

Face Recognition in Mobile Devices

Mattias Junered

Luleå University of Technology
MSc Programmes in Engineering
Media Technology
Department of Computer Science and Electrical Engineering
Division of Signal Processing

Face Recognition in Mobile Devices

Mattias Junered
Luleå University of Technology

March 2, 2010

Abstract

Recent technological advancements have made face recognition a very viable identification and verification technique and one reason behind its popularity is the non-intrusive nature of image acquisition. A photo can be acquired easily without the person even being aware of the process.

The interest in biometrics by several governments for identifying possible criminals or verifying users for access control is steadily increasing. Other industries are also finding uses for face recognition techniques such as in entertainment systems and for robots that interact with humans.

Mobile phones are constantly improving and the majority are currently equipped with a digital camera. This facilitates taking a large amount of photos every day with a camera phone instead of a stand-alone digital camera. Using face recognition techniques on these images makes it possible to perform so called face tagging to tag images with the names of the photographed persons. This is convenient for sorting photos, creating albums or retrieving images of only a specific person.

Having a stand-alone mobile application on the phone that performs these face recognition tasks on recently captured images is an interesting concept. The system can be trained on a set of images containing faces to become capable of automatically recognizing a person from the training set. However, many users have up to hundreds or even thousands of images on their mobile phones and training a system on the phone is prohibitively time-consuming on such devices today. Instead, face recognition can be performed on the client using already trained data transferred from a computer (server).

This approach shows promising results and very good success rates. This article covers several methods that can improve results by making the system more robust.

Acknowledgements

This work would not have been possible without the support of Apostolos Georgakis as external supervisor at Ericsson AB. Many thanks to Jiong Sun at the EAB/TVV section for connecting the pieces in the phone application and additional help and support. The author would also like to thank the whole EAB/TV department for their cooperation in creating the internal image database and all the interesting discussions. Finally, thanks to Josef Hallberg internal supervisor, Kåre Synnes examiner at Luleå University of Technology and everyone else who helped out.

Contents

1	Introduction	4
1.1	Background	4
1.2	Chapter outline	5
1.3	Objective	5
1.4	Goal	5
2	Related work	6
2.1	Face Recognition Vendor Test	6
2.2	Algorithm categorization	8
2.2.1	Projection methods	9
2.2.2	Statistical methods	9
2.2.3	Graph matching methods	10
2.2.4	Neural network methods	11
3	Theory	12
3.1	History	12
3.2	Pre-processing	13
3.2.1	Illumination normalization	13
3.3	Face detection	14
3.3.1	Skin color detection	15
3.4	Post-processing	16
3.4.1	Eye detection	16
3.4.2	Face alignment	16
3.5	Face recognition	17
3.6	Image databases	18
3.6.1	ORL	18
3.6.2	Color FERET	19
3.6.3	LFW	19
3.6.4	EAB/TV	19
4	Method	21
4.1	System description	21
4.2	Suitable algorithms	21
4.2.1	LBP	23

4.2.2	Mean LBP	30
4.2.3	DCV	31
4.2.4	Kalmanfaces	34
4.3	Illumination normalization	35
4.3.1	Histogram equalization	35
4.3.2	Three step method	36
4.4	Face detection	37
4.4.1	OpenCV	37
4.4.2	RGB skin pixel model	38
4.5	Pose normalization	39
4.5.1	In-plane rotation	39
4.5.2	Background noise removal	40
4.6	Training	41
4.7	Testing	42
4.7.1	Creating the test sets	42
4.7.2	Adding additional training samples	42
5	Experimental results	44
5.1	Recall/Precision graphs for LBP	44
5.2	Overall first match scores	46
5.3	Recall/Precision graphs under various settings	46
6	Demo	49
7	Discussion	51
7.1	Conclusions	51
7.2	Optimal algorithm	52
8	Future improvements	53
8.1	Face synthesization	53
8.2	Artificial lighting	53

Chapter 1

Introduction

1.1 Background

One of the increasingly popular trends in image analysis is face recognition. The need for this kind of technology in several different areas keeps pushing the research forward every year. Most commonly used in practice by law enforcement for human identification purposes, face recognition systems can also be used at home for more trivial tasks such as logging in on a computer. There are some entertainment applications evolving as well, for example in virtual reality and human-robot-interaction, but they are still far from common.

Early systems used specific face scanners to detect and store information about the geometry of a person's face, but current algorithms for detection and recognition of faces use captured images/photos. Although the identification performance for face recognition is more sensitive than other simpler identification methods like fingerprint matching, face recognition is considered to be a very user-friendly verification technique because it lacks demand of direct participant interaction. Recent advances have even opened up the possibility for computers to rival the human ability to recognize a face in any natural condition. Computers have almost always been able to memorize more faces than humanly possible.

A fairly new concept for this topic is the use in mobile devices. The mobile phone has become an every day tool in most people's lives packed with functions beyond what could be imagined only a few years ago. Since most mobile phones currently are equipped with a digital camera, it might be interesting for users to have the ability to put names or tags on photos of their family and friends. Users can more easily handle their photos and also be able to take new ones where the faces are automatically recognized by the mobile phone. This requires a system that is fast and efficient but also robust enough to actually work under the varying conditions that can be expected from normal every day use.

1.2 Chapter outline

This thesis is divided into eight major chapters. Chapter one gives an introductory background to the work, an objective and a goal. In the second chapter, some independent evaluations are described and different existing techniques. The third chapter will discuss some brief history, the face recognition process, some related methods for improving the system and available databases. Chapter four will move on to the methodology by describing the system, the tested algorithms and additional attempted improvements. It also contains a description of how the test sets were built to compare the selected algorithms against each other. The fifth chapter shows some experimental results using the constructed test sets and also various parameters for the different algorithms. A quick explanation of the demo application on the actual phone is done in chapter six. Chapter seven and eight concludes the report with conclusions and possible future work.

1.3 Objective

A client-server architecture needs to be developed with all the relevant parts of a face recognition system. The client- and server-side are required to be very similar because the processed data will have to be comparable.

The server-side will need to be implemented specifically for creating training data for example in Matlab. Some data compression could also be beneficial to reduce the memory requirements before transfer to the client.

A client-side face detector is available from Ericsson [8] implemented in J2ME so the client should also be implemented for that platform. All comparisons are meant to be made on the client when the data is transferred from the server. This means that the client needs to be able to perform distance calculations on the vectors used to describe the faces in order to classify them. The closest matches will then be represented by sorting in order of smallest distance.

If time allows, some extra improvements could be tested such as normalization of the input images or optimizing the algorithms.

1.4 Goal

The purpose of this master's thesis is to investigate the current state-of-the-art techniques for face recognition and to determine the most suitable method for mobile devices. A client application on a mobile phone will be trained by a server application running on a stationary computer using the selected method. With the help of images from albums or social networks for training it might be possible for the client to accurately recognize familiar faces when taking new photos. The client will give a few suggestions of names for each photo to the user so that the user can correct the system if it fails. The main goal is to have at least 50% correct names in the top three suggestions.

Chapter 2

Related work

A comprehensive survey of different face recognition techniques has been written by Zhao *et al.* [1]. Their survey also has more detailed descriptions of algorithms used for both still- and video-based recognition research. Other related areas are also covered, such as psychosocial studies and issues of image variations. Abate *et al.* [2] later did a survey aimed at the trends in 2D imagery and 3D model-based algorithms. The effects of terrorist attacks and security flaws are described as a big motivation for government agencies to invest large amount of resources into this kind of research. Again, the varying conditions encountered in images especially encountered outdoors are also described. Section 2.2 only contain a quick summary of these surveys where a wide array of algorithms are mentioned.

First however, the Face Recognition Vendor Test will be explained in section 2.1. The latest test from 2006 follows five previous face recognition technology evaluations that helped advance face recognition from infancy to the prototype system stage and after that also commercially. Researchers and developers in the area from companies, research institutions and academia are eligible to participate. Accurate assessments are guaranteed by providing both new test data and test environments to the participants. Results are publically presented on the official website after each evaluation [17].

2.1 Face Recognition Vendor Test

Several independent evaluations have been conducted in an attempt to compare the effectiveness of different algorithms. The Face Recognition Technology (FERET) [17] evaluation is one of the older evaluations held back in 1994, 1995 and 1996. A new series of tests for both prototype systems and commercial techniques called Face Recognition Vendor Test (FRVT) [17] is now being held regularly. Their purpose is to provide the U.S Government and law enforcement agencies with information about the current state of the face recognition technology and how it can best be deployed. FRVT has so far been held in 2000, 2002 and most recently in 2006 by the National Institute of Standards and Technology (NIST).

False reject rate (FRR) and false accept rate (FAR) is the most common error mea-

asures when dealing with verification systems. A false reject is a known person that is rejected by the system and a false accept is an unknown face accepted by the system. Both should optimally be as low as possible, but since for example security systems are much more sensitive to false acceptance it is better to have a low maximum allowed FAR. This is why it is common to only measure the FRR at a certain level of FAR. The improvement after each evaluation using this measure with a FAR limit of 0.001 is shown in figure 2.1.

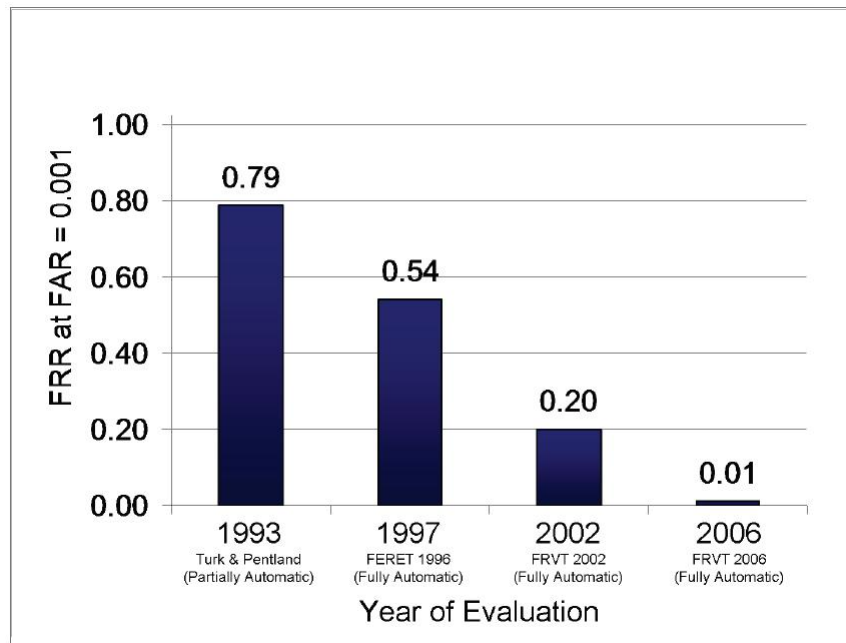


Figure 2.1: Reduction in FRR over the years (source: www.frvt.org)

The Face Recognition Grand Challenge (FRGC) [18] is another algorithm development project designed to promote and advance face recognition technology to support existing efforts in the U.S. Government. It ran from 2004 to 2006 and the goal was to decrease the error rate of the face recognition algorithms by an order of magnitude.

In the FRVT 2006 a decrease by at least an order of magnitude over the FRVT 2002 was reported [17]. A FRR of 0.01 percent was given at a FAR of 0.001 by a representative state-of-the-art algorithm. However, this was achieved on quite high-resolution images in a controlled environment.

Two different test sets of high- and very high-resolution images were also used in the FRVT 2006 to test uncontrolled lighting conditions. This time the top performing algorithms reported error rates below 20%. One of them was V-Norm developed by the Viisage group and the other was ST-Norm developed by SAIT. All of the tested algorithms performance is shown in figure 2.2.

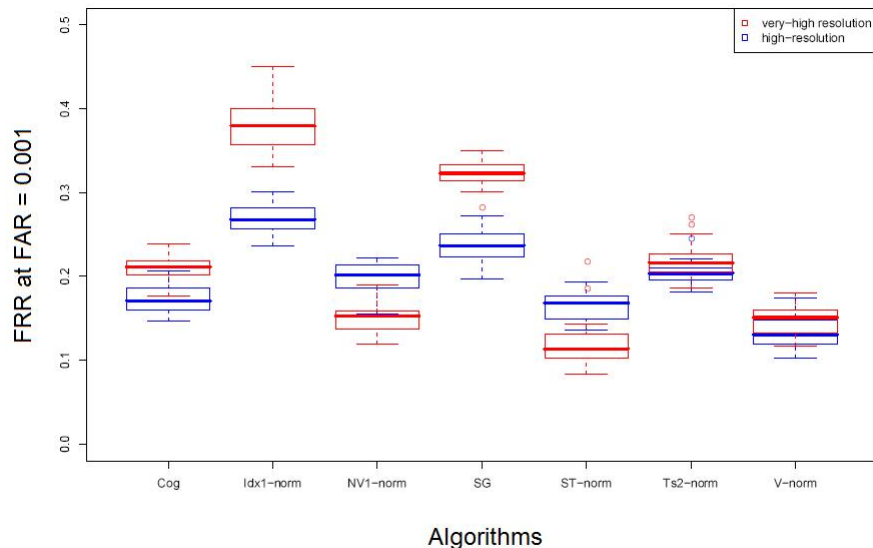


Figure 2.2: FRR results from FRVT 2006 under uncontrolled lighting conditions (source: www.frvt.org)

2.2 Algorithm categorization

Several different categorizations of the existing face recognition algorithms have been suggested. Either by how the methods generally work or what they have in common.

