Face Analysis and Recognition in Mobile Devices

Mauricio Villegas Santamaría

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Valencia, November 2008

Cover designed by Carolina Vallejo.
Photograph *Cell Phone Cameras* by Jeramey Jannene, used under the Creative Commons License Attribution 2.0 Generic.

Preface

This document is a masters thesis for the program *Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital* submitted to the *Departamento de Sistemas Informáticos y Computación* at the *Universidad Politécnica de Valencia*. In this masters program the thesis could be either oriented to research or it could be professionally oriented. Although I am working on research which in the future will lead to my Ph.D. thesis, this masters thesis has been professionally oriented so that it somewhat complements my ongoing education.

Abstract

Face recognition is currently a very active research topic due to the great variety of applications it can offer. Moreover, nowadays it is very common for people to have mobile devices such as PDAs or mobile phones which have an integrated digital camera. This gives the opportunity to develop face recognition applications for this type of devices. This thesis treats the problem of face analysis and recognition in mobile devices. The problem is discussed and analysed, observing which are the difficulties that are encountered. Some algorithms for face recognition are analysed and optimised so that they are better suited for the resource constrains of mobile devices. An implementation of the algorithms in Java ME are presented, and these are used to create a demonstration application that works in commercial phones.

Acknowledgements

First of all, I would like to thank Dr. Roberto Paredes, professor of the Master and adviser for this thesis. He has taught me most of the things that were required to complete this work and has always given me useful advises and tips for improving the things that I do.

My most sincere gratitude to Prof. Enrique Vidal, director of the Pattern Recognition and Human Language Technology (PRHLT) group at the Universidad Politécnica de Valencia, for believing in me and supporting me to join the PRHLT and the Instituto Tecnológico de Informática (ITI).

I would also like express my great gratitude to the Generalitat Valenciana - Consellería d'Educació that have financially supported most of my work by giving me an FPI scholarship which without non of this could have ever been possible.

Special thanks to Carolina Vallejo, first for designing the cover of this thesis, but most of all for encouraging me throughout the development of this work.

Finally I would like to thank my parents, close family and friends, they are the reason why I am here and for the person I have become.

Face Analysis and Recognition in Mobile Devices

by

Mauricio Villegas Samtamaría

Masters thesis submitted to

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Thesis adviser: Roberto Paredes Palacios

Valencia, November 2008

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Biometrics is the study of methods for recognising humans based on some intrinsic physical or behavioural trait. These methods have proved to be very useful for different tasks which are common in the current society. Among the different modalities used on biometric systems, face images are very popular because it is an unobtrusive method, well tolerated by users, and with a wide range of applications. Moreover, nowadays it is very common for people to have mobile devices such as PDAs or mobile phones which have an integrated digital camera. This gives the opportunity to develop face recognition applications for this type of devices.

The main objective of this work was to develop a face verification application which worked on commercial mobile devices. However the idea was not only to have a working software, but also to consider the difficulties that needed to be addressed. The basic problem that face recognition has is that it generally requires a lot of computational resources, something which is not common to have on a mobile device. In this work, some algorithms for face recognition are analysed, and some optimisations for these are proposed. The proposed optimised algorithms were implemented and an extensive empirical analysis was performed.

The thesis is organised as follows. The next chapter is a review of the work that has been done on the area of biometrics and face recognition on resource constrained devices. This review includes the research publications that are related to this topic and some of the products that are currently available. Chapter 3 describes the face recognition algorithms considered in the work and the respective proposed optimisations. Followed by this, the chapter 4 presents an empirical analysis of the algorithms. The results first show the effectiveness of the proposed optimisations, followed by an estimate of how well the algorithms will perform in such circumstances and what is the amount of resources that is actually required. The final demonstration

applications developed to work on mobile devices that support Java ME are described in chapter 5. The final chapter states the conclusions, making emphasis on which objectives were fully reached and which are the areas that need further improvement.

Chapter 2

Review of Related Work

The current demand for security and surveillance systems, and the ever increasing computing power that makes room for new applications, are some of the reasons that have resulted in a large amount of research on face recognition and biometrics. A complete literature review on this subject would take several pages, which is not the current objective. For more detailed literature surveys refer to [ZCRP03, TEBEH06, LJ04]. This chapter is intended to be a short summary of the most important aspects in face recognition research followed by a more specific review of research related to face recognition in mobile or resource constrained devices.

The amount of research on face recognition can be observed on several important conferences specialised on this subject like the International Conference on Biometrics (ICB) and the International Conference on Automatic Face and Gesture Recognition (FG). Also on more general conferences about pattern recognition, computer vision or image processing, there are generally many papers related to face recognition. Apart from the numerous publications available, over the last few years there have been a several face recognition competitions, to name a few there are some competitions using the XM2VTS and BANCA databases and also the important NIST competitions, the Face Recognition Grand Challenge (FRGC), the Face Recognition Vendor Test (FRVT) and the upcoming Multiple Biometrics Grand Challenge (MBGC).

In the computer science literature the term *face recognition* is generally used to encompass the two subjects *face verification* and *face identification*. Face verification, also known as face authentication, is the task of deciding if a given face truly belongs to a certain claimed identity. That is, it is only decided if the face belongs or it does not belong to a claimed identity. On the other hand, face identification is the task of deciding among several known identities which, if any, is the one that corresponds to a certain input face.

Although the term *face recognition* generally refers to face verification and face identification, the research on face recognition involves a lot more of topics. Among these topics the first one that must be mentioned is face detection, or face tracking if an image sequence is involved. Other topics on face recognition are extraction of different types of information from a face image. Some examples can be the identification of gender, ethnicity, state of emotion, age, etc.

2.1 Face Detection

Irrespectively of the application, as input to a face recognition system there is always some form of data which can be for example an image, an image sequence or a three-dimensional image. Therefore the first process to preform is to find out if in the input there is a face (or faces) present. This process is achieved by a face detection algorithm. If the input is a sequence of images, the task is to do face tracking, however the basis is still a face detection algorithm, the difference is that the positions of the faces in previous frames and motion estimation are used to assist the detection in the current frame.

The most common approach to face detection is to treat it as a binary pattern classification task and use a *scanning window*. Some algorithms based on this approach are found in [SK00, JKF01, RBK98]. This approach has the limitation that it is intractable to scan all of the possible scales and positions in an image, and therefore a discretisation must be introduced. The most successful scanning window face detection algorithm is the one introduced by Viola and Jones [VJ04, LM02, RF08]. This algorithm uses a cascade of classifiers which are based on simple haar-like features which can be efficiently computed at any scale or position using the so called integral image.

A different approach to face detection is the one based on image segmentation, or more specifically on skin colour detection. In this approach the idea is to find the regions of the image that correspond to skin and afterwards these regions are analysed to decide if they are faces or not. Some examples of this type of algorithms are [SB02, SR02, JS08] and a good review on this subject is [KMB07]. Another family of face detection algorithms are the ones that represent the face as a constellation of feature points. These algorithms are capable of giving a higher accuracy of the position of the face, however they need higher resolution images [HKK$^+$05].

2.2 Face Recognition Algorithms

As was mentioned in the start of the chapter, the abundant research on face recognition has left in the literature numerous face recognition algorithms. In the survey by Zhao et al. [ZCRP03], they divide the approaches in three different groups which are holistic approaches, feature based structural matching approaches and hybrid approaches. The holistic approaches take into account the whole face, examples of these are eigenfaces [Tur91], fisherfaces [BHK97], Support Vector Machines (SVM) [JMKL00], Bayesian face recognition [MJP00] and discriminative common vectors [CNWB05].

The feature based structural matching approaches determine the positions of different facial features such as nose, mouth, eyes, and use the individual features for structural matching. Examples of these are Hidden Markov Models (HMM) [NI98], Active Appearance Models (AAM) [ECT98], and the Elastic Bunch Graph Method (EBGM) [WFKvdM97]. It is worth mentioning that Gabor wavelets, play an important role for facial representation in these graph matching methods. The coefficients of these wavelets are robust to illumination change, translation, distortion, rotation, and scaling.

Finally the hybrid approaches analyse the entire face as well as its local features. Although these three groups can be used to categorise most of the face recognition algorithms, there are others that do not fit perfectly into any of them. An example of one of these algorithms which gives very good recognition results is the local features approach [PPJV01]. This approach uses local information, however the features extracted do not correspond to predetermined facial features and also the structural information is not used.

2.3 Face Recognition in Mobile Devices

All of the research done on face recognition could be applied to resource constrained devices. However the problem lies in the fact that the algorithms require a significant amount of resources. This problem can make the algorithms too slow for them to be useful in a real application. The research on face recognition has been aimed more at obtaining low errors and in few occasions the algorithms are analysed by the amount of resources they require. Unfortunately, there are very few publications specifically dealing with face recognition on resource constrained devices.

The initial efforts to develop face recognition for mobile devices applications have the idea that the mobile device is only used to capture the image used for recognition. This image is then transferred through a wireless network to a server which does the actual recognition, so the cost of the

recognition algorithm is not a concern. These systems commonly target face identification applications, which for instance can be very useful for law enforcement agencies. One example in this direction is the work of [WHH+02] in developing a face identification system for pervasive computing. Another example is the work of [ABBAQ05] also for face identification but based on GPRS. In the work of Hazen the same approach is used, however it includes the face and speech for doing identification [HWP03].

In this work, the interest is on doing all of the computing on the mobile device, without depending on a communication network. This is obviously a harder challenge. In [SHC05] a simple face recognition system based on 2D PCA is proposed, however it is only a preliminary analysis and it does not present any results. The work of Schneider [SEKK06], describes a face recognition system developed in C++ for a Symbian platform. It includes a face detector based on skin colour segmentation and a face recognition algorithm based on the position of facial features. The paper includes results with very few images, 21 to assess the face detection and 14 to assess the recognition, therefore the results are not reliable. In [yHPC+07] a face recognition system for mobile phones is presented however it requires a special camera because it is based on Near-Infrared light. The work presented in [NSK05], describes a system which does face detection using Viola and Jones and recognises faces by means of correlation filters. A complete analysis is made about the computation requirements and how much can be gained by using fixed point arithmetic.

Although there are few publications on the subject, there are several systems being developed and offered for face recognition on mobile devices. Since 2005 the Japanese firm Omron has been offering face authentication technology for PDAs and mobile phones [ISL06]. Another Japanese company which offers face recognition technology for mobile phones is Oki Electric Industry, and this technology has been selected by Vodafone KK for its mobile devices. On the other hand, Motorola also announced to be incorporating face recognition on mobile devices, however targeted at face identification for law enforcement agencies. The software used by Motorola is FaceIt ARGUS from i-Secure Group of Oakwell Engineering Ltd. These are some examples that show that for several years several companies have been working on and offering face recognition technologies for mobile phones. However up to now this technology has not been widely spread.

An industry that is more rapidly adopting face recognition technology is the digital camera industry. All of the most important digital camera producers now have models which incorporate face detection technology. The face detection is mainly used for automatically adjusting the image so that the people in the picture appear correctly focused. This application has the

advantage that if the detection fails, it does not affect much the usability of the camera.

