Embedded face recognition using cascaded structures

Citation for published version (APA):

DOI:
10.6100/IR613013

Document status and date:
Published: 01/01/2006

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Embedded Face Recognition
Using Cascaded Structures

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen op
dinsdag 3 oktober 2006 om 16.00 uur

door

Fei Zuo

geboren te Xi’an, China
Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. P.H.N. de With
en
prof.dr.ir. J. Biemond
# Contents

1 Introduction ................................................................. 1  
1.1 Background .............................................................. 1  
1.2 Automatic face recognition ........................................... 2  
1.3 Problem statement ...................................................... 5  
1.4 Research objectives and contributions .......................... 7  
  1.4.1 Contributions to face detection ............................... 8  
  1.4.2 Contributions to facial feature extraction ................. 8  
  1.4.3 Contributions to face identification ....................... 9  
1.5 Thesis outline and scientific background ...................... 9

2 Face Detection ............................................................. 13  
2.1 Introduction ............................................................ 13  
2.2 Key elements of face-detection techniques .................... 16  
2.3 Heuristics-based detectors ......................................... 18  
  2.3.1 Color-based detectors ......................................... 18  
  2.3.2 Template-based detectors .................................... 19  
2.4 Learning-based detectors ........................................... 22  
  2.4.1 Basic strategies for designing a classification function 22  
  2.4.2 Distribution-based detectors ................................ 23  
  2.4.3 Neural-network-based detectors ......................... 25  
  2.4.4 SVM-based detectors ........................................ 26  
2.5 Cascaded detector architecture ................................. 28  
2.6 Heterogeneous cascaded detector: a case study ............. 30  
  2.6.1 Skin-color detector ........................................... 31  
  2.6.2 Feature-geometry-based detector .......................... 31  
  2.6.3 Neural-network-based detector ............................. 33  
  2.6.4 Experimental results and discussions .................... 33  
2.7 Summary and conclusions .......................................... 34
3 Cascaded Face Detection Using Neural-Network Ensembles 35
3.1 Introduction .............................................. 35
3.2 Overview of the detector architecture .................. 36
3.3 Neural-network ensemble .............................. 37
   3.3.1 Basic principles of ensemble design .............. 37
   3.3.2 Ensemble architecture ......................... 40
   3.3.3 Training algorithms .......................... 42
   3.3.4 Performance analysis ......................... 46
3.4 Cascaded neural ensembles for fast detection ....... 54
   3.4.1 Formulation and optimization of cascaded ensembles .. 54
   3.4.2 Implementation of a CE detector ............... 57
   3.4.3 Performance analysis of the CE face detector .... 57
3.5 Application to real-world face detection ............... 59
   3.5.1 Strategies for real-world face detection ....... 60
   3.5.2 Performance analysis .......................... 61
3.6 Summary and conclusions .................................. 63

4 Facial Feature Extraction 69
4.1 Introduction .............................................. 69
4.2 Overview of facial feature extraction techniques ...... 71
4.3 Model-based algorithms for locating features .......... 73
   4.3.1 Deformable geometric templates ................. 73
   4.3.2 Statistical shape model ....................... 75
4.4 H-ASM: Haar-based shape model .................... 78
   4.4.1 Introduction to Active Shape Model (ASM) .... 78
   4.4.2 Enhanced Haar-based shape model .............. 80
4.5 Summary and conclusions .................................. 84

5 Cascaded Model-Based Facial Feature Extraction 87
5.1 Introduction .............................................. 87
5.2 Performance metrics for model-based algorithms ...... 88
   5.2.1 Basic concepts ................................ 89
   5.2.2 Performance metrics .......................... 89
5.3 Problem statement and design considerations .......... 92
   5.3.1 Problem statement ............................ 92
   5.3.2 Design considerations ........................ 92
5.4 Overview of cascaded extraction algorithm ............. 94
5.5 Sparse graph search (SGS) .............................. 96
   5.5.1 Sparse graph model ............................ 97
   5.5.2 Cost-function definition for fitting .......... 98
   5.5.3 Model-fitting strategy ........................ 99
   5.5.4 Efficient node selection ..................... 100
   5.5.5 Implementation and evaluation of SGS .......... 104
5.5.6 Short summary of SGS .......................... 108
5.6 Component-based texture fitting (CTF) .......................... 108
  5.6.1 Component-based feature model .......................... 109
  5.6.2 Cost-function definition for fitting ......................... 110
  5.6.3 Model fitting by direct parameter prediction ............. 111
  5.6.4 Prediction functions .................................. 112
  5.6.5 Implementation and evaluation of CTF ...................... 114
  5.6.6 Short summary of CTF .................................. 119
5.7 Component-based direct fitting (CDF) .......................... 121
  5.7.1 Component-based appearance model ......................... 121
  5.7.2 Direct optimization of the model parameters ............. 123
  5.7.3 Implementation and evaluation of CDF ...................... 124
5.8 Performance summary ...................................... 125
5.9 Summary and conclusions .................................. 128

6 Face Identification .......................... 131
  6.1 Introduction ........................................... 131
  6.2 Key elements of face-identification techniques ............. 134
  6.3 Feature representation .................................. 134
    6.3.1 Holistic features .................................. 135
    6.3.2 Structural local features ............................ 135
  6.4 Classification function .................................. 137
    6.4.1 Multi-class formulation for classification-function design 137
    6.4.2 Two-class formulation for classification-function design 141
  6.5 Summary and conclusions ................................ 142

7 Face Identification by Cascaded Selection 145
  7.1 Introduction ........................................... 145
  7.2 Cascaded face-identification scheme ......................... 146
  7.3 Cascaded LDA ........................................... 149
  7.4 Cascaded feature selection and classification .............. 150
    7.4.1 Motivation for adaptive feature selection ............ 151
    7.4.2 Class-specific feature selection ........................ 151
    7.4.3 Similarity function .................................. 155
    7.4.4 Complete cascaded algorithm ........................... 156
    7.4.5 Implementation and experimental results ............... 158
  7.5 Conclusions and discussions ................................ 162

8 Conclusions, Applications and Outlook 163
  8.1 Thesis summary and conclusions ............................ 163
  8.2 Evaluated applications .................................. 166
    8.2.1 Face recognition for smart homes ....................... 166
    8.2.2 Face recognition for secure biometric identification ... 169
Chapter 1

Introduction

1.1 Background

In the 1980s, Arnold Schwarzenegger played in The Terminator a human-looking, futuristic robot that was equipped with powerful voice and face recognition functions. It had the ability to precisely identify its surrounding people and interact with them in a smart way. Nowadays, professional face-recognition systems are already used in markets, targeting for applications such as airport security and border control. These systems belong to a category of content analysis systems that perform analysis on images or video sequences captured from cameras, with the purpose of raising the level of understanding of machines.

Different from traditional video processing such as compression and enhancement, video content analysis aims at extracting the semantic meaning from the video images. For example, for surveillance video, content analysis has the objective to find answers to typical questions like: who is present in the monitored scene and what activities are going on?

In order to derive the semantic interpretation of image content as described above, usually we first resort to basic image characteristics (e.g. by analyzing colors and edges). However, it is not straightforward to obtain high-level semantics from such analysis results. In most cases, this gap is bridged by techniques addressed in computer vision and machine learning. Although tremendous efforts have been spent in the past decades in these areas, a fully automatic content analysis system for general purposes still does not exist. For some applications, special requirements are imposed such as unconstrained
operation environments and high processing efficiency, which adds another dimension to the difficulties of the problem.

A starting point to approach the general content analysis problem is to constrain the problem and examine specific objects in video scenes. An active research topic here is to detect and identify a specific kind of objects which has a set of common characteristics while still having a large variability in appearances. Some examples of these objects are cars, pedestrians, etc. Among these, human faces have recently received significant attention in both academia and industry. This is because face is the personal communication center and the related research results can facilitate numerous applications [84], such as user identification and interaction, etc. Furthermore, human faces are typical examples of highly non-rigid objects, and techniques and insights obtained in face analysis can be beneficial for other object analysis tasks. In this thesis, we focus on the problem of face recognition and related topics such as face detection and facial feature extraction, with the aim to improve the efficiency and performance of the related techniques. We especially look into performance-related issues for face recognition and propose a number of techniques for high-performance face analysis.

The purpose of this chapter is to first give a brief overview of the face-recognition framework and define the problem statement of this thesis. The second half of this chapter highlights our major contributions and provides the content overview of this thesis.

The chapter is organized as follows. In Section 1.2, we briefly introduce a general face-recognition framework, where we divide the face recognition into a sequence of stages, namely, face detection, facial feature extraction and face identification. Section 1.3 discusses the performance requirements for designing face-recognition systems for consumer/embedded use. Furthermore, we point out the performance deficiencies of current face-recognition systems and introduce our approach for improvements. Section 1.4 presents our research objectives and specifies our major contributions in the three stages of face recognition. In Section 1.5, we outline the thesis structure and explain the related scientific background.

1.2 Automatic face recognition

The intriguing question of how to recognize faces has aroused great interest in scientific research for a long time. Human beings at a very early age are already capable of recognizing familiar faces without any difficulty. This ability shows remarkable robustness w.r.t. variable face appearances. Many studies have been carried out in psychophysics and neuroscience to figure out the brain mechanisms underlying this phenomenon [50].
1.2. Automatic face recognition

Figure 1.1: General framework of a face-recognition system.

However, it is an extremely difficult task to mimic the human visual perception system by computers. Despite this aspect, with the enduring efforts spent in the past decades, significant progress has been made in automatic face recognition, which has achieved reasonably good results under constrained situations. In the following, we first present an overview of a general face-recognition framework.

A typical face-recognition system usually consists of a series of processing stages, as depicted in Fig. 1.1. These processing stages are: face detection, facial feature extraction and face identification. Each processing stage forms a sub-system of general face analysis having its own characteristics. Let us now discuss these sub-systems in more detail.

1. **Face detection**
   Face detection locates face areas within input images or video sequences. It is a crucial preprocessing step of face recognition that takes arbitrary images as input. A central question in face detection is how to optimally separate face areas from arbitrary non-face background regions (see the example shown at the left of Fig. 1.2).

   As a sub-system, face detection has a number of specific applications. For example, face detection can facilitate object-based coding such as MPEG-4 [34], where face areas can be segmented from the video scene and encoded independently from their backgrounds [85][7]. Another popular application is face-based camera control in video telephony. For instance, a PTZ (Pan-Tilt-Zoom) camera can be automatically controlled by the location of the detected
faces in the captured scene, which enhances the user experience during the video communication.

2. Facial feature extraction
Facial feature extraction locates important feature positions within a detected face. The features which are most interesting are eyes, mouth, nose and eyebrows. In some cases, it is necessary to obtain more detailed descriptions of these features, such as their contours (see the example shown in the middle of Fig. 1.2). Accurate feature extraction facilitates face region normalization, where the detected face can be aligned to a common coordinate framework. This can significantly reduce the large variances introduced by different face scales and poses, which alleviates the difficulty for face identification. In addition, for some face-identification techniques, the accurate locations of feature points sampling the contours of facial features provide important input parameters for the face identification.

Furthermore, the accurate localization of facial features also facilitates some other face analysis tasks, such as facial expression analysis [69], face animation and face synthesis [41][1]. For example, the facial expression of a cartoon face can be driven by the expression of a real person based on the correspondences between their facial feature points.

3. Face identification
Face identification generates the final output of the complete face-recognition system: the identity of the given face image. More specifically, based on the normalized face image and the (optional) facial feature locations derived from the previous stages, a suitable representation (e.g. a feature vector) is generated from the given face and compared with a database of known faces. If a close match is found, the algorithm returns the associated identity. A key problem in face identification is the large differences between face images from the same person (intra-personal variances) as compared to those from differ-
ent persons (inter-personal variances). Therefore, it is important to choose a suitable face representation and classification technique that can provide a good separability between different persons.

Face identification has a wide range of applications. Because it offers a non-intrusive way for human identification, the face is used as an important biometric in security applications. Recently, face recognition has received wide interest in border control, and a number of countries are integrating facial information into the electronic passport in addition to several other biometrics such as fingerprints and iris [30]. In addition to security and law enforcement, face recognition is also applied in entertainment and consumer electronics as a means for a natural user interface. By recognizing the existence of the user and his identity, consumer devices can offer customized services, thereby creating an enhanced user experience.

In order to achieve a high-performance face-recognition system, each processing stage in the system has to be carefully designed to satisfy specific application requirements, which are discussed in the next section.

1.3 Problem statement

In this thesis, we aim at face recognition for embedded/consumer applications. With an embedded perspective, we mean that the system is actually a sub-system that is a part of and interacting within a larger system, so that it is usually constrained by several factors, such as cost, efficiency and applied technology, etc. In consumer applications, we have similar constraining factors and a good trade-off between cost and performance is required. As compared to traditional face recognition as used in security applications, face recognition in a consumer/embedded environment has the following characteristics:

1. Large variability of operating environments, e.g. complex image backgrounds, different illuminations, various imaging devices, etc.

2. Use of low-cost hardware (processing units and cameras), which provides only limited computation power and moderate image quality.

Correspondingly, face-recognition techniques designed for consumer applications should pay special attention to the following technical requirements:

1. Accuracy

As the basic requirement for face recognition, the final face-identification results should be as close to the ground-truth as possible. This requirement implies that each processing stage in face recognition should provide accurate results. For instance, the face detection should locate existing faces in an input image with minimum false detections. The facial feature extraction should
find precise locations of relevant feature points. The final face identification should be able to locate the ground-truth identity of the given face. It should be noted that the accuracy of the final identification is heavily influenced by that of the face detection and the feature extraction.

2. Robustness and stability
The robustness of a face-recognition system refers to the system’s ability to generate stable outputs even with large input variations. For example, the face detection should be able to cope with unconstrained image inputs. The feature extraction should handle possible large deviations from the face-detection outputs. The final identification should give consistent results even when the faces for the identification undergo certain changes (e.g. aging, illumination changes) as compared to the reference faces stored in the database.

It should be noted that we do not attempt to address all aspects of robustness for face recognition. For example, our work is limited to (near) frontal view only and does not explicitly address variable face poses [48]. However, many techniques proposed in the thesis can be extended to cope with multiple poses by incorporating additional models such as a 45-degree side-view model and a profile-view model, and each model can be derived using similar techniques as used in the frontal-view model.

3. Processing efficiency
For consumer and embedded applications, (near) real-time performance is usually required for low-cost hardware. Therefore, the algorithm should have sufficient efficiency to satisfy these requirements. In this thesis, we always consider high efficiency together with high accuracy as our major design objectives.

As mentioned earlier, tremendous efforts have been spent in the past years on face-recognition research, and many techniques and commercial systems have been developed. In accordance with these developments, many evaluation datasets and protocols have been proposed to benchmark the performance of face-recognition techniques. Several well-known test sets are the CMU+MIT sets for face detection [60] and the FERET test protocols for face identification [54][55]. The Face Recognition Vendor Tests (FRVT) have provided independent government evaluations of commercially available and prototype face-recognition technologies [67] and a new test-round is being organized in 2006. It has been concluded from these tests that the current automatic face-recognition systems are capable of handling large databases that far exceed the human brain capacity. However, their accuracy and robustness are still very limited.

For real-life field tests such as airport security and criminal tracking, current face-recognition systems have received substantial criticism in recent
1.4 Research objectives and contributions

During our study of the state-of-the-art techniques, we have found that current techniques for face recognition usually adopt a single algorithm at each stage of the face recognition, which is insufficient to satisfy the performance requirements in real-life applications. To this end, in this thesis we aim at using multiple algorithms that coordinate with each other for overall enhanced performance. More specifically, for each major processing stage of face recognition (face detection, facial feature extraction and face identification), we propose a series of novel algorithms and further construct them in a cascaded structure. The use of the cascaded structure offers joint improvements w.r.t. accuracy, efficiency and robustness. This is explained in more detail for each stage later in this section.

For the three stages, we have the following common research objectives:

1. Design of individual algorithms to satisfy system requirements on accuracy, efficiency, etc;

2. Optimization of algorithm structure (e.g. optimized ordering of and coupling between constituting algorithms) to improve the overall system performance.
The basic idea is to use a *divide-and-conquer* strategy, where we design an algorithm that is efficient for a particular task and compensate for missing aspects with supplementary algorithms. By appropriately combining a set of algorithms together, a high overall performance can be achieved.

In the following, we present our contributions more specifically for each stage within face recognition.

### 1.4.1 Contributions to face detection

A major problem in face detection is the conflict between the detection accuracy/robustness and processing efficiency. In order to achieve all the performance requirements without having to trade-off one for the other, in this thesis, we adopt a pruning cascade structure for face detection. More specifically, we propose the following two novel solutions for the cascaded face detection.

1. **Heterogeneous cascaded detector based on various image features.** The cascaded detector is composed of the following sub-detectors: a color-based detector, a feature-geometry-based detector and a neural-network-based detector. The three detectors look at different image characteristics of facial areas and have an increasing complexity and detection capability. By cascading them into a chain, we can achieve both high processing efficiency and detection accuracy.

2. **Homogeneous cascaded detector based on neural-network ensembles.** The proposed detector is constructed by the following steps. Firstly, for improved detection accuracy, we propose a novel ensemble technique to build a coordinated set of neural networks. A new training strategy is proposed which outperforms other state-of-the-art techniques. Secondly, for improved detection efficiency, we build a cascade of neural-network ensembles by increasing complexity of base networks. The approach achieves one of the best detection accuracies in literature with a significantly reduced computation cost.

### 1.4.2 Contributions to facial feature extraction

Our first contribution in this part is an enhanced version of one representative extraction algorithm, the Active Shape Model (ASM). We improve the local feature representation in the conventional ASM by using 2-D Haar feature modeling. The 2-D patterns contain richer and more reliable local feature descriptions, resulting in a doubled convergence area as compared to the conventional ASM. Simultaneously, the Haar modeling of the 2-D features also facilitates real-time implementation.

Our second contribution in this part is a cascaded feature-extraction technique with incremental accuracy. We propose a set of principles for construct-
1.5. Thesis outline and scientific background

Fig. 1.3 depicts the structure of the thesis. Besides the introduction and conclusion chapters (Chapter 1 and Chapter 8), the thesis consists of three parts based on the major processing stages in face recognition: face detection, facial feature extraction and face identification. Chapters 3, 5, and 7 contain our major contributions for each stage, and Chapters 2, 4, and 6 contain the
Chapter 1. Introduction

Introduction of the face-recognition problem and summary of the research contributions.

Chapter 2. Face detection
1. Overview of face-detection techniques.
2. Introduction of cascaded detector-structure and a case study of a heterogeneous cascaded detector.

Chapter 3. Cascaded face detection using neural-network ensembles

Chapter 4. Facial feature extraction
1. Overview of facial-feature-extraction techniques.
2. An enhanced model-based extraction algorithm (H-ASM).

Chapter 5. Cascaded model-based facial feature extraction
1. Introduction of the cascaded extraction framework.
2. A new three-stage cascaded extraction algorithm (sparse graph search, texture fitting, direct fitting).

Chapter 6. Face identification
Overview of face-identification techniques.

Chapter 7. Face identification by cascaded selection
1. Introduction of the cascaded identification framework.
2. Cascaded LDA and cascaded feature selection and classification.

Chapter 8. Conclusions and future work
Thesis summary, applications and future directions.

Figure 1.3: Structure of the thesis reflecting the three stages of face recognition, containing the specific overview and contribution chapters.

overview of corresponding techniques and some preliminary/additional contributions from our side. In the following, we briefly summarize the contents of each chapter.

Chapter 1 introduces a generic face-recognition framework consisting of three processing stages: face detection, facial feature extraction and face iden-
tification. Afterwards, the problem statement for embedded applications of face recognition is addressed, followed by the primary research objectives and a detailed summary of our contributions to each stage.

In Chapter 2, we first present an overview of the state-of-the-art face-detection techniques, which are classified into heuristics-based and learning-based algorithms. We point out a key problem in achieving high-performance face detection: the conflict between the detection accuracy/robustness and the processing efficiency. To this end, we propose a heterogeneous cascaded detector to satisfy both performance goals, which consists of a skin-color detector, a feature-geometry detector and a neural-network detector. The concept of combining an efficient color-based detector and a more elaborate feature-geometry detector was published in the Proceedings of the 23rd Symposium on Information and Communication Theory in the Benelux [86]. Later, a third detector based on neural networks is added to the processing chain and the corresponding result was published in the Proceedings of the SPIE Visual Communications and Image Processing conference in 2003 [88].

Chapter 3 discusses a new homogeneous detector using a cascade of neural-network ensembles. Two architectural issues are addressed: (1) parallel ensembles of neural networks for improved detection accuracy, and (2) cascading of neural-network ensembles for improved processing efficiency. We also evaluate the proposed technique and compare its performance in real-life detection tasks to the state-of-the-art techniques in literature. This contribution has been published in the Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems [97] in 2005 and an extended version has been recently submitted to the IEEE Transactions on Systems, Man and Cybernetics (Part B), special issue on recent advances in biometric systems.

In Chapter 4, we first present an overview of the state-of-the-art facial feature extraction techniques. We especially look into a well-known model-based algorithm, Active Shape Model (ASM), and present an improvement of ASM by incorporating Haar-feature modeling. The idea of using 2-D patterns for an enhanced local feature representation was published in the Proceedings of the SPIE Visual Communications and Image Processing conference in 2004 [92]. The further improvement by using Haar features was published in the Proceedings of the IEEE International Conference on Image Processing [91] in 2004.

Chapter 5 analyzes the performance limitation when using a single algorithm to achieve robust and accurate facial feature extraction. To this end, we propose a cascaded framework using a set of model-based algorithms to improve the overall performance. The framework consists of three component algorithms: sparse graph search, component-based texture fitting and component-based direct fitting. The design of the last two algorithms was first published in the Proceedings of the International Conference on Image
Analysis and Recognition [93] in 2004. Later in 2005, we extended the processing by adding a sparse graph model to ensure globally-correct estimation of feature positions. The complete system was published in the Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance [96].