Zhao *et al.* [1] used a high-level categorization based on psychological studies of humans consisting of holistic, feature-based and hybrid methods. The most common approach in face recognition is to just process face images in a holistic manner to capture the general relation between features. Feature-based methods on the other hand, are focused on dividing the face into important separate features. While Hybrid methods attempt to do both at the same time and combine the results somehow. However, one problem when creating hybrid methods is how to combine them in a good way to be able to get the best out of both worlds.

A slightly different categorization was chosen for this thesis based on four groups: projection, statistical, graph matching and neural network methods. Face recognition in general is a pattern matching problem and Abate *et al.* [2] also suggest to think of it as a template matching problem. The higher the dimension space the more computations are needed to find a match. Another important aspect in face recognition is specifying important features and representing classes (persons) in such a way that an optimal subset can be derived leading to high classification performance. This is one factor behind the upcoming of the projection methods that mainly try to reduce the dimensions and use discrimination for better classification. Statistical models are a good way to remove redundant information and is the basic idea behind the statistical

methods while the graph matching methods were developed to better model transformations of the faces in the images. Neural networks have been used a lot in other areas like machine learning and can effectively be used for any complex input-output system because of its learning capabilities.

2.2.1 Projection methods

One very popular technique behind many projection methods for face recognition is Principle Component Analysis (PCA) which is a dimensionality reduction method. The traditional PCA algorithm used in the Eigenfaces approach by Turk and Pentland[24] computes the eigenvectors and eigenvalues of a covariance matrix and solves an eigenvalue system problem. This method performs no discrimination but instead the eigenvectors that correspond to the largest eigenvalues are used as base vectors in a low-dimensional space. Matching a new test face can then be performed by projecting the image into this reduced space and finding the closest match by Euclidean distance. The combination of Eigenfaces and Eigenmodules into a hybrid method called Modular Eigenfaces has also been tested. Results are combined from eigeneyes, eigen noses and eigenmouths that are calculated in addition to the eigenfaces in a similar manner.

From the lack of discrimination in Eigenfaces, a new method emerged that is often called Fisher Linear Discriminant (FLD) because it is based on a method by Fisher[25]. His method referred to as Linear Discriminant Analysis (LDA), is used to improve the low-dimensional space by finding the most discriminant extracted features. This method defines a projection that minimizes the within-class scatter matrix and at the same time maximizes the between-class scatter matrix. Within-class means images of the same person and between-class means images of different persons where the scatter matrices describe the differences in the data scatter from the mean value. These discrimination criterion techniques require many training examples per face to get a good generalization, but there are ways to artificially increase the number of examples even in a small training set. As a face recognition algorithm, this method is simply called Fisherfaces.

A very effective binary classification method for general purpose pattern matching is the Support Vector Machines (SVM) that was first applied to face recognition by Phillips[27]. The input data is projected into a high dimensional feature space where the mapped data could become linearly separable again in the transformed space. An optimal hyperplane is determined by classifying all the positive and negative samples using a non-linear kernel function. This method is often used in conjunction with other methods to improve the classification.

2.2.2 Statistical methods

A statistical approach also based on PCA is called probabilistic eigenfaces by Moghadam and Pentland[28]. However, this method also use the eigenspace decomposition for estimating the complete density functions in high-dimensional image spaces. Matching is then performed using these density estimates in a maximum likelihood formulation instead of the usual distance measure.

Another method similar to PCA is Independent Component Analysis (ICA) that tries to estimate independent characteristics of objects such as human faces. Bartlett *et al.* [26] describes that there are many approaches to perform ICA and they used two different architectures, one which treated the images as random variables and the pixels as outcomes, and a second the other way around. Human faces are correlated in many ways, but finding the independent basic faces could improve the representation. Spatially localized feature vectors are used for their statistical independence which makes them less susceptible to occluded parts of the face. Another difference from PCA is that the calculated vectors in ICA are not necessarily orthogonal which makes them more flexible and better at reconstructing the original data.

The most common statistical feature-based approach is called Hidden Markov Models (HMM). It is built on using a Markov chain with a finite number of states, a state transition probability matrix, an initial state probability distribution and a set of probability density functions associated with each state. First, images are scanned in 1D or 2D and a set of blocks are extracted. Features are then extracted as they appear in natural order from top to bottom and each facial region is assigned a state in the HMM. Finally a maximum selection procedure is used to find the most probable match. Nefian and Hayes[29] managed to significantly reduce the computational complexity of previous methods of this kind.

Local Feature Analysis (LFA) is a biologically inspired method used to extract topographic local features to remove statistical redundancy. This method is similar to the way the brain has to discover what objects are in the field of view from the activity of sparsely distributed receptors. Normally the feature representation is done with so called Gabor wavelets. Each feature can be described as a Gabor jet which is a set of convolution coefficients for kernels of different orientations and frequencies at each pixel. The features can then be classified by using holistic discriminant methods such as LDA. Ersi and Zelek[30] showed that automatically extracted features at locations with highest deviations from the expectations was more powerful than using pre-defined locations.

2.2.3 Graph matching methods

The use of Gabor wavelets for capturing the salient visual features such as eyes, nose and mouth has also been introduced along with a framework called Dynamic Link Architecture (DLA). From this another method emerged called Elastic Bunch Graph Matching (EBGM) that further generalized the graph representation. Wiskott *et al.* [31] introduced the bunch graph as a novel data structure which is constructed from a small set of sample image graphs. The graph matching is normally done in two consecutive steps where the first is a rigid alignment of the grid to account for global transformations of the graph. Secondly, the grid nodes are compared using a graph similarity function. Graphs are generally rotation invariant so they are robust against pose changes, but the downside is that they are also very computationally expensive.

Shape modeling is another form of methods used in flexible appearance models (FAM) or active appearance models (AAM). These models are iteratively deformed to fit examples of the shapes in new images using different fitting procedures. The

within-class variations are classified using discriminant analysis and a global shape-free gray-level model is constructed.

Lanitis *et al.* [33] deployed a fully automatic robust face identification system using flexible models. Component-based recognition such as Huang *et al.* [35] used can further enhance the flexible geometrical models using a set of facial components that are interconnected also using gray-scale components. The idea is that components can model head pose and other changes just by changing positions within the model.

An active appearance model scheme was used by Cootes *et al.* [34] using statistical appearance models. Their model required a close initial starting position of the facial points to achieve reliable and rapid convergens. However, such information is seldom available beforehand and without it the search procedure can take a lot of time.

2.2.4 Neural network methods

In the search for even better generalization of face classes, neural networks have also been tested such as the Probabilistic Decision-Based Neural Network (PDBNN) and Evolutionary Pursuit (EP). These types of learning procedures improve the decision boundary with the help of fitness functions.

The PDBNN was deployed by Lin *et al.* [36] consisting of three modules: a face detector, eye localizer and face recognizer. It adopts a hierarchical network structure with non-linear basis functions where feature intensity and edge information is used for recognition.

EP on the other hand, implements the characteristics of genetic algorithms (GAs) for searching the space of possible solutions and determine the optimal basis. The data is first projected into a lower dimensional space similar to PCA and then random rotations of the basis vectors are searched by the GAs. Liu and Wechsler[37] first presented the method as a novel and adaptive representation method for image encoding and classification.

Another neural network system has also been deployed by Lawrence *et al.* [38] called Convolutional Neural Network (CNN) that uses a Self-Organizing Map (SOM). The SOM is used to quantize the image samples into a topological space so that similar features end up near each other. This has the effect of making the matching procedure invariant to small changes in the images.

Chapter 3

Theory

General theories and subjects closely related to face recognition are covered in this chapter starting with some brief history. Different techniques for improving the face detection normally needed before doing recognition using images are then discussed. After that, the following section contains a general description of face recognition by comparison to authentication. Finally, a walkthrough of the considered image databases that are helpful when running tests is given.

3.1 History

Research on automatic machine recognition of faces basically started evolving in the 1970s. Most studies before that were focused on the human perception system in the field of psychology.

Machine recognition of faces can generally be formulated as the identification or verification of one or more persons in a set of images using a stored database of faces, Zhao *et al.* [1].

The interest in this kind of technology grew substantially after the increasing importance of surveillance technology in airports and other security related applications.

Early approaches treated it as a 2D pattern recognition problem of 3D objects, probably since the current technology was not capable of even capturing or processing in 3D at the time. Today, the need for better 2D systems is still a pressing issue even though there have been recent advances on 3D face recognition. One reason for this is that new application areas have arisen where computational capacity is still limited, such as mobile devices. The image acquisition process is one of the major obstacles for 3D face recognition since it requires expensive scanner equipment that is not very mobile.

A solution model is presented in [1] involving face detection in cluttered scenes, feature extraction from the face regions and finally face recognition or verification. Adding a pre-processing step and some kind of normalization step after the face detection is another slightly less generic way to model a face recognition system. One way

to model such a generic system is shown in figure 3.1.

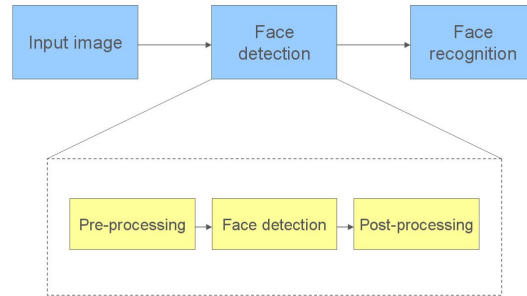


Figure 3.1: Generic face recognition system

3.2 Pre-processing

3.2.1 Illumination normalization

Strong variation in illumination is a big problem factor for many face recognition methods when important edges and discriminating features are lost. It has been shown by Adini *et al.* [39] that illumination actually cause more variation in face images than pose or even images of different persons. Facial appearance can be modeled as in the book by Delac *et al.* [41] using four components: diffuse reflection, specular reflection, attached shadow and cast shadow as demonstrated in figure 3.2.

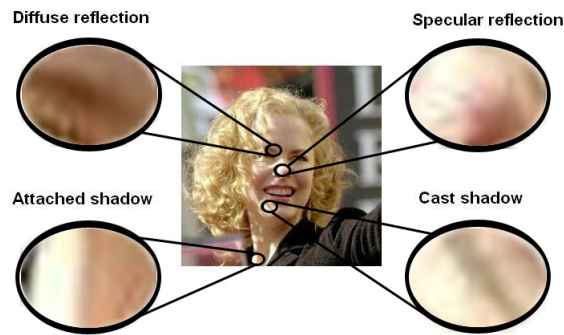


Figure 3.2: Facial appearance classified into four main components

A natural approach to reduce this problem is using so called canonical form methods. The goal for these methods is to reduce images to a more canonical form in which the variations are suppressed. Another approach is to train a global model of the possible variations and then generalize the variation in new images as in Shan *et al.* [21], but this is only possible when a large training set is available. Direct modelling of light

sources as for example by Basri and Jacobs *et al.* [40] is neither considered practical in real applications because of the complexity.

3.3 Face detection

Faces need to be detected and cropped before training and recognition can be performed. A technique that has almost become standard in this area is to use so called Haar-cascades and scan images using an increasing window. It is referred to as the Viola-Jones face detector[15] and became popular because of its robust real-time processing capabilities.

The system is first trained on a set of labeled images containing faces using the AdaBoost learning algorithm by Freund and Schapire[19] that uses weak classifiers. During the learning, a set of rectangular features as those shown in figure 3.3 are classified for optimal detection.



Figure 3.3: The two most prominent rectangular features

Images are also represented in a special way as so called integral images for rapid processing. Each pixel in the integral image describes the sum of the pixel values above and to the left of that pixel. This makes it possible to calculate any rectangular sum in an image only using four corner values from its integral image. An example is shown in figure 3.4.

