computer Vision* (C. H. Chen, L. F. Pau, and P. S. P. Wang, eds.), pp. 667–683, World Scientific Publishing Company, 2nd ed., 1998.

[67] C. Kotropoulos, A. Tefas, and I. Pitas, "Frontal Face Authentication Using Variants of Dynamic Link Matching Based on Mathematical Morphology," in *Proc. Int'l Conf. on Image Processing (ICIP 98)*, Chicago, USA, Oct. 1998, vol. 1, pp. 122–126.

[68] B. Duc, S. Fischer, and J. Bigun, "Face Authentication with Gabor Information on Deformable Graphs," *IEEE Trans. on Image Processing*, vol. 8, no. 4, pp. 504–516, 1999.

[69] C. Kotropoulos, A. Tefas, I. Pitas, C. Fernandez, and F. Fernández, "Performance Assessment of Morphological Dynamic Link Architecture under Optimal and Real Operating Conditions," in *Proc. of the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP '99)*, Antalya, Turkey, June 1999, pp. 355–359.

[70] C. Kotropoulos, A. Tefas, and I. Pitas, "Morphological Elastic Graph Matching Applied to Frontal Face Authentication under Optimal and Real Conditions," in *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems (ICMCS 99)*, Florence, Italy, July 1999, vol. 2, pp. 934–938.

[71] C. Kotropoulos, A. Tefas, and I. Pitas, "Morphological Elastic Graph Matching Applied to Frontal Face Authentication under Well-Controlled and Real Conditions," *Pattern Recognition*, vol. 33, no. 12, pp. 1935–1947, 2000.

[72] M. A. Fischler and R. A. Elschlager, "The Representation and Matching of Pictorial Structures," *IEEE Trans. Computers*, vol. 22, pp. 67–92, Jan. 1973.

[73] F. Samaria and F. Fallside, "Face Identification and Feature Extraction Using Hidden Markov Models," in *Image Processing: Theory and Applications* (G. Vernazza, A. N. Venetsanopoulos, and C. Braccini, eds.), pp. 295–298, Elsevier, 1993.

[74] A. Nefian and M. Hayes III, "Hidden Markov Models for Face Recognition," in *Proc. IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP'98)*, Seattle, WA, USA, May 1998, vol. 5, pp. 2721–2724.

[75] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Black, "Face Recognition: A Convolutional Neural-Network Approach," *IEEE Trans. Neural Networks, Special Issue on Neural Networks and Pattern Recognition*, vol. 8, no. 1, pp. 98–113, 1997.

[76] Y. Gao and M. Leung, "Face Recognition Using Line Edge Map," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 764–779, 2002.

[77] S. Arca, P. Campadelli, and R. Lanzarotti, "A Face Recognition System Based on Local Feature Analysis," in *Proc. 4th Int'l Conf. on Audio- and Video-based Biometric Person Authentication (AVBPA '03)*, Guildford, UK, June 2003, pp. 182–189.

[78] G. G. Gordon, "Face Recognition from Frontal and Profile Views," in *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, Zurich, Switzerland, June 1995, pp. 47–52.

[79] K. Yu, X. Jiang, and H. Bunke, "Face Recognition by Facial Profile Analysis," in *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, Zurich, Switzerland, June 1995, pp. 208–213.

[80] Z. Liposcak and S. Loncaric, "A Scale-Space Approach to Face Recognition from Profiles," in *Proc. of the 8th Int'l Conf. on Computer Analysis of Images and Patterns (CAIP '99)*, Ljubljana, Slovenia, Sep. 1999, pp. 243–250.

[81] K. Bowyer, K. Chang, and P. Flynn, "A Survey of Approaches to Three-Dimensional Face Recognition," in *Proc. 17th Int'l Conf. on Pattern Recognition (ICPR 2004)*, Cambridge, UK, Aug. 2004, vol. 1.

[82] S. Kong, J. Heo, B. Abidi, J. Paik, and M. Abidi, "Recent Advances in Visual and Infrared Face Recognition — A Review," *Computer Vision and Image Understanding*, vol. 97, no. 1, pp. 103–135, 2005.

[83] H. Wechsler, *Reliable Face Recognition Methods: System Design, Implementation and Evaluation*, Springer, 2007.

[84] J. Zhang, Y. Yan, and M. Lades, "Face Recognition: Eigenface, Elastic Matching, and Neural Nets," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1423–1435, 1997.