Chapter 6 presents an overview of face-identification techniques based on different feature representation and classification function.

In Chapter 7, we improve the current face-identification techniques by using a selective cascade. Our first realization of the selective cascade is based on Linear Discriminant Analysis, which was published in the Proceedings of the 25th Symposium on Information and Communication Theory in the Benelux [95]. A second realization is based on adaptive feature selection and classification, which has achieved more efficient performance for large-scale databases. This result was published in the Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems [100] in 2005.

Chapter 8 summarizes the thesis and discusses several applications where our proposed face-recognition system was embedded. We also analyze the status of the current work and point out future directions. This chapter is related to several publications addressing the system aspects of embedded face recognition. For example, based on our integrated system implemented within the project HomeNet2Run [59], we published our first results in the Proceedings of the 24th Symposium on Information and Communication Theory in the Benelux [87]. Later, a more complete system description for consumer device interaction was published in the Proceedings of the IEEE International Conference on Consumer Electronics [99] and the IEEE Transactions on Consumer Electronics [98] in 2005. The face-identification work addressed in this thesis was also integrated with a privacy-protected face biometrics system at Philips Research Labs, and the related publications were in the Proceedings of the IEEE workshop on Automatic Identification Advanced Technologies [32] in 2005 and the SPIE Proceedings on the Security, Steganography, and Water-marking of Multimedia Contents conference [72] in 2006.
Chapter 2

Face Detection

2.1 Introduction

Face detection is the first processing stage in a face-recognition system (see Fig. 2.1). It refers to the process which segments possible faces from their background. This process takes an image (or a video sequence) as input, and returns the location and scale of each face, if there exists any. This is of crucial importance for unconstrained face recognition, where no a-priori knowledge is available as to the locations and scales of possible faces in an image. In this thesis, we consider detectors that are capable of finding multiple faces in one image.

In addition to face recognition, face detection is also an important research topic for a number of reasons.

1. Face detection is an indispensable preprocessing step for a number of face-analysis applications, including face identification, facial expression analysis and face coding [80].

2. Faces are typical examples of highly non-rigid objects. The insights and techniques obtained from face-detection research can broadly facilitate the detection of deformable objects in visual scenes.

For face detection, generally we need first to find a suitable representation (called feature representation) of the original image/video signals by using e.g. (multidimensional) vectors. Various image/video processing techniques can be applied at this stage to derive an effective feature representation.
Based on those feature representations, the second step in face detection is to define a decision criterion for separating faces and non-face backgrounds, which should be able to optimally discriminate faces from non-face background image signals. However, it is a challenging problem to define such a discrimination criterion, due to different factors related to the varying face appearances.

1. **Intrinsic variation of faces**: The human face itself varies strongly in its appearance, making the characterization of the human face difficult. The highly variable appearances of faces can be attributed to ethnic diversity, expressions, poses and aging, etc.

2. **External conditions**: Environments also exert significant influences on facial appearances in the image, such as imaging sensors and illumination conditions. Furthermore, the diversity of background images adds another dimension of difficulty to the discrimination between face and non-face images.

In the past decade, extensive research has been carried out on face detection, and significant progress has been achieved to improve the detection performance with respect to two goals: detection accuracy and efficiency.
2.1. Introduction

Detection accuracy. The accuracy of face detection can be described using a Receiver Operating Characteristic (ROC), which is a curve widely adopted in signal-detection theory. An ROC is essentially a scatterplot that shows the relationship between the false acceptance rate and the true acceptance rate. The false acceptance rate measures the likelihood of a face detector to incorrectly accept a background image as face. The true acceptance rate measures the likelihood of a face detector to correctly identify a face. In this and the following chapters, the true acceptance rate is also referred to as face detection rate.

The ROC curve of a given face detector shows its performance as a trade-off between the false acceptance rate and the face detection rate by varying its discrimination criterion (e.g. a threshold parameter) [17]. Note that different detectors can have different ROC curves. For example, in Fig. 2.2, we show two typical ROC curves for two detectors. It can be seen that detector A has a higher accuracy than B, since for any given false acceptance rate, detector A achieves a higher face detection rate than B.

Detection efficiency. The detection efficiency is often characterized by the operation speed of the face detector for a given platform. An efficient face detector should have a low computation cost for a predetermined performance required by the corresponding application. This is especially important for real-time embedded applications, where the face detector is required to process one image at sub-second level.

\[ \text{false acceptance rate} \]

\[ \text{true acceptance rate} \]

\[ \text{face detection rate} \]

\[ \text{hit rate} \]

\[ \text{false alarm rate} \]

Figure 2.2: Receiver Operating Characteristics (ROC) curves used for measuring the face-detection accuracy.
In the current face-detection literature, a performance trade-off still exists between the detection accuracy and efficiency. Some earlier detectors (e.g. [28][61]) based on color and shape features give fast performance, however, they are usually vulnerable to environment changes (e.g. illumination or imaging devices). In order to cope with the discrimination complexity involved in face detection, many techniques such as the use of neural networks [60] and Support Vector Machines [25], often resort to computationally intensive solutions, which unfortunately hinder the detection efficiency. In the first part of this chapter, we aim at giving a brief overview of existing face-detection techniques and analyze their merits and pitfalls. We pay special attention to the performance of these techniques in terms of detection accuracy and efficiency. The discussion reveals that it is difficult to achieve both performance goals with a single detection algorithm.

In order to cope with the accuracy-efficiency trade-off, a so-called cascaded face-detection structure is proposed in literature. By properly cascading a number of detection algorithms, the overall detection performance can be improved in both accuracy and efficiency. Examples of cascaded face detectors are found in [75] and [40]. In the second part of this chapter, we present our earlier work [88] as a case study, which is among the earliest work on cascaded face detection. The detector utilizes a color-based detector, a feature-geometry detector and a neural-network-based detector. The three subdetectors are arranged in such a way that the detection efficiency can be greatly improved while maintaining the overall detection accuracy.

The remainder of this chapter is organized as follows. In Section 2.2, we first present a general overview of face-detection techniques and their constituents. These techniques are classified into two categories: heuristics-based techniques and learning-based techniques, which are discussed in more detail in Section 2.3 and Section 2.4, respectively. Roughly characterizing, heuristics-based detectors are relatively fast, but they are less accurate in dealing with complex, real-world environments. In contrast with this, learning-based detectors offer more accurate detection results, but they usually require more computation cost. In order to cope with this controversy, in Section 2.5 and Section 2.6, we illustrate the cascaded detection structure and present one of our earlier contributions of a heterogeneous cascaded detector. Section 2.7 summarizes and concludes this chapter.

2.2 Key elements of face-detection techniques

Tremendous effort has been spent on designing face detectors in the past decade. Various techniques have been proposed in literature, which range from simple heuristics-based algorithms to advanced machine-learning-based
2.2. Key elements of face-detection techniques

A basic classification unit represented by $x$, classification function (rule) $f(x) = \begin{cases} \text{face area} \\
\text{non-face area} \end{cases}$, input image $f(x)$

Search strategy

Figure 2.3: Three basic elements in a face detector.

algorithms [80]. However, most techniques can be characterized by three basic elements (see Fig. 2.3):

1. *Feature representation*: For each technique, specific features have to be defined first to represent a basic classification unit (a pixel or an image window). For example, in color-based detection techniques [56][28], a pixel is a basic classification unit, which is represented by its associated color components. In neural-network-based detection techniques such as in [60], an $M \times M$ image window is a basic classification unit, which is represented by a vector of its pixel intensities.

2. *Classification function (rule)*: A classification function (rule) takes features as input and defines how face areas can be discriminated from their backgrounds. This function can be defined explicitly or implicitly. For example, it can be heuristically defined as a simple threshold function, or it can be defined based on statistical properties derived from a trained neural network.

3. *Search strategy*: A classification function decides whether a basic classification unit is a face area or not. Since an input image usually consists of a large number of basic classification units, a search strategy has to be defined to traverse these units to locate all potential face areas.

Since the classification function is the most important constituent in a face detector, we classify the current face detectors into two categories based on the classification function adopted: (1) *heuristics-based face detectors*, where the classification functions are explicitly defined to incorporate direct knowledge of face characteristics. This knowledge may concern face geometry, skin color and edges occurring in face areas; (2) *advanced learning-based detectors*, which implicitly derive separating boundaries between face and non-face areas by using (statistical) pattern-classification techniques, such as neural networks and Support Vector Machines (SVM). It should be noted that there exists an overlap between these categories. For example, some basic heuristic knowledge about face geometry can be embedded into the topology design of a neural network.
2.3 Heuristics-based detectors

Heuristics-based detectors define classification functions (rules) by using empirical knowledge about face characteristics. For example, face areas have mostly a skin-tone, so that skin color can be used to separate faces from their backgrounds. Another widely-used rule is based on facial feature geometry. For example, a face is always comprised of two symmetric eyes, a nose in the middle and a mouth underneath. Accordingly, a face area can be verified by detecting the existence of these features. Other rules concerning face detection include prominent intensity gradients around facial feature areas, etc.

Generally speaking, heuristics-based detectors are simple, easy to implement and usually do not require much computation cost. However, one major problem with these approaches is the difficulty of translating empirical knowledge into well-defined classification rules [80]. In the following, we review two well-known techniques that mainly make use of heuristic knowledge about face appearances: color-based face detection and template-based face detection.

2.3.1 Color-based detectors

In color-based detectors, an image pixel is regarded as the basic classification unit, which is represented by its associated color components in certain color spaces. Commonly-used color spaces are normalized $RGB$, $YCBCr$ and $HSV$ [56]. It is assumed that the skin-color across various ethnic groups forms a condensed cluster in these color spaces. Therefore, the values of the color components can be used as basic measures for discriminating face areas from their backgrounds.

A typical classification function used for color-based detectors is based on the Bayes decision rule (see Appendix A). More specifically, assume that vector $x$ contains the color components of a pixel, $x$ is classified as a skin pixel if

$$\frac{p(x|\text{skin})}{p(x|\text{non-skin})} > \beta,$$  \hspace{1cm} (2.1)

where $p(x|\text{skin})$ and $p(x|\text{non-skin})$ are class-conditional probability density functions and $\beta$ is a constant.

The class-conditional probability densities can be estimated by various techniques, such as histograms, parametric modeling based on uni-modal Gaussians or mixture of Gaussians [56]. Suppose a uni-modal Gaussian is used to characterize the skin-color distribution, the class-conditional pdf becomes

$$p(x|\text{skin}) = (2\pi)^{-d/2}|C|^{-1/2}e^{-\frac{1}{2}(x-m)\T C^{-1}(x-m)},$$  \hspace{1cm} (2.2)

where $C$ denotes the covariance matrix, $m$ is the mean color vector and $d$ is the dimension of the color feature vector. Furthermore, it is assumed that the
2.3. Heuristics-based detectors

Figure 2.4: Skin-color classification by uni-modal Gaussian.

non-skin color distribution \( p(x|\text{non-skin}) \) is uniform. Using this assumption, the following formula can be derived based on Eqn. (2.1):

\[
(x - m)^T C^{-1} (x - m) < \alpha,
\]

where \( \alpha \) is a constant. This formula actually represents an ellipse in 2D or an ellipsoid in 3D. In Fig. 2.4(a), we show an elliptic modeling of the skin-color distribution in the \( C_b - C_r \) space. Since \( C \) and \( m \) can be estimated from a set of skin-color samples (see Fig. 2.4(a)), and \( \alpha \) can be empirically chosen, Eqn. (2.3) can be used as an explicit function to classify a pixel to face/non-face area.

By scanning the whole input image and applying the above classification function to each pixel, a skin map can be generated (as shown in Fig. 2.4(b)), where all potential face areas are marked. In order to extract faces from these areas, some postprocessing steps are usually required, such as pixel clustering (grouping) and shape verification [86][28].

Color-based face detectors use simple classification functions and only require a single scan of the image for locating potential face areas. Consequently, they achieve a high detection efficiency. However, for images containing skin-color-like background regions, noisy detection results can frequently occur, leading to an increased false acceptance rate. Furthermore, color-based face detectors are relatively sensitive to the change of lighting conditions or imaging devices. Therefore, they are mostly suited for well-constrained situations.

2.3.2 Template-based detectors

Template-based detectors employ a pre-defined face template (model) to match possible faces from an input image. A representative approach is the edge-orientation template proposed in [19]. It is assumed in [19] that human faces
are characterized by prominent edges around facial feature areas, such as eyes and mouth, therefore it is attractive to exploit edge features for discriminating face/non-face areas.

The basic classification unit used in [19] is a $40 \times 40$ image window, which is represented by two edge maps of the same size, an Edge Orientation Map (EOM) and an Edge Intensity Map (EIM). Given an image window with raw intensity $I(x, y)$, where $(x, y)$ is the coordinate within the image window, the EOM and EIM of the image window can be derived by the following steps. For each $(x, y)$, we convolve $I(x, y)$ with $3 \times 3$ horizontal and $3 \times 3$ vertical Sobel edge filters, resulting in $S_x(x, y)$ and $S_y(x, y)$, respectively. Based on these convolution results, each element of the EOM ($E_o(x, y)$) is calculated by

$$E_o(x, y) = \arctan \frac{S_y(x, y)}{S_x(x, y)} + \frac{\pi}{2}, \quad (2.4)$$

which is further normalized into the range of $[0, \pi]$. Additionally, each element of EIM ($E_s(x, y)$) is generated by

$$E_s(x, y) = \sqrt{S_y^2(x, y) + S_x^2(x, y)}. \quad (2.5)$$

The edge features are used as the basis for the representation due to its relative insensitivity to illumination changes. Therefore it offers more reliable performance than using image intensities. In the following, we illustrate the classification function adopted by [19] to determine whether a given image window with edge features $E_o(x, y)$ and $E_s(x, y)$ is a face area or not.

More specifically, the classification function of an image window is based on the distance between the edge maps of the window and pre-defined templates. The templates are generated by averaging the EOMs and EIMs of a set of face images of size $40 \times 40$, denoted as $\bar{E}_o(x, y)$ and $\bar{E}_s(x, y)$. Given a basic classification unit with edge maps $E_o(x, y)$ and $E_s(x, y)$, the classification criterion is defined as

$$\sum_x \sum_y d_e(x, y) < \gamma, \quad (2.6)$$

where $\gamma$ is an empirically chosen threshold. Distance $d_e(x, y)$ is defined by

$$d_e(x, y) = \begin{cases} \sin(|E_o(x, y) - \bar{E}_o(x, y)|), & \text{if } E_s(x, y) > T \text{ and } \bar{E}_s(x, y) > T; \\ 1, & \text{otherwise}, \end{cases} \quad (2.7)$$

where $T$ is a threshold. Here the difference of the edge orientation is used, because it is less sensitive to variances in image contrast than the edge intensity.

In the above, we have defined the classification function on a basic classification unit of fixed size ($40 \times 40$). Since we have no \textit{a-priori} knowledge
2.3. Heuristics-based detectors

Algorithm Multi-resolution search for face detection

**Input:** Image \( P \) and face template (model) \( T \) of size \( M \times M \).

**Output:** A list \( L \) of detected faces.

1. Let image \( F = P \).
2. while size_of(\( F \)) > size_of(\( T \))
3. for each location \( (x, y) \) in image \( F \)
4. Let \( W \) be an image window of size \( M \times M \) from \( F \) centered at \( (x, y) \).
5. Classify \( W \) as face/non-face area, e.g. by comparing \( W \) with template \( T \).
6. if \( W \) is a face
7. add \( W \) to \( L \).
8. Downsample image \( F \) to \( F' \) by a scaling factor of \( k \).
9. Let \( F = F' \).

| Table 2.1: Multi-resolution search algorithm for face detection. |

about the sizes of the faces in the input image, in order to select face candidates of various sizes, we need to scan the image at multiple scales. In this way, potential faces of any size can be matched to the face template at (at least) one of the image scales. Table 2.1 portrays a general search strategy for the template-based detectors. In this strategy, exhaustive search has to be performed at multiple image resolutions and positions, such that all possible face candidates can be located.

Due to the simplicity of the matching criterion defined in Eqn. (2.6), the edge-template-based face detection can be performed efficiently. One problem with this approach is the relatively high false acceptance rate when a reasonably good detection rate is required.

It is worth pointing out that in addition to the template-based detectors, the multi-resolution search strategy as presented in Table 2.1 has also been widely applied in many other face detectors. The computation complexity involved during the search poses a key problem when a real-time face detector is concerned. This issue is revisited in Section 2.5.

In this section, we have reviewed two types of face detectors based on heuristic knowledge of face appearances. In literature, many other heuristics-based detectors are also proposed. For example, in [35], the horizontal/vertical projections of image intensities are used to discriminate a face area from its background. In general, these face detectors have difficulties in separating faces from complex backgrounds. Furthermore, the detection may not be robust against changes in the imaging environment.
2.4 Learning-based detectors

A statistical or learning-based detector consists of three basic elements as presented in previous sections. The basic classification unit in these detectors is an image window of \( M \times M \) pixels, which is usually represented by an \( M^2 \) dimensional vector \( x \) containing the raster-scanned intensities of the image window. The classification function is typically characterized by a statistical model or a learning machine derived from training samples. Given an input image, a multi-resolution search is performed, which is similar to the procedure as described in Table 2.1. During the search, for each candidate window, the classification function is applied to \( x \) to decide whether the candidate window is a face area or not.

Since most statistics or learning-based detectors adopt similar features and search strategies, in the following discussion, we focus primarily on the design of the classification function.

2.4.1 Basic strategies for designing a classification function

A classification function \( f \) assigns a given feature vector \( x \) to face or non-face classes. The theoretically optimal classification based on Bayes decision rule (see Appendix A) would lead to the minimization of the classification-error probability. The Bayes decision rule defines the following classification:

\[
f(x) = \begin{cases} 
  \text{face,} & \text{if } p(x|\text{face})P(\text{face}) > p(x|\text{non-face})P(\text{non-face}), \\
  \text{non-face,} & \text{if } p(x|\text{face})P(\text{face}) < p(x|\text{non-face})P(\text{non-face}),
\end{cases}
\]  

(2.8)

where \( p(x|\text{face}) \) and \( p(x|\text{non-face}) \) are class-conditional probability density functions of feature vector \( x \) for face and non-face.

In order to apply Eqn. (2.8) to face detection, the probability densities \( p(x|\text{face}) \) and \( p(x|\text{non-face}) \) have to be reliably estimated. However,
in practice, the feature vector $x$ usually has very high dimensionality (e.g. \( \text{dim}(x) = 625 \) for a $25 \times 25$ image window), and it would be impractical to directly estimate the probability densities in such a high-dimensional feature space. Furthermore, it is not known whether there exist natural parameterized forms of $p(x|\text{face})$ and $p(x|\text{non-face})$. In the current literature, these probability densities are usually estimated based on certain empirical assumptions and simplifications. For example, it is assumed in [47] that $p(x|\text{face})$ forms a multivariate Gaussian, which is approximated by an efficient subspace density estimation method.

Furthermore, it should be noted that the Bayes classification function (Eqn. (2.8)) actually partitions the feature space into two disjoint decision regions $R_{\text{face}}$ and $R_{\text{non-face}}$, which are separated by a so-called decision (hyper)surface (see Fig. 2.5). This surface is usually represented by a discriminant function. It has also been customary in literature to directly seek a decision surface between face and non-face classes, without explicit probability-density estimation as required in Eqn. (2.8). These approaches usually optimize alternative costs other than those used by the Bayes approach and lead to different classification functions. Examples of face detectors using this approach include neural-network-based detectors (Section 2.4.3) and SVM-based detectors (Section 2.4.4). In the following, we briefly review several representative learning-based detectors.

### 2.4.2 Distribution-based detectors

In [47], it is proposed to directly estimate density $p(x|\text{face})$, followed by a standard Bayes classification (Eqn. (2.8)). In this approach, it is first assumed that feature vector $x$ forms a uni-modal Gaussian distribution, specified by

$$p(x|\text{face}) = (2\pi)^{-d/2} |C|^{-1/2} e^{-\frac{1}{2} (x - \bar{x})^T C^{-1} (x - \bar{x})},$$  \hspace{1cm} (2.9)$$

where $C$ is the covariance estimation of feature vector $x$, $d$ is the dimension of $x$ (in this case, $d = M^2$), and $\bar{x}$ is the mean estimation of $x$. Due to
the high dimensionality of $x$, a reliable estimation from Eqn. (2.9) requires a large amount of feature samples, which usually cannot be satisfied in practice. Furthermore, the computation cost involved is formidable.

In [47], an efficient density-estimation method is proposed. In this approach, the feature space is divided into two complementary subspaces: the principal subspace $F$, which is spanned by the first $k$ principal components derived from the PCA (Principal Component Analysis) decomposition of the original feature space (see Appendix B), and its orthogonal complement $\bar{F}$, which contains the residual of the expansion. Correspondingly, each feature vector $x$ is decomposed into two components in subspaces $F$ and $\bar{F}$, referred to as the distance in feature space (DIFS) and the distance from the feature space (DFFS), respectively (see Fig. 2.6).