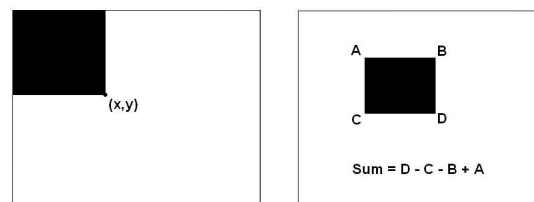


Figure 3.4: Integral image and rectangle calculation example

A cascade detection process is finally used with the structure of a degenerate decision tree to quickly discard background regions and classify actual faces. The number of feature classifiers used decrease the number of false positives detected by the system. Figure 3.5 shows an example of a cascade classifier using three features.

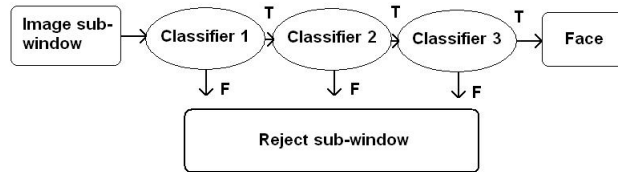


Figure 3.5: Cascade classifier example using three rectangular features (T = True, F = False)

3.3.1 Skin color detection

Another rather common approach for face detection is to detect skin in color images. Quite a lot of work has been done in this area mainly for human-robot interaction. Different color space transformations such as normalized RGB, HSI and YCrCb from television standards and other image processing methods are commonly used to isolate the skin color region. Vezhnevets *et al.* [13] have a more detailed description of these methods in their survey on this topic. Figure 3.6 shows what an image might look like if only the skin pixels in a face are shown on a black background. This region can also be found by training a system on manually selected skin pixels from a large image set. Geometric transformations do not affect the outcome of the detection since the classification of skin and non-skin pixels is very straightforward. The most common difficulties occur when the skin tone changes to be outside the color region boundaries. This could be reduced by having larger skin color regions. However, increasing the boundaries also increase the number of misclassifications of pixels. These techniques then use a face template to distinguish faces from other body parts after the skin detection has been performed.



Figure 3.6: Example images of a good skin detection compared to a lossy detection

3.4 Post-processing

3.4.1 Eye detection

Many techniques exist for eye detection and localization. Detection can be made quite efficient by narrowing down the search to a specifically chosen smaller area in an image. Inside the top half of the bounding box from a detected face is one good example assuming a face detector is used. Figure 3.7 shows an example of an eye detection and the resulting eye boxes marking the most probable area for each eye.

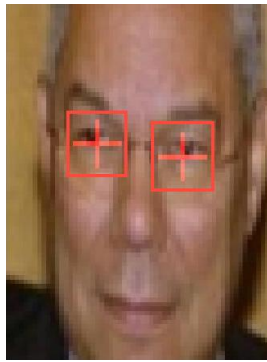


Figure 3.7: The resulting detected eyes shown as eye boxes where the pupil centers are marked by plus signs

The position of the eyes can then be used to normalize the position of the head and also to get an estimation of the rotation and view direction. Usually a higher resolution makes it easier to find a more precise location of the eyes. However, comparable accuracy of an eye detection system compared to systems requiring high-resolution images is claimed by Asteriadis *et al.* [12].

The usual way of measuring the eye location is with the pupil center. The biggest errors occur if the eye positions are hard to locate because of overlaying shadows, the head is in profile or if the eyes are closed.

3.4.2 Face alignment

Only a handful of techniques exist specifically for face alignment and most of them are very computationally intensive. Especially if the face is not centered, rotated and in the wrong scale all at once. Figure 3.8 shows an example of a rolled head that is normalized by in-plane rotation.



Figure 3.8: Example of an unaligned face that is aligned by in-plane rotation

The main idea in general object alignment is to use a lot of images and then try to find a common way to align them with each other. These methods iterate through a set of translations, rotations and scaling to find the best transformation that minimizes the difference of a set of images.

One example of this kind of technique is used for hand-written character recognition in digital images called congealing. Huang *et al.* [7] introduced it for other objects like faces and images of cars for number plate identification. A probability measure is used for the binary value of each pixel stack from an image set. The alignment is achieved by transforming one image at a time and minimizing the total entropy sum. New additions to the set can then be aligned using the saved transformation set from each iteration, this process is also called image funneling. Most methods would probably benefit from image alignment, unless the image acquisition process is normalized so that very little variation occurs among the images. It is claimed in [7] that many current methods ignore this important issue that has become its own sub-problem.

Wang *et al.* [23] suggest a method that adjusts face alignment online. An alignment candidate is selected corresponding to the largest performance metric extracted from an analysis of face recognition similarity scores. They showed that more exact (manually) marked eyes were better for recognition than when they would add small errors from uniform random noise. This confirmed that improvements can be made by an accurate aligning procedure.

3.5 Face recognition

Recognition, as in this case, usually means finding the distance to the trained classes using some kind of distance measure. A selection of the smallest distances is finally presented in order to the user. This is different from for example face authentication where the user have to claim an identity and the system will determine if it accepts the claim. A threshold is commonly used to decide if the distance between the input face is close enough to the face in the stored database. Face authentication systems are therefore employed as automatic since no user input is required after authentication. Face recognition systems on the other hand, are semi-automatic in the way that they require the user to evaluate the final results. The user will either accept one of the suggestions of who the person could be or manually input a different name. A simple example di-

agram in figure 3.9 shows how the face recognition is performed using already trained data vectors.

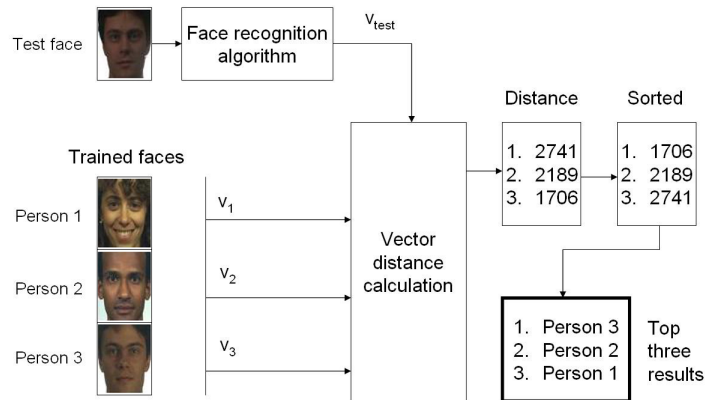


Figure 3.9: Simple example diagram of how face recognition is performed using already trained data

3.6 Image databases

3.6.1 ORL

This database contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge, UK. There are 10 different images of 40 distinct subjects. For some of the subjects, the images were taken at different times, varying lighting slightly, facial expressions (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). All the images are taken against a dark homogeneous background and the subjects are in up-right, frontal position (with tolerance for some side movement).



Figure 3.10: ORL sample images

3.6.2 Color FERET

This larger database contains a total of 11338 facial images. They were collected by photographing 994 subjects at various angles, over the course of 15 sessions between 1993 and 1996. This database is largely a color version of the original Facial Recognition Technology (FERET) Database which was released in 2001 and consisted of 14051 grayscale images. A regular frontal image was captured for every subject at every capture session. In almost every case, a second frontal was also captured, and half- and profile- left and right images were usually captured as well.



Figure 3.11: Color FERET sample images

3.6.3 LFW

Labeled faces in the wild is a database containing 13233 images of faces collected from the web. In total there are 5749 subjects where 1680 of them have two or more distinct photos in the data set. All images are unconstrained face photos since it was designed for studying the problem of performing face recognition under those conditions. The only constraint on these faces is that they were detected by the Viola-Jones face detector [15].



Figure 3.12: LFW sample images

3.6.4 EAB/TV

A new database was created at Ericsson using a mobile phone camera to have a even better representation of the possible variations using such devices. Three sample images were gathered for 22 persons resulting in a database of 66 images. Since the face detector only works on small pose-variations only frontal faces were captured but with varying lighting and background.



Figure 3.13: EAB/TV sample images

Chapter 4

Method

The methodology will be explained in detail in this chapter. First the general system description will be shown to get an overview of the whole architecture and the different parts. After that the suitable methods will be discussed a bit and also how the selection process was executed. All attempts at improving the input faces to the system are then covered before explaining how the algorithms were tested. The chapter ends with how the test sets were constructed from the image databases explained in section 3.6 and some additional optimization tests.

4.1 System description

The system used was built on a server-client architecture as shown in figure 4.1. An offline training procedure on a stationary computer (server) was performed on a stored database of images containing faces. Training basically means running the selected face recognition algorithm to create unique data to represent the persons in the database. The output data from the training was then transferred to a mobile phone (client). Comparable data is then also calculated from newly captured images taken with the clients camera. A distance is calculated to each of the trained classes using a suitable distance measure depending on the face recognition algorithm. The top three closest matches are finally presented on the client screen using the closest distance.

4.2 Suitable algorithms

Face recognition continuously adopts state-of-the-art techniques from research areas like image/signal processing, computer vision, pattern recognition and machine learning to name a few and has produced a vast number of algorithms and methods over the years. All methods come with their own advantages and disadvantages which makes choosing the best one suited for a certain task a time-consuming job.

With the aim of running the recognition on a mobile device, it becomes a search for efficiency and low complexity. The limited processing power and memory capacity, in a mobile phone for example, puts constraints on feasible methods.

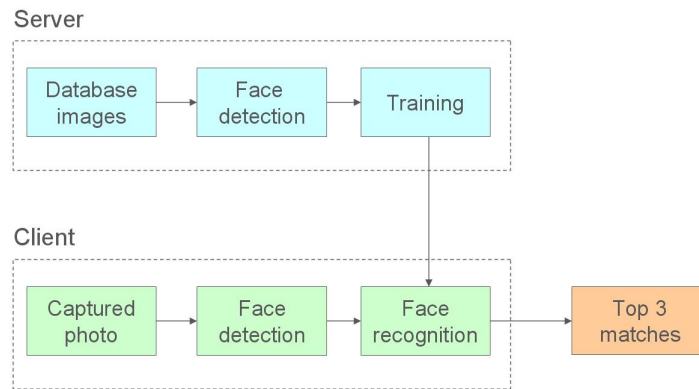


Figure 4.1: System description: client-server architecture

Despite the recent advances in feature-based methods, holistic methods is claimed to still give better performance overall by Zhao *et al.* [1]. It seems that the spatial information in the face plays an important role that is often lost when looking at specific features separately. However, improved results have been claimed [1] by detecting facial features and then using this information to normalize faces. Most algorithms just assume the face is properly aligned and this significantly degrades the performance of the recognition.

Variations in illumination, head pose and partial occlusion are other problems affecting the performance as well. Several proposed pre/post-processing techniques [5, 6, 7, 21] attempt to remove the effects of these problems, but most of them are not adaptive enough to work in any situation and others are just too complex or inefficient for mobile devices. One possibility is to process the database images before training to at least have normalized training data. Although depending on the face recognition method, it might result in losing discriminant information that could be valuable for different head-pose and illumination variations when matching a new face. However, the training set cannot be expected to contain every possible variation that might be encountered in future test images. Naturally it would be easier if the exact set of possible variations was known, but this will depend on the user and also vary from one camera to another.

The main consideration when looking at all the proposed methods was that the algorithms would work under all the limitations present on a mobile system. However, some conditions were also rated higher than others.

Order of importance:

- Computational complexity and memory requirements
- Implementation difficulty
- Recognition results under various conditions

After selective elimination of everything but a few methods in each algorithm category

(projection, statistical, graph matching and neural networks), the search was narrowed down to just two options since the graph matching and the neural network methods were discarded right away because of their complexity. The first was a novel method called Local Binary Patterns (LBP) originally intended for image textures but tested for face recognition as well by Ahonen *et al.* [3]. This is a holistic method that was found separately during the literature search. The second was a method called Discriminant Common Vectors (DCV) which is based on Fisher's Linear Discriminant Analysis from the holistic methods. Cevikalp *et al.* [4] proposed two different algorithms to obtain the discriminative common vectors. One using the within-class scatter matrix and the other using difference subspaces and a Gram-Schmidt orthogonalization procedure.

Both of these methods have very low computational complexity and reasonable memory requirements which is the biggest concern for mobile devices. DCV was considered slightly more difficult to implement than LBP, but comparable recognition results are claimed in frontal face recognition tests. Another good feature of the LBP method is that it is monotonic gray-scale invariant and therefore less sensitive to illumination differences than DCV. A self-made modification to LBP was also tested to try to reduce both memory and the amount of comparisons needed for matching called mean LBP.

When these methods had been implemented and tested there was some time left to look for a third option so a method called Kalmanfaces was selected. Eidenberger[10] showed that this method has low complexity, low memory requirements and was built to handle large variation in the input images.