The software developed for this thesis was programed in the Java language to be used in mobile phones supporting J2ME. On the Internet there is some face recognition software available for J2ME. At `www.drhu.org` there is an application called J2MEFaceSearch which lets the user select an image which is uploaded to a server. Afterwards another image can be uploaded to be recognised. Another application called J2MEFaceMatch is used for user authentication. These applications do not perform face detection, however there is a third application, J2MEFaceDetect for doing this. Finally it is worth mentioning that there is an open source project called JJIL (Jon's Java Imaging Library) which is a Java image processing library targeted towards mobile devices. It includes code for doing object detection using the Haar cascade models trained with the OpenCV library.

Chapter 3

Face Recognition in Mobile Devices

This chapter is dedicated to describing in somewhat detail the algorithms that have been chosen for doing face analysis and recognition in a mobile device. The first sections describe different algorithms, and for each, the reasons for choosing this particular algorithm are stated and discussed. The final section of the chapter gives some modifications, that apply all of the described algorithms, so that they are optimised for greater speed and less memory usage and thus are more suited for a mobile device.

3.1 Face Recognition Using Eigenfaces

In the literature the most cited and popular face recognition algorithm is eigenfaces [Tur91]. Eigenfaces is the first proposed appearance based face recognition algorithm and lots of work has been based on it. However doing face recognition using eigenfaces gives very poor results compared to state of the art algorithms. Nonetheless eigenfaces is still used, most of all as a reference. In other words eigenfaces is the baseline generally used for comparing recognition results, and this is the same reason why it is included in this work.

A common misconception around the eigenfaces is that it is a complete face recognition algorithm, however it is simply a representation of the face image information. Because the faces are originally represented as images, they inherently have a very high dimensionality, although the information actually lies in a lower dimensional space. What eigenfaces does is to reduce the dimensionality of the face images using a base learned by Principal Component Analysis (PCA) from a set of example face images. After the images

are projected to this low dimensional *face space*, it is still needed to use some sort of classifier in order to do face verification or face identification. The key advantage of the eigenfaces is that by doing dimensionality reduction, any pattern recognition algorithm may perform better because it helps to deal with the curse of dimensionality.

From the perspective of face recognition in a resource constrained device, eigenfaces is very favourable. The low dimensional representation of the face images is obtained by a simple linear projection. This means that it only requires simple multiplications, and no other more complex mathematical functions are used. Also, because the images are projected to a lower dimensional space, the amount of information that the algorithm needs to process is considerable low.

Eigenfaces can be summarised as the following. As input, it is assumed to have a face image that has already been detected, cropped and resized to a predetermined image size $W \times H$. First the face images are represented as a vector $\mathbf{x} \in \mathbb{R}^D$ were $D = WH$ and each pixel of the image is assigned to a different dimension of the vector. Afterwards, the vectors are dimensionally reduced to $\mathbf{y} \in \mathbb{R}^d$ using a projection base $B_d \in \mathbb{R}^{D \times d}$ obtained by PCA. The sub-index d indicates that the first d principal components are kept. The resulting vectors are obtained by

$$\mathbf{y} = \mathbf{B}_d^T(\mathbf{x} - \mu) \ . \tag{3.1}$$

To compute the projection base, a training set of face images is used $X = \{\mathbf{x}_1 \dots \mathbf{x}_N\}$. First the face mean, which is estimated by the empirical mean μ, is computed by

$$\mu = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \ . \tag{3.2}$$

Afterwards, the covariance matrix $\mathbf{\Sigma}$ is also estimated empirically and is computed by

$$\mathbf{\Sigma} = \frac{1}{N} \mathbf{X}_o \mathbf{X}_o^T \ , \tag{3.3}$$

where $\mathbf{X}_o = (\mathbf{x}_1 - \mu \ \mathbf{x}_2 - \mu \ \dots \ \mathbf{x}_N - \mu) \in \mathbb{R}^{D \times N}$. Finally the columns of the projection base are the eigenvectors $\mathbf{b}_i$ of $\mathbf{\Sigma}$ that correspond to the highest d eigenvalues λ_i, that is $\mathbf{\Sigma}\mathbf{b}_i = \mathbf{b}_i \lambda_i$. For more details on eigenfaces and PCA refer to [Tur91, Fuk90].

As mentioned earlier, eigenfaces can be used with several classification techniques which lead to different face recognition algorithms. A very simple face verification algorithm which works a bit better than the one originally proposed by Turk and Pentland [Tur91] would be the following. A set of face

images from people that are not users of the system is used as a world model. The images from the world model and the user are dimensionally reduced using eigenfaces. As a score to verify an image $\mathbf{x}'$, an approximation of the posterior probability of the nearest neighbour classifier is used

$$p(c|\mathbf{x}') = \frac{d^2(\mathbf{y}', \mathbf{y}_{w,nn})}{d^2(\mathbf{y}', \mathbf{y}_{w,nn}) + d^2(\mathbf{y}', \mathbf{y}_{c,nn})} \ , \tag{3.4}$$

were $\mathbf{y}_{w,nn}$ and $\mathbf{y}_{c,nn}$ are the nearest neighbours in the face space of the world model and the user model respectively.

3.2 Face Recognition Using Fisherfaces

Another very popular appearance based face recognition algorithm is fisherfaces [BHK97]. Just like with eigenfaces, fisherfaces is not a complete face recognition algorithm, it is just a method of representing face images in a low dimensional space. It has the same benefits as eigenfaces, the low dimensional representation is obtained by a simple linear projection, and it does not require a large amount of data to be stored. The difference is that fisherfaces is a discriminative approach, and therefore compared to eigenfaces it can give better recognition results. However the state of the art algorithms perform even better than fisherfaces.

Learning the projection base for fisherfaces is very similar to eigenfaces, the images are also represented as a vector $\mathbf{x} \in \mathbb{R}^D$ and are dimensionally reduced to $\mathbf{y} \in \mathbb{R}^d$. The main difference is the way the projection base is computed, which uses a combination of PCA and Linear Discriminant Analysis (LDA).

First a PCA projection base is computed $\mathbf{P}_{d'} \in \mathbb{R}^{D \times d'}$. Using the projected training set $\mathbf{Z} = \mathbf{P}_{d'}^T \mathbf{X}_o$, a second projection base $\mathbf{L} \in \mathbb{R}^{d' \times d}$ is obtained using LDA.

For doing LDA the between $\mathbf{S}_b$ and within $\mathbf{S}_w$ scatter matrices are needed and these are computed by

$$\mathbf{S}_b = \sum_{c=1}^{C} N_c (\nu_c - \nu)(\nu_c - \nu)^T \ , \tag{3.5}$$

$$\mathbf{S}_w = \sum_{c=1}^{C} \sum_{n \in N_c} (\mathbf{z}_n - \nu_c)(\mathbf{z}_n - \nu_c)^T \ , \tag{3.6}$$

were C is the number of classes, N_c is the number of vectors of class c and

$$\nu_c = \frac{1}{N_c} \sum_{n \in N_c} \mathbf{z}_n \ . \tag{3.7}$$

The vector ν is the global mean of Z, however because of the previous subtraction of μ in the PCA projection then $\nu = 0$.

The columns of the LDA projection base are the generalised eigenvectors $\mathbf{a}_j$ of $\mathbf{S}_b$ and $\mathbf{S}_w$ that correspond to the highest d generalised eigenvalues κ_j, that is $\mathbf{S}_b\mathbf{a}_j = \mathbf{S}_w\mathbf{a}_j\kappa_j$. For more details on fisherfaces and LDA refer to [BHK97, Fuk90].

The two projection bases can be combined to make a single projection base, this is done by

$$\mathbf{B}_d = \mathbf{P}_{d'}\mathbf{L}_d \ . \tag{3.8}$$

The first projection by PCA is a necessary because generally the dimensionality of the images is higher than the number of images in the training set. If this happens, the within scatter matrix is singular and LDA cannot be performed. Also by first doing dimensionality reduction with PCA a filtering effect is obtained that removes some of the noise in the images. For this reason it is a good idea to use the number of dimensions used by PCA as another parameter which can be varied to optimise the performance of the algorithm.

Fisherfaces can also be used to develop several face recognition algorithms. For face verification the same methodology as presented in the previous section for eigenfaces could be used.

3.3 Face Recognition Using Local Features

The only face recognition algorithm which was considered in this work which has a recognition accuracy comparable with other state of the art algorithms is the one known as Local Features [PPJV01]. Between the current best face recognition algorithms, it is very difficult to choose which one is better suited for a constrained device. All of the algorithms have very high requirements either in the amount of processing or the amount memory or even both. The local features algorithm requires lots of processing and memory, however just like the two previous algorithms described, it is based on a simple linear projection and distance calculations. Furthermore the amount of resources required depend completely on a few parameters which can be easily adjusted making a compromise between performance and computational requirements.

In the local features face recognition algorithm, each image is represented by a set of feature vectors $f = \{\mathbf{x}_1, \ldots \mathbf{x}_F\}$ $\mathbf{x}_f \in \mathbb{R}^{w^2}$. The set of feature vectors is obtained from all of the possible regions of the image of a window size $w \times w$. Depending on the size of the image and the size of the features, the amount of feature vectors that are extracted can be very large, and this is normally the case in order to obtain a high recognition accuracy. This

amount of feature vectors can be reduced by using only the regions spaced by s pixels, and also by selecting the F highest features according to a certain criterion, for example the variance of the feature.

As can be observed, the information in the extracted local features is redundant because the features can overlap. Therefore the space required to store this information is larger than the space required for the original image. This is somewhat alleviated by using PCA to dimensionally reduce the local features, however it still requires more space than was originally required. The PCA base is learned using the extracted local features from all of the images in a training set, that is the training set would be $X = \{\mathbf{x}_{1_1}, \ldots \mathbf{x}_{1_F}, \ldots \mathbf{x}_{N_F}\}$, were N is the number of images and F the number of extracted features per image.

The local feature representation can be used for face identification or face verification, however in this work the only interest is on face verification algorithms. In the literature the standard approach of doing face verification using local features is the following [PPJV01, VP07, VPJV08]. A set of face images from people that are not users of the system are used to construct a world model which are all of the local features from these images. The user models are the extracted features from the training images of that particular subject. When a test image is presented to the system, as a verification score the following approximation to the posterior probability of the client c given the image $\mathbf{x}'$ is used

$$p(c|\mathbf{x}') = \frac{1}{F}\sum_{i=1}^{F} H[d^2(\mathbf{y}'_i, \mathbf{y}_{c,nn}) - d^2(\mathbf{y}'_i, \mathbf{y}_{w,nn})] \ , \tag{3.9}$$

were $\mathbf{y}_{w,nn}$ and $\mathbf{y}_{c,nn}$ are the nearest neighbours of the world model and the user model respectively, and $H[]$ is the Heaviside step function.

3.4 Discriminant Projections and Prototypes

The Learning Discriminant Projections and Prototypes (LDPP) algorithm is a classifier learning approach which has been recently proposed by the author of this thesis [VP08]. Although LDPP is a very general pattern recognition approach, it has been specifically designed so that it works well in the recognition of images.

As was mentioned before in the section on eigenfaces, images have a very high dimensionality even though the objects that the images represent will lie in a much lower dimensional space. This leads to a problem while trying to learn a pattern recognition model, which is the so called curse

of dimensionality. The curse of dimensionality in the context of machine learning means that in order to obtain a reliable solution, the number of parameters that need to be estimated must be low compared to the number of data samples available. For example if an image has a dimensionality higher than the number of training samples, then just a model as simple as a single layer perceptron has more parameters than the number of data samples.