More specifically, we first define $m(x) = (x - \bar{x})^T C^{-1} (x - \bar{x})$, which is the most relevant part in Eqn. (2.9) as far as Bayes decision rule is concerned. According to the subspace decomposition mentioned above, we have:

$$m(x) = (x - \bar{x})^T C^{-1} (x - \bar{x})$$

$$= (x - \bar{x})^T (\Phi \Delta^{-1} \Phi^T) (x - \bar{x}) \quad \text{(Eigenvalue decomposition)}$$

$$= y^T \Delta^{-1} y \quad \text{(Eigenvalue)}$$

$$= \sum_{i=1}^{d} \frac{y_i^2}{\lambda_i} \approx \sum_{i=1}^{k} \frac{y_i^2}{\lambda_i} + r \left( \sum_{i=k+1}^{d} y_i^2 \right) \quad \text{(Eigenvalue decomposition)}$$

In the above, $\Phi$ contains the Eigenvectors derived from the Eigenvalue decomposition of $C$, and $\Delta$ is a diagonal matrix containing all Eigenvalues. The first term in Eqn. (2.13) can be calculated by PCA, while the second term can be easily estimated by the residual reconstruction error [47]. More detailed explanation of the calculation can be found in [47]. Eqn. (2.13) can be used as the basis for the Bayes classification. Sometimes, the second term in Eqn. (2.13) can be discarded and only the principal components in feature space $F$ are used [70]. However, this gives a less accurate result than the full estimation in Eqn. (2.13).

In addition to the uni-modal Gaussian assumption employed in the above modeling, a mixture of Gaussians can be also adopted as the basic assumption for the estimation of $p(x|\text{face})$ [47]. In this case, similar line of reasoning as above can be followed to derive the estimation.

Another distribution-based face detector is proposed in [63] using the naive Bayes classification. In this approach, the face region represented by $x$ is divided into a number of subregions which are assumed to be independent from each other. The global density $p(x|\text{face})$ is then represented by the joint statistics of each subregion, which are described by its local appearance and position. These can be then estimated by using histograms.
A common characteristic of the distribution-based detectors is that the class-conditional probability densities have to be explicitly estimated, from which the discriminant function can be derived. In the following, we review another class of approaches that directly derives discrimination functions not imposed by the probability densities of face and non-face.

2.4.3 Neural-network-based detectors

Neural networks have been demonstrated to be effective tools for many pattern-classification problems, such as character recognition, etc. Various architectures and training methodologies have been proposed in literature to address different problems. Generally speaking, a neural network contains a number of interconnected units (neurons), as motivated by the human-brain structure. The relationship (connection properties) of these neurons are learned from a large set of training samples. The trained network can be seen as a realization of the discriminant function to separate face and non-face feature vectors. In Appendix C, we give a brief review of a commonly-used neural-network architecture: the multi-layer feedforward neural network, which has been applied in a number of face-detection techniques [66][60]. In the following, we briefly review one of the most representative solutions proposed by Rowley et al. [60].

In [60], a multi-layer feedforward neural network is used to realize a non-linear decision function $N$ (see Fig. 2.7). The network receives a feature vector $\mathbf{x}$ corresponding to a $20 \times 20$ image window, and outputs a score ranging from $-1$ to $1$. By choosing a proper threshold $T (-1 \leq T \leq 1)$, $\mathbf{x}$ is classified as face if $N(\mathbf{x}) > T$ and non-face otherwise.

As shown in Fig. 2.7, the hidden neurons of the network are partially connected to local receptive fields from the input layer, allowing the network
to capture local feature structures important for face detection. Three groups of hidden neurons are adopted: 4 neurons look at $10 \times 10$ pixel subregions, 16 neurons look at $5 \times 5$ pixel subregions and 6 neurons look at $20 \times 5$ pixel subregions (horizontal stripes). The first two groups of hidden neurons are supposed to capture facial features such as individual eyes, the nose and the corners of the mouth in different resolutions. The third group of neurons (the horizontal stripes) are designed to capture mouths and pairs of eyes. Experiments from [60] show that better detection accuracy can be achieved by incorporating these heuristics to network structures. Furthermore, in order to improve the reliability of the final decision, multiple (2 to 3) networks of the same architecture can be used and the final decision is based on an arbitration of all these networks.

The network as depicted in Fig. 2.7 is trained by the backpropagation algorithm (Table C.1) using a large set of face and non-face training samples. Since it is impossible to cover all kinds of non-face patterns in a limited training set, a bootstrapping procedure is adopted. In this procedure, an initial set of randomly selected non-face patterns are used to train the network for the first iteration. After that, randomly chosen scenery images which contain no faces are fed to the network and those patterns that are misclassified by the network are dynamically added to the non-face training set. In this way, more difficult samples that lie close to the decision surfaces are reinforced during the training. It has been demonstrated in [60] that the bootstrapping training effectively limits the size of the non-face training set and improves the system performance.

It is reported in [60] that the above-mentioned face detector has achieved a good detection accuracy even with the difficult MIT+CMU test sets. However, one primary drawback of the technique is the computation complexity. A reported example states that 383 seconds are required to process a $320 \times 240$ pixel image. By incorporating additional filtering and heuristics, the computation time can be reduced to 7.2 seconds per image. However, this is still insufficient for a (near) real-time application.

In addition to the multi-layer backpropagation network, other network structures have also been proposed in literature for face detection, such as a convolutional network detector in [20] and a SNoW (Sparse Network of Winsnows) detector in [81]. It has been reported that these detectors also achieve high performance with respect to detection accuracy.

### 2.4.4 SVM-based detectors

Support Vector Machine (SVM) is another well-known statistical learning technique for generating complex decision boundaries between face and non-face patterns. Appendix D presents a brief review of some basic theory on SVM classification.
SVM was first applied to face detection in [49]. In this approach, a $19 \times 19$ window is used as the basic classification unit, and a large set of such windows containing face and non-face patterns are collected to form a training set. In order to capture the complex nonlinear decision boundaries between face and non-face classes, nonlinear kernel mapping (e.g. 2nd-degree polynomials) is adopted in the SVM formulation (see Appendix D).

One specific problem to apply SVM classification to face detection is the large datasets involved, which usually requires a lot of computation resources (CPU and memory) to obtain the solution to the optimization problem in SVM. In [49], an efficient training method is proposed to apply SVM to large-scale problems. In Fig. 2.8, a conceptual nonlinear decision surface found by SVM is depicted, together with some example support vectors lying close to the decision surface. Some more recent results using SVM for face detection can be found in [25]. Compared to neural-network-based detectors, the training and classification of an SVM-based face detector demand more computation resources.

The statistics or learning-based face detectors usually achieve more accurate and robust detection performance as compared to heuristics or knowledge-based face detectors. However, they are usually far more computationally expensive. To further reduce the computation cost, an emerging interest in literature is to study structured face detectors employing multiple subdetectors. In the following, we examine a cascaded detector structure, which is particularly effective in improving the detection efficiency without having to sacrifice the detection accuracy.
2.5 Cascaded detector architecture

From the review of the face-detection techniques in the previous sections, we can see that it is generally difficult for a single detector to achieve both high detection accuracy and efficiency. Therefore, it is useful to exploit multiple detectors to maximize overall performance in both detection accuracy and efficiency. An attractive multiple-detector architecture is a cascade of face detectors with incremental computation complexity and detection capability. In such a cascade, each detector can progressively filter out non-face candidates, thereby improving the overall detection performance in a step-wise manner. Informally speaking, the following guidelines are applied for building a cascaded detector.

1. The first face detector(s) in the cascade should be simple and fast, while they are able to prune large areas of non-face regions in the backgrounds.

2. The final face detector(s) in the cascade may involve higher computation complexity, while they should have better discrimination capability, such that the remaining non-face candidates can be correctly identified.

Special considerations need to be paid as how to design each subdetector and effectively combine them to form an effective cascade. The overall performance of a cascade depends on a number of factors such as the detection accuracy and complexity of each detector and their correlations. In the following, we look at several design issues in more detail.

Suppose we need to perform a multi-resolution search for all possible faces from an input image, which consists of a total of $W$ candidate windows. Correspondingly, the classification has to be performed $W$ times for complete verification. As shown in Fig. 2.9, we can use a cascade of $N$ detectors $\{D_1, D_2, \ldots, D_N\}$, where each detector can reject a fraction of $P_i$ of all candidate windows as non-faces, while allowing (almost) all true face candidates to pass to the next detector.
Detection accuracy of a detector cascade. Since the first detectors in the cascade aim at filtering out as many as possible non-face candidates while retaining (almost) all face candidates, they should have a near 100% face detection rate and a reasonably low false acceptance rate. The overall face detection rate of the detector cascade reflects the pass rate of faces through the face cascade, which is largely determined by that of the final subdetector. The false acceptance rate reflects the fraction of non-face candidates that is incorrectly identified as faces, which is a joint contribution of all subdetectors.

Computation complexity of a detector cascade. We further assume that the computation cost spent on classifying one image window by detector \( D_i \) is \( c_i \), where \( c_i \) is expressed in some cost-related parameter\(^2\). In order to illustrate the efficiency gain that can be obtained by using the cascade structure, in the following, we take a simple detector cascade as an example, which consists of two subdetectors. The cases of cascades with more subdetectors can be derived in a similar way.

Based on the above assumptions, the total computation cost \( C \) for a cascaded detector is

\[
C = (c_1 + (1 - P_1)c_2)W. \quad (2.14)
\]

In comparison, the total computation cost \( C_2 \) when using the second (final) detector alone is

\[
C_2 = c_2W, \quad (2.15)
\]

For an effective detector cascade, the subdetectors are normally arranged in such a way that:

\[
P_1 > 1 - P_1, \quad \text{and} \quad c_1 < c_2. \quad (2.16)
\]

Suppose that \( P_1 = 90\% \) and \( c_2 = 5c_1 \), we can easily derive that \( C = 0.3C_2 \), indicating up to 70\% efficiency gain by using the detector cascade. On the other hand, suppose we have \( P_1 = 30\% \) and \( c_2 = 2c_1 \), indicating that the first subdetector has a larger false acceptance rate and is also less efficient. In this case, we can derive that \( C = 1.2C_2 \), which means that the overall processing of the cascade is 1.2 times slower than using the second subdetector alone. This indicates a design failure and should be avoided in practice. In conclusion, during the construction of a detector cascade, it is of crucial importance that each subdetector is designed with appropriate \( P_i \) and \( c_i \), such that the overall cascade can yield an improved performance.

Examples of cascaded face detection include [75], [38], [26] and [43]. In [75], first a so-called Adaboost algorithm\(^3\) is used to select a set of Haar-like feature classifiers to form a single detector. In order to improve the overall detection

\(^2\)In practice, \( c_i \) can be measured based on the execution time.

\(^3\)The Adaboost algorithm is revisited in Chapter 3.
speed, a set of such detectors with different characteristics are cascaded into a chain. Detectors consisting of smaller numbers of feature classifiers are relatively fast, and they are used at the first stages in the cascade to filter out regions that most likely do not contain any faces.

The Viola-Jones face detector in [75] achieves real-time processing speed. The detection accuracy, however, is sacrificed to a certain extent (see comparison of several well-known detectors in Table 3.8 in Chapter 3). Furthermore, the feature selection (training) stage in [75] is quite time-consuming in practice. It is reported that several weeks are needed to completely train a cascaded detector.

Following [75], a number of variants of the Viola-Jones detector have also been proposed in literature, such as the detector with extended Haar features [40] and FloatBoost-based detector [38]. In addition, in [43], hierarchical Support Vector Machines are proposed, which use a combination of linear SVMs to fastly exclude most non-faces in images, followed by a nonlinear SVM to further verify possible face candidates.

In the following, we present our early work to contribute to cascaded face detection [88]. The proposed detector makes use of various features to form a heterogeneous cascade. This is used as an illustrative case study to demonstrate the performance gain by using a cascaded structure.

### 2.6 Heterogeneous cascaded detector: a case study

The original motivation in our earlier work [88] is to reduce the computation cost involved in an advanced learning-based face detector, e.g. neural-network-based detector [60]. To this end, we propose a heterogeneous cascaded face detector, which makes use of various feature representations. More specifically, the cascade consists of a skin-color-based detector, a face-geometry-based detector and a neural-network-based detector. The detector cascade offers the following advantages.

1. Various images features are used including color, geometry and image luminance. These features complement each other and capture different properties of face images.
2.6. Heterogeneous cascaded detector: a case study

2.6.1 Skin-color detector

Color has been proven to be an effective feature for coarsely locating potential facial regions (see Section 2.3.1). The skin-color subdetector that we adopted, is based on a uni-modal Gaussian modeling of skin-color distribution in the $C_b - C_r$ color space (see Fig. 2.4). By applying the skin-color detector to an input image, we can obtain a skin map, which is further smoothened as shown in Fig. 2.11. Each connected region in the skin map is then marked as a skin blob and passed on to the next detector.

Note that the color-pruning criterion used in this stage is very relaxed, resulting in a higher false acceptance rate than a more intricate color-based detector. However, for an initial detector in the complete detector cascade, we intend to put more emphasis on speed than accuracy and the overall detection accuracy is reinforced by the succeeding detectors.

2.6.2 Feature-geometry-based detector

In order to be able to select potential face candidates from the skin blobs acquired from the previous detector, a multi-resolution search similar to that given in Table 2.1 is applied within each skin blob to enumerate all possible candidate windows of $20 \times 20$ pixels. A simple geometry-based classification rule is applied to each candidate window, which is briefly summarized in Table 2.2.

In Table 2.2, the algorithm uses a set of heuristic rejection rules to verify the existence of prominent facial features such as eyes and mouth. Firstly, the vertical profile of the candidate window is generated, and each point in
Algorithm Geometry-based face classification
1. Generate the vertical profile of the candidate window (Fig. 2.12), and select local minima as candidate positions of eyes and mouth.
2. if no proper minima are found, return non-face
3. for each candidate location
4. Apply a sliding window to search horizontally for most probable eye-pair (or mouth) based on the average intensity of the covered region.
5. if no possible eyes (mouth) are found return non-face
6. if no feature group (eyes + mouth) exists that forms an approximate equilateral triangle, return non-face
7. else return face

Table 2.2: Geometry-based face classification.

the profile corresponds to the average luminance value of pixels from the same horizontal line. The first rule states that local minima should exist in the upper and lower parts of the candidate window, corresponding to eyes and mouth, respectively.

If proper minima exist in the vertical profile, we search horizontally for the possible eyes and mouth by examining the average intensity values within a pre-defined template window (see Fig. 2.12). We assume that eyes and mouth regions have relatively lower intensity distributions than other parts of the face area. Therefore, if two sufficiently dark regions can be found from the left and right parts along a candidate line, then these regions are stored in an eye-pair candidate set. Similarly we can store possible mouth locations in a mouth candidate set. If both candidate sets are empty, the candidate window is safely discarded as a non-face.

The final rule concerns the verification of an approximate equilateral triangle relationship between an eye-pair and a mouth from the two candidate sets. If no such candidate group (eye-pair + mouth) can be found to satisfy the triangle relationship, then the candidate window is discarded as a non-face.

The feature-geometry-based detector progressively rejects a large portion of non-face skin-areas by applying a set of simple heuristic rules. It can be also seen as a subcascade employing a set of rule-based detectors. Only the candidates satisfying all of these rules are retained and passed on to the final detector.
2.6. Heterogeneous cascaded detector: a case study

2.6.3 Neural-network-based detector

The neural-network-based subdetector is quite similar to Rowley’s approach in [60]. Since the previous detectors in the cascade already filter out a large portion of the background regions, only a very small number of ambiguous candidates are passed on to the neural-network detector for the final verification. This significantly reduces the potential high computation cost involved for the final detector.

2.6.4 Experimental results and discussions

To evaluate the performance of the proposed detector cascade, we use a test set composed of color images downloaded from the Internet (most of them from the Yahoo News photo gallery), 100 randomly selected sample images from the FERET face database [55][54], and a number of self-recorded video sequences capturing both indoor and outdoor environments. We obtain an average of 92% face detection rate (slightly lower than using the neural-network detector alone). On the average, the proposed detector can process a $320 \times 240$ pixel image in about 0.3-0.4 seconds. This is a significant improvement as compared to using the final neural-network detector alone, which usually takes over one minute on the same computing platform.

The heterogeneous face detector described in this section significantly reduces the computation cost of a single neural-network detector. However, for some test images with relatively poor lighting conditions, the detection performance is restricted by the first color-based detector, which is more sensitive to the lighting conditions. In the next chapter, we propose a homogeneous cascaded face detector employing a set of neural-network ensembles. Our experiments show that the approach is more robust for environment changes such as illumination while still being computationally efficient.
2.7 Summary and conclusions

In this chapter, we have reviewed some major techniques used in face detection. We have found that a key problem in the face-detector design is the conflict between the detection accuracy (robustness) and the operation efficiency. Heuristics or knowledge-based face detectors, such as color-based and template-based detectors, use direct knowledge about faces and often give fast performance, but they are usually less robust with respect to large face variances and background interferences. On the contrary, statistical or learning-based face detectors, like the neural-network and SVM-based detectors, make use of powerful pattern-classification algorithms and provide better performance in discriminating face and non-face patterns. However, these learning-based algorithms often involve high processing complexity, which can be too costly for applications in embedded systems.

The cascaded detector structure as presented in Section 2.5 provides a way to reduce the total processing cost while still maintaining the high detection accuracy of advanced algorithms. In particular, we have presented a heterogeneous face detector as a case study to demonstrate the effectiveness of such an algorithm architecture. The detector first employs simple and fast detectors (e.g., color-based detector and geometry-based detector) to prune large areas of non-face backgrounds, which significantly reduces the computation burden for the final neural-network-based detector. The experimental results show that this strategy effectively improves the processing speed from minutes (by neural-network detector) to the sub-second level.

However, the proposed detector can only work with color images, and the color-based detector is not robust under relatively poor lighting conditions. Such conditions inevitably result in a lower face-detection rate. In the next chapter, we propose a high-performance face detector built as a cascade of neural-network ensembles. The use of a neural-network ensemble effectively improves the performance of a single network, leading to an overall enhanced accuracy. By cascading a set of different neural-network ensembles, we can optimize the overall detection performance of the detector in both accuracy and efficiency at the same time.
3.1 Introduction

In this chapter, we propose a neural-network-based face detector by using two architectural principles: *ensembles* and *cascades*, in order to achieve a fast, yet accurate performance [97]. First, we employ an *ensemble* (or *committee*) of neural-network classifiers to jointly perform the detection task. The neural-network classifiers are constructed such that they complement each other in recognizing face patterns. The experimental results show that the detection accuracy of our proposed ensemble is significantly improved compared to traditional neural-network-based detectors. Furthermore, in order to reduce the total computation cost for face detection, we cascade a series of neural-network ensembles with increasing complexity. In this way, simpler and more efficient ensembles are used at earlier stages in the cascade, which are able to reject a majority of non-face patterns from image backgrounds. Consequently, the detector cascade significantly improves the overall detection efficiency while maintaining the detection accuracy.

The major contribution in this chapter is a new training paradigm to form an ensemble of neural networks, which are subsequently used as the building blocks of the cascaded detector. The training strategy is very effective as compared to existing techniques and significantly improves the face-detection
accuracy. We also adapt this ensemble structure into the cascaded framework as introduced in the previous chapter by proposing a set of ensembles with scalable complexity. This yields a significant gain in efficiency with (near) real-time detection speed.

The remainder of the chapter is organized as follows. In Section 3.2, we present an overview of the cascaded detector architecture employing a set of neural-network ensembles. Section 3.3 explains the construction of a neural-network ensemble, which is used as the basic element in the detector cascade. In Section 3.4, a cascaded detector is formulated consisting of multiple neural-network ensembles. In Section 3.5, we apply the cascaded detector to face detection in real-world images and video sequences. A number of implementation issues are discussed and the final performance is analyzed. Section 3.6 gives a short summary of this chapter.

#### 3.2 Overview of the detector architecture

The detector proposed in this chapter uses an image window of $24 \times 24$ pixels as the basic classification unit, which is represented by the raster-scanned intensity vector $\mathbf{x}$. We aim at deriving a classification function $f(\mathbf{x})$ for the discrimination between face and non-face patterns. Once $f(\mathbf{x})$ is obtained, a multi-resolution search of the image can be performed to locate all possible faces in an image.

The classification function $f(\mathbf{x})$ is derived from a cascaded neural-network structure as depicted in Fig. 3.1. This structure contains two layers.
3.3 Neural-network ensemble

**Ensemble layer.** We construct an ensemble of neural networks consisting of $N$ base networks $h_i$ and a decision network. The base networks are trained sequentially, each of which learns to discriminate patterns that fail to be classified by its preceding networks. In order to classify a given feature vector $x$, we feed $x$ simultaneously to the base networks, resulting in $N$ outputs $h_i(x)$ ($1 \leq i \leq N$). These outputs are then fed to a decision neural network to make the final combined decision. The aim of using the ensemble layer is to optimize the detection accuracy of neural networks with limited cost.

**Cascade layer.** In order to construct a cascaded face detector for improved efficiency, we train a number of neural-network ensembles with varying base-network topology. Initial ensembles in the cascade adopt base networks that only receive a coarse representation of feature $x$. They usually have fewer nodes and connections, leading to simpler decision boundaries. However, since these networks can be executed with extreme efficiency, a large portion of candidate windows that contain no faces can be quickly rejected during the multi-resolution image search. Subsequent ensembles adopt relatively complex base networks, which have the capability of capturing more detailed pattern information and constructing more precise decision boundaries. These ensembles are only invoked for difficult cases that fail to be rejected by its previous ensembles in the cascade. By optimizing the cascade structure, the overall computation cost involved can be significantly reduced.

In the next section, we illustrate the ensemble layer in more detail, followed by a supplementary explanation on the cascade design in Section 3.4.

### 3.3 Neural-network ensemble

#### 3.3.1 Basic principles of ensemble design

For complex real-world classification problems such as face detection, using a single classifier may not be sufficient to capture the complex decision surfaces between the face and non-face patterns. In the machine-learning community, a class of promising techniques is being studied, which makes use of a combination of classifiers to improve the classification accuracy, generally denoted as classifier ensemble or pooled classifiers [17]. Each constituting classifier in an ensemble is called a component classifier, which usually takes the same general form (e.g. all are neural networks, or all are Bayes classifiers, etc).