[85] C. Kotropoulos, A. Tefas, and I. Pitas, "Frontal Face Authentication Using Discriminating Grids with Morphological Feature Vectors," *IEEE Trans. Multimedia*, vol. 2, no. 1, pp. 14–26, 2000.

[86] M. Yang, D. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[87] E. Hjelmas and B. Low, "Face Detection: A Survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236–274, 2001.

[88] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Algorithm in Low Dimensions," *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[89] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, Inc., 2nd ed., 2001.

[90] S. Bengio, J. Mariéthoz, and S. Marcel, "Evaluation of Biometric Technology on XM2VTS," *IDIAP Research Report 01-21*, IDIAP, Martigny, Switzerland, 2001.

[91] D. A. Pollen and S. F. Ronner, "Visual Cortical Neurons as Localized Spatial Frequency Filters," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 13, no. 5, pp. 907–916, 1983.

[92] T. S. Lee, "Image Representation Using 2D Gabor Wavelets," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 959–971, Oct. 1996.

[93] A. Jain and F. Farrokhnia, "Unsupervised Texture Segmentation Using Gabor Filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[94] T. Ebrahimi and M. Kunt, "Image Compression by Gabor Expansion," *Optical engineering*, vol. 30, no. 7, pp. 873–880, 1991.

[95] J. Daugman, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1169–1179, July 1988.

[96] G. Matheron and J. Serra, "The Birth of Mathematical Morphology," in *Proc. 6th Intl. Symp. Mathematical Morphology (ISMM 02)*, 2002, pp. 1–16.

[97] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, 2nd ed., 2001.

[98] H. Heijmans, "Mathematical Morphology: Basic Principles," in *Proc. of Summer School on Morphological Image and Signal Processing*, Zakopane, Poland, Sep. 1995.

[99] P. Jackway and M. Deriche, "Scale-Space Properties of the Multiscale Morphological Dilation-Erosion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, pp. 38–51, 1996.

[100] A. Witkin, "Scale-Space Filtering: A New Approach to Multi-Scale Description," in *Proc. IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP'84)*, San Diego, USA, Mar. 1984, vol. 9, pp. 150–153.

[101] A. Tefas, C. Kotropoulos, and I. Pitas, "Face Verification Based on Morphological Shape Decomposition," in *Proc. 3rd IEEE Int'l Conf. on Automatic Face and Gesture Recognition (FG '98)*, Nara, Japan, Apr. 1998, pp. 36–41.

[102] A. Tefas, C. Kotropoulos, and I. Pitas, "Face Verification Using Elastic Graph Matching Based on Morphological Signal Decomposition," *Signal Processing*, vol. 82, no. 6, pp. 833–851, 2002.

[103] C. Sanderson and K. Paliwal, "Polynomial Features for Robust Face Authentication," in *Proc. IEEE Int'l Conf. on Image Processing (ICIP '02)*, Rochester, NY, USA, June 2002, vol. 3, pp. 997–1000.

[104] C. Sanderson and S. Bengio, "Robust Features for Frontal Face Authentication in Difficult Image Conditions," in *Proc. 4th Int'l Conf. on Audio- and Video-based Biometric Person Authentication (AVBPA '03)*, Guildford, UK, June 2003, pp. 495–504.

[105] J. Bracamonte, M. Ansorge, F. Pellandini, and P.-A. Farine, "A Low Complexity Change Detection Algorithm Operating in the Compressed Domain," in *Proc. 6th COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*, Thessaloniki, Greece, May 2004, pp. 88–93.

[106] N. Pavešić, I. Fratrić, and S. Ribarić, "Degradation of the XM2VTS Database Face Images," in *Proc. 2nd COST 275 Workshop on Biometrics on the Internet*, Vigo, Spain, Mar. 2004, pp. 15–19.

[107] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maître, "XM2VTSDB: The Extended M2VTS Database," in *Proc. 2nd Int'l Conf. on Audio- and Video-based Biometric Person Authentication (AVBPA'99)*, Washington, D.C, USA, Mar. 1999, pp. 72–77.

[108] J. Luettin and G. Maître, "Evaluation Protocol for the extended M2VTS Database (XM2VTSDB)," *IDIAP Communication 98-05*, IDIAP, Martigny, Switzerland, 1998.

[109] R. Sedgewick, *Algorithms*, Addison-Wesley, 1988.

[110] C. A. R. Hoare, "Quicksort," *The Computer Journal*, vol. 5, no. 1, p. 10, 1962.