The LDPP algorithm tries to handle the high dimensionality of the feature vectors in the following way. Suppose that the input feature vectors have a dimensionality of D. LDPP simultaneously learns two things, first it learns a linear projection base $\mathbf{B} \in \mathbb{R}^{D \times R}$ for dimensionality reduction, were R is the number of dimensions of the reduced space. It also learns a reduced set of prototypes $P = \{\mathbf{p}_{1,1} \dots \mathbf{p}_{1,M_1}, \mathbf{p}_{2,1} \dots \mathbf{p}_{C,M_C}\}$ for nearest neighbour classification, were C is the number of classes, $\mathbf{p}_{c,m} \in \mathbb{R}^R$ and $M = \sum_{c=1}^{C} M_c$ is the total number of prototypes.

Although the matrix $\mathbf{B}$ has a size of R times the number of dimensions of the input vectors, if this matrix is restricted to be orthonormal, then the actual number of degrees of freedom for this matrix is R. As a result to this orthonormal restriction, the number of parameters that LDPP estimates is $R(1 + M)$. Reasonable values for R and M generally give a number of parameters which is considerably lower than the number of data samples available.

For more details on the LDPP algorithm refer to [VP08].

This algorithm has been used in this work to learn several recognition models. One model was learned to do face detection and quality of the detected face. Other models were learned to do face gender recognition and face expression classification. This algorithm can also be used for face verification. A projection base is learned using some training images, however the prototypes obtained are discarded [VP08]. Afterwards it is the same as for eigenfaces and fisherfaces. The recognition results are similar to the ones obtained with fisherfaces so there are no results included for this approach.

3.5 Algorithm Optimisations

This section discusses some particular properties of resource constrained devices and how can the algorithms be modified so that they are optimised for this type of environments. The type of devices that are being considered in this work are mobile phones, Personal Digital Assistants (PDAs), digital cameras, or any smaller device, even laptop computers which can benefit from lower power consumption and longer battery spans. There are

two key restrictions that mobile devices have which need to be addressed for improvement of the algorithms. First of all, mobile devices have small amounts of memory, both for program execution and for storage of program data. The other restriction is that the processors are slower than the ones that are found in general purpose computers. Also the processors generally lack a native floating point arithmetic unit, and therefore the execution of operations involving real numbers are even slower.

In the previous sections some reasons were given for choosing each of the algorithms. One common reason for choosing the previous algorithms was that in the testing phase they all rely on very simple mathematical operations. Namely the only operations that the algorithms require are multiplications and additions or subtractions. In fact all the algorithms have in common that they first perform a dimensionality reduction using a linear projection base, and afterwards a nearest neighbour search using the euclidean distance is carried out. These are the two aspects that are going to be optimised both in speed and in memory requirements, and therefore it improves all of the algorithms.

3.5.1 Memory Usage Reduction

Because dimensionality reduction using a linear projection base requires the use of floating point arithmetic, it is common to use 32-bit or 64-bit floating point variables to store the feature vectors. To optimise the memory usage, in this work it is proposed to use for each value only one 8-bit signed integer (byte) to store all of the feature vectors, both for permanent storage and for storage during program execution. By representing the data using 8-bit values, in contrast to 32-bit variables the space required to store the feature data is reduced to a quarter. This improvement is without taking into account other data that is needed to represent matrices or other more complex data structures. However it is still a great improvement and it does make difference not only in the amount of memory needed but also in the seed of computation, as can be observed in the results in the next chapter.

The decision to store the data using only 8-bit signed integer values obviously has some consequences. The first consequence that comes to mind is that the precision of the represented values is greatly diminished and this can affect the recognition performance of the algorithms. In the following chapter it is shown that at least for the algorithms that were tested, the performance of recognition is not significantly affected by this reduction of precision. This is precisely one important contributions of this thesis. The real question that needs to be answered is how to convert a floating point value to an 8-bit integer so that the limited precision is well used. This

question will be answered in the following section.

3.5.2 Dimensionality Reduction Optimisation

If the feature vectors are all stored as 8-bit integers, including the vectors before and after the dimensionality reduction, then the question remains on how to perform this process if it requires floating point arithmetic. One option would be to convert the original vectors to float variables, if it was necessary, then do the dimensionality reduction and finally convert the result to 8-bit integers. However this is not an optimal solution, in part because it requires more computation than it was originally necessary, and also because using floating point arithmetic is too slow in the resource constrained devices that are being targeted. The solution that is needed is, to store the input feature vectors as 8-bit integers, do the dimensionality reduction process using only integer arithmetic and directly obtain a result that is limited to the 8-bit integer range.

As input to the algorithms it can expected to have images with 8-bits of depth per channel. When having as input a face image it is common to do some type of processing so that the effects of different illumination conditions are somewhat removed. Some of these techniques give as output real valued numbers, and in such a case it would be necessary to convert back to 8-bit. However among the different illumination normalisation techniques, there are ones that are based on integer arithmetic and perform very well. The best one is the Local Histogram Equalisation (LHE) [VP05], which it is known to work well with eigenfaces and fisherfaces. For local features face recognition, a simple Histogram Equalisation per feature is enough to obtain a good result [VP07].

Up to now it can be expected to have as input to the dimensionality reduction 8-bit valued vectors. So the current question is how can the dimensionality reduction be performed using integer arithmetic. For this it is necessary to analyse some of the properties of the projection bases. The bases obtained from PCA and LDA come from the use of an eigenvalue decomposition. This assures that the bases are orthonormal. The same can be said about the base obtained from LDPP because it is being forced to be orthonormal. The interesting fact that this tells about the bases is that all of the values are between one and minus one. The actual projection is done by an inner product of the vector with the base vector. This can be approximated with integer arithmetic by multiplying the base vector by a large number K, so that it can be rounded to an integer (but not an 8-bit integer), and afterwards the result is divided by this factor. In mathematical

terms, this is

$$\mathbf{y} = \mathbf{B}^T(\mathbf{x} - \mu) \approx \frac{round(K\mathbf{B}^T)(\mathbf{x} - \mu)}{K} \ . \tag{3.10}$$

In the previous equation the mean μ has been subtracted even though it is not necessary because when computing distances between vectors the means cancel out. The reason to keep the mean is so that when the vectors are projected, the values will be centred around the origin and thus making more predictable which will be the output range.

If the input vector $\mathbf{x}$ is composed of integer values, which will be the case, then all of the operations of the projection are done using integer arithmetic. However even if the input vector is represented by an 8-bit integer, the result is not guarantied to be in the same range as the input. If the maximum value of the input is $M = max(|x_i - \mu_i|)\ i = 1, 2 \ldots D$ then the maximum value of a projected component is given by $M_p = M\sqrt{D}$. It can be seen that for high dimensional vectors M_p can be a lot higher than the original maximum M. This value could be used for scaling the projected output to the desired range.

Although M_p is the theoretical projected maximum, it is quite pessimistic because it is an extreme value that does not necessarily contain relevant information, in fact it is possible that the data never reaches these values. A better way to rescale the projected output and take the most advantage of the limited range is to use the actual training data to choose an optimum value. The final projection is given by

$$\mathbf{y}' = \frac{\mathbf{B}_d^T(\mathbf{x} - \mu)}{2^m} \approx fitrange\left[\frac{round(2^n\mathbf{B}_d^T)(\mathbf{x} - \mu)}{2^{n+m}}\right] \ , \tag{3.11}$$

were n and m are positive integers and $fitrange$ is the operation that sets to the maximum or minimum the values that fall outside the range or otherwise the values is not modified. Choosing the factors to be powers of two is so that the operation can be done a bit faster using bit displacement operations.

For choosing the value of m from the training data, a further analysis of the projection base must be made. In PCA the first component is the one that captures most of the variance of the data, this means two things. First, that all the other components will have lower values, and second that the first component is the most important one to represent the whole vector. For LDA the situation is a bit different, the first component is the one that maximises the between scatter and minimises the within scatter. Anyhow for both techniques, the first component is the most important one, so it is a good choice to use only this component to find an optimal value for m. For the bases obtained from LDPP there is no information about which is

the most important component, so in this case the component which has the most variance in the training data is used. In the experiments, the value of m was chosen so that three standard deviations of the chosen component are kept within the range.

3.5.3 Efficient Nearest Neighbour Search Optimisation

In the previous sections it was proposed that in order to optimise the amount of memory needed, the features should be stored and used as 8-bit signed integers. After the dimensionality reduction, the next step in the algorithms is to do a nearest neighbour search using the euclidean distance. Going on with the idea that the computations should be done using integer arithmetic and the feature vectors in 8-bit integers, then the same must be done for the nearest neighbour search. Other metrics different from the euclidean distance could give better recognition results, however this would make it more difficult to optimise.

To find the nearest neighbour using the euclidean distance it is actually only necessary to compute the square euclidean distance. This is very convenient because the computation of a square root is a much slower operation. This way it is straight forward to compute the distance using only integer arithmetic because it only requires subtractions multiplications and additions. Furthermore given that the input are integer valued vectors, the squared euclidean distance is always an integer.

The approach could be to use brute force and compute the distance to all of the reference vectors and find the smallest one. However the nearest neighbour search can be done a lot more efficiently using a *kd*-tree structure. To build a good *kd*-tree from some feature vectors, it is necessary to use floating point arithmetic however the structure can be represented using only integer values.

For the local feature face recognition algorithm, it is sufficient to have an approximate nearest neighbour search. Therefore the nearest neighbour search can be performed by a fast approximate nearest neighbour search algorithm which also uses a *kd*-tree structure to store the set of features from the training objects. In [AMN+98], the concept of $(1+\epsilon)$-approximate nearest neighbour query is introduced. A point p is a $(1+\epsilon)$-approximate nearest neighbour of q if the distance from p to q is less than $(1+\epsilon)$ times the distance from p to its nearest neighbour. This concept works exactly the same if the features are composed of integer values, and there is no need to use floating point arithmetic for doing the nearest neighbour search.

Chapter 4

Results

One of the main objectives of this work was to assess if the optimisations proposed in the chapter 3 result in a significant performance improvement and how much is actually gained. With this aim, in this chapter some analytical and experimental results are presented.

To be able to assess the proposed optimisations, all of the algorithms were implemented in Java using for the feature vectors 64-bit and 32-bit floating point variables (double and float) and 32-bit, 16-bit and 8-bit signed integers (int, short and byte). The results for 64-bit floating point and in some places the 32-bit integers variables were omitted because it did not contribute any meaningful results.

4.1 Effect of Optimisations on Recognition Performance

This section presents the results of an experiment aimed at showing how much is the recognition performance of the algorithms affected by doing an integer based dimensionality reduction and an integer based approximate nearest neighbour search. In this experiment the XM2VTS multi-modal database was used [MMK+99]. This database is composed of eight images for each of the 295 subjects, which were taken at four sessions distributed over a period of four months (two shots per session). This database is mainly intended to assess the performance of identity verification algorithms, for which the Lausanne protocol has been defined. This protocol specifies the performance measures to be used and two different configurations that indicate which images are to be used as training, evaluation and test. For more details on the Lausanne protocol, refer to [LM98]. The experiments were carried out only for configuration 1 of this protocol.

Table 4.1: Total error rates (%) for face verification on the XM2VTS database with the Lausanne protocol configuration I for different data types.