An ensemble of classifiers are particularly useful when its component classifiers are diverse. Two classifiers are diverse if they implement different classification functions, thereby making potentially different decisions on a given input. The diversity is a necessary condition for an ensemble of classifiers to
be more accurate than any of its component classifiers [16]. This can be easily seen by the following example. Suppose we have a set of component classifiers with (almost) the same behavior, then if one of them makes a classification error, the remaining classifiers most likely make the same error. In this case, it is not useful to employ multiple classifiers. On the contrary, if the outputs from the component classifiers are uncorrelated, we can use a certain strategy to combine these individual outputs. For example, we can use majority voting or choose the output from the most reliable component classifier for a given input. In this way, a classifier ensemble is expected to give more accurate classification results than any single component classifier.

There are various ways to construct diverse classifiers. Here we briefly introduce two commonly adopted methods: injecting randomness to the training procedure and manipulating the training data [16].

A. Injecting randomness to the training procedure.
One of the simplest ways to generate diverse classifiers is to make use of the stochastic nature of the classification techniques, such as neural networks. With this respect, we recall the backpropagation training algorithm from Appendix C (Table C.1), where the initial weights of the network are set randomly. If the network is trained with the same training data but with different initial weights, the resulting decision surfaces can be quite different [17]. This strategy has been applied in Rowley’s approach [60] in face detection as mentioned in Chapter 2, where three networks with different initial weights are trained and the final output is based on the majority voting of these networks.

B. Manipulating the training data.
Another method for generating ensembles is to manipulate the training data. Each component classifier is trained with a different subset of the training data. According to [17][16], this method works especially well for unstable learning algorithms, where the classification results undergo large changes in response to small changes in the training data used. Two commonly adopted algorithms for manipulating the training data are bagging and boosting.

1. **Bagging (Bootstrap aggregation):** Bagging trains each component classifier in the ensemble with a subset of the training data randomly drawn from the complete training set. The final decision is based on an averaging (or voting) of the outputs from all component classifiers. In this strategy, each component classifier is trained on a differently sampled data set (with possible overlapping samples), leading to diverse classifier behaviors. Note that in bagging, the component classifiers have no correlation with each other and can be trained independently.

2. **Boosting:** Boosting sequentially trains the component classifiers by selecting those parts of the training data that are most informative. During
the training of each component classifier, the samples that are misclassified by the current set of component classifiers, have higher probabilities to be selected. The final output is based on a linearly-weighted combination of the outputs from all component classifiers. The Viola-Jones face detector [75] is an example of how the boosting is applied for selecting a subset of Haar-feature classifiers (component classifiers).

In this chapter, we use an ensemble of neural network classifiers for face detection. Each component neural network in the ensemble is randomly initialized with different weights. More important is that we manipulate the training data such that each component network is specialized in a different region of the training data space. This conforms to the second principle as mentioned above (classifier diversity). Our proposed ensemble has the following new characteristics that are different from existing approaches in literature.

1. The component neural networks in our proposal are sequentially trained, each of which uses training face samples that are misclassified by its previous networks. Our approach differs from the boosting approach in that the training samples that are already successfully classified by the current network are discarded and not used for the later training. This gives a ‘hard’ partitioning of the training set, where each component neural network characterizes a specific subregion.

2. The final output of the ensemble is determined by a decision neural network, which is trained after the component networks are already constructed. This offers a more flexible combination rule than the voting or linear weighting as used in bagging (boosting).

The experimental evidence from Section 3.3.4 shows that our proposed
approach gives good performance in face detection and outperforms the traditional ensemble techniques.

### 3.3.2 Ensemble architecture

We depict the structure of our new proposed neural-network ensemble in Fig. 3.2. The ensemble consists of two layers: a set of sequentially trained
component neural classifiers \( \{h_k | 1 \leq k \leq N\} \), and a decision neural network \( g \).

**Component neural network**

Each component classifier \( h_k \) is a multi-layer feedforward neural network, which has inputs receiving certain representations of the input feature vector \( \mathbf{x} \) and one output ranging from 0 to 1. The network is trained with a target output of unity indicating a face pattern and zero otherwise. Each network has locally connected neurons, as motivated by [60]. It is pointed out in [60] that by incorporating heuristics of facial feature structures in designing the local connections of the network, the network gives much better performance (and higher efficiency) than a fully connected network.

We present here four novel base-network structures employed in this chapter: FNET-A, FNET-B, FNET-C and FNET-D, which are extensions of [60] by incorporating scalable complexity. These networks are used as the basic elements in the final face-detector cascade. The design philosophy for these networks are partially based on heuristic reasoning. The motivation behind the design is illustrated below:

1. We aim at building a complexity-scalable structure for all these base networks. The networks are constructed with similar structures.

2. The complexity of the network is controlled by the following structural parameters: the input resolution, the number of hidden layers, and the number of hidden units in each layer.

3. When observing Fig. 3.3, FNET-B (FNET-D) enhances FNET-A (FNET-C) by incorporating more hidden units which specifically aim at capturing various facial feature structures. Similarly, FNET-C (FNET-D) enhances FNET-A (FNET-B) by using a higher input resolution and more hidden layers.

In this way, we obtain a set of networks with scalable structures and varying representation properties. In the following, we illustrate each network in more detail.

As shown in Fig. 3.3(a), FNET-A has a relatively simple structure with one hidden layer. The network accepts an \( 8 \times 8 \) grid as its inputs, where each input element is an averaged value of a neighboring \( 3 \times 3 \) block in the original \( 24 \times 24 \) input features. FNET-A has one hidden layer with \( 2 \times 2 \) neurons, each of which looks at a locally neighboring \( 4 \times 4 \) block from the inputs.

FNET-B (Fig. 3.3(a)) shares the same type of inputs as FNET-A, but with extended hidden neurons. In addition to the \( 2 \times 2 \) hidden neurons, additional \( 6 \times 1 \) and \( 2 \times 3 \) neurons are used, each of which looks at a \( 2 \times 8 \) (or \( 4 \times 3 \)) block from the inputs. These additional horizontal and vertical stripes are used to capture corresponding facial features such as eyes, mouths and noses.
Table 3.1: Summary of the component network topology adopted in this chapter.

<table>
<thead>
<tr>
<th>Network structure</th>
<th>Number of hidden layers</th>
<th>Number of weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNET-A</td>
<td>1</td>
<td>73</td>
</tr>
<tr>
<td>FNET-B</td>
<td>1</td>
<td>265</td>
</tr>
<tr>
<td>FNET-C</td>
<td>2</td>
<td>713</td>
</tr>
<tr>
<td>FNET-D</td>
<td>2</td>
<td>2033</td>
</tr>
</tbody>
</table>

The topology of FNET-C is depicted in Fig. 3.3(b), which has two hidden layers with $2 \times 2$ and $8 \times 8$ hidden neurons, respectively. The FNET-C directly receives the $24 \times 24$ input features. In the first hidden layer, each hidden neuron takes inputs from a locally neighboring $3 \times 3$ block of the input layer. In the second hidden layer, each hidden neuron unit takes a locally neighboring $4 \times 4$ block as input from the first hidden layer.

FNET-D (Fig. 3.3(b)) is an enhanced version of both FNET-B and FNET-C, with two hidden layers and additional hidden neurons arranged in horizontal and vertical stripes.

From FNET-A to FNET-D, the complexity of the network is gradually increased by using a finer input representation, adding more layers or adding more hidden units to capture more intricate facial characteristics. Therefore, the networks have an increasing number of connections and consume more computation power. In Table 3.1, we give a summary of all the component network structures we have mentioned so far.

Decision neural network

For the decision network $g$ (see Fig. 3.2), we adopt a fully-connected feedforward neural network, which has one hidden layer with eight hidden units. The number of inputs for $g$ is determined by the number of component classifiers in the network ensemble. The decision network receives the outputs from each component network $h_k$, and outputs a value $y$ ranging from 0 to 1, which indicates the confidence that the input vector represents a face. In other words,

$$y = g(h_1(x), h_2(x), ..., h_N(x)). \quad (3.1)$$

In the following, we present the training paradigm for our proposed network ensemble.

3.3.3 Training algorithms

The ensemble training consists of the following two stages:
1. Sequentially train \( N \) component classifiers \( h_k \) (\( 1 \leq k \leq N \)) with feature sample \( x \) drawn from a training data set \( T \). \( T \) contains a face sample set \( F \) and a non-face sample set \( N \).

2. Train decision neural network \( g \) with samples \( \langle h_1(x), h_2(x), ..., h_N(x) \rangle \), where \( x \in T \).

Let us now present the training algorithm for each stage in more detail.

Training algorithm for component neural networks

<table>
<thead>
<tr>
<th>network</th>
<th>training set</th>
<th>correctly classified samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_1 )</td>
<td>( F_1 = F )</td>
<td>( F_1^f (F_1^f \subset F_1) )</td>
</tr>
<tr>
<td>( h_2 )</td>
<td>( F_2 = F \setminus F_1^f )</td>
<td>( F_2^f (F_2^f \subset F_2) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( h_N )</td>
<td>( F_N = F \setminus \bigcup_{i=1}^{N-1} F_i^f )</td>
<td>( F_N^f (F_N^f \subset F_N) )</td>
</tr>
</tbody>
</table>

Table 3.2: Partitioning of the training set for component networks.

One important characteristic of the component network training is that each network \( h_k \) is trained on a subset \( F_k \) of the complete face set \( F \). \( F_k \) contains only face samples misclassified by the previous \( k-1 \) component classifiers. More specifically, suppose the \((k-1)\)-th component network is trained over sample set \( F_{k-1} \). After the training, the network is able to correctly classify samples \( F_{k-1}^f \) (\( F_{k-1}^f \subset F_{k-1} \)). The next component network (the \( k \)-th network) is then trained over sample set \( F_{k} = F_{k-1} \setminus F_{k-1}^f \). This procedure can be iteratively carried out until all \( N \) component networks are trained. This is also illustrated in Table 3.2.

In this way, each component network is trained over a subset of the total training set and is specialized in a specific region within the face space. This is illustrated in Fig. 3.4. For each \( h_k \), the non-face samples are selected in a bootstrapping manner similar to the approach used by Rowley et al. [60]. According to the bootstrapping strategy, an initial set of randomly chosen non-face samples are used, and during the training new false positives are iteratively added to the current non-face training set. In this way, more difficult non-face samples are reinforced during the training process.

Up to now, we have explained the training-set selection strategy for the component networks. The actual training of each network \( h_k \) is based on the standard backpropagation algorithm as introduced in Appendix C. The network is trained with unity for face samples and zero for non-face samples. During the classification, a threshold \( T_k \) is chosen, such that the classification
function becomes
\[ f_k(x) = \begin{cases} 1, & \text{if } h_k(x) > T_k, \\ 0, & \text{otherwise}. \end{cases} \] (3.2)

By adjusting \( T_k \), the network has different false acceptance rates and false rejection rates, resulting in an ROC curve as introduced at the start of Chapter 2.

In the following, we elaborate on how the combination of neural networks (\( h_1 \) to \( h_N \)) can yield a reduced classification error over the training face set.

First we define the face learning ratio of component network \( h_k \) as
\[ \alpha_k = \frac{|\mathcal{F}_k^f|}{|\mathcal{F}_k|}, \] (3.3)
where \( |\cdot| \) denotes the number of elements in a set. Further we define \( \beta_k \) as the fraction of the face samples successfully classified by \( h_k \) w.r.t. the total training face samples, given by
\[ \beta_k = \frac{|\mathcal{F}_k^f|}{|\mathcal{F}|}. \] (3.4)

We can see that
\[ \beta_k = \frac{|\mathcal{F}_k|}{|\mathcal{F}|} \cdot \alpha_k = (1 - \sum_{i=1}^{k-1} \beta_i) \alpha_k \quad (\text{since } |\mathcal{F}_k| = |\mathcal{F}| - \sum_{i=1}^{k-1} |\mathcal{F}_i^f|) \] (3.5)
\[ = \beta_{k-1} \frac{\alpha_k}{\alpha_{k-1}} (1 - \alpha_{k-1}). \quad (\text{since } |\mathcal{F}_k| - |\mathcal{F}_k^f| = |\mathcal{F}_{k+1}|) \] (3.6)

By recursively applying Eqn. (3.6), we derive the following relation between \( \beta_k \) and \( \alpha_k \):
\[ \beta_k = \alpha_k \times \prod_{i=1}^{k-1} (1 - \alpha_i). \] (3.7)
Algorithm Training algorithm for component neural network

Input: Training face set $F = \{x_i\}$, number of component neural networks $N$, decision threshold $T_k$, an initial non-face set $N$, and a set of downloaded scenery images $S$ containing no faces.

1. Let $k = 1$, $F_1 = F$
2. while $k \leq N$
   3. Let $N_k = N$
   4. for $j = 1$ to Num_Epochs /* Number of training iterations */
      5. Train neural classifier $h^j_k$ on face set $F_k$ and non-face set $N_k$ using the backpropagation algorithm.
      6. Compute the false rejection rate $R^j_f$ and false acceptance rate $R^j_n$.
      7. Feed $h^j_k$ with randomly-cropped image windows from $S$ and collect misclassified samples in set $B_j$.
      8. Update $N_k \leftarrow N_k \cup B_j$.
   9. Select $j$ that gives the maximum value of $(1 - R^j_f)/R^j_n$ for $1 \leq j \leq Num_Epochs$, and let $h_k = h^j_k$.
10. Feed $h_k$ with samples from $F_k$, and let $F_k^f = \{x | h_k(x) > T_k\}$.
11. $F_{k+1} = F_k \setminus F_k^f$
12. $k = k + 1$

Table 3.3: The training algorithm for component neural classifiers.

The $(k+1)$-th component classifier $h_{k+1}$ thus uses a percentage of $P_{k+1}$ of all the training samples, and

$$P_{k+1} = 1 - \sum_{i=1}^{k} \beta_i = 1 - \sum_{i=1}^{k} (\alpha_i \times \prod_{j=1}^{i-1} (1 - \alpha_j)).$$

During the sequential training of the component networks, each network has a decreasing number of available training samples $P_k$. To ensure that each component network has sufficient samples to learn some generalized facial characteristics, $P_k$ should be larger than a performance-critical value (e.g. 5% when $|F| = 6,000$).

Given a fixed topology of component networks, the value of $\alpha_k$ is inversely proportional to threshold $T_k$. Hence, the larger $T_k$, the smaller $\alpha_k$. Eqn. (3.8) provides guidance to the selection of a proper $T_k$ for each component network, such that $P_k$ is large enough to provide sufficient statistics.

In Table 3.3, we give the complete training algorithm for component neural-network classifiers.
Algorithm Training algorithm for the decision neural network

Input: Sets $F$, $N$ and $S$ as used in Table 3.3. A set of $N$ trained component networks $h_k$, a validation face set $V_f$, a validation non-face set $V_n$, and required face detection rate $R_f$.

1. Let $N_t = N$
2. for $j = 1$ to Num_Epochs /* Number of training iterations */
3. Train decision network $g_j$ on face set $F$ and non-face set $N_t$ using the backpropagation algorithm.
4. Compute the false rejection rate $R_j^f$ and false acceptance rate $R_j^n$ over the validation set $V_f$ and $V_n$, respectively.
5. Feed the current ensemble $(h_k, g_j)$ with randomly-cropped image windows from $S$ and collect misclassified samples in $B_j$.
6. Update $N_t ← N_t \cup B_j$.
7. Let $g = g_j$, so that $R^n_0$ is the minimum value for all values of $j$ with $1 \leq j \leq$ Num_Epochs that satisfy $R^f_j > R_f$.

Table 3.4: The training algorithm for the decision network.

Training algorithm for the decision neural network

In Table 3.4, we present the training algorithm for the decision network $g$. During the training of $g$, the inputs are taken from $\langle h_1(x), h_2(x), ..., h_N(x) \rangle$, where $x$ is drawn from face set $F$ or non-face set $N_t$. We use subscript $t$ for $N_t$ because the non-face set grows over time, whereas the face set $F$ remains constant. The training also makes use of the bootstrapping procedure as in the training of the component networks to dynamically add non-face samples to the training set (Line 5 in Table 3.4). In order to prevent the overfitting problem as mentioned in Appendix C, we use here an additional face set $V_f$ and a non-face set $V_n$ for validation purposes.

3.3.4 Performance analysis

To demonstrate the performance of our proposed ensemble technique, we evaluate four ensembles (FNET-A to FNET-D) that are employed in the cascaded detection later in this chapter. The evaluation of the detection accuracy of these ensembles is based on the ROC (Receiver Operating Characteristic) curves. In the case of a neural-network ensemble, the ROC curve shows its performance as a trade-off between the false acceptance rate and the face detection rate by varying the decision threshold $T_g$ of the decision network.

Firstly, we define several data sets used in our training experiments. Our training face set $F$ consists of 6,304 highly variable face images, all cropped
to the size of 24 × 24 pixels. Some of the faces used for the training are depicted in Fig. 3.5. Furthermore, we build up an initial non-face training set $\mathcal{N}$ consisting of 4,548 non-face images of size 24 × 24. Set $\mathcal{S}$ comprises of around 1,000 scenery pictures containing no faces. For each scenery picture, we further generate five scaled versions of it, thereby acquiring altogether 5,000 scenery images. For validation purposes, we use a separate validation face set $\mathcal{V}_f$ consisting of 488 faces and a validation non-face set $\mathcal{V}_n$ consisting of 23,573 non-faces. Each 24 × 24 sample is preprocessed to zero mean and one standard deviation to reduce the influence of global illumination changes.

Let us first quantitatively analyze the performance gain by using an ensemble of neural classifiers. We vary the number of constituting components $N$ and derive the corresponding ROC curve of each ensemble. The evaluation is based on the validation sets $\mathcal{V}_f$ and $\mathcal{V}_n$. In Fig. 3.6, we depict the ROC curves for ensembles based on networks FNET-A and FNET-C, respectively. In Fig. 3.6(a), we can see that the detection accuracy of FNET-A ensemble consistently improves by adding up to three components. However, no obvious improvement can be achieved by using more components. Similar results also hold for the FNET-C ensemble (see Fig. 3.6(b)).

Since using more component classifiers in a neural-network ensemble inevitably increases the total computation cost during the classification, for a given network topology, we need to select $N$ with the best trade-off between the detection accuracy and the computation efficiency.

As a next performance-evaluation step, we compare our proposed classifier ensemble for face detection with two other popular ensemble techniques, bagging and boosting.

As introduced in Section 3.3.1, the bagging injects diversity to different component classifiers by sampling the training set with replacements. There
Figure 3.6: ROC curves of various network ensembles w.r.t. different $N$. 
is no correlation between the different subsets used for the training of different component classifiers. In bagging, we train \( N \) component neural classifiers independently on randomly-selected subsets of the original face training set. The non-face samples are selected in a bootstrapping fashion similar to Table 3.3. The final output \( g_a(x) \) is based on the average of outputs from component classifiers, given by

\[
g_a(x) = \frac{1}{N} \sum_{k=1}^{N} h_k(x). \tag{3.9}
\]

As mentioned in Section 3.3.1, boosting sequentially trains a series of classifiers by emphasizing difficult samples. We adopt the basic scheme of AdaBoost [17]. During the training of the \( k \)-th component classifier, AdaBoost alters the distribution of the samples, such that those samples misclassified by its previous component classifier are emphasized. The final output \( g_o \) is a weighted combination of the outputs from the component classifiers. The algorithm we used for training a set of component neural networks is briefly illustrated in Table 3.5. Note that this version of the AdaBoost training differs from the conventional AdaBoost algorithm w.r.t. the choice of the non-face training set. According to the conventional Adaboost algorithm, the training procedure uses fixed non-face set and face set to train a set of classifiers. However, we found from our experiments that this strategy does not lead to satisfactory results. Instead, we minimize the training error only on the face set. The non-face set is dynamically formed using the bootstrapping procedure.

**Difference between our proposed technique and bagging/boosting**

Different from bagging, our proposed ensemble technique sequentially trains a set of interdependent component classifiers. In this sense, it shares the basic principle with boosting. However, the proposed ensemble technique differs from boosting in the following aspects.

1. Our approach uses a ‘hard’ partitioning of the face training set. Those samples already correctly classified by the current set of networks are not reused for subsequent networks. In this way, face characteristics already learned by the previous networks are not included in the training of subsequent components. Therefore the subsequent networks can focus more on a different class of face patterns during their corresponding training stages.

As a result of the hard partitioning, the subsequent networks are trained on smaller subsets of the original face training set. We have to ensure that each network has sufficient samples that characterize a subclass of face patterns. This has also been discussed in Section 3.3.3.
Algorithm AdaBoost algorithm for a neural-network ensemble

Input: Training face set $F = \{x_i\}$ containing $M$ faces; $N$, $T_k$, $N$ and $S$ are the same as defined in Table 3.3.