4.2.1 LBP

Originally, the local binary pattern operator was designed for texture description. Ahonen *et al.* [3] then tried it for face recognition because of its efficiency and fast feature extraction. They claim superiority over all the other algorithms they compared to in tests on the FERET database.

The main idea is to threshold a small area around each pixel with the pixels intensity value to build a binary code where values greater than or equal to the center pixel are assigned to one and smaller values to zero. Depending on the choice of the surrounding area, it will be possible to detect different kinds of edges in an image. The default area is the 3x3 neighborhood with the current pixel in the center shown to the left in figure 4.2. Local texture can then be described using histograms of the binary values for a certain block in the image. The number of blocks and size of each block determines the level of retained spatial information.

Image histogram:

$$H_{i,j} = \sum_{x,y} I\{f_l(x,y) = i\}I\{(x,y) \in R\}, i = 1, \dots, n-1, j = 1, \dots, m-1 \quad (4.1)$$

Where $f_l(x,y)$ is a labeled image and i is the gray-value index ranging from 0 to 255 (n is normally 256). R is the image regions and j is the region index (m is number of blocks).

The operator was extended to use other areas than the default 3x3 neighborhood [3]. For example circular neighborhoods by interpolating the pixel coordinates, one example is shown to the right in figure 4.2. Even though larger texture can be detected using these neighborhoods it will also increase the binary code length and in turn memory usage. One idea was to somehow quantize the binary codes to achieve the same memory usage, but that would have similar effect as to just use a smaller neighborhood.

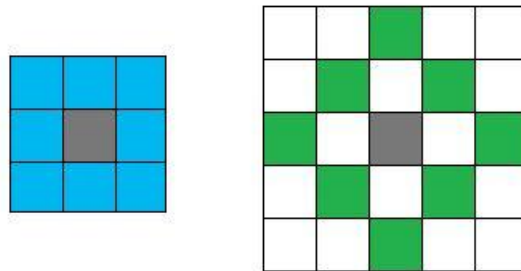


Figure 4.2: Default 3x3 and circular example neighborhood

Another extension was the introduction of so called uniform patterns [3]. A pattern is called uniform if it contains at most two bitwise transitions from one to zero or vice versa. Tests showed [3] that a bit less than 90% of the important patterns for face recognition are uniform while the other patterns mostly describe noise. Using only uniform patterns also reduces the size of the histograms so less memory is used in addition to increased performance. However, the non-uniform patterns are not totally discarded. Instead they are summed up and added in a separate bin. The number of uniform patterns use 58 bins so a total of 59 bins are used in each histogram compared to the default 256. If only the uniform and non-uniform patterns would be calculated and plotted as intensity values in an image it would look like in figure 4.3.

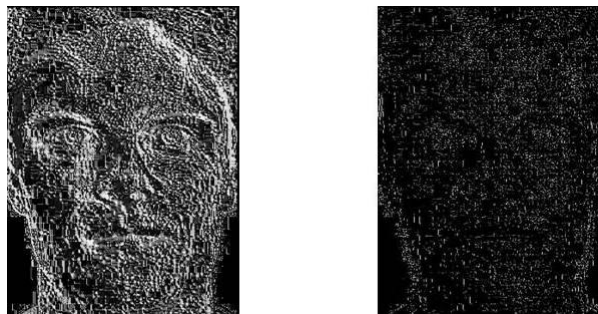


Figure 4.3: Uniform patterns and non-uniform patterns image visualization

Further reduction of the number of patterns used could be made by treating the binary codes with the same number of connected ones and zeros as rotated versions.

These rotation invariant uniform patterns suggested that changes in images of the same person could appear as a rotation of the patterns. Unfortunately, tests showed that this was not always the case since the recognition rates would actually decrease. It seems that the lost information contained discriminative information so there was a trade-off between memory reduction and performance that made it sub-optimal.

A histogram similarity measure called chi-square distance was suggested [3] to match the processed faces. Testing also revealed that this measure is more optimal than traditional distance measures like the Euclidean distance for comparing histograms.

Chi-square distance:

$$X^2(S, M) = \sum_{i,j} \frac{(S_{i,j} - M_{i,j})^2}{S_{i,j} + M_{i,j}} \quad (4.2)$$

Histograms S and M are compared by gray-scale i and region index j .

Attempts were made to optimize the pixel neighborhood and number of blocks that the face is divided into. After comparison, it was discovered that 5x5 blocks giving a total of 25 histograms was the optimal in respect to overall performance. An example image divided like this is shown in figure 4.4. This block division requires almost one fourth of the memory per image sample compared to 10x10 blocks even though it had similar performance. Overlapping blocks was not found to be any better than default side-by-side. Small changes in widths and heights were arbitrary and not as important as the overall division of the image. A measure of how the performance would change by increasing the block division simultaneously in both width and height is shown in figure 4.5.

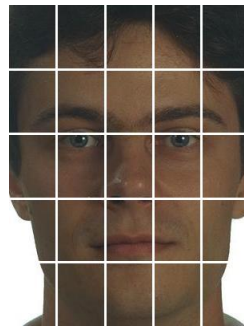


Figure 4.4: Example image divided into 25 blocks (5x5)

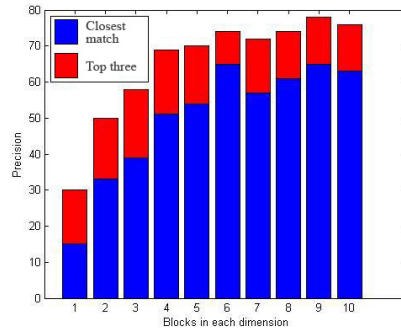


Figure 4.5: Performance of LBP when increasing the number of blocks from 1 (1x1) up to 100 (10x10)

The default pixel neighborhood was also tried at different offsets from the center pixel, but tests showed that there was no real big difference. Except if you increased the offset to more than three pixels that would gradually lower the recognition rates. However, a two pixel offset was finally chosen since it would most of the time give slightly higher recognition rates than the standard offset of one pixel. An illustration of how the pixel neighborhood expands with the offset is shown in figure 4.6.

3			3			3
	2		2			2
		1	1	1		
3	2	1	1	1	2	3
		1	1	1		
	2		2			2
3			3			3

Figure 4.6: Default neighborhood using different pixel offsets

Another paper by Liao *et al.* [20] suggested a new approach to the surrounding pixel neighborhood called Multi-scale Block LBP. Instead of directly using the local neighboring pixels to calculate the binary patterns a larger neighborhood was used. The neighborhood width and height should be dividable by three creating nine areas (nine 3x3 areas is one example), as shown in figure 4.7, where the center pixel is in the middle area. Each area is then converted to a single value by taking the mean value of the pixels contained in that area. These nine mean values are then treated as the default 3x3 neighborhood and the LBP code can be calculated as normal. The idea is that the larger the neighborhood is the bigger features are captured by the binary patterns. However, no big improvement was shown compared to the original LBP and this method also resulted in increased computation because of the larger neighborhoods and mean values. Also, the uniform patterns are not the same for this method and cannot be used in the same manner as described in [20].

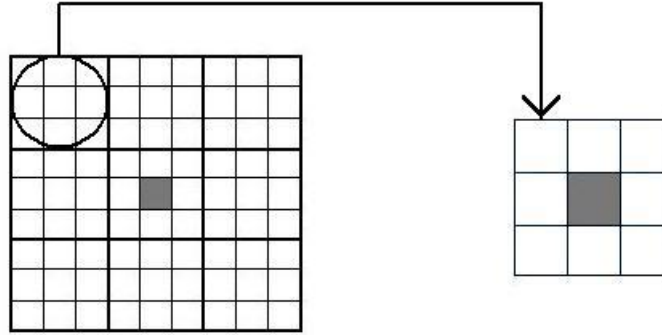


Figure 4.7: Multi-scale block neighborhood converted to default 3x3

In [3] they also tested weights for each block since some blocks might be more important than others for recognition. Although they state that the weights they used were most definitely not the optimal, it would still show increased performance so a similar block setup was tested with weights assigned accordingly. One problem is that the cropping they used was very close to the face and only using frontal faces which can explain why no real improvement could be noted. Attempts at creating the same conditions as they had showed a minor improvement but only under controlled variations in the images.

Face symmetry when dealing with frontal faces gave another idea that some blocks might also be almost symmetrical. This was used in an attempt to improve the extracted data from each image by performing a flipping procedure. The images were flipped horizontally resulting in new images that could be processed and merged with the data from the original images. It seems that important edges would be enhanced in the histograms that often showed improved recognition rates. A similar test was also done based on the fact that a flipped image would invert the shadows in the face. So by adding an extra set of histograms for each image in the database it was possible to account for more possible lighting variations. However, more memory was also needed using this procedure. An example image and its flipped counter-part is shown in figure 4.8.



Figure 4.8: Original image and horizontally flipped version

Block weights also gave the idea that memory usage could be reduced further if some blocks could be ignored entirely without losing important discriminant information. For this purpose a block mask was constructed after iteratively searching for important blocks. The tested masks are displayed in figure 4.9 where the black areas are not used. Between 6-23 of the most important blocks were kept from the total of 25. This reduced memory by up to a factor of 5 using the least amount of blocks while still showing decent performance in recognition.

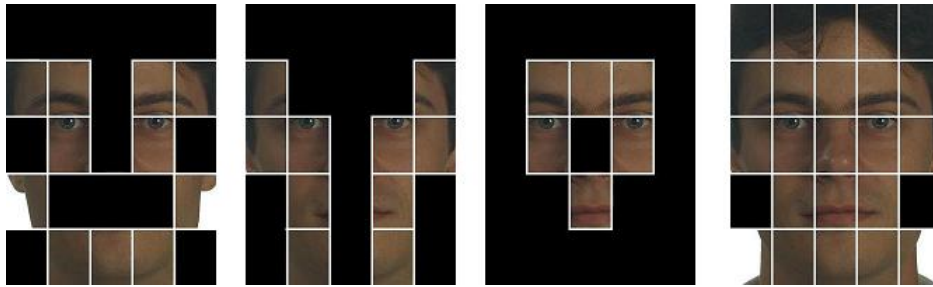


Figure 4.9: Tested block masks

The optimal blocks were found by looking at two difference measures called intra-person and extra-person difference. A mean intra-person difference was plotted for each block, shown to the left in figure 4.10, by comparing each possible pair of images containing the same person. Similarly a mean extra-person difference was also plotted, shown to the left in figure 4.11, using each possible pair of images containing different persons. Respective plots for the variance difference of each block were then plotted, shown to the right in figure 4.10 and 4.11. These differences show what blocks are good at representing images of the same person by having a low mean distance. Although they are also required to have a high mean distance for images of different persons for optimal classification. The most stable blocks were determined by looking at the block variance. Stability is needed to ensure that the blocks are not susceptible to noise.

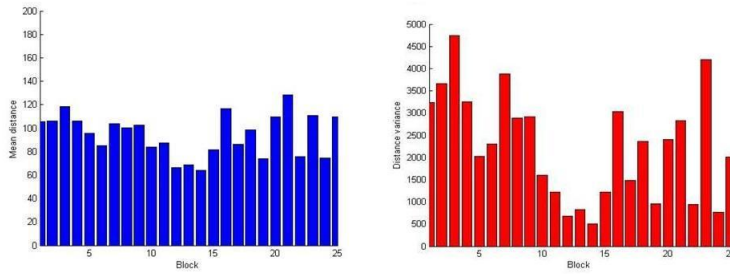


Figure 4.10: Intra-person mean block distance and variance

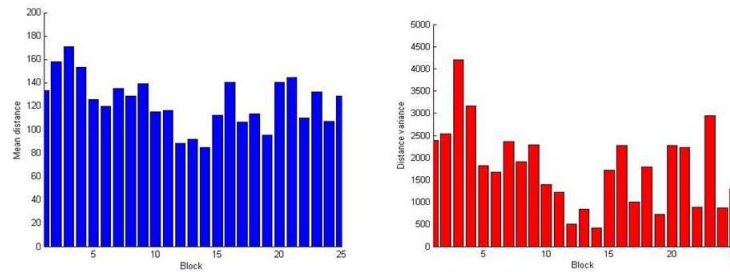


Figure 4.11: Extra-person mean block distance and variance

The found block masks were then tested against variations of the face location in the image and also the face size. This could simulate the effects of different bounding boxes acquired from a face detector. First the original bounding box was detected and then moved in both x and y-direction from -200 to +200 pixels on an arbitrary test image as shown to the left in figure 4.12. A match could be calculated for each displacement to see what boundaries each block mask could handle. After that they were also tested by increasing and decreasing the size of the bounding box using the normal detected center position fixed. Both the original box width and height were adjusted by -50 to +200 pixels in size as shown to the right in figure 4.12. The major conclusion that could be drawn from these tests was that block masks using more centered blocks would tolerate more movement than masks with many blocks near the image borders.