Algorithm	**float**		**short**		**byte**	
	eval	test	eval	test	eval	test
Eigenfaces	24.3	19.0	24.3	19.0	24.0	19.1
Fisherfaces	15.0	11.0	15.0	11.0	15.0	11.0
Local Features	3.9	4.7	3.9	4.7	3.9	4.8

Each of the face verification algorithms presented in the chapter 3 was tested. The results for float, short and byte can be observed in the table 4.1. There is a difference between the results for short and byte because for short the $fitrange$ operation is not needed for approximating the projection, therefore it was not applied.

These results for the total error rates could be compared with other algorithms found in the literature [MKS+03, MKS+06], however the results presented here are not the best ones that can be achieved with the algorithms. The parameters of the algorithms were chosen arbitrarily because the current objective was not obtain the best result possible but to see the effect of the approximations.

The results for eigenfaces and fisherfaces were obtained using images of size 24×24 and globally histogram equalised. For eigenfaces, the images were reduced to 64 dimensions. For fisherfaces, the images were reduced to 64 dimensions by PCA and afterwards reduced to 48 dimensions by LDA.

For the results with local features, the images were of size 75×75 and preprocessed using local histogram equalisation for a window of size 5×5 [VP05]. The extracted local features were for a window of size 13×13 with a sub-sampling of 1 for the training images and 2 for the test images. The 50% of the features with highest variance were kept. Finally the features were reduced to 30 dimensions.

As can be observed in the table, there is very little difference between the results for the different data types. For fisherfaces, the is no difference up to the second decimal. The differences for all the algorithms are so small that they are not statistically significant. This result confirms that the proposed approximations do not make a noticeable negative impact on the recognition performance of the algorithms.

Table 4.2: Face verification results on the BANCA database for the Matched Degraded experiment.

Algorithm	**TER@EER** [%]	**TER@FAR=0.1%** [%]	**AUC**
Eigenfaces	27.9	57.0	0.932
Fisherfaces	26.9	37.2	0.940
Local Features 48×48	20.2	29.8	0.970
Local Features 64×64	13.7	17.8	0.988

4.2 Face Verification on Low Quality Images

In most of the literature related to face verification, the experiments are done using high quality images taken under controlled illumination. The idea in this work is to use the algorithms in mobile devices were the cameras available are of much lower quality and the images are taken in uncontrolled environments. There are few results published using low quality images and there is no public database of images taken from mobile devices. However there is a public database in which the images were taken using a webcam in a somewhat uncontrolled situation. It is the BANCA database [BBBB$^+$03]. In the BANCA database 208 subjects were recorded in three different scenarios, controlled, degraded and adverse, taken over a period of three months. The degraded scenario was the one that used the webcam. The associated BANCA protocol [BBBB$^+$03] defines several experiments for assessing face verification algorithms.

The experiment in the BANCA protocol that best matches the situation of recognition in a mobile device is the Matched Degraded, which uses for training and testing only images from the webcam. This experiment was used to estimate the face verification performance of the algorithms. For each of the algorithms the parameters were varied and different illumination normalisation techniques were tried. Three different performance measures were computed, namely the total error rate at the equal error rate point (TER@EER), the total error rate at a false acceptance rate of 0.1% (TER@FAR=0.1%) and the area under the ROC curve (AUC). The results are reported on the table 4.2.

Since it is common for a verification system to require a low false acceptance rate, the parameters of the algorithms are the ones that minimised the TER@FAR=0.1% in the development sets. The results in the table for eigenfaces and fisherfaces are for images of size 32×32 using local histogram equalisation (LHE) for a window of size 4 [VP05]. For eigenfaces the features

were reduced to 48 dimensions, and for fisherfaces reducing to 96 dimensions by PCA and to 29 dimensions by LDA.

Two results are presented for local features. The first one is faster and produces smaller models. The image size was 48×48, the feature window size was 9 pixels extracted every 2, the normalisation was histogram equalisation per feature and the dimensionality was reduced to 24. The second configuration for local features was for images of size 64×64, features of 13 pixels extracted every 2, LHE for a window of size 6 and dimensionality reduced to 24.

As expected, the best results are for local features, followed by fishefaces and eigenfaces. For the Matched Degraded experiment of the BANCA some comparable results can be found in [SKKM03]. The best algorithm reported is for fisherfaces with a support vector machine as classifier which has a TER@EER of 15.1%. The result obtained here for local features is better, although it is somewhat an unfair comparison because the parameters here were optimised for the FAR=0.1% operating point. However the configuration which gives the lower error rate may be unpractical for a mobile device because it requires a lot of computation. The results in the following sections for the local features are only for the faster configuration.

4.3 Memory Requirements

In order to reduce the amount of memory required, both for processing and for storage of data, in the chapter 3 it was proposed to represent all of the feature vectors using 8-bit integers. Assuming that the baseline alternative is to represent the feature vectors using 32-bit float variables, then storing them using 8-bits reduces the space required to one quarter of what was needed. This is a correct estimate of the reduction of memory needed to store data such as the user models, nonetheless this is leaving out a possible final step of data compression. From this we can estimate the space required to store the world model and user models given a certain configuration of the algorithm.

Take for example that we were to use eigenfaces reducing dimensionality to 30, having 300 images to learn the world model, and 3 images to learn the user models. Then the size of the models using 8-bit integers would be $300 \times 30 = 9000$ bytes for the world model and $3 \times 30 = 90$ bytes. As a general rule, using 8-bit integers the size of the models for eigenfaces and fisherfaces would be $N \times d$ and for local features would be $F \times N \times d$, were N is the number of images used for training, d the number of dimensions in the reduced space and F the number of local features extracted per image.

Estimating the amount of memory required for the processing of the al-

gorithms is a bit more complicated, and it actually is implementation dependent. The gain obtained by passing from 32-bit floats to 8-bit integers is not a quarter. This is because if there is a large amount of feature vectors, a *kd*-tree structure is required to do an efficient nearest neighbour search. As mentioned before a *kd*-tree structure for discrete data can be generated using only integers, however not all of the information can be stored using only 8-bit precision. Furthermore, a *kd*-tree structure requires the use of a lot of pointer data which depend on the architecture of the system and the structure of the generated tree.

Using the implemented software in Java, several *kd*-tree structures were built using real data extracted from face images of the XM2VTS database [MMK+99]. The dimension of the feature vectors and the amount of vectors in the structure were varied. The graphs in figures 4.1, 4.2 and 4.3 show the size of the *kd*-tree structure for 16, 32 and 64 dimensions respectively. Obviously the size of the *kd*-trees is the same for float and int, for short is smaller and for byte is the smallest and they all increases linearly with the number of vectors.

As can be observed in the graphs the curves are linear, this means that for the different data types the relative size of the *kd*-trees is constant for any number of vectors and varies with the dimensions of the feature vectors. This relationship can be observed in the figure 4.4. This graph gives a better idea of how much the memory usage is reduced by representing the data as 8-bit integers.

4.4 Computation Time of the Algorithms

As was noted in the section 3.5, all of the face recognition algorithms rely on first doing a dimensionality reduction by a simple linear projection and afterwards a nearest neighbour search is performed. Apart from this, each algorithm has to do some image processing and a feature extraction process. Each of these tasks have been evaluated and the computation times were measured first on a desktop computer and afterwards on a mobile phone.

4.4.1 Efficient Nearest Neighbour Search on a PC

If there is a large amount of feature vectors, which is the case for the world models of all of the algorithms and the user models for local features, the nearest neighbour search can take a significant amount of time. For this reason an efficient nearest neighbour search algorithm based on a *kd*-tree structure was implemented.

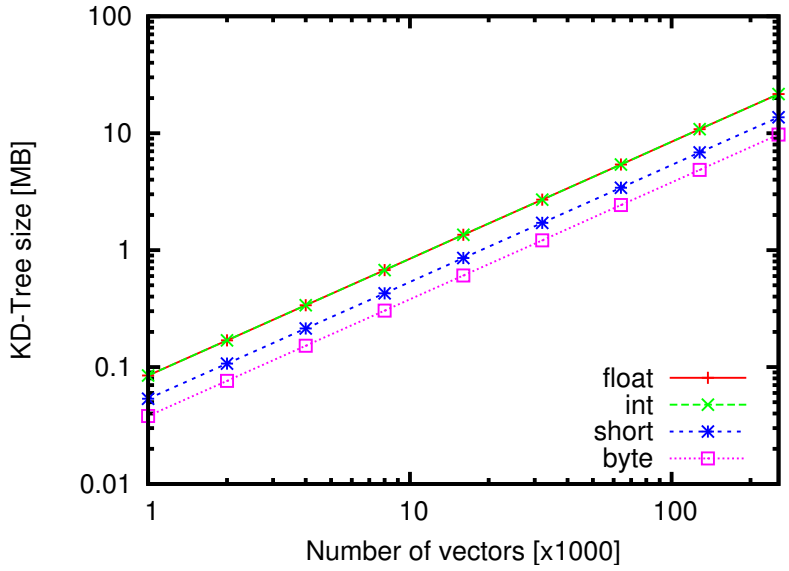

Figure 4.1: Size of the *kd*-tree structure for feature vectors of 16 dimensions for different data types.

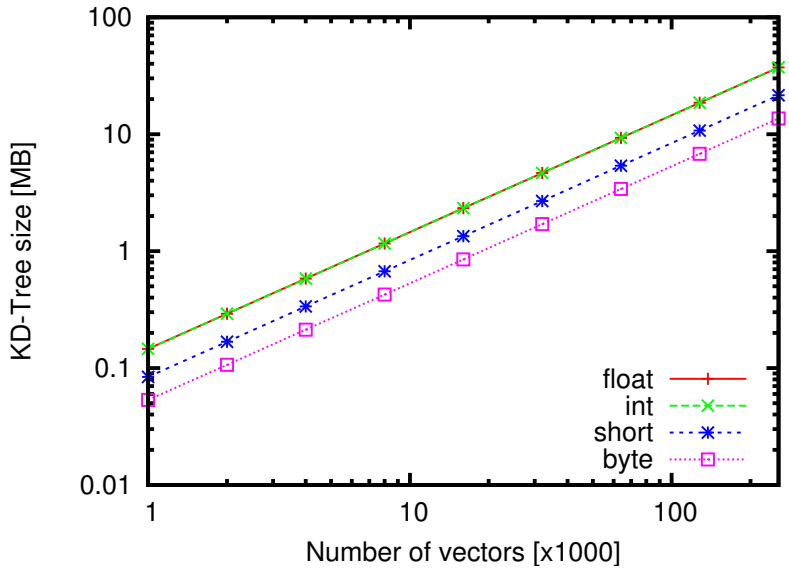

Figure 4.2: Size of the *kd*-tree structure for feature vectors of 32 dimensions for different data types.