1. Let $k = 1$
2. Initialize face sample distribution $w_1(i) = 1/M$ ($1 \leq i \leq M$).
3. while $k \leq N$
   4. Let $N_k = N$. 
5. for $j = 1$ to $Num\_Epochs$ /* number of training iterations */
     6. Train component classifier $h_j^k$ based on face samples selected by distribution $w_k$ and non-face samples $N_k$. 
    7. Calculate the face classification error $\epsilon_j^k = \sum_{i=1}^{M} w_k(i) e_j^k(i)$, where $e_j^k(i)$ is defined as a unit step function $e_j^k(i) = \begin{cases} 1, & \text{if } h_j^k(x_i) \geq T_k, \\ 0, & \text{otherwise}. \end{cases}$ (3.10)
   8. Calculate the false acceptance rate $R_j^k$. 
    9. Feed $h_j^k$ with randomly-cropped image windows from $S$ and collect misclassified samples in set $B_j$. 
   10. $N_k = N_k \cup B_j$. 
   11. Select $j$ that gives the maximum value of $(1 - \epsilon_j^k)/R_j^k$ with $1 \leq j \leq Num\_Epochs$, and let $h_k = h_j^k$, $\epsilon_k = \epsilon_j^k$, $e_k(i) = e_j^k(i)$ for $1 \leq i \leq M$. 
   12. Update the sample distribution by
       $$w_{k+1}(i) = \frac{w_k(i) \beta_k^{1-e_k(i)}}{Z_k},$$ (3.11)
       where $\beta_k = \epsilon_k/(1 - \epsilon_k)$, and $Z_k$ is a normalization factor, such that $w_{k+1}$ forms a distribution.
   13. Let $k = k + 1$
   14. The final combined classifier output $g_o$ is defined as
       $$g_o(x) = \begin{cases} 1, & \text{if } \sum_{k=1}^{N} \alpha_k h_k(x) \geq \frac{1}{2} \sum_{k=1}^{N} \alpha_k, \\ 0, & \text{otherwise}, \end{cases}$$ (3.12)
       where $\alpha_k = \log(1/\beta_k)$.

Table 3.5: AdaBoost training for a neural-network ensemble.
3.3. Neural-network ensemble

Figure 3.7: ROC curves of network ensembles using different training strategies.
Figure 3.8: ROC curves of network ensembles using different training strategies (cont.).
2. We use a decision neural network to make the final classification based on individual outputs from component networks. This results in a more flexible decision function than the linear combination rule used by bagging or boosting.

In Fig. 3.7 and Fig. 3.8, we compare the performance of the resulting neural-classifier ensembles trained with different strategies. It can be seen that for complex base-network structures such as FNET-C and FNET-D, our proposed neural-classifier ensemble produces the best results. For a base network with relatively simple structures, such as FNET-A and FNET-B, our proposed ensemble gives comparable results w.r.t. the boosting-based algorithm. It is worth mentioning that for the most complex network structure FNET-D, bagging or boosting only gives a marginal improvement as compared to using a single network, while our proposed ensemble gives much better results than the other techniques. This can be explained by the following reasoning.

The training strategy adopted by the boosting technique is mostly suitable for combining weak classifiers that may only work slightly better than random guessing. Therefore, during the sequential training as shown in Table 3.5, it is beneficial to reuse the samples that are correctly classified by its previous component networks to reinforce the classification performance. For a neural network with simple structures, the use of boosting can be quite effective in improving the classification accuracy of the ensemble. However, when training strong component classifiers, which can already give quite accurate classification results in stand-alone operation, it is less effective to repeatedly feed the samples that are already learned by the preceding networks. Neural networks with complex structures (e.g. FNET-C and FNET-D) are such strong classifiers, and for these networks, our proposed strategy is more effective and gives better results in practice.

To summarize this section, we have introduced a novel approach by creating an ensemble of cooperating neural classifiers for face detection. Each component neural network in the ensemble is trained on a different region in the face space, and the final decision rule is learned by a decision neural network based on the outputs from the component classifiers. We have evaluated the performance of the ensemble based on four architectures (FNET-A to FNET-D). The detection results demonstrated the advantages of using our proposed approach for face detection.

In the remaining part of this chapter, we use neural-network ensembles as the ‘building blocks’ for a pruning cascade for fast, yet accurate face detection.
3.4 Cascaded neural ensembles for fast detection

In this section, we apply the ensemble technique into a cascading architecture for face detection, such that both the detection accuracy and efficiency are jointly optimized.

Fig. 3.9 depicts the structure of the cascaded neural-network ensembles for face detection, following the basic cascading principle as introduced in Chapter 2. In this structure, we cascade a set of neural-network ensembles, where non-face candidate windows during the multi-resolution search are gradually pruned at each stage. More efficient ensemble classifiers with simpler base networks are used at earlier stages in the cascade, which are capable of rejecting a majority of non-face patterns, thereby boosting the overall detection efficiency.

In the following, we introduce a notation framework in order to come to expressions for the detection accuracy and efficiency of cascaded ensembles. Afterwards, we propose a technique to jointly optimize the Cascaded-Ensemble (CE) face detector for both accuracy and efficiency. Following that, we introduce an implementation of a CE face detector using five ensembles.

3.4.1 Formulation and optimization of cascaded ensembles

As shown in Fig. 3.9, we assume a total of \( L \) neural-network ensembles \( g_i \) \((1 \leq i \leq L)\) with increasing base-network complexity. The behavior of each ensemble classifier \( g_i \) can be characterized by face detection rate \( f_i(T_i) \) and false acceptance rate \( d_i(T_i) \), where \( T_i \) is the output threshold of the decision network in the ensemble. By varying \( T_i \) in the interval \([0, 1]\), we can obtain different pairs \( \langle f_i(T_i), d_i(T_i) \rangle \) which actually constitute the ROC curve of ensemble \( g_i \). Now the question arises how we can choose a set of appropriate values for \( T_i \), such that the performance of the cascaded classifier is optimal.

Suppose we have a detection task with a total of \( I \) candidate windows, and \( I = F + N \), where \( F \) is the number of faces and \( N \) is the number of non-faces. The first classifier in the cascade takes \( I \) windows as input, among which \( F_1 \) windows are classified as faces and \( N_1 \) windows are classified as non-faces. Hence, \( I = F_1 + N_1 \). The \( F_1 \) windows are passed on to the second classifier.
for further verification. More specifically, the $i$-th classifier ($i > 1$) in the cascade takes $I_i = F_{i-1}$ input windows and classifies them into $F_i$ faces and $N_i$ non-faces. At the first stage, it is easy to see that

$$F_1 = f_1(T_1)F + d_1(T_1)N,$$

(3.13)

More generally, it holds that

$$F_i = f_i(T_1, T_2, ..., T_i)F + d_i(T_1, T_2, ..., T_i)N,$$

(3.14)

where $f_i(T_1, T_2, ..., T_i)$ and $d_i(T_1, T_2, ..., T_i)$ represent the face detection rate and false acceptance rate, respectively, of the subcascade formed jointly by the first to the $i$-th ensemble classifiers. Note that it is difficult to express $f_i(T_1, T_2, ..., T_i)$ explicitly using $f_i(T_i)$ and $d_i(T_i)$, since the behaviors of different ensembles are usually correlated. In the following, we first define two target functions for maximizing the detection accuracy and efficiency of the cascaded detector. Following this, we propose a solution to optimize both objectives.

**Detection accuracy.** The detection accuracy of a face detector is characterized by both its face detection rate and false acceptance rate. For a specific application, we can define the maximally-allowed false acceptance rate. Under this constraint, the higher the face detection rate, the more accurate the classifier. More specifically, we use cost function $C_p(T_1, T_2, ..., T_L)$ to measure the detection accuracy of the $L$-ensemble cascaded classifier, which is defined by the maximum face detection rate of the classifier under the condition that the false acceptance rate is below a threshold value $T_d$. Therefore,

$$C_p(T_1, T_2, ..., T_L) = \max f_L(T_1, T_2, ..., T_L)$$

subject to: $d_L(T_1, T_2, ..., T_L) < T_d$.

(3.15)

**Detection efficiency.** We define the detection efficiency of a cascaded classifier by the total amount of time required to process the $I$ input windows, denoted as $C_e(T_1, T_2, ..., T_L)$. Suppose the classification of one image window by ensemble classifier $g_i$ takes $t_i$ time. To classify $I$ candidate windows by the complete $L$-layer cascade, we need a total amount of time

$$C_e(T_1, T_2, ..., T_L) = \sum_{i=0}^{L-1} F_i t_{i+1}$$

(with $F_0 = I$)

$$= \sum_{i=0}^{L-1} (f_i(T_1, T_2, ..., T_i)F + d_i(T_1, T_2, ..., T_i)N)t_{i+1},$$

(3.16)
Algorithm Parameter selection for the CE face detection

**Input:** $F$ test face patterns and $N$ test non-face patterns. A classifier cascade consisting of $L$ neural-network ensembles. Maximally-allowed false acceptance rate $T_d$.

**Output:** A set of selected parameters $(T^*_1, T^*_2, ..., T^*_L)$.

1. Select $T^*_L = \text{argmax}_{T_L} f_L(T_L)$, subject to $d_L(T_L) \leq T_d$.
2. for $k = L - 1$ to 1
3. Select $T^*_k = \text{argmax}_{T_k} C(T_k, T^*_k+1, ..., T^*_L)$.

Table 3.6: Parameter selection for the CE face detection.

where the last step is based on Eqn. (3.14) and we define the initial rates $f_0 = 1$ and $d_0 = 1$.

The performance of a cascaded face detector should be measured by both its detection accuracy and efficiency. To this end, we combine cost functions $C_p$ (Eqn. (3.15)) and $C_e$ (Eqn. (3.16)) into a unified function $C$, which measures the overall performance of a cascaded face detector. There are various combination methods. One example is based on a weighted summation of Eqn. (3.15) and Eqn. (3.16):

$$C(T_1, T_2, ..., T_L) = C_p(T_1, T_2, ..., T_L) - wC_e(T_1, T_2, ..., T_L).$$ (3.17)

We use a subtraction for the efficiency (time) component to trade-off against accuracy. By adjusting $w$, the relative importance of desired accuracy and efficiency can be controlled\(^1\).

In order to obtain a CE face detector of high performance, we aim at maximizing the performance goal as defined by Eqn. (3.17). For a given cascaded detector consisting of $L$ ensembles, we can optimize over all possible $T_i$ ($1 \leq i \leq L$) to obtain the best parameters $T^*_i$. However, this process can be computationally prohibitive, especially when $L$ is large. In the following, we propose a heuristic suboptimal search to determine these parameters.

**Sequential backward parameter selection.** In Table 3.6, we present the algorithm for selecting a set of parameters $(T^*_1, T^*_2, ..., T^*_L)$ that maximizes Eqn. (3.17). Since the final face detection rate $f_L(T^*_1, T^*_2, ..., T^*_L)$ is upper-bounded by $f_L(T^*_L)$, we first ensure a high detection accuracy by choosing a proper $T^*_L$ for the final ensemble classifier (Line 1 in Table 3.6). Following that, we add each ensemble in a backward direction and choose its threshold

\(^1\)Factor $w$ also compensates for the different units used by $C_p$ (detection rate) and $C_e$ (time).
3.4. Cascaded neural ensembles for fast detection

3.4.1 Parameter selection

parameter \(T_k^*\) such that the partially-formed cascade from the \(k\)-th to the \(L\)-th ensemble gives an optimized \(C(T_k^*, T_{k+1}^*, ..., T_L^*)\).

The experimental results show that this selection strategy gives very good performance in practice.

3.4.2 Implementation of a CE detector

We build a five-stage cascade of classifiers with increasing order of topology complexity. The first four stages are based on component network structures FNET-A to FNET-D, as illustrated in Section 3.3.2. The final ensemble consists of all component networks of FNET-D, plus a set of additional component networks that are variants of FNET-D. These additional component networks allow overlapping of locally connected blocks, so that they offer slightly more flexibility than the original FNET-D. Although in principle a more complex base-network structure can be used and the final ensemble can be constructed following the similar principle as FNET-A to FNET-D, we found in our experiments that using our proposed strategy for the final ensemble construction already offers sufficient detection accuracy while still keeping the complexity at a reasonably low level.

In the following, we analyze the performance of the five proposed neural-network ensembles and also study the performance gain by using the cascaded structure.

3.4.3 Performance analysis of the CE face detector

As a first step, we first evaluate the individual behavior of each trained ensemble. Using the same training sets and validation sets as in Section 3.3.4, we obtain the ROC curves of different ensemble classifiers \(g_i\) as depicted in Fig. 3.10. The plot at the bottom part of the figure is a zoomed version where the false acceptance rate is within \([0, 0.015]\).

Afterwards, we form a cascade of neural-network ensembles from \(g_1\) to \(g_5\). The decision threshold of each network ensemble is chosen according to the parameter-selection algorithm given in Table 3.6. We depict the ROC curve of the resulting cascade in Fig. 3.11, and the performance of the \(L\)-th (final) ensemble classifier is given in the same plot for comparison. It can be noticed that for false acceptance rate below \(5 \times 10^{-4}\) for the given validation set, which is normally required for real-world applications, the CE face detector has almost the same face detection rate as the most complex \(L\)-th stage classifier. However, we have found that when it is tested on the validation sets \(\mathcal{V}_f\) and \(\mathcal{V}_n\), the computation time drastically drops to less than 5% compared to using the \(L\)-th stage classifier alone.

In our implementation, we train each ensemble independently and then build up a cascade. A slightly different strategy is to sequentially train the
ensembles, such that the subsequent ensemble detectors are only fed with the non-face samples that are misclassified by the previous ensemble detectors. This strategy was adopted by the Viola-Jones detector in [75]. When this strategy is used in the neural-ensemble cascade in our case, our experiments show that such a training scheme leads to slightly worse results than with
the independent training. This may be due to the relatively good learning capability of subsequent ensemble classifiers, which is less dependent on the relatively ‘easy’ non-face patterns to be pruned. More study is still needed to arrive to a solid explanation.

Another benefit offered by the independent training is the saving of the training time\(^2\). This is because during the cascaded training, it takes longer time to collect non-face samples during the bootstrapping training for more complex ensembles, considering the relatively low false acceptance rate of the partially formed subcascade.

In the following, we apply the above-obtained CE detector to real-world face-detection tasks.

### 3.5 Application to real-world face detection

In this section, we first discuss several implementation issues of applying the CE detector to find faces from arbitrary input images (video sequences). Following that, we discuss the performance of the detector on several test sets and compare it to other well-known face detectors.

\(^2\)The complete training takes roughly one hour in our experimental setup (P-IV PC 3.0 GHz).
3.5.1 Strategies for real-world face detection

In this subsection, we discuss the search strategy used by the CE detector and related preprocessing and postprocessing techniques.

Multi-resolution face scanning. Given a still image (or a video frame from a sequence), the detector needs to perform a multi-resolution search for faces as already mentioned in Chapter 2. Here we use a scaling factor of 1.2 between adjacent image scales during the search (refer to Table 2.1). In Fig. 3.12, we give an example of the search results by using the new CE detector.

Fast preprocessing using integral images. Our proposed face detector accepts an image window preprocessed by zero mean and unity standard deviation, with the aim to reduce the global illumination influence. To facilitate efficient image preprocessing during the multi-resolution search, we compute the mean and variance of an image window using a pair of auxiliary integral images of the original input image. The integral image of an image with intensity $P(x, y)$ is defined as

$$I(u, v) = \sum_{x=1}^{u} \sum_{y=1}^{v} P(x, y). \quad (3.18)$$

Given the integral image $I$ of the original image, for an image window with its top-left corner $(x_1, y_1)$ (point 1) and bottom-right corner $(x_2, y_2)$ (point 4) (see Fig. 3.13), the summation of the pixel intensity in this subwindow can be quickly obtained by

$$S = I(x_1, y_1) + I(x_2, y_2) - I(x_2, y_1) - I(x_1, y_2). \quad (3.19)$$

A fast algorithm is proposed in [75] to obtain the integral image of a given image in only one pass over the image. By using integral images, the mean of an image window can be easily obtained using Eqn. (3.19). Similarly, a
3.5. Application to real-world face detection

3.5.2 Performance analysis

In this subsection, we apply our CE face detector on a number of test sets and evaluate its detection accuracy and efficiency. Three test sets containing various images and video sequences are used for our evaluation purposes, which
Chapter 3. Cascaded Face Detection Using Neural-Network Ensembles

(a) Face-detection results before the merging of overlapping boxes.  
(b) Face-detection results after the merging of overlapping boxes.

**Figure 3.14:** Merging overlapping detections.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Num. of images (sequences)</th>
<th>Num. of faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMU+MIT</td>
<td>130</td>
<td>507</td>
</tr>
<tr>
<td>WEB</td>
<td>98</td>
<td>199</td>
</tr>
<tr>
<td>HN2R-DET</td>
<td>46</td>
<td>50</td>
</tr>
</tbody>
</table>

**Table 3.7:** Data sets used for the evaluation of our proposed face detector.

are listed in Table 3.7. The CMU+MIT set is the most widely-used test set for benchmarking face-detection algorithms [60], and many of the images included in this data set are of very low quality. The WEB test set contains various images randomly downloaded from the Internet. The HN2R-DET set contains various images and video sequences we have collected using both a DV camera and a web camera during several test phases in the HN2R project [59].

First, we compare our detection results to reported results from the literature on the CMU+MIT test set. The comparison results are given in Table 3.8⁴. It can be seen that our approach for face detection is among one of the best performing techniques in terms of detection accuracy. In Fig. 3.15, we give some example results using our approach for the CMU+MIT data set.

Using the WEB data set, we achieve a face detection rate of 93% with a total of 29 false positives. For the HN2R-DET set, which captures indoor scenes with relatively simple background, a total of 98% detection rate is achieved with zero false positives. Some detection results using these two data sets are shown in Fig. 3.16.

⁴Techniques 3, 4, 7, and 8 use a subset of the test sets excluding hand-drawn faces and cartoon faces, leaving 483 faces in the test set. We reported our results on 479 faces excluding further 4 faces using face masks or having poor resolution, since we do not consider these situations in the construction of our training sets.
3.6. Summary and conclusions

<table>
<thead>
<tr>
<th>Detector</th>
<th>Detection rate</th>
<th>Num. of false positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Single neural network [60]</td>
<td>90.9%</td>
<td>738</td>
</tr>
<tr>
<td>2. Multiple neural networks</td>
<td>84.4%</td>
<td>79</td>
</tr>
<tr>
<td>3. Bayes statistics [63]</td>
<td>94.4%</td>
<td>65</td>
</tr>
<tr>
<td>4. SNoW [81]</td>
<td>94.8%</td>
<td>78</td>
</tr>
<tr>
<td>5. AdaBoost [75]</td>
<td>88.4%</td>
<td>31</td>
</tr>
<tr>
<td>6. FloatBoost [38]</td>
<td>90.3%</td>
<td>8</td>
</tr>
<tr>
<td>7. SVM [25]</td>
<td>89.9%</td>
<td>75</td>
</tr>
<tr>
<td>8. Convolutional network [20]</td>
<td>90.5%</td>
<td>8</td>
</tr>
<tr>
<td>9. Our approach [97]</td>
<td>94.4%</td>
<td>61</td>
</tr>
</tbody>
</table>

Table 3.8: Comparison of different face detectors for the CMU+MIT data set.

sets are shown in Fig. 3.16.

Next we evaluate the efficiency gain by using our proposed detector. For the CMU+MIT test set, the five ensembles in the cascade reject 77.2%, 15.5%, 6.2%, 1.1% and 0.09% of all the background image windows, respectively. For a typical image of size 320 × 240, using a cascade can significantly reduce the computation of the final ensemble by 99.4%, bringing the processing time from several minutes to a subsecond level. When processing video sequences of 320 × 240 resolution, we achieve a 4-5 frames/second detection speed on a Pentium-IV PC (3.0 GHz). The detection is frame-based without the use of any tracking techniques. Several example frames are shown in Fig. 3.17 when the detector is applied to real-time face detection from live cameras.

3.6 Summary and conclusions

In this chapter, we have presented a face detector using a cascade of neural-network ensembles, with the following two advantages.

First, we have used a neural-network ensemble for improved detection accuracy, which consists of a set of component neural networks and a decision network. The experimental results have shown that our proposed ensemble technique outperforms several existing techniques such as bagging and boosting, with significantly better ROC performance for more complex neural-network structures. For example, as shown in Fig. 3.8(b), by using our proposed technique, the false rejection rate has been reduced by 23% (at the false acceptance rate of 0.5%) as compared to bagging/boosting.

Second, we have used a cascade of neural-network ensembles with increasing complexity, in order to reduce the total computation cost of the detector. Fast ensembles are used first to quickly prune large background areas, while
subsequent ensembles are only invoked for more difficult cases to achieve a refined classification. Based on a new weighted cost function incorporating both detection accuracy and efficiency, we use a sequential parameter-selection algorithm to optimize the defined cost. The experimental results have shown that our detector has effectively reduced the total processing time from minutes to a fraction of a second, while maintaining similar detection accuracy as compared to the most powerful subdetector in the cascade.

When used for real-world face-detection tasks, our proposed face detector in this chapter is one of the best performing detectors in detection accuracy, with 94.4% detection rate and 61 false positives on the CMU+MIT data set (see Table 3.8). In addition, the cascaded structure has greatly reduced the required computation complexity. The proposed detector has been applied in a real-time face-recognition system operating at 4-5 frames/second.

It is also worth pointing out the architectural advantages offered by the proposal. In our detector framework, each subdetector (ensemble) in the cascade is built upon similar structures, and each ensemble is composed of base networks of the same topology. Within one ensemble, the component networks can simultaneously process an input window. This structure is most suitable to be implemented in parallelized hardware architectures, either in multiprocessor layout or with reconfigurable hardware cells. Additionally, the different ensembles in a cascade can be implemented in a streamlined manner to further accelerate the cascaded processing. It is readily understood that these features are highly relevant for embedded applications.

The face detection is the very first step to achieve a complete face-recognition system. The outputs of the face detection are bounding boxes of existing faces in an input image, indicating the coarse position and scale for each face. In the following two chapters, we address a subsequent topic in face recognition: facial feature extraction, which aims at locating important facial structures such as eyes, nose and mouth. Accurate extraction of facial features can facilitate precise face alignment and normalization for the final face identification.
Figure 3.15: Examples of detection results on the CMU+MIT data set.
Figure 3.16: Examples of detection results on the HN2R-DET and WEB data sets.
Figure 3.17: Frame samples from a live face-detection system.
4.1 Introduction

Whereas the previous chapters described the first stage of face recognition, in the upcoming two chapters, we focus on techniques related to the second stage in face recognition: facial feature extraction.