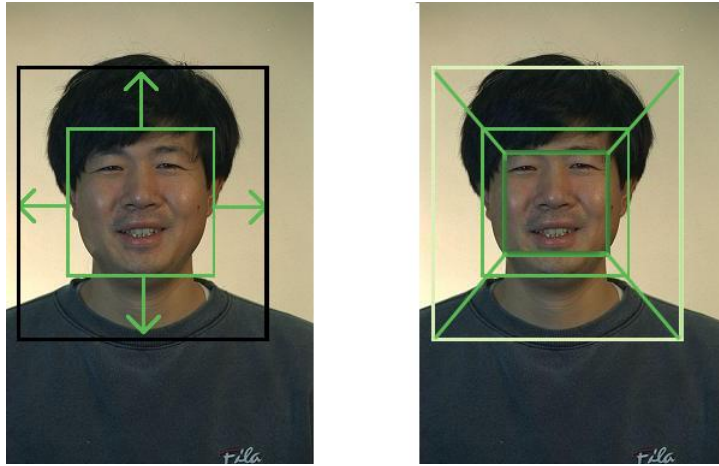


Figure 4.12: Moving the bounding box and increasing/decreasing the box size

4.2.2 Mean LBP

The modification of the LBP operator originated from the capabilities in DCV to only require a single test per class when performing recognition. As a first attempt, the mean intensity image of all the samples of each class was used in the LBP calculation creating a set of histograms for each class. However, the mean intensity image of all the samples removed a lot of discriminant information which made some classes very similar. An example mean intensity image is shown in figure 4.13 where the blurry and noisy effects are apparent.



Figure 4.13: Mean intensity image of a class (person)

This resulted in high error rates so another method was considered by doing it in reversed order. When first calculating the LBP for each sample of a class and then combining the histogram bins into one set of mean histograms instead, it seems that most of the discriminant information was kept. This can be explained by the nature of the histogram representation of each block in the image. The mean intensity image

of each class decreases the variation of gray-scale values in each block which often hold important information. A mean histogram will keep more information about the variation since LBP is describing different edges in the image by frequency in the histograms. Chi-square distance was kept as distance measure for this method since it still uses the same type of histograms as the normal LBP.

An additional idea was tested considering the zeros in the histograms instead of the actual values as normal. While the mean histograms for each person would describe the type of edges most commonly found in a persons face. The idea sprung from the fact that the edges not found at all could hold similar information. Although this seemed to be the case it would not increase the recognition rates, but they were surprisingly close and often using less data. One problem with this change is that the more training samples that are added the more edge types are typically found reducing the number of zero bins. This in turn reduces the amount of data that can be compared which also degrades performance.

4.2.3 DCV

The discriminative common vector method is based on a variation of Fisher's linear discriminant analysis for the small sample size case. Unless the images used for face recognition are very small, then the sample size is typically larger than the number of samples in the training set. This creates a problem since the within-class scatter matrix will be singular which means that LDA will not work. Figure 4.14 shows an example of two data scatters for two classes while the within-class scatter is visualized as the arrows from the center of the data sets. The idea of common vectors was introduced especially for this kind of problem which was first encountered in isolated word recognition. Cevikalp *et al.* [4] claim that during their tests, DCV was superior in terms of recognition accuracy, efficiency and numerical stability.

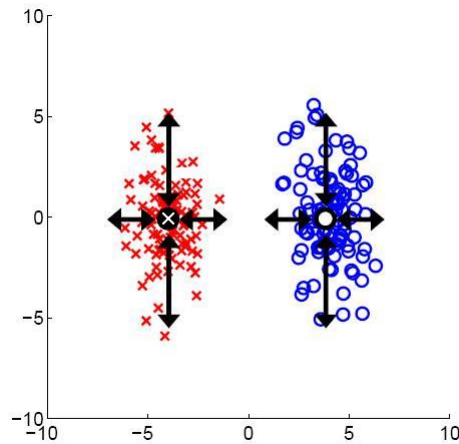


Figure 4.14: Within-class scatter visualization for two classes

The common properties of classes are extracted by eliminating the differences of the samples in each class. These properties are then described as a common vector after removing all the features that are in the same direction as the eigenvectors corresponding to the nonzero eigenvalues of the within-class scatter matrix. So called feature vectors are finally obtained from the common vectors by performing basically the same procedure again, but this time on the common vector scatter matrix. These feature vectors are the discriminant common vectors used for classification of new images. The calculations can be divided into three major steps.

Step 1:

$$S_w = AA^T \quad (4.3)$$

Where S_w is the within-class scatter matrix expressed using the image data scatter matrix A.

$$A = [x_1^1 - \mu_1 \dots x_N^1 - \mu_1, x_1^2 - \mu_2 \dots x_N^2 - \mu_2 \dots x_1^C - \mu_C \dots x_N^C - \mu_C] \quad (4.4)$$

Matrix A describes the scatter by subtracting the mean intensity μ from all the image samples x.

Step 2:

$$x_{com}^i = x_m^i - QQ^T x_m^i, m = 1, \dots, N, i = 1, \dots, C \quad (4.5)$$

Common vector x_{com} expressed using the orthonormal matrix Q which basically is the eigenvectors, image sample index m and class index i.

Step 3:

$$A_{com} = [x_{com}^1 - \mu_{com} \dots x_{com}^C - \mu_{com}] \quad (4.6)$$

A_{com} is the common vector matrix with subtracted corresponding mean value μ_{com} .

$$\Omega_i = W^T x_m^i, m = 1, \dots, N, i = 1, \dots, C \quad (4.7)$$

The discriminative common vectors or feature vectors Ω expressed using the orthonormal common matrix W, image sample x, image sample index m and class index i. The standard Euclidean distance was selected as a vector distance measure for this method as suggested in [4].

Euclidean distance:

$$E(U, V) = \sqrt{\sum_{i,j} (U_{i,j} - V_{i,j})^2} \quad (4.8)$$

Feature vectors U and V.

An attempt was made to reduce memory usage by removing the least significant eigenvectors. A projection matrix need to be stored since every new test image is projected into a reduced image space before comparison. The size of this matrix is dependant on how many of the calculated eigenvectors are used. However, reducing the size would quite drastically degrade the recognition rates for any meaningful reduction in memory usage.

The idea of important features was instead tested by dividing the images into two sub-regions. One for both eyes and the other containing the remaining parts of the image as shown to the left in figure 4.15. The two areas would then be calculated separately and combined using a weighing procedure. A higher weight was given to the eyes to see if this area would be more important for classification. However, this idea might work for other methods it was not satisfactory to use with DCV and no improvement could be noted.

Another image block division was tested for DCV inspired by the blocks used in LBP as shown to the right in figure 4.15. The image was divided into equally sized blocks where each block was calculated separately using DCV as normal. What could be noted was that smaller individual blocks were harder to classify and the standard large single block (whole image) was the optimal so no further block tests were tried.

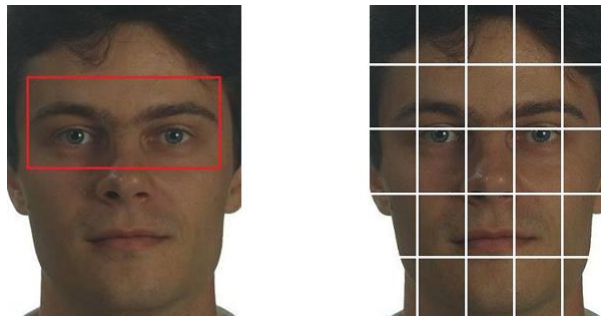


Figure 4.15: Eye mask sub-regions and LBP like blocks

Instead of manipulating the way of calculating the vectors some different types of input images was used. A processed image using the LBP code outputs as intensity values turned out to be really bad as input image. The biggest reason for this is the granular texture as can be seen in the example to the right in figure 4.16 hinting that DCV is sensitive to noise. From this an idea to use a mean filter-mask was tested to reduce noise which indeed showed improvement. A small 3x3 mask was found to be optimal, probably since bigger masks would also remove discriminative information in the face. This filter-mask would just smoothen the edges in the image as shown to the left in figure 4.16.



Figure 4.16: Mean-filtered and lbp-code input image

4.2.4 Kalmanfaces

A simplified version of the Kalman filter is used to detect invariant face features in this face recognition algorithm by Horst[10]. Kalman filters are more commonly employed for face detection in video sequences because of its powerful linear data processing capabilities.

Each face class is extracted as a single feature vector with sufficiently invariant luminance variance using downsized images. The vector is computed on the temporal sample sequence and then the pixels with a variance below a certain threshold are selected as features. An example of this is shown in figure 4.17.



Figure 4.17: Image sample series example and downsized image output showing the resulting features in gray-scale

This means that the length of the feature vector is not entirely dependent on the number of pixels.

Filter:

$$x_t = x_{t-1} + k_t(x_{t-1} - l_t) \quad (4.9)$$

For each pixel x , kalman coefficient k and luminance l at time t and $t-1$.

Kalman coefficient:

$$k_t = \frac{\sigma_{t-1}}{\sigma_{t-1} + \sigma_t} \quad (4.10)$$

The kalman coefficient calculated from the luminance standard deviation σ at time t and $t-1$.

A minkowski distance measure (or city-block distance) was chosen to match two images as in [10]. This measure is more suited for this method than others since the actual pixel values are used for comparison.

Minkowski distance:

$$d(f, c) = \frac{\sum_i^{n_c} |f_i - c_i|}{n_c} \quad (4.11)$$

Face f and class c with value index i and vector length n .

All sample images were reduced to the same size and several different combinations were tested. Both rectangular and square dimensions were tried but keeping the original aspect ratio was optimal. An image size of around 15x20 pixels was the smallest used since any further downsizing was sub-optimal. It seems reasonable that there is a minimum image size where it is still possible to discriminate between classes.

The variance threshold based on a percentage of the maximum found in the sample sequence was also varied for each test. In [10] they claimed that a very low threshold percentage was optimal since that would remove the unstable pixels often describing noise. Although this is true, it seems that a suitable threshold is very dependent on the database used. Images with high variation for example were poorly represented if the threshold was too low. An optimal threshold was therefore hard to find, but somewhere between 40-60% (0.4-0.6) would work decent in most cases.

4.3 Illumination normalization

4.3.1 Histogram equalization

The simplest technique to normalize the variations in illumination was considered to be normal histogram equalization. Images that are very dark or very bright become more normal because the histogram becomes more balanced instead of being centered in the low or high parts of the spectrum. A re-balancing example is shown in figure 4.18. Eramian and Mould[5] made a refinement of the original technique claiming better results with only a small increase in computation time.

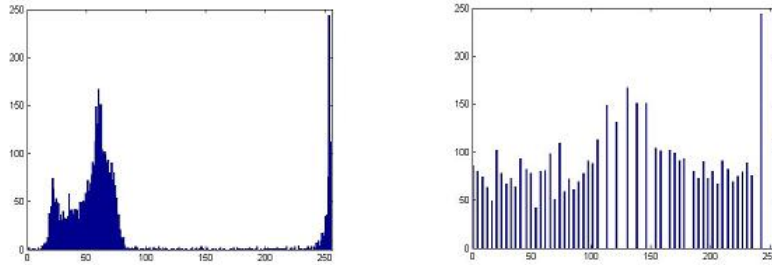


Figure 4.18: Original and histogram equalized gray-scale spectrum from an arbitrary dark face image

A modification to divide the image into blocks and then perform histogram equalization on each of them was also considered. However, it was rather difficult to find an optimal way of choosing these blocks to see any improvements, mostly because shadows are cast very differently on different faces. The face model can for example be simplified as a round object being lit up by surrounding light sources, but this would not be accurate enough. Another more straight forward approach using four blocks was instead tested centered around the nose. What would often happen using this method was that the block edges would be very sharp. Examples of processed images using these techniques are shown in figure 4.19.



Figure 4.19: Original, histogram equalized and local histogram equalized images

4.3.2 Three step method

Another more complex function was also created using the method described in Tan and Triggs[6]. The normalization is performed by a three step process starting with gamma correction, then applying a difference of Gaussian filter (DoG) and finally performing contrast equalization. Gamma correction enhances the dynamic range of the dark regions while Gaussian filtering reduces aliasing and noise. The contrast equalization then rescales the image intensities to a more robust overall contrast variation.