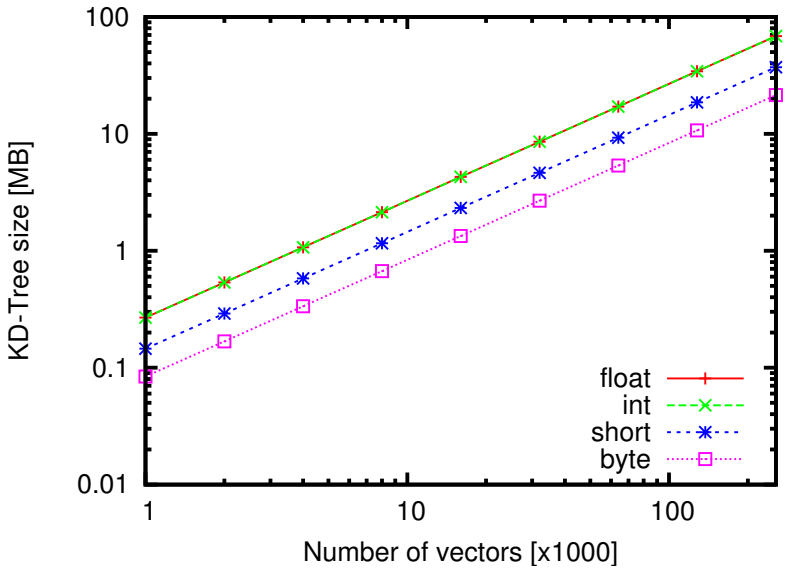

Figure 4.3: Size of the *kd*-tree structure for feature vectors of 64 dimensions for different data types.

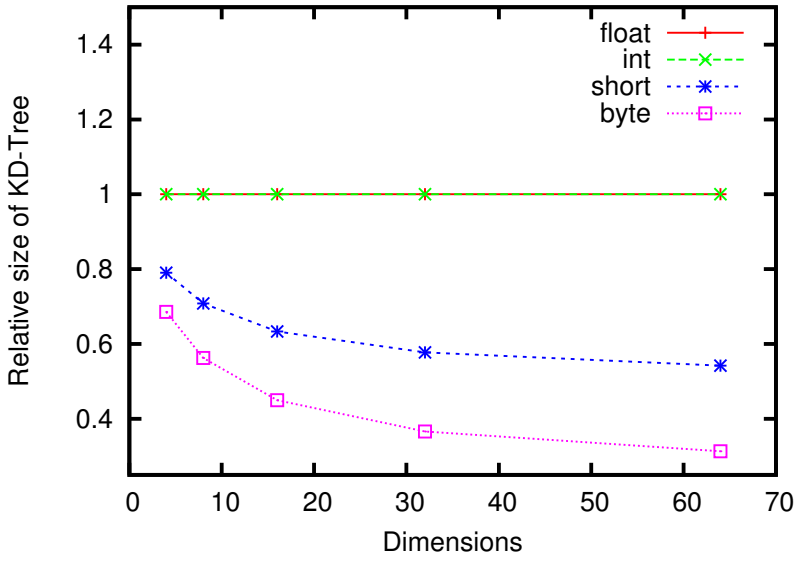

Figure 4.4: Relative size of the *kd*-tree structure for different data types.

The implementation of the *kd*-tree nearest neighbour search was first tested on a desktop computer running Linux with an Intel(R) Pentium(R) 4 CPU 3.20GHz processor. The Java virtual machine used was the Sun Java 2(TM) Runtime Environment Standard Edition (build 1.5.0_11-b03). For this experiment 1400 images from the XM2VTS database [MMK+99] were used, 1000 for training and 400 for testing. The face in each image was cropped using the eye coordinates that are publicly available, resized to 32×32 pixels and converted to grayscale. Using the methodology explained in the section 3.3, local features were extracted for a window size of 13×13 pixels. In order to obtain training sets of different sizes, the amount of local features extracted from each image was varied.

Figures 4.5, 4.6 and 4.7, show the results of the average nearest neighbour search time for features of 16, 32 and 64 dimensions respectively. The experiments were repeated ten times and the results averaged. The graphs also include the 95% confidence intervals, although they are too small to be seen.

In all of the graphs it can be observed that there is a large gap between the search time for float and the others. This is mostly due to the difference in speed of floating point arithmetic and integer arithmetic. When there are more dimensions, the gain in speed seems to be larger. For 32 and 64 dimensions the integer based search is about twice as fast as with float. Finally it is interesting to note that for a large amount of training vectors, byte becomes slightly faster than int and short, and the gap with float stays constant.

4.4.2 Face Verification on a PC

On the same desktop computer used to compute the nearest neighbour search times, the average time for each step of the verification process was estimated. This estimation was done using the XM2VTS configuration 1 for the same parameters of the algorithms as in the section 4.1. In the table 4.3 are the average times in milliseconds for a single verification of an image. The extraction process includes the preprocessing and the feature extraction. These times are for each algorithm and data type. The behaviour is as expected, being for byte about 1.5 times faster than float.

As mentioned earlier, the parameters were chosen somewhat arbitrarily because the objective was to compare between the use of the different data types. Furthermore, it is difficult to find the verification times of other algorithms found in the literature, so neither can these be compared.

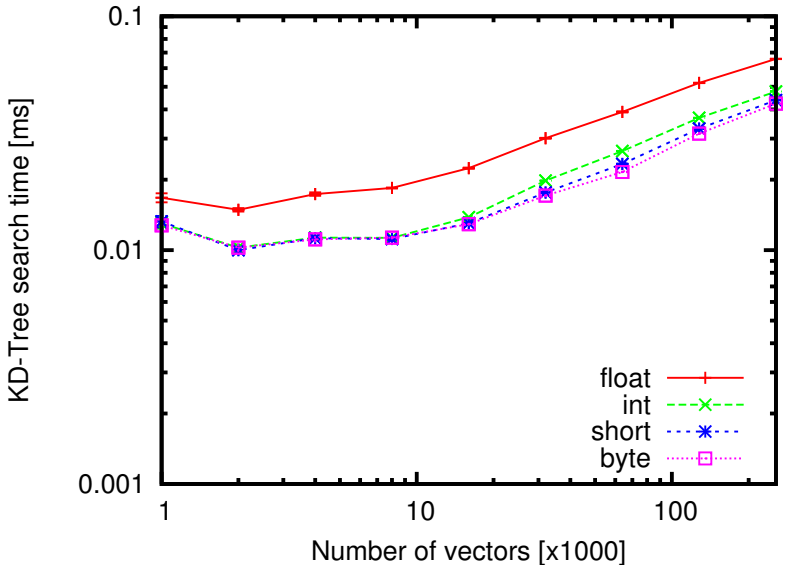

Figure 4.5: Average nearest neighbour search time for $\epsilon = 2$ and feature vectors of 16 dimensions for different data types on a PC.

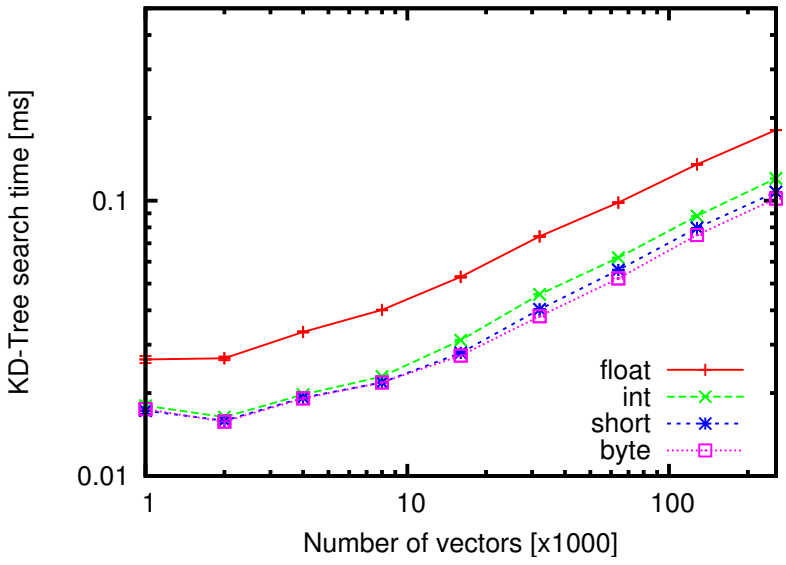

Figure 4.6: Average nearest neighbour search time for $\epsilon = 2$ and feature vectors of 32 dimensions for different data types on a PC.

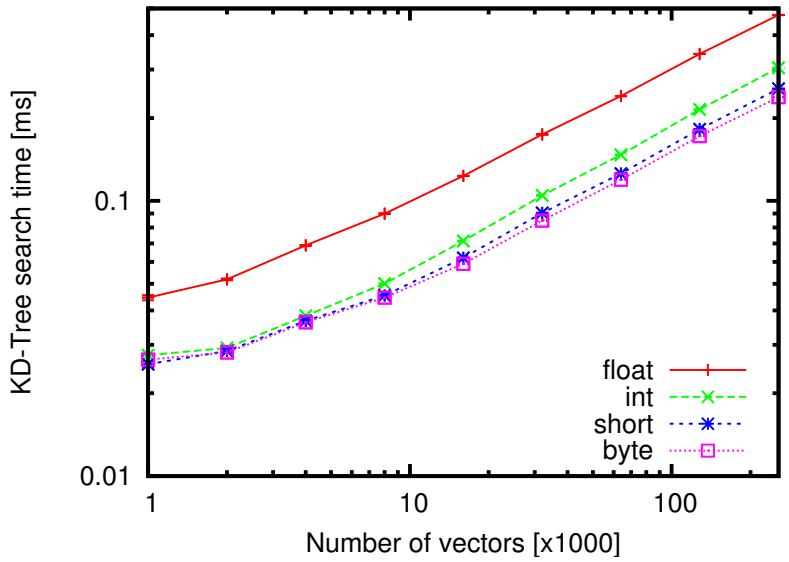

Figure 4.7: Average nearest neighbour search time for $\epsilon = 2$ and feature vectors of 64 dimensions for different data types on a PC.

Table 4.3: Verification times (ms) for the different algorithms and data types on a PC.

Algorithm		extraction	projection	score	Total
Eigenfaces	float	0.040	0.320	0.220	0.580
	short	0.035	0.245	0.135	0.415
	byte	0.028	0.288	0.135	0.451
Fisherfaces	float	0.040	0.243	0.168	0.451
	short	0.035	0.188	0.105	0.328
	byte	0.028	0.220	0.105	0.353
Local Features	float	45.43	22.43	1026.67	1094.50
	short	44.03	16.16	765.76	825.95
	byte	43.97	16.88	653.72	714.57

4.4.3 Face Verification on a Mobile Phone

Up to now the processing times presented have been for the implemented software running on a desktop computer. These results have proved that the proposed optimisations really work under these circumstances. Nonetheless the objective is to determine how much is gained when the algorithms are running on a mobile device and if the processing times are adequate for a real world application. The ideal solution would be to test the software on several mobile devices, however during the realisation of this work only one device was available for testing. All of the results presented here are for a Nokia N70 mobile phone.

Doing experiments on a mobile phone involving thousands of images or feature vectors is not possible due to the limited memory that is available. Because of this, the results are not averages obtained for a standard experiment using a well known database. The results were obtained using real data extracted from face images, however in some cases only one image was used and the processes repeated over and over.

Projection Times:

The graph on the figure 4.8 shows the average projection time for features of an original dimensionality of 1024 for the different data types. The original dimensionality is the same as the one that would be required for the parameters of eigenfaces and fisherfaces presented in the section 4.2.