Let us briefly recall the overall architecture of a face-recognition system as introduced in Chapter 1. As shown in Fig. 4.1, the output of a face detector usually contains coarse indications of positions and scales of existing faces from the input images (video). Given this knowledge, facial feature extraction aims at locating important facial feature structures, such as eyes, nose and mouth. Depending on specific applications, the outputs of facial feature extraction can be the center locations of facial features. Alternatively, they can refer to more intricate descriptions such as contours of these facial features. The outputs of the facial feature extraction can be utilized by the final identification stage in the following ways.

1. **Face alignment and normalization**: Based on the positions of facial features, the detected face can be geometrically aligned to a uniform pose w.r.t. position, scale and rotation. This normalization step is crucial for the final face identification because it largely reduces the interference of different scales and rotations existing in the face-detection outputs. It is reported in literature [44][37] that significant degradation in identification performance can be observed without careful normalization.
2. **Feature generation for identification**: Since facial features usually carry important information to discriminate different faces, they are often used to represent faces in many face-identification systems (e.g., in [77][3]). It has been shown in [84] that the accurate localization of facial features plays an important role for achieving a high identification rate in these systems.

In addition to face recognition, it is worth pointing out that facial feature extraction is also a key step in many other applications such as facial-expression analysis [69], face tracking and coding [2] and face synthesis [41].

For achieving an automatic facial feature extraction system, a number of technical challenges should be addressed.

- The facial feature appearances and their geometric relationships are considerably influenced by individual variations and also affected by various facial expressions.
- Especially for non-professional applications, the image-capturing environments are more variable than for professional cases. Particularly cameras and illumination conditions exert difficulties to the feature extraction procedure.
A good feature-extraction algorithm should provide accurate feature locations while being robust against large feature variances. Furthermore, for embedded applications, computation cost is also an important factor that should be taken into account during the algorithm design.

The purpose of this chapter is to present a brief review of a number of state-of-the-art techniques for facial feature extraction, with special emphasis on model-based techniques. Following that, we propose an improved new algorithm on statistical shape model, denoted as H-ASM[94][91].

The structure of this chapter is organized as follows. In Section 4.2, we present a brief review of current facial feature extraction techniques in literature, which are roughly classified into three categories: heuristic techniques, pattern-classification-based techniques and model-based techniques. Section 4.3 focuses particularly on model-based techniques and reviews several representative techniques. In Section 4.4, we present an improved algorithm for feature extraction based on a statistical shape model. Section 4.5 summarizes this chapter.

4.2 Overview of facial feature extraction techniques

In the past decade, a number of techniques have been proposed in literature on automatic facial feature extraction. In the following, we classify these techniques into three categories: heuristic techniques, pattern-classification-based approaches (referred to as classification-based techniques) and model-based approaches. It should be noted that some feature-extraction\(^1\) algorithms may belong to multiple categories.

1. **Heuristic techniques**: Earlier techniques find the positions of facial features based on empirical knowledge about facial feature appearances (e.g. [28][61]), which is usually explicitly defined as rules. For example, the positions of two eyes should be found in regions with relatively low intensities on the upper part of the face, and they should be symmetric about the center line of the face. Another commonly-used knowledge is that the color distribution of feature components should have larger differences than the remaining parts of the face (e.g. [28]), which are mostly of skin-tone. Although straightforward, it is difficult to define adequate rules based on heuristic knowledge alone, due to the large feature variances under varying situations.

2. **Classification-based techniques**: Classification-based techniques are most suitable for coarsely locating feature components. Usually they derive a

\(^{1}\)For simplicity, in this and the following chapter, we use *feature* as an abbreviation for *facial feature*.
Figure 4.2: Example of a model-based feature-extraction scheme.

template or learning model for each feature component and search the face region for the best match. In a general view, feature localization in this context can be seen as a two-class pattern-classification problem, where the classes are feature and non-feature. Basically, the techniques used for face detection from Chapter 2 and 3 can also be applied on a smaller scale to the feature detection. Typical examples include the Eigenfeature-based techniques in [70] and Adaboost-based techniques in [14]. Classification-based techniques are relatively robust for large feature variances. However, they usually provide only coarse localization results.

3. Model-based techniques: Model-based techniques adopt feature models that can adapt to the underlying image. These techniques explicitly or implicitly incorporate \textit{a-priori} knowledge of facial features into the model and search for optimal model parameters for a pre-defined cost function. In general, the following elements are required for a model-based technique.

(a) Feature model: A feature model is an abstract representation of facial feature structures. It should capture inherent characteristics of facial features and at the same time still have the flexibility to allow for individual feature variances. In literature, facial features can be modeled in various ways, such as the graph structure as shown in Fig. 4.2, where the \textit{nodes} represent the important facial features.

A feature model usually has a set of parameters. By varying these parameters, the model can adapt to different face instances. A cost function is usually defined to measure the matching quality of the model to an input face image. The aim of feature extraction is then to find a set of optimal model parameters, such that the cost function can be minimized. This optimization is usually accomplished by using a \textit{model-fitting algorithm}. 

4.3 Model-based algorithms for locating features

(b) Model-fitting algorithm: Such an algorithm can be seen as an optimization procedure, which defines how the optimal model parameters can be obtained for the best matching to an input image. For example, in Fig. 4.2, the model can be deformed in such a way that the facial features of an underlying face image can be accurately located.

In this thesis, we focus mainly on model-based feature extraction, since it offers both accurate description of features and flexibility to adapt to individual feature variances. In the following section, we take a closer look at several representative model-based techniques for feature extraction.

4.3 Model-based algorithms for locating features

Generally, the performance of a model-based algorithm is characterized by the following two important aspects.

1. Capture range: A model-based algorithm usually needs an explicitly-initialized model (or initial search range) to start with the model-fitting procedure. In order to characterize the algorithm’s ability to converge to correct feature locations starting with different initializations, we define a region called capture range, which includes all parameter initializations that lead to correct convergence.

2. Extraction accuracy: For converged feature-extraction cases, i.e. when the model initialization is within the capture range, we use the extraction accuracy to indicate how precise the extracted feature locations are, as compared to the ground-truth information. The extraction accuracy is usually expressed by the deviation between a fitted model and the manually-labeled feature positions.

The capture range and extraction accuracy are two closely-related concepts. We have observed that in current literature, these concepts are usually empirically defined and no clear criteria exist for deriving these metrics. In the next chapter, we define these performance metrics in a more formal way.

In the sequel, we first review several representative model-based extraction algorithms.

4.3.1 Deformable geometric templates

Facial features belong to a category of objects that share common shape characteristics while still having variances due to individual diversity, facial expressions, etc. A promising technique to model this kind of objects is to use deformable models, which can adapt to individual instances with certain model
Figure 4.3: Deformable templates used by Yuille et al. [82]. Left: eye template; Right: mouth template.

constraints. These constraints can be based on a-priori knowledge of object properties. For example, active contour models (snakes) [5] can be used to model individual facial features, e.g. an eye. A ‘snake’ is actually a smooth curve to characterize the contour of an eye, which has to be initialized near the target positions. This curve can be iteratively refined to optimize a pre-defined energy function encouraging e.g. strong edges and curve smoothness.

Another technique used in literature is deformable geometric templates [82], which incorporate more object-specific a-priori knowledge about facial features. In [82], two parameterized feature templates (models) are pre-defined for matching the eyes and mouth individually (see Fig. 4.3). In order to find the best parameters for the model to match a given image, the technique first chooses several image representations (valley map, peak map and gradient map) to extract properties such as peaks, valleys and gradients in the grayscale values of the image. Based on these representations, an energy function $E$ is then defined as a weighted sum of terms reflecting the knowledge about the feature characteristics. For instance, suppose that $V(x, y)$ is the valley map of the image, an energy term $E_v$ can be defined as

$$E_v = \frac{1}{\text{Area}} \int \int_{\text{Circle--area}} V(x, y)d(x, y), \quad (4.1)$$

where the integral above is performed over the interior of the eye circle.

Given an initialized model near the target position, the fitting algorithm applies an iterative procedure such as a gradient-descent algorithm to minimize the energy function $E$. It is reported in [82] that the algorithm has achieved accurate extraction results. However, it can be difficult in practice to find a set of suitable weights for each energy term. Furthermore, the fitting procedure may converge to incorrect local minima due to improper model initializations and large feature variances, resulting in a limited convergence performance.

To improve the deformable template modeling, in a preliminary work [89] we proposed a multistage feature-localization technique incorporating (1) a preprocessing step to select multiple initializations for feature template match-
4.3. Model-based algorithms for locating features

...ing; and (2) a postprocessing step to verify the feature locations using a set of heuristic rules. The proposed method improves the reliability of the template matching of [82].

The a-priori knowledge about facial features in deformable templates as used by [82][89] are specifically defined for eyes and mouth, which lacks flexibility to deal with more general feature specifications. An alternative approach is to implicitly define facial feature properties by using statistics, which automatically derives feature models by the use of a set of training samples. The statistics-based approach for feature extraction offers more flexibility, which also can be easily applied to other object structures. Typical techniques using statistical models are Active Shape Model (ASM) [9] and Active Appearance Model (AAM) [10], both of which are based on the so-called statistical shape model [11].

### 4.3.2 Statistical shape model

Statistical shape model is the basis for a number of popular feature-extraction algorithms, such as ASM, AAM and their variants [91][64][27]. It differs from earlier approaches in that the feature model is implicitly derived from statistics instead of explicitly defined. Another noteworthy feature of statistical shape model is that it is a structural technique, which considers the constraints on the relations between various facial features. The use of structural matching often yields more robustness in terms of handling variations in image intensity...
and individual diversity [11].

More specifically, in a statistical shape model, \( k \) key points are defined which mark the contours of prominent facial features. An example is shown in Fig. 4.4(a). A shape \( s \) is defined as a vector containing the concatenated coordinates of these \( k \) points, given by \( s = (x_1, x_2, ..., x_k, y_1, y_2, ..., y_k)^T \), where \((x_i, y_i)\) is the coordinate of the \( i \)-th feature point. It is easy to see that the shape vector \( s \) resides in a \( 2k \)-dimensional shape space. In order to capture the essential shape variability of different faces, Principal Component Analysis (PCA, see Appendix B) can be applied to a set of manually-labeled shape samples that have been aligned to a common coordinate system. Therefore, any normalized (w.r.t. position, scale and rotation) shape vector \( s_n \) can be approximated by

\[
s_n \approx \bar{s} + \Phi \cdot b. \tag{4.2}
\]

In the above, \( \bar{s} \) is the mean of normalized shape samples, and \( \Phi \) contains \( L \) (\( L < 2k \)) Eigenvectors with the largest Eigenvalues after the Eigen-decomposition of the shape covariance matrix (PCA). Vector \( b \) is the projection of \( s_n - \bar{s} \) in the reduced shape space spanned by \( \Phi \), which actually defines a set of deformation parameters for the given class of shape vectors. The model variation corresponding to the \( i \)-th parameter \( b_i \) is called the \( i \)-th mode of the model, which has variance \( \lambda_i \) (Eigenvalue of the PCA decomposition). Fig. 4.4(b) depicts several synthesized shapes by varying the first three shape modes.

If we further incorporate the geometric transformation (translation, scale and rotation), any shape vector \( s \) (not normalized) can be approximated by the expression

\[
T_{x,y,r,\phi}(\bar{s} + \Phi \cdot b), \tag{4.3}
\]
where $T_{x,y,r,\phi}$ is a geometric transformation function parameterized by translation $(x, y)$, scale $r$ and rotation $\phi$.

In this framework, the model of facial features is derived from the statistical analysis of a set of samples. Based on this model, a class of algorithms has been proposed in literature, such as ASM and AAM. These techniques add more elements to the shape model such as texture representations and employ various fitting algorithms. For example, the ASM [9] fits a shape model to an input image by using a local deformation process constrained by the global variances. The ASM forms the basis for the H-ASM algorithm [91], and is explained in more detail in the next section.

Different from ASM, the AAM [10] further incorporates texture analysis in addition to a statistical analysis of face shapes. More specifically, the AAM first performs a Delauney triangulation of the face image based on the shape (see the left of Fig. 4.5). The enclosed texture region of the triangle mesh is then normalized and sampled into a texture vector. Both the shape and texture information are modeled using PCA as in the statistical shape model, and the combined variation modes are demonstrated at the right of Fig. 4.5.

In order to match the model to a given image, an optimal vector of model parameters are searched by minimizing the difference between a synthetic image (by the model) and the given one. In Fig. 4.6, we exemplify the iterative searching/optimization procedure used by AAM.

In addition to the above-mentioned scheme, a variety of related approaches to AAM have also been proposed in literature, such as Shape AAM [64], Direct AAM [27], etc. Interested readers are referred to [11][12] for a more complete review of these techniques.

In the next section, we present a new enhanced version of ASM by incorporating Haar-based texture modeling, which has significantly improved performance as compared to the conventional ASM in both the capture range and the extraction accuracy.

**Figure 4.6:** Example texture-fitting procedure of AAM.
4.4 H-ASM: Haar-based shape model

In this section, we propose a Haar-feature shape model (H-ASM) for feature extraction. As a preliminary, we first give in Section 4.4.1 a brief review of the Active Shape Model.

4.4.1 Introduction to Active Shape Model (ASM)

The Active Shape Model uses the same statistical shape model as defined in Section 4.3.2, which is parameterized by $(x, y, r, \phi, b)$. In addition, each key point in a face shape is associated with a texture vector describing the local feature appearance, which plays an important role in the subsequent model-fitting procedure.

**Model definition.** In ASM, the face is modeled by two parts: (1) a set of *key feature points* constituting a face shape, which is described by the statistical model in Eqn. (4.3), and (2) a set of *profile vectors*, each of which characterizes the local texture of a key feature-point. A profile vector $g_i$ is defined by the sampled image intensities perpendicular to the shape boundary (see Fig. 4.7). In practice, in order to reduce the sensitivity to global illumination, $g_i$ actually contains the normalized derivatives of these sampled intensities instead of the original values.

In order to derive a statistical description of vector $g_i$, for each feature point, a set of sampled profile vectors $\{g_i\}$ are first obtained from a set of training images. It is further assumed in ASM that these vectors are distributed as a multivariate Gaussian, and the mean $\bar{g}$ and covariance matrix $S_g$ can be estimated from the sample set $\{g_i\}$ and stored for further reference.

**Model-fitting algorithm.** We depict the model-fitting algorithm of ASM in Table 4.1. Given an initialized shape model with certain parameters, three alternating steps (see Line 2, 3 and 4 in Table 4.1) are carried out iteratively to
Algorithm  Model fitting in ASM

Input: An initialized shape $s$ and face image $f$.

Output: Fitted shape $s_f$.

1. repeat
2. For each feature point $(x_i, y_i)$ in $s$, search in its neighborhood in $f$ for a local best match $(x'_i, y'_i)$ based on profile matching, resulting in new shape $s'$.
3. Update the parameters $(x, y, r, \phi, b)$ to best fit $s'$.
4. Apply constraints (e.g. $\pm 3\sqrt{\lambda_i}$) to each $b_i$ to ensure that $s'$ is a plausible face shape.
5. $s = s'$
6. until convergence.
7. $s_f = s$

Table 4.1: Model-fitting algorithm of ASM.

fit the model to the facial features in an input image. These steps are further illustrated as follows.

1. Step 1: The position of each feature point in the current model is updated by searching for a best match in its neighborhood. The matching is based on the quality of fit of a profile vector $g_s$ of a candidate point to the mean profile vector $\bar{g}$, which is defined by the following:

$$m(g_s) = (g_s - \bar{g})^T S_g^{-1} (g_s - \bar{g}). \quad (4.4)$$

This is actually the Mahalanobis distance between $g_s$ and the model mean $\bar{g}$.

2. Step 2: The model parameters are updated to best align to the newly generated shape $s'$ after the first step. This is achieved by first estimating the parameters $(x, y, r, \phi)$ and then projecting the new shape to the shape space $\Phi$ to obtain $b$.

3. Step 3: The global constraints are applied to each deformation mode $b_i$ in the statistical shape model. By applying limits, e.g. $\pm 3\sqrt{\lambda_i}$ to parameter $b_i$, the generated shape is constrained to a plausible face-like shape.

In Fig. 4.8, we show an example fitting procedure by ASM.

One important limitation of ASM is that only 1-D profile information is used to characterize local textures during the model deformation, which is
insufficient to correct large model deviations. In order to improve the robustness of the shape model, we proposed to use 2-D texture attributes to characterize each feature point in a shape model [92][90]. However, since using 2-D textures inevitably increases the computation complexity, we further adopt Haar-wavelets for modeling these local texture attributes, which offers both high processing speed and fitting robustness. In the next subsection, we illustrate the proposed extension of ASM (denoted as H-ASM) in more detail.

4.4.2 Enhanced Haar-based shape model

Different from the conventional ASM, we use 2-D block patterns around each feature point to characterize local textures, which contain richer and more reliable information than 1-D profiles. In our case, the 2-D local texture is modeled by extracting an $M \times M$ block around each feature point from the image, which is subsequently transformed using the Haar transform for robustness and processing efficiency (see Fig. 4.9). A closer examination of the local feature patterns in face images shows that they usually contain relatively simple patterns, which match quite well with the basis images of the 2-D Haar transform. Therefore, it is attractive to exploit them for efficient signal representation. Furthermore, the simplicity of Haar transform supports the requirement of real-time implementation. In the following, we explain the steps needed to obtain a Haar-based feature vector to characterize each feature point in a face shape.

A. Fast computation of Haar features

During the ASM search, the matching of a local point has to be performed on many candidates in order to select the best match. In order to facilitate fast matching, we need a way to derive Haar features for each candidate point in an efficient way. We have noticed that the Haar decomposition mainly involves summations of pixel subblocks (see the Haar basis images shown in Fig. 4.10), which can be efficiently computed by using two auxiliary ‘integral images’ as
introduced in Section 3.5.1. More specifically, for a block with intensity \( B(x, y) \), we can compute its Haar feature \( H(u, v) \) using the expression

\[
H(u, v) = \frac{\sum_{i=0}^{N_B-1} \{ \text{Sgn}(B_i) \sum_{x=0}^{M_{B_i}-1} \sum_{y=0}^{M_{B_i}-1} [B_i(x, y) - \mu_B] \}}{N(u, v) \cdot \sigma_B}. 
\]

(4.5)

Note that the above equation already incorporates the illumination correction of the block by zero mean and unity standard deviation. In Eqn. (4.5), \( B_i \) refers to subblocks corresponding to nonzero coefficient areas in the basis images (see Fig. 4.10). The number of these subblocks is denoted as \( N_B \). The function \( \text{Sgn}(B_i) \) refers to the sign of the coefficient part corresponding to subblock \( B_i \), while the size of subblock \( B_i \) is \( M_{B_i} \times M_{B_i} \). Since coefficient \( H(0, 0) \) only contains the average intensity value of the block, it is zero for all illumination-corrected block images and can be ignored during the matching. For all remaining basis images, the total area of +1 signed subblocks is equal to the area of -1 signed subblocks. The parameter \( N(u, v) \) is the normalization factor for coefficient \( H(u, v) \). Since the summation of subblock data can be efficiently computed using the integral images, we can derive each coefficient by only a few table-lookups of the integral images.

**B. Haar-feature selection**

From the above, the local texture of each feature point can be represented by a transformed data vector

\[
T_i = (H_1(i), H_2(i), ..., H_{c-1}(i))^T, 
\]

(4.6)

where \( H_k(i) \) gives the \( k \)-th Haar coefficient (in zigzag order) of the \( M \times M \) block surrounding the \( i \)-th feature point\(^2\). Only a part, say \( c \) where \( c \ll M \times M \), of the total coefficients are selected, due to the energy-compacted Haar-transform\(^2\). We leave out the DC coefficient \( H_0(i) \).

\(^2\)
description of the local feature structure. For most feature points, less than 4% of the total coefficients are required to retain up to 95% of the signal energy. This significantly increases the computation efficiency of the algorithm and reduces the interference of image noise. Note that $c$ can also be variable to adapt to different feature structures.

C. Haar-feature weighting

The selected coefficients $H_k(i) \ (1 \leq k \leq c - 1)$ may not be of equal importance to characterize the $i$-th feature point. Coefficients with large variances for different face images may contain ‘noisy’ features and should be suppressed. Therefore, a feature-weighting scheme is adopted, which assigns larger weights (higher importance) to those coefficients with less variance over a set of training samples. To this end, Equation (4.6) is rewritten to:

$$T_i = \frac{H_1(i)}{\sigma_1}, \frac{H_2(i)}{\sigma_2}, ..., \frac{H_{c-1}(i)}{\sigma_{c-1}} \quad (4.7)$$

where $\sigma_k \ (1 \leq k \leq c - 1)$ is the standard deviation of $H_k(i)$ over a set of training samples.

The model-fitting procedure of H-ASM is similar to ASM as depicted in Table 4.1, except for the following aspects.

1. In H-ASM, for each feature point, an optimal local position is searched along a trace composed of eight radial lines originating from the current model point. This is different from the single search line used by ASM that is perpendicular to the shape contour.

2. The local search is based on the Euclidean distance between the Haar coefficients (Eqn. (4.7)) of the current block and the mean values of the coefficients from a set of labeled training faces.
Performance evaluation of H-ASM

To demonstrate the performance gain of H-ASM, we use a training set of 200 face images to derive the shape model and a separate test set containing another 200 face images to test the extraction performance. Both data sets are extracted from the FERET face database \cite{54, 55}.