Gamma correction:

$$I = I^\gamma \tag{4.12}$$

Image I corrected using dynamic range parameter γ .

DoG filter:

$$D = G_1 - G_2 \quad (4.13)$$

$$G_i(x, y) = \frac{1}{2\pi\sigma_i^2} e^{-\frac{x^2+y^2}{2\sigma_i^2}} \quad (4.14)$$

Filtered image D which is the difference of the two filtered images G_1 using σ_1 and G_2 using σ_2 .

Contrast equalization:

$$I(x, y) = \frac{I(x, y)}{(\text{mean}(\min(\tau, |I(x', y')|)^\alpha))^{1/\alpha}} \quad (4.15)$$

Image intensity $I(x, y)$ equalized using truncation threshold τ and compressive component α .

The dynamic range parameter in the first step was set to 0.2 as the suggested default value [6]. In the second step the two Gaussian parameters were set to 1 and 2. A truncation threshold of 10 and the compressive component of 0.1 was used. A normalized example image is shown in figure 4.20.



Figure 4.20: Original and three step processed image

4.4 Face detection

4.4.1 OpenCV

A separate face detector on the client and the server was required since the environments are very different. A J2ME face detector from Ericsson [8] was used on the client. To get results as similar as possible to the clients face detector on the server, it was decided to use the Open Source Computer Vision Library (OpenCV) developed

by Intel [9]. OpenCV contains programming functions mainly aimed at real-time computer vision including an object detector. Face detection is done by using a so called Haar-classifier previously mentioned in section 3.3. The classifier was loaded from an xml-file trained to detect frontal faces in images. An example bounding box is displayed as a green rectangle in figure 4.21.

Similarity is important since the training data can affect the recognition accuracy on the phone if the images are slightly different from the captured test images. Exactly identical results are hard to achieve because of the round-off errors in J2ME compared to Matlab, but the principal is practically the same. Each detected face was cropped to a size of 60x80 to get normalized image dimensions.

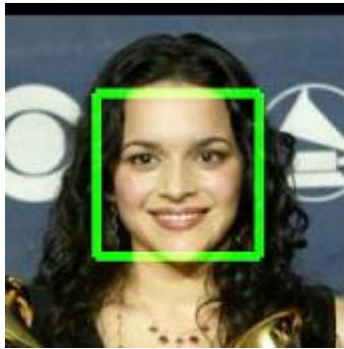


Figure 4.21: Detected face bounding box

4.4.2 RGB skin pixel model

Skin pixel detection was also considered as an alternative to face detection or maybe to improve the accuracy of another method. It felt like a natural decision to stay in the RGB color-space instead of doing any costly conversion since it was already used for both the server and client environments. A model built from a large database used for head tracking by Al Haj *et al.* [11] was found suitable for the task because of its simplicity. It is built on the assumption that the blue component is not as important for thresholding skin pixels as the red and green. In their tests over 94% of the skin pixels matched all of the found conditions. The detected skin pixels are set to 1 and the rest to 0 in a binary image as shown in figure 4.22.

Model properties:

$$R > 75 \quad (4.16)$$

$$20 < R - G < 100 \quad (4.17)$$

$$\frac{R}{G} < 2.5 \quad (4.18)$$

Red component R, green component G, red and green component difference and quotient were thresholded.



Figure 4.22: Original and detected skin pixels

Another addition to this model was tested in an attempt to find the largest ellipse in the image and thereby also the rotation of the detected face. A model for real-time face tracking by Hong *et al.* [14] suggested an ellipse fitting technique by calculating the second order moments of a binary image. However, such a method would increase the computational load of the system and needs to be accurate enough to not reduce performance by incorrect rotations. Unfortunately, no stable method was found that could handle the errors from the skin pixel model. The ellipse rotation ended up skewed when parts of a face was missing in the binary image. It seems more operations would be necessary to fill in the missing pieces of the face in order for the ellipse fitting to work correctly.

4.5 Pose normalization

4.5.1 In-plane rotation

Rotating the whole image as it is, also called in-plane rotation, was used to align the faces in an upright position. Using the height difference of the eyes made it possible to calculate the angle of rotation since they are most often unaligned when the head is tilted to the side. If the eye detection is inaccurate then the rotation can in fact make the pose even worse than before so robustness was the most important when looking at different eye detection techniques.

Functions from the OpenCV library can be used for more than just face detection by loading a different xml-file for the cascade. The good performance of the face detection gave hope of similar results by switching to a file intended for eyes. Using it in the same manner as when doing face detection was disappointing and it could hardly detect anything. The idea of running it inside a detected face box was also tested but with even worse results and so it was abandoned. Instead, rectangular features similar to the ones used for face detection but intended for the eyes were tested. Figure 4.23

shows examples of how the rectangles were built. The white areas were summed up and then the black areas were subtracted from that sum to detect areas with either a very high or very low total sum. Eye regions normally have a very low sum because of dark lines, dark eyebrows and the black pupil center. A problem was that unaligned eyes for example could often be missed since the features could not include both eyes at once.

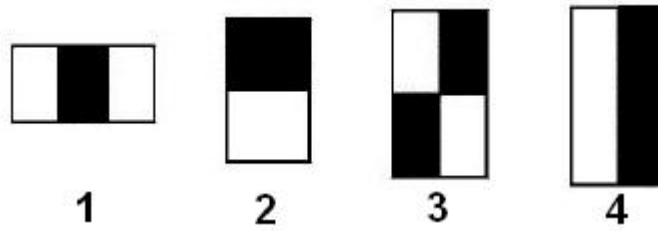


Figure 4.23: Rectangular features

The most suitable function for detecting eyes was based on a similar template matching procedure as in Asteriadis *et al.* [12]. A left eye template of 17x17 pixels was constructed and flipped horizontally to create a corresponding right eye template as well. Example templates are shown in figure 4.24. Then the closest matching area was found in the upper half of the image by sliding the templates and using the pixel intensities as a similarity measure. Both the templates and the test area were normalized to avoid brightness scale variations using the mean and standard deviation.

Template normalization:

$$I = \frac{I - \mu}{\sigma} \quad (4.19)$$

Image segment I normalized using mean intensity μ and intensity standard deviation σ .



Figure 4.24: Left and right eye template (17x17 pixels)

4.5.2 Background noise removal

One problem with the detected faces returned by the face detector comes from the fact that the classifier is trained on frontal faces only. This means that non-frontal faces can often be detected but including some background noise as shown in figure 4.25. The

most apparent way to minimize background noise is to improve the cropping so that the bounding box contains the most skin as possible ignoring pose.

A first attempt was using normal edge detection techniques to find the face edges and then re-size the bounding box. However, only looking at large intensity changes is quite subjective to noise so that a lot of background would still be included in the cropping. Instead the skin pixel model was used as an edge detector with more stable results. The left and right sides of the head was found by averaging the first detected skin pixel column on every row in the image. Then the top was also found by detecting the first pixel in the middle column of the image by starting at the first row and scanning downwards. However in some cases, background would be mistaken as skin and then the bounding box would still be incorrectly shaped.



Figure 4.25: Detected non-frontal face with background noise and improved cropped version

4.6 Training

The training phase of the face recognition system requires information about how many images correspond to each class respectively. However, not all systems take advantage of this when iterating through the database as long as each comparable output is linked to a certain class. As an example, LBP creates one output for each image sample from all the classes while DCV only creates one output per class. This implies that performance will be affected differently when the number of samples and classes in the training database increases. Even though all the training is performed on the server with a fast processor and plenty of available memory, it is still important since it needs to be processed down the line on the client.

Binary files were chosen as a way to more efficiently store all the training data before it was transferred to the client. This also made it relatively fast and easy to access on the client side. Even though more efficient ways of data storage exist, it served the purpose well enough for testing this application.

4.7 Testing

4.7.1 Creating the test sets

Two kinds of subsets were used for each test of the algorithms. The first subset was some randomly picked images of each test person from the training database i.e. already seen images. In the second subset, only images not in the training database was selected i.e. not previously seen images. Each test image was randomly selected from all the images of each class and both classes with a high as well as low number of samples were selected. This was intended to give a good representation of an average case situation and the possibility to compare the effects of having different number of samples. Four test sets with two subsets each was created from the gathered databases.

Test sets:

- ORL test set. For this small scale test set, 100 images displaying 15 different persons from the ORL database were selected for the training phase. Five of those persons were selected at random and three test samples of both seen and unseen images were picked out for each.
- FERET test set. The second test set from the FERET database was the first large scale set. A total of 274 images displaying 28 different persons were gathered for training. Five randomly selected test persons were selected, but this time with five seen and unseen images per person for the test phase.
- LFW test set. This test set was built to really see the effects of a very large database and also high variation that is found in the LFW database. A total of 1000 images displaying 54 persons were gathered for training. The randomly selected test persons were increased to 10 and 10 different seen and unseen images was gathered per person.
- EAB/TV test set. A similar test set was also built from the in-house database to see how different images taken with a mobile phone camera would compare to the other databases. This time all 66 of the images displaying 22 persons were used since it was still a quite small set. The whole database was selected as the seen subset while 25 new images displaying 17 of the persons were gathered as the unseen subset.

4.7.2 Adding additional training samples

In many cases there will only be a limited number of samples per person in the database on the client. Therefore a test was constructed to see how the performance varies by step-wise adding more samples for one test subject. The starting point was having three samples per person and a total of 22 persons in the database. A test set of 46 images of the test subject was gathered with strong variation in lighting and both indoor and outdoor images. The goal was to reach 100% correct suggestions in the top three closest results. Images from the test set were added until the goal was met. As shown in figure 4.26, the minimum amount of added samples needed to reach 100% top three correct result was three. Although most cases will require even more.

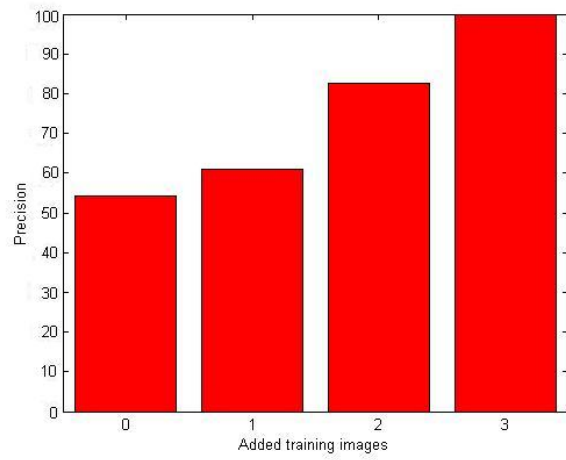


Figure 4.26: Improvements for LBP when adding additional training samples

Chapter 5

Experimental results

Test images were first sent as input to the face detector. The faces were then cropped and processed to be comparable with the training data for face recognition. Two different performance measures were used to compare the algorithms. Recall/Precision curves could be acquired from LBP since every sample of each class is measured using this technique. However, for the other methods only one measurement for each class is performed so this was more suitably represented as an average percentage or first match score.

5.1 Recall/Precision graphs for LBP

Recall can be described as the amount of training images returned by the system for a test person when going through the closest matches in order. The precision at each recall point describes the percentage of correct matches returned at that recall percentage. So a new precision is calculated for each training sample of the current test person returned by the system. It can be used to compare the total error the system makes for each test person described by the area above the lines. Test results on the different test sets are shown for seen images in figure 5.1 and unseen images in figure 5.2.