As can be observed in the graph, the processing times for byte are more than five times faster than for float. This is a very encouraging result, however in eigenfaces and fisherfaces a verification of an image involves only one projection and therefore it does not account for much of the total processing time. The graph on the figure 4.9 shows the projection time for features of 81 dimensions, a value common for local feature face recognition. The gain from float to byte is slightly lower than the previous case, however it is still very high. Furthermore for local features several feature projections are needed for verifying one image, therefore the impact on the total time can be higher.

Nearest Neighbour Search Times:

The graphs on the figures 4.10 and 4.11 show the average nearest neighbour search time for features of 16, 24 and 32 dimensions respectively. In this case the nearest neighbour searches are about six times faster than the floating point searches. It is somewhat unexpected that the gain from integer arithmetic is higher for nearest neighbour searches than for feature projections.

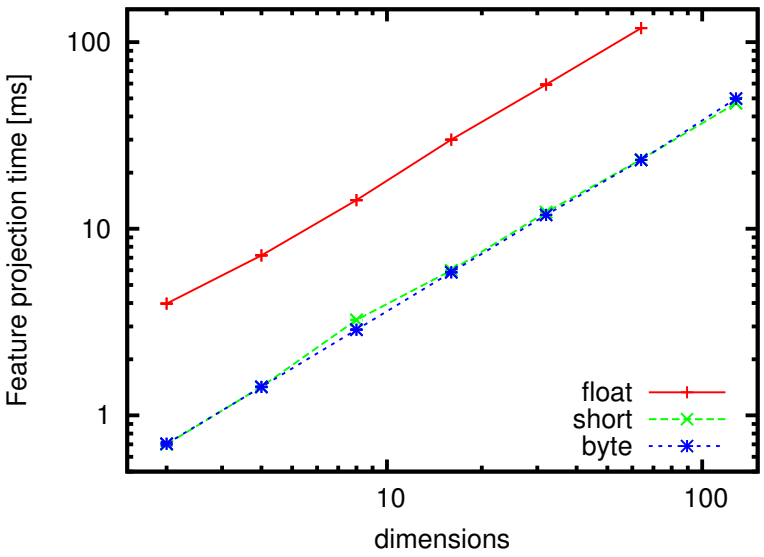

Figure 4.8: Average feature projection time for $D = 1024$ for different data types on a Nokia N70.

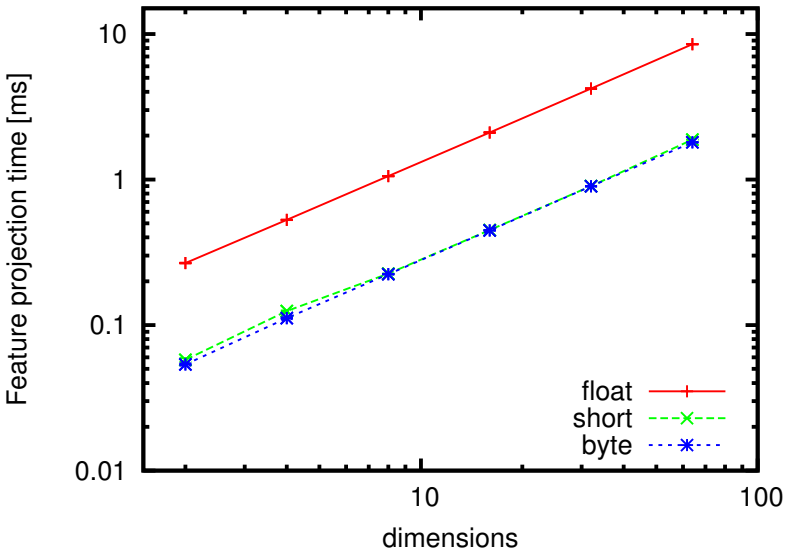

Figure 4.9: Average feature projection time for $D = 81$ for different data types on a Nokia N70.

Projections require exclusively floating point operations unlike the nearest neighbour searches.

kd-tree Structure Construction Times:

Depending on the amount of reference feature vectors, the nearest neighbour searches may or may not require a kd-tree structure for being more efficient. The user models for eigenfaces and fisherfaces only have a handful of vectors, and therefore a kd-tree is not necessary. On the other hand, the world models for all of the algorithms and the user models for local features do require a kd-tree, and this means that these structures have to be constructed. The world models are fixed, and therefore the corresponding kd-trees do not have to be built on the mobile device. However, this is not the case for the user models in the local features algorithm. Figures 4.12 and 4.13 show the average kd-tree construction times for each data type and varying the number of vectors.

Face Verification Times:

In the table 4.4 are the execution times for each data type and algorithm using the parameters from the experiment on the section 4.2. For local features, the faster configuration was used.

For eigenfaces and fisherfaces the gain due to the optimisations is very little. For these configurations most of the processing time is taken by the local histogram equalisation which is integer based and was not possible to optimise. On the other hand, the gain obtained for the local features is almost five times better. This takes the verification time from almost four seconds, which can be very frustrating for a user of the system, to less than one second. The 95% confidence intervals are included which confirm the significance of the result.

4.5 Face Verification Statistics

Using the results from the previous sections, some statistics of the face verification algorithms have been estimated and organised, see table 4.5. The statistics are for the parameters from the experiment on the section 4.2 and only for the optimised versions of the algorithms. The data gathered in table are the total error rate at a false acceptance rate of 0.1%, the size of the verification user models, the memory required by the algorithms, and the enrolment and verification times on a Nokia N70. The user model sizes and enrolment times were estimated for three training images.

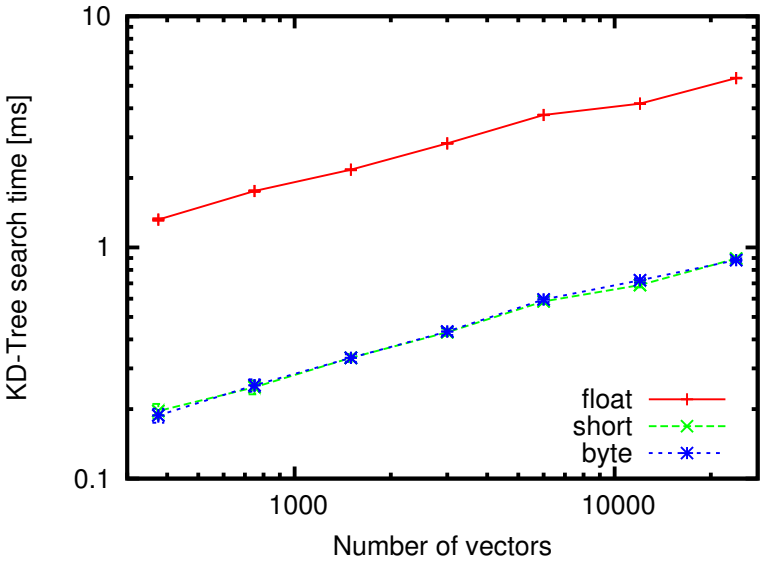

Figure 4.10: Average nearest neighbour search time for $\epsilon = 2$ and feature vectors of 24 dimensions for different data types on a Nokia N70.

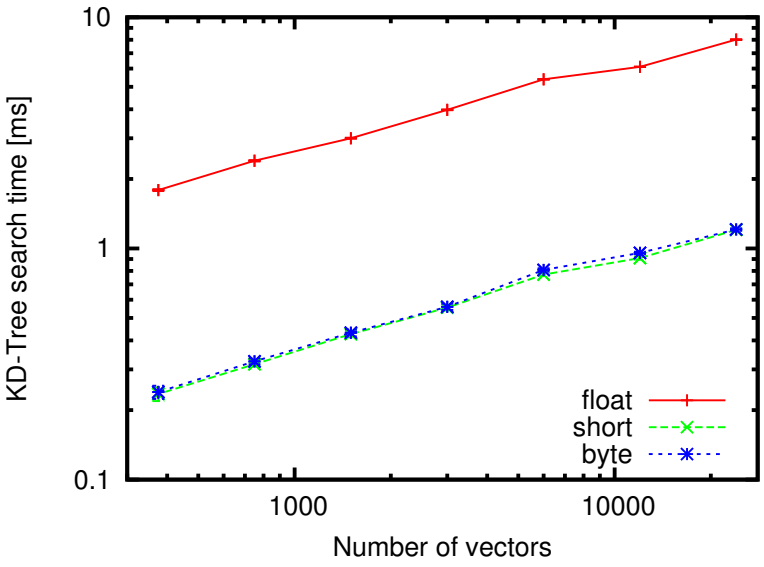

Figure 4.11: Average nearest neighbour search time for $\epsilon = 2$ and feature vectors of 32 dimensions for different data types on a Nokia N70.

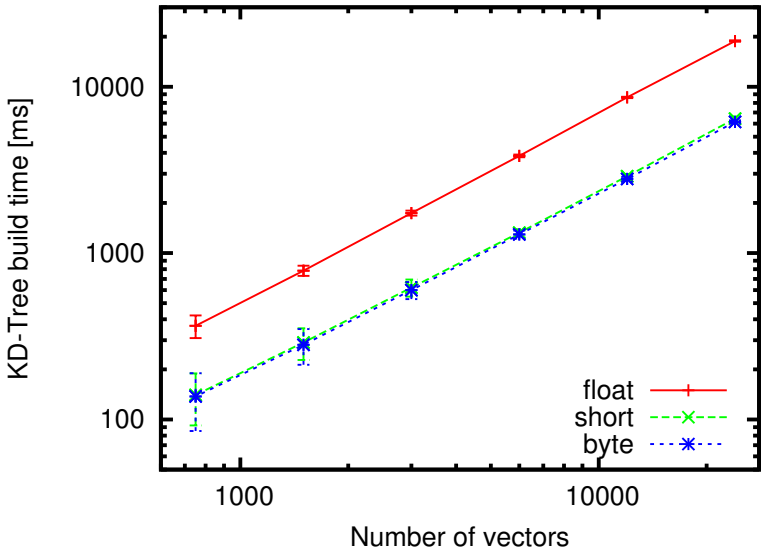

Figure 4.12: Average KD-Tree construction time for bucket= 10 and feature vectors of 24 dimensions for different data types on a Nokia N70.

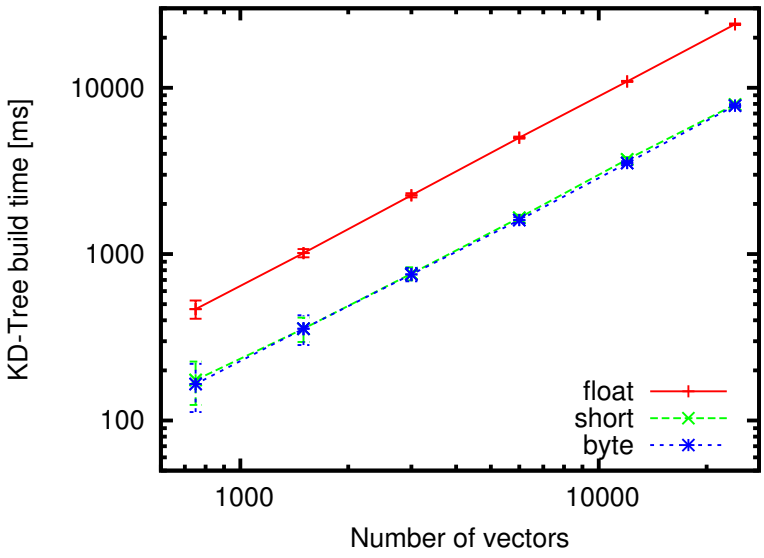

Figure 4.13: Average KD-Tree construction time for bucket= 10 and feature vectors of 32 dimensions for different data types on a Nokia N70.