For each test image, we randomly perturb the center position of the initial shape from its ground-truth position by up to 50 pixels. We perform the model fitting using both ASM and H-ASM. The results are depicted in Fig. 4.11. At the left of Fig. 4.11, we show the average point-to-point error between the fitted model and the manually-labeled shape, given different horizontal shape deviations. The flat area on the bottom of each curve indicates a stable (converged) output. At the right of Fig. 4.11, we give the convergence rate of two algorithms with different horizontal shape deviations. The two plots reflect the convergence capability of the algorithm. We can see that H-ASM has a larger capture range (with ±30 pixels in x-deviation) than that of ASM (±15 pixels).

Furthermore, for converged cases, we measure the frequency of successful searches that have point-to-point errors below a given threshold, of which the results are depicted in Fig. 4.12. The H-ASM clearly has an improved extraction accuracy as compared to ASM. For example, up to 70% of the test cases achieve an accuracy of less than 4 pixels, while for ASM, only 40% of the test cases achieve the same accuracy.

\footnote{The face images are scaled such that the distance between the two eyes is roughly 80 pixels, and the shape model consists of 51 feature points as defined in Fig 4.4.}
4.5 Summary and conclusions

In this chapter, we have presented a brief review of existing feature-extraction techniques in literature, which is summarized in Table 4.2. Generally speaking, heuristic techniques such as color-based approaches, are relatively fast and simple to implement. However, they are less reliable with feature diversity and varying imaging conditions. Classification-based techniques such as neural network and Eigenfeature-based extraction, can be very robust for the independent detection of individual feature regions, such as eyes and mouth. Unfortunately, they usually do not provide an accurate description of feature details, such as contours. The use of deformable templates offers a more accurate description of features, but this modeling relies heavily on a good initialization of the model. Another disadvantage is the high computation complexity. By using a structural representation, statistical shape models, such as ASM and AAM, have a better convergence capability. It should be remembered that the model still has to be explicitly initialized close to the ground-truth position to ensure correct convergence.

We have presented in this chapter an improved modeling technique called H-ASM, which uses 2-D local features to guide the extraction process. These features are further processed by a Haar transform for more efficient representation and matching. H-ASM outperforms ASM by a doubled convergence area and a 17% improvement in accuracy (on the average). Unfortunately, erroneous convergence still occurs in a few cases due to inappropriate model initializations or matching failures.

**Figure 4.12:** Cumulative distribution of the pt-pt error of H-ASM as compared to ASM.
4.5. Summary and conclusions

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Robustness</th>
<th>Complexity</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristics-based</td>
<td>unreliable for large feature variances</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>Classification-based</td>
<td>robust with feature variances</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Deformable templates</td>
<td>very limited convergence</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>ASM &amp; AAM, etc</td>
<td>limited convergence</td>
<td>medium</td>
<td>high</td>
</tr>
</tbody>
</table>

Table 4.2: Summary of techniques discussed in this chapter.

It is also worth noticing that from Table 4.2, we can see that it is generally difficult for a single extraction algorithm to achieve robustness and accuracy simultaneously. Robust techniques usually have good convergence capability, but they may not offer sufficient fitting accuracy. On the contrary, techniques offering more accurate results often rely heavily on a good initialization of the model. Furthermore, robust techniques usually require more computation power.

In order to cope with the above problem, in the next chapter, we aim at designing a feature-extraction system that has a large capture range and a high extraction accuracy with comparable computation complexity w.r.t. representative algorithms in literature. Instead of finding a trade-off between the robustness and accuracy, we approach this problem by proposing a cascaded model-based framework. In this framework, a set of component algorithms are cascaded to sequentially perform the extraction. The design of each algorithm has to be tailored to its specific role in the cascade, so that the overall extraction performance is optimized and the computation complexity is manageable.
Cascaded Model-Based Facial Feature Extraction

5.1 Introduction

In the previous chapter, we have evaluated a number of well-known model-based feature-extraction algorithms and shown the dilemma between the robustness and the accuracy in these techniques. In order to achieve both large capture ranges for robustness and a high extraction accuracy at the same time, this chapter presents a facial feature extraction approach using a cascade of model-based algorithms. In this framework, a set of algorithms are cascaded into a chain with increasing model flexibility and extraction accuracy. By using a cascaded approach, the facial feature model can be gradually ‘pulled’ to the optimal position to match the facial features in an input face image, yielding an overall large capture range and high extraction accuracy. In addition, to maximize the efficiency of the cascaded approach, fast techniques can be used first to provide a coarse estimation of facial feature locations, while more elaborate algorithms can be used for refinements in constrained local areas, thereby retaining the overall processing efficiency.

In one implementation of the cascaded framework, we employ three new algorithms that use models with increasing flexibility. The first algorithm concentrates mainly on the feature locations, the second algorithm further incorporates scale and rotation variations and the final algorithm also considers feature deformations. In this incremental modeling, we gradually constrain the parameter search space, leading to both improved reliability and accuracy.
The major contribution in this chapter is the exploration of cascaded modeling for facial feature extraction. This is different from traditional multistage approaches (e.g. [15]) because we propose some basic principles of cascading with the purpose of enhancing the coordination between the constituting algorithms. More specifically, the following differences become apparent.

1. In a multistage approach such as [15], several techniques are combined heuristically without quantitatively addressing their relation in performance. In our approach, each algorithm is tuned according to the output statistics of its previous algorithm, thereby providing a well-coordinated cascade.

2. We present in Section 5.2 and 5.3 a unified framework to address the performance of model-based algorithms. Within this framework, we propose two design principles to guide the algorithm design. In the sequel of the chapter, these principles are followed precisely to ensure the effectiveness and efficiency of the approach.

In addition to the cascading framework, we also propose in this chapter three novel algorithms to constitute a extraction cascade. The key features and advantages of each algorithm are explained in detail in this chapter.

The remainder of the chapter is organized as follows. In Section 5.2, we first define several performance metrics to characterize model-based algorithms. These metrics are extensively used in this chapter to quantitatively measure and compare the algorithm performance. In Section 5.3, we start with the problem statement and present the cascaded feature-extraction framework, where two important principles are proposed to provide guidance for the subsequent algorithm design. Following that, Section 5.4 gives an overview of an implementation of the extraction cascade consisting of three component algorithms, namely, sparse graph search, prediction-based texture fitting and appearance-based feature refinement. From Section 5.5 to Section 5.7, we illustrate in detail these constituting algorithms in the cascade. Section 5.8 summarizes our approach and analyzes the overall performance of the system. A short conclusion of the chapter is given in Section 5.9.

5.2 Performance metrics for model-based algorithms

In this section, we introduce several performance metrics for model-based algorithms for facial feature extraction. These metrics and related concepts are used extensively in the sequel of this chapter to characterize different algorithms.
5.2. Performance metrics for model-based algorithms

5.2.1 Basic concepts

As mentioned in the previous chapter, the key feature points in a face model can be defined by marking important image structures of prominent facial features. Depending on the required granularity of an application, these points may correspond to contours or just centers of facial features (see Fig. 4.4(a)).

Suppose we have defined $k$ key facial feature points of our interest, a feature vector $\theta$ can then be defined to represent the locations of these $k$ points $\Delta x_i$, so that $\theta = (x_1, x_2, ..., x_k, y_1, y_2, ..., y_k)^T$, where $(x_i, y_i)$ is the coordinate of the $i$-th feature point. Let us assume that $\theta_f = (x_{f1}, x_{f2}, ..., x_{fk}, y_{f1}, y_{f2}, ..., y_{fk})^T$ gives the optimal fit to an input image $f$. We define the model deviation as $\delta = \theta - \theta_f = (\Delta x_1, \Delta x_2, ..., \Delta x_k, \Delta y_1, \Delta y_2, ..., \Delta y_k)^T$, where $\Delta x_i = x_i - x_{fi}$ and $\Delta y_i = y_i - y_{fi}$. In practice, $\theta_f$ is usually obtained by the manual labeling of the feature points. In this chapter, we use the average point-to-point distance $\|\delta\|$ as the basic metric to measure the fitting accuracy of $\theta$ (the closeness of $\delta$ to $0$), given by

$$\|\delta\| = \frac{1}{k} \sum_{i=1}^{k} \sqrt{\Delta x_i^2 + \Delta y_i^2}. \quad (5.1)$$

Given the initial feature point locations represented by $\theta_i$ and an input face image $f$ with ground-truth feature-point locations represented by $\theta_f$, we can compute the input deviation $\delta_i = \theta_i - \theta_f$. The aim of a feature-extraction algorithm is to reduce this deviation, such that feature positions can be located as close to the ground-truth as possible. In other words, given input $\delta_i$, we should seek output feature locations $\theta_o$, such that the difference between $\theta_o$ and $\theta_f$ is minimized.

More specifically, we define a model-based feature-extraction algorithm $A$ as $A(\delta_i, f) = \delta_o$, which maps input deviation $\delta_i$ of image $f$ to output deviation $\delta_o$, where $\delta_o = \theta_o - \theta_f$. Note that $f$ is usually taken from a certain face dataset $F$. In general, for a given $\delta_i$, $\delta_o$ can be variable for different input face images $f (f \in F)$. In practice, the mean value $E_{\theta}(\delta_i) = E(\|\delta_o\| | \delta_i)$ is often adopted to measure the performance of an algorithm at given input deviation $\delta_i$. In the following, the $\delta_i - E_{\theta}(\delta_i)$ plot is used as a starting point to derive the performance metrics of a feature-extraction algorithm.

5.2.2 Performance metrics

A feature-extraction algorithm should be robust with large model deviations and at the same time provide accurate extraction results. In the previous chapter, we use the capture range and the extraction accuracy in an informal way to characterize the robustness and accuracy of a feature-extraction algorithm.

---

1. $\theta$ is equivalent to the shape vector $s$ used in the statistical shape model (Section 4.3.2).
2. In some cases, the median value is used instead of the mean value.
algorithm. The capture range measures the convergence capability of an algorithm. If the fitting converges correctly, then we use the extraction accuracy to measure how precise the fitted result is as compared to the ground-truth information.

Although these performance metrics have been used already in literature for evaluating the feature-extraction algorithms, the exact definition of these metrics still remains vague. This can be attributed to the fact that the conceptual definition of the two metrics (capture range and extraction accuracy) are related to each other and can hardly be defined separately. For example, in order to measure the capture range of an algorithm, we first need to define when a model fitting procedure can be called *converged*. This is often determined by the final extraction precision (accuracy). On the other hand, the extraction accuracy has to be measured on only *converged cases*, since the inclusion of unconverged cases causes a large bias in the estimation which is not desirable. As a consequence, in practice, the measure of the capture range and the extraction accuracy is often determined in an ad-hoc way.

In the following, we propose a more precise way to define the performance metrics of a feature-extraction algorithm. We have observed from experiments that a model-based algorithm usually has a relatively stable output deviation $\delta_o$ if the input deviation $\delta_i$ is within an area surrounding the origin. This can be seen from examples as shown in Fig. 4.11. More specifically, if we depict the $\delta_i - E_s(\delta_i)$ plot of the algorithm, this *stable area* usually corresponds to an almost flat area at the bottom of a U-shaped curve. We use this flat area to characterize the *converged* outputs, where the algorithm usually successfully locates the target facial features.

Base on the previous observation, we need first to estimate the level of the flat area in the $\delta_i - E_s(\delta_i)$ plot, which is called the accuracy threshold $S_A$. Based on $S_A$, the capture range and the extraction accuracy can be derived. This procedure is illustrated by the following consecutive three steps:

1. Estimate the accuracy threshold $S_A$;
2. Estimate the capture range $R_c$;
3. Compute the average localization error $X_A$ as an indication of the extraction accuracy.

**Accuracy threshold $S_A$**

The $\delta_i - E_s(\delta_i)$ plot is used as a starting point to evaluate the performance of an extraction algorithm. A threshold $S_A$ for $||\delta_o||$ can be selected to characterize the flat area in the plot, and $\delta_o$ is considered as a *converged output* if and only if $||\delta_o|| < S_A$ (see Fig. 5.1).
5.2. Performance metrics for model-based algorithms

The selection criterion for $S_A$ is ambiguous and can vary with the type of algorithm. Therefore, we introduce here a procedure to select $S_A$ for consistent evaluation of extraction algorithms. Since the flat converged area in the $\delta_i - E_s(\delta_i)$ plot corresponds to a stable level, in practice, we can randomly generate input instances $\delta_i$ within a small perturbation region $R^\Delta$ (e.g. $R^\Delta$ contains input deviation $\delta_i$ such that $\|\delta_i\| < 5$ pixels), and we specify

$$S_A = E(\|\delta_o\| | \delta_i \in R^\Delta) + c \cdot \sigma(\|\delta_o\| | \delta_i \in R^\Delta), \quad (5.2)$$

where $\sigma$ is the standard deviation and $c$ is a constant. $S_A$ is an error parameter that defines the level at which the algorithm $A$ is allowed to deviate from the ground-truth when it is converged. We define set $R^c_A = \{\delta_o | \|\delta_o\| < S_A\}$ as the converged output region.

**Capture range $R^c_A$**

Based on threshold $S_A$, it is straightforward to choose the input area in the $\delta_i - E_s(\delta_i)$ plot corresponding to $E_s(\delta_i) < S_A$ as the capture range. However, in our formulation, instead of using the mean value $E_s$, we measure the capture range in a probabilistic way for a more precise definition. More specifically, the capture range $R^c_A$ of the algorithm can be determined by the plot at the right of Fig. 5.1. $R^c_A$ contains all inputs $\delta_i$, such that their corresponding output $\delta_o$ has a high probability to satisfy $\|\delta_o\| < S_A$, given by

$$P_A(\delta_o \in R^c_A | \delta_i) = P_A(\|\delta_o\| < S_A | \delta_i) > T_A, \quad (5.3)$$

where $T_A$ is a probability threshold. Typical values for $T_A$ are between 95% to 99%. The size of $R^c_A$ can be used as a metric to evaluate the robustness (convergence capability) of an extraction algorithm. A robust algorithm should have the property of a large capture range.
Average localization error $X_A$

$X_A$ can be defined as statistical expectation $\mathcal{E}(\|\delta_o\| | \delta_i \in \mathcal{R}_A^c)$, which is the average localization error for inputs within the capture range. It is often used to express the accuracy of an extraction algorithm.

Having introduced the basic concepts, in the following section, we propose a cascaded framework for model-based algorithms. We show how each component algorithm within the cascade can be designed such that the cascaded algorithm can yield an overall improved performance.

5.3 Problem statement and design considerations

5.3.1 Problem statement

As we have mentioned in the previous chapter, it is generally difficult for a single extraction algorithm to achieve both robustness and high extraction accuracy. For example, the classification-based techniques are fairly robust, but have a very limited accuracy. Deformable templates have a very small capture range due to the limitation of the optimization techniques adopted. ASM/AAM has an improved capture range but is still subject to wrong convergence for large input deviations. Therefore, it is attractive to use a coarse-to-fine procedure to gradually approach an accurate description of facial features. For example, an algorithm with a large capture range can be used first to coarsely locate the right features and a more accurate algorithm can be applied at a later stage to refine the results. In addition to this, we address in this chapter a more important problem which is the optimization of the overall performance of the extraction cascade. In the following, we focus on the optimization of the individual stages and an elegant coupling of them within the cascade to come to an overall performance optimization.

To this end, we propose in this section some basic design principles to guide the construction of a feature-extraction cascade, based on the performance measurements of individual algorithms. In this framework, each constituting algorithm in the cascade receives inputs from the preceding algorithm and refines the extraction. The computation cost of a cascaded algorithm can be kept at a low level, because fast techniques can be used first to provide a rough estimation of feature locations, while more complex algorithms can be used for a refinement in a constrained local area with a reduced search space.

5.3.2 Design considerations

In the following, we discuss several principles that should be observed during the design of such a cascaded extraction algorithm. First, we introduce some
We first define a cascaded algorithm $A$ for feature extraction as consisting of $m \ (m \geq 1)$ atomic algorithms $A_i$, given by

$$A = A_1 \otimes A_2 \otimes \cdots \otimes A_m,$$

(5.4)

where $\otimes$ symbolizes the concatenation of two succeeding algorithms $A_i$ and $A_{i+1}$ such that the output of $A_i$ acts as the input of $A_{i+1}$. Since a single atomic algorithm $A_i$ is a special case of cascaded algorithm when $m = 1$, we use the term *algorithm* in the following to refer to both the atomic algorithm and the cascaded algorithm. The performance of an algorithm can be measured based on the metrics defined in Section 5.2.

In Fig. 5.2, suppose algorithm $A_1$ has a large capture range but low extraction accuracy, while algorithm $A_2$ has a small capture range but high extraction accuracy, the cascaded algorithm $A = A_1 \otimes A_2$ can provide both a large capture range and high extraction accuracy under certain conditions.

### Principle 1  Cascade coupling: $\mathcal{R}^s_{A_i} \subseteq \mathcal{R}^s_{A_{i+1}}$.

According to this principle, the capture range of the subsequent algorithm $A_{i+1}$ should be able to cover the converged output region of the preceding algorithm $A_i$. Therefore, by appending algorithm $A_{i+1}$ to $A_i$, the capture range of the cascaded algorithm can be maximally retained. Note that for the first algorithm $A_1$ in the total cascade, it is required that $\mathcal{R}^I \subseteq \mathcal{R}^s_{A_1}$. Here $\mathcal{R}^I$ is the range for the input model deviation, which is largely determined by the application requirements. Furthermore, it should be noted that the converged output region should be declining and always be a subset of the converged output region of the preceding algorithm, hence, $\mathcal{R}^s_{A_{i+1}} \subseteq \mathcal{R}^s_{A_i}$.
Principle 2 **Cascade effectiveness**: $X_{A_i} > X_{A_{i+1}}$.

According to this principle, the average localization error of the subsequent algorithm $A_{i+1}$ should be smaller than that of the preceding algorithm $A_i$. Principle 2 ensures that by appending algorithm $A_{i+1}$ to $A_i$, the extraction accuracy of the cascaded algorithm is improved.

By complying with the above principles during the cascade design, we can gradually decrease the localization error while maximally retaining the robustness. In addition to the capture range and localization error, efficiency is also a critical factor determining the overall performance of a cascaded algorithm. In order to ensure the overall operational efficiency, each component algorithm in the cascade needs to be designed to maximize certain performance goals while keeping the computation cost as low as possible. In our case, the total computation time on a standard PC (P-IV, 3.0 GHz) should be constrained to 0.5 second to ensure that the whole chain of the face-recognition system is still executing at a feasible speed of 1 full recognition cycle per second (order of magnitude).

In the following, we propose a new cascaded feature-extraction system consisting of three component algorithms with varying capture ranges and extraction accuracy. Although the performance of a single algorithm in the cascade is limited, we aim at optimizing the cascade structure to boost the overall system performance. In the next section, let us first present an overview of the proposed technique.

### 5.4 Overview of cascaded extraction algorithm

We propose three novel algorithms for constituting an extraction cascade, which are depending on each other and jointly perform the extraction. In Fig. 5.3, we give an overview of the extraction framework. In this framework, each algorithm uses a more flexible model than the previous algorithm incorporating more parameters. The aim is to obtain a more accurate description of facial features by each stage. The three constituting algorithms are briefly summarized as follows.

1. **Sparse Graph Search (SGS)**. SGS is the first algorithm in the cascaded framework, and aims at finding the *center locations* of six facial features, namely, eyes (left and right), eyebrows (left and right), nose and mouth. It uses a graph model consisting of six nodes corresponding to these facial features. A fast search algorithm is applied to locate a coarse feature structure. To maximize the robustness of the search algorithm, the following techniques are employed:

   a. **Exhaustive pattern search**. The locations of each facial feature (instead of specific feature points) are first estimated using exhaus-
5.4. Overview of cascaded extraction algorithm

Figure 5.3: Overview of our proposed feature-extraction system.

ative pattern search within pre-defined search ranges (e.g. based on the face-detector outputs). The exhaustive search ensures that the global optimum for the best-matching feature can be reached to avoid incorrect local convergence.

(b) Structural topology regulation. A number of structural constraints are used to ensure that the located features constitute a plausible face. The constraints are based on geometric relations between facial features, which help to improve the reliability of the search algorithm.

The SGS algorithm uses a coarse model representation with the feature-location information only, where the extraction accuracy is traded-off against large capture ranges and reliability.

2. Component-based Texture Fitting (CTF). In CTF, in addition to location, the shape of each facial feature, e.g. an eye or a mouth, is described by a point-based model with additional scale and rotation parameters. Supplementary to this shape description, we also define texture descriptions. In order to find the best shape that fits to the feature in an input image, we use a direct prediction scheme by training a prediction function, such that given an initial model estimation, the correct shape parameters including location, scale and rotation, can be directly obtained based on the associated texture description. We propose two non-linear schemes for estimating the prediction function based on neural networks and support vector regression, which have shown to significantly outperform the linear estimation approach such as the one used in AAM [10].

3. Component-based Direct Fitting (CDF). In addition to the geometric parameters used by CTF (location, scale and rotation), the CDF incorporates an additional deformable parameter to the feature model
to account for more feature variations. This offers more flexibility for the model to adapt to individual feature instances. In order to find the optimal parameters for the model to fit to an input image, a generic optimization is used in CDF. Generally, a generic optimization scheme (e.g. Simplex algorithm as used in our implementation) involves high computation cost and tends to converge to local minima. However, when the initial position of the facial feature is relatively close to the optimal position, which can be ensured by the previous algorithms in the cascade, the optimization can further improve the extraction accuracy with limited computation cost.

In the following, we illustrate each algorithm in more detail and also quantitatively analyze their extraction performance.

### 5.5 Sparse graph search (SGS)

The aim of the sparse graph search is to estimate facial feature locations at a coarse granularity. More specifically, we define six facial features corresponding to six prominent feature regions, i.e. eyes (left and right), eyebrows (left and right), nose and mouth. Given a coarsely estimated face region, SGS aims at finding the rough locations of these features, which can be used as inputs for the subsequent algorithms working at finer granularities.