[111] K. Messer, J. Kittler, J. Short, G. Heusch, F. Cardinaux, S. Marcel, Y. Rodriguez, S. Shan, Y. Su, W. Gao, and X. Chen, "Performance Characterisation of Face Recognition Algorithms and Their Sensitivity to Severe Illumination Changes," in *Proc. 2nd Int'l Conference on Biometrics (ICB 2006)*, Honk Kong, Jan. 2006, pp. 1–11.

[112] J. Matas, M. Hamouz, K. Jonsson, J. Kittler, Y. Li, C. Kotropoulos, A. Tefas, I. Patas, T. Tan, H. Yan, F. Smeraldi, J. Bigun, N. Capdevielle, W. Gerstner, S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz, "Comparison of Face Verification Results on the XM2VTS Database," in *Proc. Int'l Conf. on Pattern Recpognition (ICPR '00)*, Barcelona, Spain, Sep. 2000, vol. 4, pp. 4858–4863.

[113] K. Messer, J. Kittler, M. Sadeghi, S. Marcel, C. Marcel, S. Bengio, F. Cardinaux, C. Sanderson, J. Czyz, L. Vandendorpe, S. Srisuk, and et al., "Face Verification Competition on the XM2VTS Database," in *Proc. 4th Int'l Conf. on Audio- and Video-based Biometric Person Authentication (AVBPA '03)*, Guildford, UK, June 2003, pp. 964–974.

[114] Y. Su, S. Shan, B. Cao, X. Chen, and W. Gao, "Multiple Fisher Classifiers Combination for Face Recognition based on Grouping AdaBoosted Gabor Features," in *Proc. 16th British Machine Vision Conference (BMVC 2005)*, Oxford, UK, Sep. 2005, vol. 2, pp. 569–578.

[115] http://www.cost.esf.org/index.php?id=118

[116] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.

[117] http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html

[118] K. Lee, J. Ho, and D. Kriegman, "Acquiring Linear Subspaces for Face Recognition under Variable Lighting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 684–698, 2005.

[119] R. H. Barnett, *The 8051 Family of Microcontrollers*, Prentice Hall, 1995.

[120] J.-L. Nagel, P. Stadelmann, M. Ansorge, and F. Pellandini, "Comparison of Feature Extraction Techniques for Face Verification Using Elastic Graph Matching on Low-Power Mobile Devices," in *Proc. IEEE Region 8 Int'l. Conf. on Computer as a Tool (EUROCON'03)*, Ljubljana, Slovenia, Sep. 2003, vol. 2, pp. 365–369.

[121] J.-L. Nagel, P. Stadelmann, M. Ansorge, and F. Pellandini, "A Low-Power VLSI Architecture for Face Verification using Elastic Graph Matching," in *Proc. XIth European Signal Processing Conference (EUSIPCO 2002)*, Toulouse, France, Sep. 2002, vol. 3, pp. 577–580.

[122] M. J. Flynn and K. W. Rudd, "Parallel Architectures," *ACM Computing Surveys*, vol. 28, no. 1, pp. 67–70, 1996.

[123] P. Stadelmann, J.-L. Nagel, M. Ansorge, and F. Pellandini, "A Multiscale Morphological Coprocessor for Low-Power Face Authentication," in *Proc. XIth European Signal Processing Conference (EUSIPCO 2002)*, Toulouse, France, Sep. 2002, vol. 3, pp. 581–584.

[124] M. Nadeem, "Design and VHDL Implementation of a Low-Power Face Verification Architecture for 3G Mobile Communicators," *MSc thesis*, Dept. of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm, Sweden, Mar. 2003.

[125] P. Stadelmann, "IRISSA: Iris Programming Environment," *Technical documentation, CTI Projet No. 4757*, Institute of Microtechnology, University of Neuchâtel, 2002.

[126] http://flex.sourceforge.net/

[127] http://www.gnu.org/software/bison/

[128] http://en.wikipedia.org/wiki/Backus-Naur_form

[129] S. Tanner, S. Lauxtermann, M. Waeny, M. Willemin, N. Blanc, J. Grupp, R. Dinger, E. Doering, M. Ansorge, P. Seitz, and F. Pellandini, "Low-Power Digital Image Sensor for Still-Picture Image Acquisition," in *Proc. SPIE Photonics West Conf. 2001*, San José, USA, Jan. 2001, vol. 4306, pp. 358–365.

[130] B. E. Bayer, “Color Imaging Array,” *U.S Patent No. 3’971’065*, 1976.