Table 4.4: Estimation of verification times (ms) for the different algorithms and data types on a Nokia N70.

Algorithm		**extraction**	**projection**	**score**	**Total**
Eigenfaces	float	379	92.4	22.8	494 ± 9
	short	373	19.5	2.9	396 ± 6
	byte	368	17.8	2.8	389 ± 5
Fisherfaces	float	375	55.1	14.2	444 ± 8
	short	366	11.2	2.0	378 ± 6
	byte	363	11.1	1.9	375 ± 6
Local Features	float	232	1266	2175	3764 ± 13
	short	119	272	357	749 ± 15
	byte	119	271	369	759 ± 14

Table 4.5: Face verification statistics for the different algorithms including average execution times on a Nokia N70.

Algorithm	**TER@ FAR=0.1%** [%]	**Model Size** [kB]	**Memory** [MB]	**Enrol. Time** [s]	**Verif. Time** [s]
Eigenfaces	57.0	0.141	0.118	0.387	0.389
Fisherfaces	37.2	0.085	0.075	0.374	0.375
Local Features	29.8	28.23	1.093	0.566	0.759

This results can be compared with the OKAO Vision face recognition engine for mobile devices. In [ISL06] they report recognition and enrolment times of 0.99 and 2.84 seconds respectively for an ARM920T 200MHz processor. This processor is similar to the one of Nokia N70, and therefore the processing times are comparable to the ones obtained here. However in memory usage the local features approach requires more than 1MB compared to 480kB required by OKAO Vision.

4.6 Face Detection

A face recognition algorithm is no use if there is no previous face detection and localisation. By face detection it is meant to detect the presence and the amount of faces in an image or video. And by face localisation it is meant the position scale and pose of the face in the image. This process is necessary because if the face is not well positioned and cropped, the performance of recognition algorithms degrades significantly. Considering the situation of face recognition in a mobile device, it is possible not to have a face detection and localisation. The idea would be to use some king of visual feedback so that the user positions its face correctly. However this is much more uncomfortable and time consuming for the user and the recognition accuracy can sill be affected.

In this work, three face detectors were implemented. The fist one only detects a single face and the others are capable of detecting multiple faces. Given that the detection is going to be done on a personal mobile device, it is reasonable to assume that the images will have only one face which will be relatively close to the camera and frontal. For the initial attempt to do face detection on the mobile device, the problem was restricted to selecting the position and scale in the image which is more probable of being a face. Very little care was taken to avoid false detections in the case that the image does not have a face. This type of detector can be reasonable in this situation if the user of the system will have a visual feedback which shows that the image was not captured correctly.

Generally detection algorithms learn to detect faces by having a labelled corpus of face and not face images. If the face detection problem is simplified to finding the most probable location of the face in the image, the classifier used can be trained differently. The classifier can be designed to discriminate between well centred faces and wrongly positioned faces and non faces. Following this idea a training corpus was generated and an LDPP classifier was trained. For the training data, the same images from the gender recognition data-set were used [VP08]. Using the eye coordinates, 1892 correctly

positioned face images and 5336 random wrongly positioned faces were extracted. To the bad positioned faces, 2500 non face images extracted from University of Washington outdoor scene data-sets [LSB05] were added. The size of the images was 24×24.

The LDPP algorithm was used to learn a 16-dimensional orthogonal projection base and eight prototypes for each of the classes. This classifier was also optimised using the 8-bit based dimensionality reduction. The detector implemented was a simple exhaustive classification of faces and non faces for different positions and scales. The position classified as a face with the highest confidence is the one used for the only detected face.

This face detector was not assessed with a well defined experiment with some standard database of face images. However this detector was tried on a mobile phone (see chapter 5) taking pictures of real people and it detects faces surprisingly well even though very few face images were used as training. On the Nokia N70 phone it takes on average 4.5 seconds to detect faces in a 160×120 image searching from faces of 60 to 120 pixels sampled every 1.2.

The second detector implemented also used the previously mentioned classifier trained with the LDPP algorithm. The difference is that it was extended to detect multiple faces. After the exhaustive search, the close-by hypothesis are fused using morphological operations. Finally the overlapping hypothesis are removed taking preference the ones with higher confidence. This post-processing is computationally expensive and therefore this second detector is slower. For this second detector the time taken to process the a 160×120 with the same scale sub-sampling as before is about 13.4 seconds. This is considerably higher and for a user would be unacceptable. This does indicate that the method for used fusing and selecting the final hypothesis is a bit too expensive for a mobile device.

The final detector that was implemented was the Viola and Jones face detector. The detection models were not trained, instead the models that come with the OpenCV library were used. After the Haar cascade gives the face hypothesis the same procedure as the previous detector for fusing and selecting the final face hypothesis was used. On average, it takes about 5 seconds to process an image under the same circumstances as before. Possibly most of the time is due to the post-processing as was noted in the previous detector. Therefore a much faster detector can be obtained by improving this part of the code.

4.7 Face Gender Recognition

The LDPP algorithm was used to learn one classifier used for face detection as mentioned in the previous section. However this algorithm can be used to for several other image recognition tasks. In the context of face recognition, an example could be gender identification.

An experiment identical to the one presented in [VP08] using the gender recognition data-set was performed. The only difference with respect to the result presented in [VP08], was that like it was explained in the section 3.4 the learned projection bases were restricted to be orthonormal. This makes the algorithm deal better with the curse of dimensionality and thus can lead to higher recognition rates. The best result obtained was for learning a 32-dimensional projection base and 8 prototypes per class. This configuration gave an error rate of 8.5% compared to the 9.5% reported in [VP08].

Chapter 5

Demonstration Applications

The final objective of this work was to have a demonstration application that worked on several commercial mobile phones. The most popular operating system used in mobile phones is Symbian, however the majority of devices use proprietary operating systems. Furthermore the new operating system Android, in the near future could gain a large portion of the market. This posed the problem regarding which architecture to target. Doing face analysis and recognition from images requires a lot of processing, therefore the best choice would be to implement the software in C language and port it to each of the platforms. Nonetheless this is not an easy task and possibly many problems would arise. To make the software more platform independent the decision was taken to implement all of the software in Java. For this same reason all of the results presented in the chapter 4 were obtained using Java.

Most of the commercial phones currently sold support the Java Micro Edition (ME) specification, which is a subset of the Java platform developed for resource constrained devices. Each manufacturer supplies its own java virtual machine, therefore some incompatibility issues can arise. The application framework used in Java ME is the MIDlet, which was adequate to make the proposed demonstration applications. The demo applications should work on devices which support Java ME with CLDC 1.1 and MIDP 2.0. The demo applications are available on the Internet and can be freely downloaded from `http://www.iti.upv.es/~mvillegas/research`.

All of the applications use the camera for capturing face images. Therefore the devices also need to have a camera which is supported by the Java Mobile Media API. Unfortunately because of the security model of the Java application each time the user tries to take a picture a confirmation message is displayed.

5.1 Face Detection

One of the demo applications developed is dedicated exclusively to face detection. The face detection demo MIDlet is a simple application that shows in the screen the output from the camera and waits for the user to take a snapshot. When the snapshot is taken, faces are detected on the image and afterwards the results and some statistics are presented. Figure 5.1 shows the output screen from an example detection.

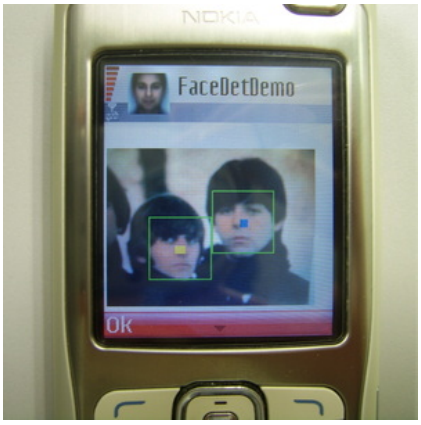

Figure 5.1: Snapshot of the face detection demo.

Several details can be configured in the application. First for mobile devices that have two cameras, the application can be configured to use either one. The algorithm used for the face detection and the parameters that control it can also be adjusted at will. The detection algorithms that can be chosen from are the ones presented in the section 4.6. The detection parameters that can be adjusted are: the minimum and maximum scale in pixels and the step between the sampled scales; the sub-sampling distance as a percentage of the face width; and the maximum number of faces to be detected.

The detection results screen first shows the original image with additional information related to the detection. In the configuration there is also a field called *Image output* which controls the amount of information that is displayed over the detected image. A value of zero only highlights the detected faces and a value of three highlights all of the considered hypothesis and the regions used for fusion of nearby hypothesis. Followed by the detected image the total detection time and the number of faced detected are displayed. Finally for each face detected the detection score, the face centre and size and a cropped version of the face is displayed.

5.2 Face Verification

The final goal of this work was to have a demo application that worked on commercial phones for doing face verification. The developed application starts displaying a menu which gives the options to capture training images, edit the training images, make a verification and change the configuration. Figure 5.2 shows a snapshot of the main menu.

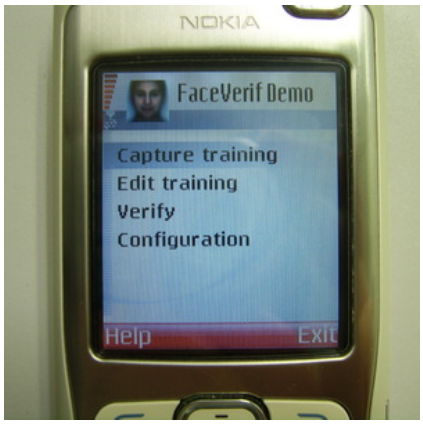

Figure 5.2: Snapshot of the face verification demo.

In the configuration screen the user can choose among the three different verification algorithms that were presented in this thesis. Also, the verification threshold can be modified by selecting one of the operating points. Finally just like in the face detection demo, for mobile devices that have two cameras, the application can be configured to use either one.

The other options of the main menu are very self explicative. The capture training option activates the camera so that the user takes images that will be included in the training of the user model. The edit training option displays the images that have been captured for training and gives the option the delete any unwanted images.

The final option of the main menu gives the option to do a face verification. By selecting this option the camera is activated waiting for the user to capture an image. When the image is captured the face is detected and verified. The output of the face verification is first a message indicating if the access was granted or not (a positive or a negative verification). Afterwards the score of the verification algorithm is displayed along with the threshold used to make the decision. Finally the times taken for detection, building the user model and each of the steps of the verification process are displayed.

5.3 Face Gender Recognition

Originally this demo application was intended to do more than just gender recognition. It was more intended to analyse a face image and printout several data, including gender, facial expression, and ethnicity. However because of a lack of adequate labelled data-sets it was only possible to do gender recognition.

This application simply gives the choice to select which camera to use and when an image is captured the face is detected and the results are presented. The result shows if the face image is of a male or a female and the confidence of the classifier. Also the execution times are shown and at the end the captured image is shown with the detected face highlighted. Figure 5.3 shows the output screen of a recognised image.

Figure 5.3: Snapshot of the face gender recognition demo.