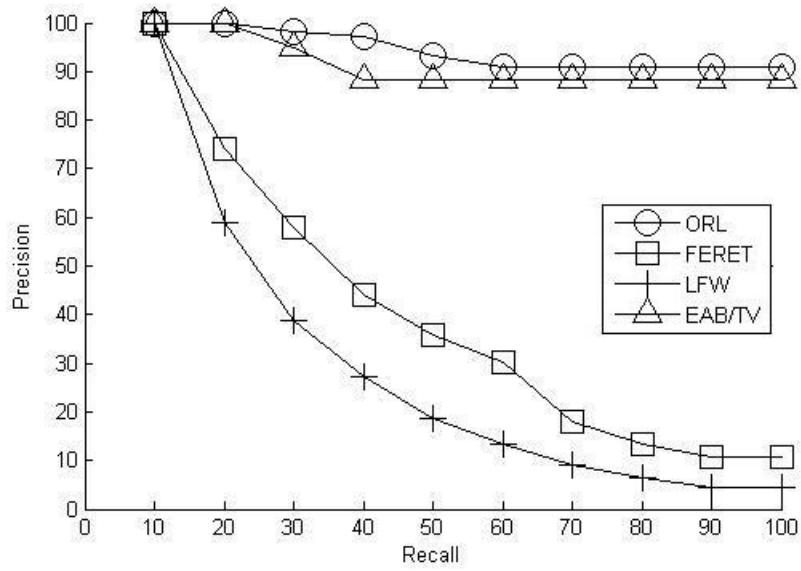


Figure 5.1: Recall/Precision results on seen images

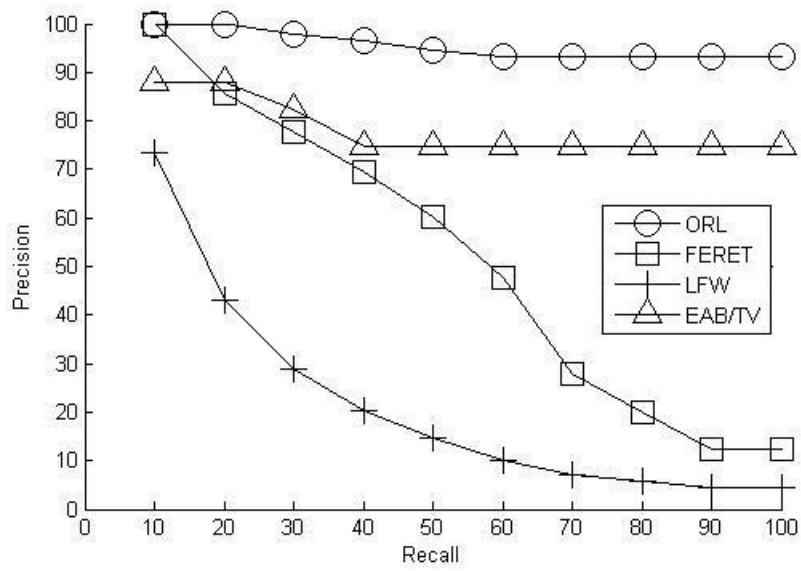


Figure 5.2: Recall/Precision results on unseen images

5.2 Overall first match scores

A different performance measure was used when comparing all the methods on the different test sets against each other. The number of correct identifications on the first returned result from the system from each test image was measured. These results could then be collected and presented as a percentage of correct first matches as shown for seen images in table 5.1 and unseen images in table 5.2.

This made it possible to see that mean LBP would perform almost as well as the normal LBP method while using a lot less memory. Another interesting notation was the increased performance from the mean filter as pre-processing for DCV.

Test sets	LBP	Mean LBP	DCV	DCV + mean filter	Kalmanfaces
ORL	100%	100%	100%	100%	100%
FERET	100%	100%	66%	100%	80%
LFW	100%	95%	31%	100%	31%
EAB/TV	100%	100%	100%	100%	95%

Table 5.1: Correct first match score on seen images

Test sets	LBP	Mean LBP	DCV	DCV + mean filter	Kalmanfaces
ORL	100%	100%	83%	100%	100%
FERET	100%	100%	63%	100%	71%
LFW	70%	67%	6%	29%	18%
EAB/TV	88%	92%	75%	83%	67%

Table 5.2: Correct first match score on unseen images

5.3 Recall/Precision graphs under various settings

The LBP method was also tested using different pre/post-processing techniques and block masks to separately to see how much each setting would improve the precision. This time only the LFW-database was used since it had the most room for improvement. Figures in this section only display unseen images since those are considered the most interesting.

The effects of the different illumination normalization techniques are shown in figure 5.3 where the most complex three step method had the highest precision. Using histogram equalization was a bit more disappointing since it actually reduced the performance a few percent compared to not using any pre-processing at all.

Figure 5.4 shows the effects of doing in-plane rotation and background removal. In-plane rotation would sometimes increase the performance by a few percent but was mostly quite close to the original performance of LBP. Background removal on the other hand, suffered from unaligned facial features as explained before in section 4.5.2 and would therefore show reduced performance.

All the tested masks and their respective performance is finally shown in figure 5.5. In most cases the precision would decrease when not using all the blocks as could be expected. However, one block mask was actually found to slightly increase performance. On the other hand, it was the one using 23 of the 25 blocks in total so not a lot of memory was saved.

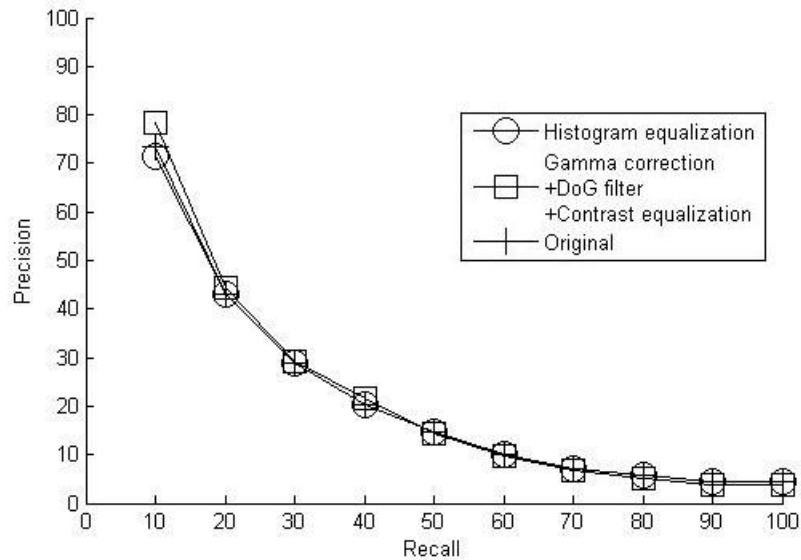


Figure 5.3: Recall/Precision results using illumination normalization techniques on unseen images

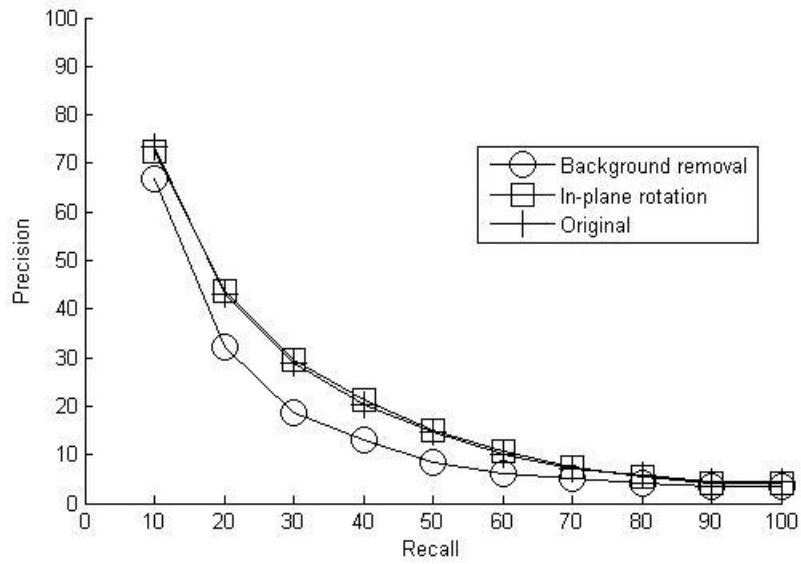


Figure 5.4: Recall/Precision results using pose normalization techniques on unseen images

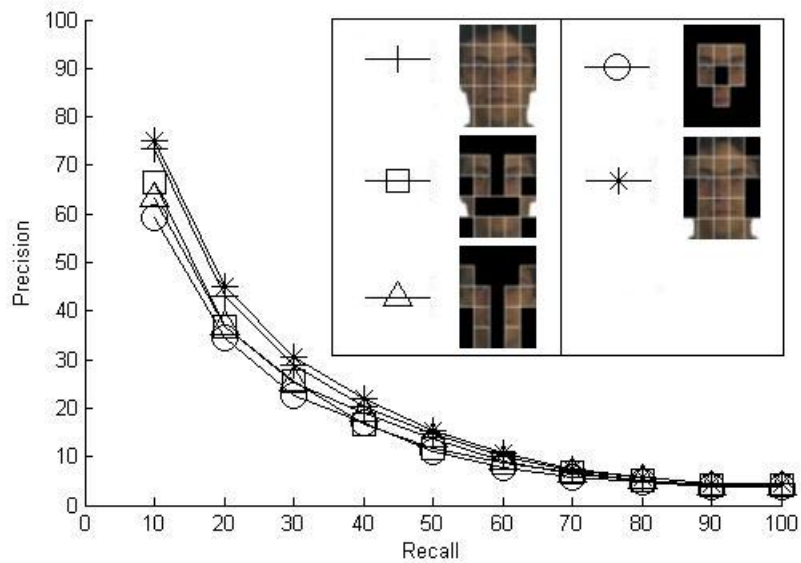


Figure 5.5: Recall/Precision results using different block masks on unseen images

Chapter 6

Demo

A demo application was created for the client and was tested on a Sony Ericsson phone as demonstrated in figure 6.1. The system is capable of showing the top three closest matches as name suggestions on the screen. The bounding box from the face detector is shown as a green rectangle.

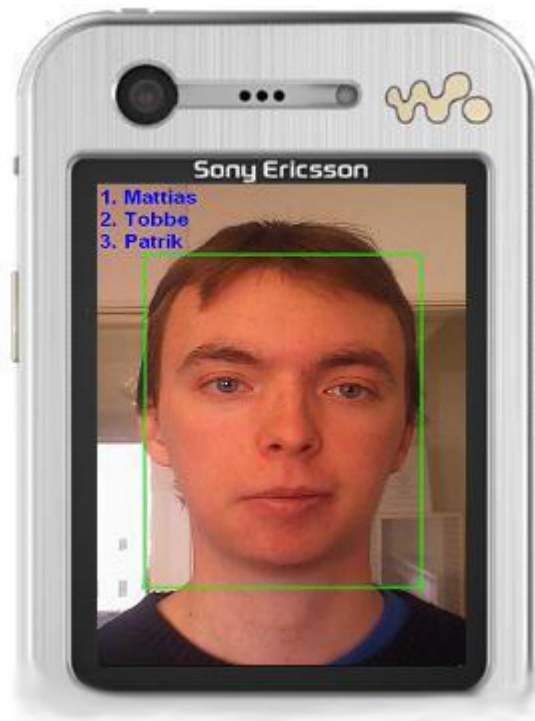


Figure 6.1: A detected face bounding box and the top three closest matches as name results on a phone

The steps that are performed on the phone in order:

- Manual photo capture
- Optional: Pre-processing such as illumination normalization, see section 4.3 for examples
- Face detection using the Ericsson face detector [8] to find all the faces in the photo, see section 4.4 for descriptions of each step
 - Calculate integral image
 - Resize the image to a smaller size
 - Find rectangular features using a moving window
 - Declare face candidates using a cascaded classifier
 - Repeat the above steps a few times
 - Return detected faces
- Optional: Post-processing such as pose normalization, see section 4.5 for examples
- Face recognition using one of the algorithms on a cropped face, see section 4.2 for algorithms
 - Process the face image to create output data/vectors depending on the algorithm used
 - Compare the calculated data to the stored training data using a suitable distance measure, for example vector distance
 - Return the closest matches by sorting the distances in order starting with the smallest
- Present the top three closest matches for the user as names in the top-left corner of the screen

The total time it takes to run the program is heavily dependent on the algorithm, number of included optional steps and also the size of the database stored on the phone. An average running time for the whole procedure with no extra steps from start to finish would take around 5-7 seconds with a database of around 100-200 images. Most users would probably accept a running time of below 10 seconds as long as the performance is high enough to motivate its use.

Chapter 7

Discussion

7.1 Conclusions

The most conclusions about an algorithm can be drawn when looking at its performance on the different databases. None of the algorithms had any trouble with the simplest ORL database but those results are to be considered as a best case scenario because of the small variations. FERET, on the other hand had a slight challenge in illumination as well as pose which clearly shows the advantage of edge-based methods like LBP. Projection methods such as DCV are more sensitive since they take no special note of neither overall nor local intensity. Kalmanfaces on the other hand, always consider the illumination variance of each pixel and remove the most unstable ones to alleviate this effect. To really separate the algorithms, the LFW-database was the hardest and most informative source of performance comparison. This database gave a good hint of how the different methods would stand against most of the really bad scenarios when using mobile digital cameras. A good mix of real background noise, blurred (out of focus), partially occluded and rotated faces also made a good benchmark for different pre/post-processing techniques.

Surprisingly, the standard illumination normalization techniques would actually decrease the performance on the LFW-database. Perhaps this could be explained by the fact that some discriminant information is also removed in the process. As expected the most computational method turned out to be the best and increased recognition rates by a few percent.