[131] Nova Engineering, *Constellation-20KE User Manual*, 2002.

[132] http://www.ethernut.de/en/index.html

[133] Nova Engineering, *Using Selectable I/O Standards in APEX 20KE*, Dec. 2001.

[134] http://tools.ietf.org/html/rfc854

[135] http://www.umc.com/English/process/d.asp

[136] T. Sakurai and A. Newton, “Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas,” *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[137] K. Nose and T. Sakurai, “Optimization of $\mathrm{V_{DD}}$ and $\mathrm{V_{TH}}$ for Low-Power and High-Speed Applications,” in *Proc. Asia South Pacific Design Automation Conference (ASP-DAC ’00)*, Yokohama, Japan, Jan. 2000, pp. 469–474.

[138] J.-M. Masgonty, S. Cserveny, and C. Piguet, “Low-Power SRAM and ROM Memories,” in *Proc. Int’l Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS 2001)*, Yverdon-Les-Bains, Switzerland, 2001.

[139] *ARM Architecture Reference Manual*, ARM Limited, 2005.

[140] http://www.secure-phone.info/

[141] R. Ricci, G. Chollet, M. Crispino, S. Jassim, J. Koreman, M. Olivar-Dimas, S. Garcia-Salicetti, and P. Soria-Rodriguez, “SecurePhone: A Mobile Phone with Biometric Authentication and e-Signature Support for Dealing Secure Transactions on the Fly,” *Proceedings of SPIE*, vol. 6250, pp. 76–86, 2006.

[142] K. Pun, Y. Moon, J. Chen, and H. Yeung, “A Face Authentication System for Mobile Devices: Optimization Techniques,” *Proceedings of SPIE*, vol. 5684, pp. 265–273, 2005.

[143] http://www.nist.gov/humanid/feret/feret_master.html

[144] Y. Moon, K. Pun, K. Chan, M. Wong, and T. Yu, “Enabling EBGM Face Authentication on Mobile Devices,” in *Proc. 2nd Int’l Workshop on MultiModal User Authentication*, Toulouse, France, May 2006.

[145] N. Aaraj, S. Ravi, A. Raghunathan, and N. Jha, “Hybrid Architectures for Efficient and Secure Face Authentication in Embedded Systems,” *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 296–308, 2007.

[146] http://www.tensilica.com/products/xtensa_overview.htm

[147] http://broadband.mb.softbank.jp/mb/en/product/vf/904sh-e-manual.pdf

[148] http://www.nttdocomo.co.jp/english/binary/pdf/support/manual/foma/f903i/F903i_E_All.pdf

[149] http://www.securimetrics.com/solutions/hiide.html

Internet links were verified to be valid as of September 30, 2008.

Publications involving the author

Publications related to biometrics

- M. Ansorge, F. Pellandini, S. Tanner, J. Bracamonte, P. Stadelmann, J.-L. Nagel, P. Seitz, N. Blanc, and C. Piguet, "Very Low-Power Image Acquisition and Processing for Mobile Communications Devices," Keynote paper, in *Proc. IEEE Int'l Symposium on Signals, Circuits and Systems (SCS 2001)*, Iasi, Romania, July 2001, pp. 289–296.
- M. Ansorge, S. Tanner, X. Shi, J. Bracamonte, J.-L. Nagel, P. Stadelmann, F. Pellandini, P. Seitz, and N. Blanc, "Smart Low-Power CMOS Cameras for 3G Mobile Communicators," Invited paper, in *Proc. 1st Int'l Conf. on Circuits and Systems for Communications, ICS'02*, St-Petersburg, Russia, June 2002, pp. 216–225.
- J.-L. Nagel, P. Stadelmann, M. Ansorge, and F. Pellandini, "A Low-Power VLSI Architecture for Face Verification using Elastic Graph Matching," in *Proc. XIth European Signal Processing Conference (EUSIPCO 2002)*, Toulouse, France, Sep. 2002, vol. 3, pp. 577–580.
- P. Stadelmann, J.-L. Nagel, M. Ansorge, and F. Pellandini, "A Multiscale Morphological Coprocessor for Low-Power Face Authentication," in *Proc. XIth European Signal Processing Conference (EUSIPCO 2002)*, Toulouse, France, Sep. 2002, vol. 3, pp. 581–584.