Chapter 6

Conclusions

This work has been oriented to the difficulties that are encountered when trying to do facial analysis and recognition in mobile devices. Some face recognition algorithms were described and analysed in detail, and this made it possible to propose some key optimisations targeted at the specific characteristics that are encountered in resource constrained devices. The objective was to reduce the system requirements in terms of memory and processing speed without affecting the recognition accuracy.

The proposed optimisations were very simple and general which could be applied to all the algorithms in question, however this ideas can also be applied to many other pattern recognition problems. The optimisations were first to use only 8-bit signed integers to store the training data and second to do as much of the processing a possible using only integer arithmetic. These ideas were used for optimising linear feature projections used in dimensionality reduction and for efficient nearest neighbour searching by means of a *kd*-tree structure.

The algorithms were implemented with and without the proposed optimisations and an extensive empirical assessment was performed. Experiments using real face images confirm that the optimisations do not have a significant impact on the recognition accuracy of the algorithms. Furthermore, experiments done on a Nokia N70 mobile phone show that the algorithms can be almost five times faster by using integer arithmetic. The improvements in memory requirements are also significant. For processing of the algorithms the memory used is reduced by a factor of two and for the storage of data the improvement is by a factor of four.

The recognition accuracy of the algorithms were estimated trying to somewhat simulate the circumstances that are generally found in mobile applications. The results obtained are comparable to existent commercial software, both in processing time or resource requirements.

The only part for which the results were not as good as was initially expected was in the face detection. Three face detectors were implemented. Two were based on prototypes and nearest neighbour classification. The first one detects only one face and the other can detect multiple faces. The third detector was the well known Viola and Jones, also for detecting multiple faces. The detection takes about five seconds, which is quite slow compared to other software available on the Internet which also runs in J2ME. The part which is making the detection slow is the post-processing done for fusing close-by hypothesis and selecting the final ones. As future work the detectors can surely be greatly improved.

As a result of this work, three J2ME MIDlet applications were developed. One for doing face detection, another for doing face verification or in other words user authentication, and the final one which does face gender recognition. They all work considerably well, however they can still be improved much more. Furthermore, as future work an application could be developed which has some real use not just being a demonstration.

Bibliography

[ABBAQ05] O. Al-Baker, R. Benlamri, and A. Al-Qayedi. A gprs-based remote human face identification system for handheld devices. *Wireless and Optical Communications Networks, 2005. WOCN 2005. Second IFIP International Conference on*, pages 367–371, March 2005. 6

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998. 18

[BBBB+03] Enrique Bailly-Bailliére, Samy Bengio, Frédéric Bimbot, Miroslav Hamouz, Josef Kittler, Johnny Mariéthoz, Jiri Matas, Kieron Messer, Vlad Popovici, Fabienne Porée, Belén Ruíz, and Jean-Philippe Thiran. The BANCA database and evaluation protocol. In *AVBPA*, pages 625–638, 2003. 21

[BHK97] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, Jul 1997. 5, 11, 12

[CNWB05] H. Cevikalp, M. Neamtu, M. Wilkes, and A. Barkana. Discriminative common vectors for face recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 27(1):4–13, January 2005. 5

[ECT98] Gareth J. Edwards, Timothy F. Cootes, and Christopher J. Taylor. Face recognition using active appearance models. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume II*, pages 581–595, London, UK, 1998. Springer-Verlag. 5

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 2 edition, 1990. 10, 12

[HKK+05] M. Hamouz, J. Kittler, J.-K. Kamarainen, P. Paalanen, H. Kalviainen, and J. Matas. Feature-based affine-invariant localization of faces. *Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1490–1495, Sept. 2005. 4

[HWP03] Timothy J. Hazen, Eugene Weinstein, and Alex Park. Towards robust person recognition on handheld devices using face and speaker identification technologies. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 289–292, New York, NY, USA, 2003. ACM. 6

[ISL06] Yoshihisa Ijiri, Miharu Sakuragi, and Shihong Lao. Security management for mobile devices by face recognition. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*, page 49, Washington, DC, USA, 2006. IEEE Computer Society. 6, 35

[JKF01] Oliver Jesorsky, Klaus J. Kirchberg, and Robert Frischholz. Robust face detection using the hausdorff distance. In *AVBPA '01: Proceedings of the Third International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 90–95, London, UK, 2001. Springer-Verlag. 4

[JMKL00] K. Jonsson, J. Matas, J. Kittler, and Y.P. Li. Learning support vectors for face verification and recognition. *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 208–213, 2000. 5

[JS08] Chia-Feng Juang and Shen-Jie Shiu. Using self-organizing fuzzy network with support vector learning for face detection in color images. *Neurocomput.*, 71(16-18):3409–3420, 2008. 4

[KMB07] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recogn.*, 40(3):1106–1122, 2007. 4

[LJ04] Stan Z. Li and Anil K. Jain. *Handbook of Face Recognition.* Springer, 2004. 3

[LM98] J. Luettin and G. Maître. Evaluation Protocol for the extended M2VTS Database (XM2VTSDB). IDIAP-COM 05, IDIAP, 1998. 19

[LM02] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. *Image Processing. 2002. Proceedings. 2002 International Conference on*, 1:I–900–I–903 vol.1, 2002. 4

[LSB05] Yi Li, L.O. Shapiro, and J.A. Bilmes. A generative/discriminative learning algorithm for image classification. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2:1605–1612 Vol. 2, October 2005. 36

[MJP00] Baback Moghaddam, Tony Jebara, and Alex Pentland. Bayesian face recognition. *Pattern Recognition*, 33(11):1771–1782, 2000. 5

[MKS+03] Kieron Messer, Josef Kittler, Mohammad Sadeghi, Sébastien Marcel, Christine Marcel, Samy Bengio, Fabien Cardinaux, Conrad Sanderson, Jacek Czyz, Luc Vandendorpe, Sanun Srisuk, Maria Petrou, Werasak Kurutach, Alexander Kadyrov, Roberto Paredes, Burcu Kepenekci, F. Boray Tek, Gozde Bozdagi Akar, Farzin Deravi, and Nick Mavity. Face verification competition on the xm2vts database. In *AVBPA*, pages 964–974, 2003. 20

[MKS+06] Kieron Messer, Josef Kittler, James Short, Guillaume Heusch, Fabien Cardinaux, Sébastien Marcel, Yann Rodriguez, Shiguang Shan, Y. Su, Wen Gao, and Xilin Chen. Performance characterisation of face recognition algorithms and their sensitivity to severe illumination changes. In *ICB*, pages 1–11, 2006. 20

[MMK+99] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. XM2VTSDB: The extended M2VTS database. In R. Chellapa, editor, *Second International Conference on Audio and Video-based Biometric Person Authentication*, pages 72–77, Washington, USA, March 1999. University of Maryland. 19, 23, 26

[NI98] Ara V. Nefian and Monson H. Hayes Iii. Hidden markov models for face recognition. In *Proc. International Conf. on Acoustics, Speech and Signal Processing*, pages 2721–2724, 1998. 5

[NSK05] Chee Kiat Ng, Marios Savvides, and Pradeep K. Khosla. Real-time face verification system on a cell-phone using advanced correlation filters. *Automatic Identification Advanced Technologies, IEEE Workshop on*, 0:57–62, 2005. 6

[PPJV01] R. Paredes, J. C. Pérez, A. Juan, and E. Vidal. Local Representations and a direct Voting Scheme for Face Recognition. In *Proc. of the Workshop on Pattern Recognition in Information Systems (PRIS 01)*, Setúbal (Portugal), July 2001. 5, 12, 13

[RBK98] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions On Pattern Analysis and Machine intelligence*, 20:23–38, 1998. 4

[RF08] G.A. Ramirez and O. Fuentes. Multi-pose face detection with asymmetric haar features. In *IEEE Workshop on Applications of Computer Vision (WACV08)*, pages 1–6, 2008. 4

[SB02] Hichem Sahbi and Nozha Boujemaa. Coarse to fine face detection based on skin color adaption. In *ECCV '02: Proceedings of the International ECCV 2002 Workshop Copenhagen on Biometric Authentication*, pages 112–120, London, UK, 2002. Springer-Verlag. 4

[SEKK06] C. Schneider, N. Esau, L. Kleinjohann, and B. Kleinjohann. Feature based face localization and recognition on mobile devices. *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, pages 1–6, Dec. 2006. 6

[SHC05] Daijin Kim Sang-Ho Cho, Bong-Jin Jun. Face recognition on a mobile device. In *Proceedings of International workshop of Intelligent Information Processing (IWIIP)*, 2005. 6

[SK00] Henry Schneiderman and Takeo Kanade. A statistical model for 3-d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2000. 4

[SKKM03] Mohammad Sadeghi, Josef Kittler, Alexey Kostin, and Kieron Messer. A comparative study of automatic face verification algorithms on the banca database. In *AVBPA*, pages 35–43, 2003. 22

[SR02] K. Sandeep and A. N. Rajagopalan. Human face detection in cluttered color images using skin color and edge information. In *ICVGIP 2002, Proceedings of the Third Indian Conference on Computer Vision, Graphics and Image Processing*, Ahmadabad, India, December 2002. Allied Publishers Private Limited. 4

[TEBEH06] A. S. Tolba, A. H. El-Baz, and A. A. El-Harby. Face recognition: A literature review. *International Journal of Signal Processing*, 2(2):88–103, 2006. 3

[Tur91] A.P. Turk, M.A.; Pentland. Face recognition using eigenfaces. *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, 3-6 Jun 1991. 5, 9, 10

[VJ04] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004. 4

[VP05] M. Villegas and R. Paredes. Comparison of illumination normalization methods for face recognition. In Mauro Falcone Aladdin Ariyaeeinia and Andrea Paoloni, editors, *Third COST 275 Workshop - Biometrics on the Internet*, pages 27–30, University of Hertfordshire, UK, October 27-28 2005. OPOCE. 16, 20, 21

[VP07] M. Villegas and R. Paredes. Illumination Invariance for Local Feature Face Recognition. In *1st Spanish Workshop on Biometrics*, pages 1–8, Girona (Spain), June 2007. -. 13, 16

[VP08] M. Villegas and R. Paredes. Simultaneous learning of a discriminative projection and prototypes for nearest-neighbor classification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008. 13, 14, 35, 37

[VPJV08] M. Villegas, R. Paredes, A. Juan, and E. Vidal. Face verification on color images using local features. In *Proceedings of the Workshop on Biometrics, in association with CVPR 2008*, Anchorage, AK, USA, June 2008. IEEE Computer Society. 13

[WFKvdM97] Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christopher von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):775–779, 1997. 5

[WHH+02] Eugene Weinstein, Purdy Ho, Bernd Heisele, Tomaso Poggio, Ken Steele, and Anant Agarwal. Handheld face identification technology in a pervasive computing environment. In *Short Paper Proceedings, Pervasive 2002*, pages 48–54, 2002. 6

[yHPC+07] Song yi Han, Hyun-Ae Park, Dal Ho Cho, Kang Ryoung Park, and Sangyoun Lee. Face recognition based on near-infrared light using mobile phone. In *ICANNGA (2)*, pages 440–448, 2007. 6

[ZCRP03] W. Zhao, R. Chellappa, A. Rosenfeld, and P. J. Phillips. Face recognition: A literature survey. *ACM Computing Surveys*, pages 399–458, 2003. 3, 5