Normalizing the face by in-plane rotation increased the algorithms by between 0-5% on average where most of the increase could be noticed on the LBP method. An automatic face alignment system is still a very complex thing to do good considering robustness when it comes to pose. Even a slight error in the pose estimation can result in transformed training data that is worse than before.

The unpredictable nature of skin detection would often remove unnecessary background noise but sometimes also big parts of the face. This would sometimes result in stretching the face when cropping and resizing it so that important facial features became misaligned. Maybe it would be beneficial coupled with an additional alignment

procedure, but that would increase the computational cost substantially. Therefore, no complex iterative face alignment was tested as it was deemed unsuitable for mobile devices.

In most cases, increasing the number of training samples for each test class would also increase the recognition rates. Some exceptions occurred for the DCV method when it had trouble distinguishing unique vectors for some classes, while at the same time successfully discriminating other classes with fewer available samples. Only increasing the size of the database would somewhat decrease the recognition rates since the chance of misclassification would go up. This is also directly connected to the similarity between testing and training images. If samples that resemble the testing images are added then the rates should naturally improve, but otherwise they might only add to the chance of failure.

7.2 Optimal algorithm

How to select the best method suited for mobile devices is a question on its own. The most straight forward option would be to use the best overall recognition rate as the determining factor. On the other hand maybe some other conditions are equally important such as scalability or robustness in certain conditions. A typical user might not even encounter any of the worst issues related to scalability, but some most definitely will. A good balance between high recognition rates, reasonable robustness and scalability is probably the way to go.

Depending on the method it will be more or less beneficial to use different pre/post-processing techniques. What needs to be considered is the processing time even if the recognition rates can in some cases slightly improve. The optimal case would of course be a real-time system for maximum convenience and usability, but there is also a limit for the minimum level of robustness that makes time seem less relevant. Settling for a system with very little pre/post-processing and reasonable computation time seems like the only sensible decision to make today. Perhaps in the future when mobile phones are even faster and better it will be feasible to include such additions for improved performance.

The most scalable method was found to be Kalmanfaces since it has very short feature vectors. A small database with the number of samples close to the number of classes makes the LBP method demand less memory than DCV. However, as the number of samples per class increases the more data is needed for LBP and then DCV could actually be the better choice in terms of scalability. This fact originates from how the methods perform training and recognition as explained in section 4.6. The LBP method matches each test image against every class while DCV only have to match once against every class. There is the possibility to cap the number of training samples from each class for LBP to make it more scalable, but that would most likely decrease the recognition rates as a consequence. Using the mean histograms as in the mean LBP method suggested in this thesis is another way to sacrifice some performance for scalability.

Based on the findings from all the different tests and highly rating recognition rate strongly declares LBP as the overall optimal algorithm among these selected ones.

Chapter 8

Future improvements

8.1 Face synthesization

One area of improvement is to enhance the training data by face synthesis. Georghiades *et al.* [16] synthesized pose by creating a 3D-model of the face, doing rotation and then converting back to 2D again. Examples of how synthesized face images can look is shown in figure 8.1. Using this technique makes it possible to create both in-plane and out-of-plane rotated training data. All calculations can be performed on the server-side so the computational complexity is not a big problem. However, the synthesized images need to be stored on the client-side which could create a memory problem when there are many persons in the database.

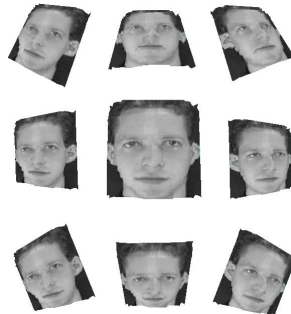


Figure 8.1: Synthesized pose variations

8.2 Artificial lighting

Different illumination conditions can be created by artificial light models. Using images of identical pose under different lighting variations makes it possible to reconstruct the shape and albedo of the face. The reconstructed shape can be used as a generative

model to render all kinds of different conditions by simulating point lights. One alternative is so called cone models based on the fact that all possible illumination conditions is a convex cone in the space of images. Georghiades *et al.* [16] suggested this method using their own version of photometric stereo to reconstruct a face. A model was built from seven images of the same pose but different lighting conditions to then predict large image changes. An example of how some rendered conditions could look like is shown in figure 8.2. As for synthesized pose, these variations need to be stored in a similar manner as extended training data requiring even more memory.

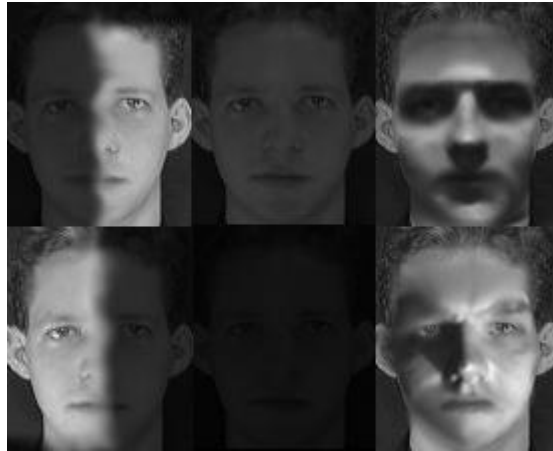


Figure 8.2: Artificial lighting conditions

Bibliography

- [1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey", *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, Dec 2003.
- [2] A. F. Abate, M. Nappi, D. Riccio, G. Sabatino, "2D and 3D face recognition: A survey", *Pattern Recognition Letters*, Vol. 28(14), pp. 1885-1906, 2007.
- [3] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns", in *Proc. 8th European Conference on Computer Vision*, ser. *Lecture Notes in Computer Science*, vol. 3021. Springer, 2004, pp. 469–481.
- [4] Hakan Cevikalp, Marian Neamtu, Mitch wilkes, atalay Barkana, "Discriminative Common Vectors for Face Recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, Jan 2005.
- [5] M. Eramian, D. Mould, "Histogram equalization using neighborhood metrics", *Proc. of the 2nd Canadian Conference on Computer and Robot Vision*, 2005, 397-404.
- [6] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions", in *Proc. AMFG'07*, 2007.
- [7] G. B. Huang, V. Jain, and E. Learned-Miller, "Unsupervised joint alignment of complex images", in *Computer Vision, 2007. Eleventh IEEE International Conference on Computer Vision*, 2007.
- [8] "Ericsson face detector", <https://labs.ericsson.com/apis/face-detector>.
- [9] "Open Source Computer Vision Library", <http://www.intel.com/technologies/computing/opencv>.
- [10] E Horst, "Kalman filtering for pose-invariant face recognition", In *Proc. of the International Conference on Image Processing, ICIP 2006*.
- [11] M. Al Haj, J. Orozco, J. Gonzalez, JJ Villanueva, "Automatic Face and Facial Features Initialization for Robust and Accurate Tracking", in *Proc. of the International Conference on Pattern Recognition, ICPR 2008*.

- [12] S. Asteriadis, N. Nikolaidis, A. Hajdu, and I. Pitas, "A novel eye-detection algorithm utilizing edge-related geometrical information", to appear in Proc. EU-SIPCO 2006, Florence, Italy, September 4-8. 2006.
- [13] V. Vezhnevets, V. Sazonov and A. Andreeva, "A survey on Pixel-Based Skin Color Detection Techniques", Proc. Graphicon-2003, pp. 85-92, 2003.
- [14] H.S. Hong, D.H. Yoo and M.J. Chung, "Real-time Face Tracker Using Ellipse Fitting and Color Look-up Table in Irregular Illumination", International Journal of Human-Friendly Welfare Robotic Systems, Vol. 3, No. 4, pp. 29-33, Dec. 2002.
- [15] P. Viola and M.J. Jones, "Robust Real-Time Face Detection", International Journal of Computer Vision 57, 137-154, 2004.
- [16] Georgiades, A.S., Belhumeur, P.N. and Kriegman, D.J. "From few to many: Illumination cone models for face recognition under variable lighting and pose", IEEE Trans. Pattern Analysis on Machine Intelligence. 23, 643-660.
- [17] "Face Recognition Vendor Test", <http://www.frvt.org>.
- [18] "Face Recognition Grand Challenge", <http://www.frvt.org/FRGC>.
- [19] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", In Computational Learning Theory: Eurocolt 95, Springer- Verlag, pp. 23-37. 1995.
- [20] S. Liao, X. Zhu, Z. Lei, L. Zhang, S. Z. Li, "Learning multi-scale block local binary patterns for face recognition", In ICB 2007, volume 4642 of Lecture Notes in Computer Science, pages 828-837, 2007.
- [21] S.G. Shan, W. Gao, B. Cao, and D. Zhao, "Illumination normalization for robust face recognition against varying lighting conditions", IEEE Intel. Workshop on Analysis and Modeling of Faces and Gestures, pp. 157 -164, 2003.
- [22] K. Peng, L. Chen, S. Ruan, G. Kukharev, "A Robust Algorithm for Eye Detection on Gray Intensity Face without Spectacles", Journal of Comp. Sci. and Tech. vol. 5, pp. 127-132, 2005.
- [23] P. Wang, L. C. Tran, Q. Ji, "Improving face recognition by online image alignment", Proc. of the 18th Inter. Conf. on Patt. Recog. (ICPR'06), 2006.
- [24] M. Turk and A. Pentland, "Eigenfaces for recognition", Journal of Cognitive Neuroscience 3, 72-86.
- [25] R. A. Fisher, "The statistical utilization of multiple measurements," Annals of Eugenics, 8:376-386, 1938.
- [26] M. Bartlett, Stewart, J. Movellan, R. Sejnowski, J. Terrence, "Face recognition by independent component analysis", IEEE Trans. Neural Networks 13 (6), 1450-1464, 2002.

- [27] P.J. Phillips, "Support vector machines applied to face recognition", *Adv. Neural Inform. Process. Syst.* 11, 803-809, 1998.
- [28] B. Moghaddam, A. Pentland, "Probabilistic visual learning for object representation", *IEEE Trans. Patt. Anal. Mach. Intell.* 19, 696-710, 1997.
- [29] A. V. Nefian and M. H. Hayes III, "Hidden Markov models for face recognition", In *Proc. Int. Conf. on Acou. Spee. and Sig. Process.* 2721-2724, 1998.
- [30] E. Fazl Ersi, J. S. Zelek, "Local Feature Matching for Face Recognition", In *Proc. of the 3rd Canadian Conf. on Comp. and Rob. Vis. (CRV'06)*, 2006.
- [31] L. Wiskott, J. M. Fellous, C. von der Malsburg, "Face recognition by elastic bunch graph matching", *IEEE Trans. Patt. Anal. Mach. Intell.* 19, 775-779, 1997.
- [32] J. Lu, N. Plataniotis, Kostantinos, N. Venetsanopoulos Anastasios, "Face recognition using LDA-based algorithms", *IEEE Trans. Neural Networks* 14 (1), 195-200, 2003.
- [33] A. Lanitis, C. J. Taylor, T. F. Cootes, "Automatic face identification system using flexible appearance models", *Image Vis. Comput.* 13, 393-401, 1995.
- [34] T. F. Cootes, G. J. Edwards, C. J. Taylor, "Active appearance models", *IEEE Trans. Patt. Anal. Mach. Intell.* 23, 681-685, 2001.
- [35] J. Huang, B. Heisele, V. Blanz, "Component-based face recognition with 3D morphable models", in *Proc. Int. Conf. on Aud. and Vid. Pers. Auth.*, 2003.
- [36] S. H. Lin, S. Y. Kung, L. J. Lin, "Face recognition/detection by probabilistic decision-based neural network", *IEEE Trans. Neural Netw.* 8, 114-132, 1997.
- [37] C. Liu, H. Wechsler, "Evolutionary pursuit and its application to face recognition", *IEEE Trans. Patt. Anal. Mach. Intell.* 22, 570-582, 2000a.
- [38] S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back, "Face recognition: A convolutional neural-network approach", *IEEE Trans. Neural Netw.* 8, 98-113, 1997.
- [39] Y. Adini, Y. Moses, S. Ullman, "Face recognition: the problem of compensating for illumination changes", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 19(7), pp. 721-732, 1997.
- [40] R. Basri, D. Jacobs, "Lambertian Reflectance and Linear Subspaces", *ICCV2001*, Beckman Institute, Vol. 2, pp. 383-390, 2001.
- [41] "Recent Advances in Face Recognition", Book edited by: Kresimir Delac, Mislav Grgic and Marian Stewart Bartlett, pp 97, I-Tech, Vienna, Austria, 2008.
- [42] S.K. Singh, D.S. Chauhan, M. Vatsa, R. Singh, "A robust skin color based face detection algorithm", *Tamkang J. Sci. Eng.*, 6, (4), pp. 227-234, 2003.