- J.-L. Nagel, P. Stadelmann, M. Ansorge, and F. Pellandini, “Comparison of Feature Extraction Techniques for Face Verification Using Elastic Graph Matching on Low-Power Mobile Devices,” in *Proc. IEEE Region 8 Int’l. Conf. on Computer as a Tool (EUROCON’03)*, Ljubljana, Slovenia, Sep. 2003, vol. 2, pp. 365–369.

- M. Ansorge, J.-L. Nagel, P. Stadelmann, and P.-A. Farine, “Biometrics for Mobile Communicators,” in *Proc. 2nd IEEE Int’l Conf. on Circuits & Systems for Communications (ICCSC’04)*, Moscow, Russia, June 2004.

- J.-L. Nagel and P. Stadelmann, “Face Authentication for Low-Power Mobile Devices,” Invited talk, *Conf. on Biometrical Feature Identification and Analysis*, Göttingen, Germany, Sep. 2007.

Other publications

- J. Bracamonte, P. Stadelmann, M. Ansorge, and F. Pellandini, “A Multiplierless Implementation Scheme for the JPEG Image Coding Algorithm,” in *Proc. 4th IEEE Nordic Signal Processing Symposium (NORSIG 2000)*, Kolmården, Sweden, July 2000, pp. 17–20.

- M. Ansorge, F. Pellandini, J. Bracamonte, S. Tanner, and P. Stadelmann, “Advances in Very Low-Power Image Sensing and Compression For Mobile Multimedia Communicators,” Invited paper, in *Proc. EURASIP Conf. on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001)*, Budapest, Hungary, Sep. 2001, pp. 141–150.

- M. Ansorge, J. Grupp, F. Pellandini, P.-A. Farine, P. Heck, S. Tanner, J. Bracamonte, and P. Stadelmann, “Microcamera Embedded in a Wristwatch,” Invited talk, *8th IEEE Int’l Symposium on Signals, Circuits, and Systems (ISSCS 07)*, Iasi, Romania, July 2007.

- S. Grassi, P. Heck, P. Stadelmann, P. Cotofrei, P. Dinissen, F. Meylan, J.-P. Mignot, J.-N. Pfeuti, F. Piccini, P. Geiser, G. Biundo, P.-A. Farine, J. Grupp, and K. Stoffel, “Watch-Dictaphone for Automatic Medical Codification,” in *Proc. XVIth European Signal Processing Conference (EUSIPCO 2008)*, Lausanne, Switzerland, Aug. 2008.

Acknowledgments

This PhD thesis could not have been successfully completed without the help and support of many people, who are all kindly acknowledged.

Firstly, I wish to express my sincere gratitude to Prof. Fausto Pellandini, for offering me the opportunity to join his research team, as well as for being my PhD adviser. I am also very grateful to his successor, Prof. Pierre-André Farine, for letting me finish this thesis in the most favorable conditions, and for being a member of my jury committee. I heartily thank them both for their understanding and patience in waiting for the completion of this work.

I would like to thank the CSEM for financially supporting the research projects that led to this thesis. I am in particular grateful to Dr. Nicolas Blanc for kindly accepting to be part of my jury committee.

I am deeply indebted to PD Dr. Michael Ansorge for the huge amount of time and energy he invested in reviewing this dissertation. I would also like to thank him wholeheartedly for the much appreciated encouragement and advice he provided when the structure of this report was being laid out, as well as for the countless comments and suggestions he offered throughout the entire redaction process.

I would like to thank all friends and colleagues from ESPLAB and PARLAB, both former and present, for the very fruitful collaborations in the framework of several projects, and / or for the numerous memories of fun and enjoyable moments they remain associated with.

I am deeply grateful to Dr. Javier Bracamonte, who supervised two of my student projects, and thusly introduced me to the captivating world of digital image processing.

In direct connection with this thesis, I am very much obliged to Dr. Jean-Luc Nagel, whom I collaborated with in the course of several projects in the field of biometrics. I am especially thankful for the support he provided during the VHDL implementation, simulation and synthesis stages. My thanks also go to Dr. Christian Schuster, who was always keen on sharing LaTeX tips, and on helping me solve issues with the design tools.

I would like to thank the people in IMT's other laboratories, as well those in the administrative department, either for their very precious help in sorting out all sorts of matters, or for the always rejoicing encounters at the building's cafeteria or at IMT's Christmas and summer's parties.

I am very much thankful to my friend and family for being very supportive and understanding, especially during the finalization of this thesis, when I sometimes was about to forget that there is still a world outside of IMT!

Finally, I am deeply grateful to my parents for their trust and indefectible support during all my years of studying.