In order to achieve a large capture range, a straightforward solution is to exhaustively search for all possible facial feature locations within the face region. The full-search pattern-matching strategy mentioned in Chapter 4 can be used, where a feature detector can be trained (e.g. using neural networks) to enumerate all candidate windows within the face region and select promising candidates for each feature. Since usually multiple positive outputs are generated for each feature, an ordering of the candidates can be performed based on the output confidence and the best-ranked feature is usually selected. However, since the detector normally does not guarantee the correctness of the
output ranking order, it is quite common that the ground-truth feature location is not ranked first among all the candidates.

An effective way to cope with this problem is to incorporate a-priori knowledge about the relations between the facial features. A good set of facial features should not only have high individual feature confidence, but also constitute a geometrically face-like shape. By imposing these constraints to the facial features, more reliable localization can be expected.

In this section, we present a new graph model for facial feature localization. The model captures both individual feature properties (represented by nodes) and their spatial relationship (represented by edges). We also propose an efficient model-fitting algorithm to reliably locate features. In the following, we illustrate in more detail the modeling technique in SGS. As shown in Fig. 5.4, we first introduce the model definition and two representation maps associated with the model. Following that, we introduce a function which defines the matching cost of the facial features with certain geometric constraints. Afterwards, we present an algorithm to optimize the cost in an efficient way.

### 5.5.1 Sparse graph model

In Fig. 5.5, the facial features are modeled by a graph $G_f = (V_f, E_f)$, where $V_f$ is the node set and $E_f$ is the edge set. We denote $|V_f| = N$ and $|E_f| = M$. Each node $v_i = (x_i, y_i)$ indicates the center position of the feature, which is also characterized by a rectangular image region centered at $(x_i, y_i)$ (see Fig. 5.5(a)). Each $e_i$ in $E_f$ is a vector connecting two nodes in $V_f$. The edges define the neighboring relations between important feature components, such as eyes/nose and nose/mouth (see Fig. 5.5(b)). They are used in the sequel for the efficient node searching.

Furthermore, we use the following two image representations for characterizing each node: the normalized intensity map and the edge maps. This is based on the observation that each facial feature has its unique intensity and edge patterns that differentiate it from other areas of a face.
Normalized Intensity Map (NIM). The NIM of a node $v_i$ is defined as a block of normalized image intensities $S^{(i)}(x, y)$ of size $w \times h$ centered at $(x_i, y_i)$, which is given by

$$S^{(i)}(x, y) = (I^{(i)}(x, y) - \mu_{I^{(i)}}) / \sigma_{I^{(i)}},$$

(5.5)

where $I^{(i)}(x, y)$ is the raw image intensity and $\mu_{I^{(i)}}$ and $\sigma_{I^{(i)}}$ are the mean and standard deviation of $I^{(i)}(x, y)$, respectively, all measured in the window $w \times h$. For each feature, we compute the average NIM of a set of samples with ground-truth node locations and store it as a reference pattern for the $i$-th node, denoted as $\hat{S}^{(i)}(x, y)$.

Edge maps. In Chapter 2, we have presented the definition of an EOM and EIM of an image region with intensity $I^{(i)}(x, y)$. Based on Eqn. (2.4) and Eqn. (2.5) in Chapter 2, we can obtain a pair of edge maps: edge-orientation map $E^{(i)}_{o}(x, y)$ and edge-intensity map $E^{(i)}_{s}(x, y)$. For each feature, we compute the average edge maps of a set of images and store it as a reference pattern, denoted as $\langle \hat{E}^{(i)}_{o}(x, y), \hat{E}^{(i)}_{s}(x, y) \rangle$.

In order to fit a graph model to the facial features in an input image, we propose in the next subsection a cost function defining the quality of the fitting.

5.5.2 Cost-function definition for fitting

The quality of fit of a graph model to an input image is measured based on the fit of individual facial features (nodes). More specifically, we define the quality of fit of a candidate node by matching its associated intensity map and edge maps with the average templates, which is defined\(^3\) as

$$C_{v}(v_i) = C_{n}(v_i) + C_{e}(v_i).$$

(5.6)

Function $C_{n}$ is defined based on the correlation between the intensity map of a candidate node and an average intensity map, and $C_{e}$ defines the difference between the edge maps of a candidate node and the average edge maps, as given by Eqn. (2.6) in Chapter 2.

Given a model with node set $V_f$, we can evaluate the overall quality-of-fit of the model to an input image as the sum of $C_{v} (v_i)$ ($v_i \in V_f$). In addition, these nodes should constitute a valid face shape. More specifically, we use a boolean function $B(V_f)$ for this evaluation purpose. If $B(V_f) = true$, $V_f$ forms a plausible face shape and otherwise $B(V_f) = false$. We derive $B(V_f)$ from a statistical analysis of a set of samples, which is further illustrated in Section 5.5.4.

\(^3\)In other situations, the designer may use a weighted sum instead of the sum.
5.5. Sparse graph search (SGS)

Algorithm Model fitting in SGS
Input: Input image $f$, search range $R_i$ for each facial feature.
Output: Best node set $\langle v^*_1, v^*_2, \ldots, v^*_N \rangle$.

1. for $i = 1$ to $N$
2. for all position $v_i = (x, y)$ within $R_i$, compute $C_v(v_i)$, yielding a cost map as shown in Fig. 5.6.
3. Select $K$ candidate nodes with the smallest local minima in the cost map and store them in candidate set $D(i)$.
4. Search for $\langle v^*_1, v^*_2, \ldots, v^*_N \rangle$ ($v^*_i \in D(i)$), such that

$$\langle v^*_1, v^*_2, \ldots, v^*_N \rangle = \text{argmin}_{i=1}^{N} C_v(v_i) \text{ and } B(\langle v^*_1, v^*_2, \ldots, v^*_N \rangle) = \text{true}. \tag{5.8}$$

5. return $\langle v^*_1, v^*_2, \ldots, v^*_N \rangle$.

Table 5.1: Model-fitting algorithm of SGS.

To summarize, the cost function for evaluating the quality-of-fit of a model can be given as

$$C_t(V_f) = \sum_{i=1}^{N} C_v(v_i), \text{ subject to } B(V_f) = \text{true}. \tag{5.7}$$

In this way, we seek a graph with minimized combined node cost, under the constraints that these nodes form a plausible face shape. In the following, we present a search strategy to find such a graph.

5.5.3 Model-fitting strategy

Table 5.1 gives the algorithm to fit the model to an input face image. For given face image $f$, we measure the quality-of-fit (Eqn. (5.6)) for all node candidates in $N$ pre-defined search regions (see Fig. 5.6). In practice, these search regions are usually defined heuristically. For example, the search range for the left eye can be a rectangular region on the top-left part of the facial region. As a result, each location in these search regions are assigned a cost value $C_v(v_i)$. This can be depicted as a matching map as shown in Fig. 5.6, where darker pixels indicate better matching results. The $K$ locations with the smallest local minima (valleys) in each matching map are selected as candidate nodes. In algorithm Line 4, the best set of $N$ nodes from these candidate sets are selected to minimize Eqn. (5.7). This is achieved by using an efficient selection algorithm illustrated in the next subsection.
Figure 5.6: Illustration of the node matching map. At the top left, the search region for the input face is indicated. Figures (a)-(f) are the visualizations of the six maps as the outcome of Eqn. (5.6). The bottom-left picture shows the outcome of the model-fitting algorithm from Table 5.1.

5.5.4 Efficient node selection

A straightforward way to define the shape-evaluation function $B$ can be based on a statistical analysis of the graph, as used in the statistical shape model in Chapter 4. The shape evaluation can then be performed by checking whether the shape deformation (e.g. deformation parameter $b$ in Section 4.3.2) has exceeded certain bounds. Given $K$ candidate nodes for each facial feature, one possible implementation of Line 4 in the model-fitting algorithm (Table 5.1) is to check all node combinations and choose the best combination which also satisfies the shape constraints. In this case, a total of $K^N$ combinations need to be evaluated, which easily leads to a combinatorial explosion even when $K$ or $N$ is only moderately large. For example, in case that $K = 10$ and $N = 6$, there are $10^6$ enumerations, which can be too costly to be applied in practice. In the following, we propose an efficient pruning-based technique to search for the best combination with a significantly reduced number of enumerations.

Basic principle for efficient node selection

The graph model as shown in Fig. 5.5(b) forms a tree structure. If a branch of a tree graph does not satisfy a local shape constraint, then the complete
5.5. Sparse graph search (SGS)

Algorithm \textit{LocateValidSubtrees}(v)

\textbf{Input:} A tree node \(v\), candidate node set \(D(v)\)

\textbf{Output:} Valid subtree set rooted by \(v\): \(L(v)\)

1. \(L(v) \leftarrow \emptyset\)
2. \(N_c \leftarrow \text{Number of child nodes}(v)\)
3. if \(N_c = 0\) (Leaf node)
4. \quad for all \(w \in D(v)\)
5. \quad \(L(v) \leftarrow L(v) \cup w\)
6. \quad return \(L(v)\).
7. else
8. \quad for \(i = 1\) to \(N_c\)
9. \quad \(L(\text{Child}(v, i)) \leftarrow \text{LocateValidSubtrees}(\text{Child}(v, i))\)
10. \quad for all \(t_j \in L(\text{Child}(v, j)), 1 \leq j \leq N_c\) and \(w \in D(v)\)
11. \quad \quad if \(B((t_1, t_2, ..., t_{N_c}, w)) = \text{true}\)
12. \quad \quad \(L(v) \leftarrow L(v) \cup \{t_1, t_2, ..., t_{N_c}, w\}\)
13. \quad return \(L(v)\)

\textbf{Table 5.2:} Search algorithm to locate valid (sub)tree structures. In the algorithm description, we use \(\text{Child}(v, j)\) to represent the \(j\)-th child node of \(v\), and \(D(v)\) contains all candidate locations for \(v\). \(L(v)\) contains all valid subtrees rooted by \(v\).

A tree structure is automatically rejected without having to perform further evaluations. In this way, the number of enumerations can be significantly reduced, yielding a more efficient implementation. This is explained by the following principle, where we use the binary classification function \(B_s\) for a subtree.

\textbf{Principle 3} Given a node set \(V\), a necessary condition for \(B(V) = \text{true}\) is that for any sub-node-set \(V_s (V_s \subseteq V)\), its corresponding evaluation function yields positive output \((B_s(V_s) = \text{true})\).

In Table 5.2, we present a pruning-based search algorithm. For given root node \(v\), the search algorithm recursively locates valid subtrees rooted by the child nodes of \(v\) (Line 9 to 10). Note that we do not have to evaluate trees with invalid subtrees. For example, in Fig. 5.7(a), \(t_1\) and \(t_2\) are valid candidate subtrees of node \(v\). Afterwards, the algorithm needs to evaluate only the validity of the combination of subtrees \((t_1, t_2)\) and each candidate root node \(w\) (Line 11 to 13). In the following, we define the binary evaluation function \(B((t_1, t_2, ..., t_{N_c}, w))\) in Line 12 (The trees \(t_1, t_2, ..., t_{N_c}\) are subtrees rooted by candidate node \(w\)) in a probabilistic framework, which can be further
decomposed into a series of partial evaluations. We see that this strategy further improves the search efficiency of the algorithm.

**Binary shape-evaluation function**

We define shape-evaluation function $B(⟨t_1, t_2, ..., t_{N_c}, w⟩)$ based on the probability that $⟨t_1, t_2, ..., t_{N_c}, w⟩$ constitutes a plausible face shape, denoted as $P(t_1, t_2, ..., t_{N_c}, w)$. More specifically,

$$B(⟨t_1, t_2, ..., t_{N_c}, w⟩) = \begin{cases} 
true, & \text{if } P(t_1, t_2, ..., t_{N_c}, w) > T_p; \\
false, & \text{otherwise.} 
\end{cases} \quad (5.9)$$

In order to find out whether the set $⟨t_1, t_2, ..., t_{N_c}, w⟩$ represents a meaningful face, we split $P(t_1, t_2, ..., t_{N_c}, w)$ into a series of partial evaluations. More specifically, this decomposition is developed as the chain rule for probabilities, so that we write

$$P(t_1, t_2, ..., t_{N_c}, w) = P(t_1, w)P(t_2|t_1, w) \cdots P(t_{N_c}|t_1, t_2, ..., t_{N_c-1}, w). \quad (5.10)$$

In order to come to an efficient computation for Eqn. (5.10), we propose a simplified pruning-rule as follows: if any intermediate probability evaluation $P(t_i|t_1, ..., t_{i-1}, w) < T_p$ ($T_p$ is a threshold), then the tree set $⟨t_1, t_2, ..., t_{N_c}, w⟩$ is automatically rejected without further evaluations. This pruning-rule can be applied to the chain rule to evaluate the probability of the meaningfulness of the whole tree. This is now clarified by an example, which is shown in Fig. 5.7.

**Example of partial shape evaluation.** In Fig. 5.7, we have two subtrees $t_1$ and $t_2$ and a root node $w$. Each subtree $t_i$ is rooted by node $v_i$. An edge
within a (sub)tree is defined as a vector between two node points of the tree. In the following, we use edge \( e_i = w - v_i \), which connects node \( w \) to the subtree root \( v_i \). The length of edge \( e_i \) is equal to the norm of the edge and symbolized by parameter \( e_i \). Correspondingly, we define the direction of the edge as the angle of the vector \( e_i \), which can be computed by the arctangent function of the ratio between the \( y \)-component and the \( x \)-component of the vector. We denote this with \( \theta_i \).

Let us now apply the chain rule of Eqn. (5.10), so that we have \( P(t_1, t_2, w) = P(t_1, w)P(t_2|t_1, w) \). Instead of computing these probabilities, we approximate them by a simplified decision rule, which is indicated in the following equation

\[
P(t_1, w) > T_p \approx (P_r(e_1/e t_1) > T_p) \land (P_r(\theta_1) > T_p),
\]

\[
P(t_2|t_1, w) > T_p \approx (P_r(e_2/e t_2) > T_p) \land (P_r(\theta_2) > T_p)
\]

\[
\land (P_r(e_2/e_1) > T_p) \land (P_r(\theta_{12}) > T_p).
\]

(5.11)

It should be noted that all probability functions \( P_r \) are different functions for the variables indicated. Also note that the influence of \( w \) at the left is covered within \( e_1 \) and \( e_2 \), as these edges start from point \( w \). The parameter \( e_t \) is the norm of an arbitrarily chosen but pre-defined edge in subtree \( t_i \). This norm parameter acts as a seed value for the scale of the subtree \( t_i \). Parameter \( \theta_{12} \) is the angle between \( e_1 \) and \( e_2 \).

The distributions of \( \theta_i, \theta_{ij} \) and \( e_i/e_j \) can be modeled by standard Gaussians and the corresponding parameters (mean and variance) can be estimated from a set of sample graphs with ground-truth information\(^4\).

The above example can be easily extended to more general cases where the root node has \( N_c \) \((N_c > 2)\) child nodes. If one of the intermediate probability approximations in Eqn. (5.11) yields a very low value, then the full shape \( \langle t_1, t_2, ..., t_{N_c}, w \rangle \) is discarded as an invalid shape configuration.

Let us now return to the implementation of the model-fitting algorithm of SGS as described in Table 5.1. The search algorithm in the table on Line 4 can now be inserted by an efficient implementation as follows:

Let \( L \leftarrow \text{LocateValidSubtrees}(v_N) \).

for all node combination \( \langle v_1, v_2, ..., v_N \rangle \in L \)

Select \( \langle v_1^*, v_2^*, ..., v_N^* \rangle \), such that \( \langle v_1^*, v_2^*, ..., v_N^* \rangle = \arg\min_{\langle v_1, ..., v_N \rangle} \sum_{i=1}^{N} C_v(v_i) \).

In the above, \( v_N \) is the root node of the face graph. In the above, we apply the search algorithm as specified in Table 5.2, which is able to efficiently find

\(^4\)We also experimented with non-parameterized distribution estimation such as using histograms. The experimental results have shown that the two techniques (histogram estimation and Gaussian estimation) yield similar performance.
all valid trees, as it discards all invalid subtrees during the traverse of the
tree. This completes our illustration of the model-fitting strategy for the SGS
algorithm.

5.5.5 Implementation and evaluation of SGS

We have implemented the SGS model (Section 5.5.1) and the model-fitting
algorithm as depicted in Table 5.1 with the previously-discussed algorithm re-
alizations from Section 5.5.3 and Section 5.5.4. We discuss two implementation
aspects here.

1. Subsampling of the input. Since SGS aims at obtaining a coarse local-
ization of facial features, we use a low-resolution representation of the
face region. This gives a higher implementation efficiency and robust-
ness to image noise. In our experiments, we work on face regions scaled
to approximately 69 × 69 pixels.

2. Multiscale templates. To compensate for scale variations due to inac-
curate localization of initial face regions and/or individual feature vari-
atations, we use a series of reference templates with various scales. The
matching selects the template with the minimum cost as computed by
Eqn. (5.6).

In the sequel, we analyze the performance of the algorithm in terms of
convergence capability, execution efficiency, and extraction accuracy. A subset
of the FERET database [55][54] is used, which consists of frontal-view faces
with large feature variances (different races, facial expressions and lighting
conditions). We use a sample set of 409 face images and a test set of another
409 images having no overlap with the sample set. Based on the sample set,
the reference templates as introduced in Section 5.5.1 were derived and the
shape statistics (e.g. the distributions of $\theta_i$, $\theta_{ij}$ and $e_i/e_j$ in Eqn. (5.11)) were
measured.

Convergence of the algorithm. Since the SGS uses exhaustive search for
all feature candidates, the convergence of the algorithm depends mainly on the
search region adopted as shown in Fig. 5.6. For most applications it suffices,
to heuristically define these regions based on the face-detection outputs. In
this case, the capture range of the SGS algorithm is equivalent to the search
region.

Execution efficiency. To evaluate the efficiency gain by using the graph
search as described in Section 5.5.4, we present in Table 5.3 the average statis-
tics of the computation cost involved for the shape evaluation. For these tests,
we have selected $K = 8$ candidates for each node (facial feature). An average
5.5. Sparse graph search (SGS)

<table>
<thead>
<tr>
<th>Root of the (sub)tree</th>
<th>Number of local eval.</th>
<th>Number of valid (sub)trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>/</td>
<td>8</td>
</tr>
<tr>
<td>$v_3$</td>
<td>/</td>
<td>8</td>
</tr>
<tr>
<td>$v_2$</td>
<td>64</td>
<td>17</td>
</tr>
<tr>
<td>$v_4$</td>
<td>64</td>
<td>22</td>
</tr>
<tr>
<td>$v_5$</td>
<td>576</td>
<td>129</td>
</tr>
<tr>
<td>$v_6$</td>
<td>547</td>
<td>256</td>
</tr>
<tr>
<td>Total</td>
<td>1251</td>
<td>256 valid graphs</td>
</tr>
</tbody>
</table>

Table 5.3: Search-space reduction: intermediate results from the model-fitting algorithm.

![Frequency vs. Relative Pt-Pt error](image)

Figure 5.8: Cumulative distribution of the relative pt-pt error of the SGS algorithm.

of 1,251 local evaluations are performed to obtain 256 valid face graphs. This reduces the exhaustive enumerations ($8^6 = 262,144$) by 99.5%.

**Extraction accuracy.** To give a consistent evaluation for all the algorithms presented in this chapter, we use the average point-to-point distance $\|\delta\|$ (as introduced in Section 5.2) to measure the fitting accuracy of the algorithm. Note that SGS uses a coarse model for the facial features. In order to use a compatible feature model with other algorithms, we apply the mean feature shapes with the estimated location and scale by SGS to synthesize a full 51-point face shape. This full shape is the basis to derive the pt-pt (point-to-point) error as defined in Section 5.2.

To eliminate the scaling interference in the evaluation within this chapter,
Figure 5.9: Parameter-estimation accuracy of SGS shown by the distributions of the horizontal position and scale deviations. The meanings of $v_3$, $v_4$, $v_5$ and $v_6$ are given in Fig. 5.5.

we employ a relative point-to-point error $e_r$ by normalizing $\|\delta\|$ with the face scale, which is represented by the absolute distance between the centers of the two eyes. This is formulated by:

$$e_r = \frac{\|\delta\|}{Distance\_Between\_Eyes},$$

(5.12)
5.5. Sparse graph search (SGS) 107

For all the test samples, we have obtained the cumulative distribution of the resulting relative error $e_r$, which is plotted in Fig. 5.8. The statistical expectation of $e_r$ is 0.066. Considering the average scale ($69 \times 69$) of the face region, the average distance between the eyes is around 30 pixels. The relative error $e_r$ then corresponds to approximately an absolute pt-pt error of two pixels.

We have also measured the estimation accuracy for individual facial features and have depicted the results in Fig. 5.9. In Fig. 5.9, we plot the distribution of two estimated feature parameters, i.e. the horizontal and scale deviations of features $v_3, v_4, v_5$ and $v_6$. The distributions of $v_1$ and $v_2$ are similar to those of $v_3$ and $v_4$, therefore they are skipped here. The horizontal deviation is measured by the relative error $x_d = (x_e - x_t)/\text{Feature.Width}$, where $x_e$ is the estimated x-coordinate of the facial feature and $x_t$ is the ground-truth value (derived from manually-labeled images). It can be seen that for most facial features, the x-center estimation has a deviation range within $[-0.3, 0.3]$ and the scale estimation has a deviation range within the interval $[-0.6, 0.6]$. The parameter-estimation accuracy is very important for the algorithm design in the upcoming sections, because it represents the output statistics of the first stage of the extraction cascade and directly influences the design of the succeeding stages in the cascade.

Fig. 5.10 shows some typical extraction results, which show that the SGS algorithm yields good results when dealing with face images of varying appearances.

**Limitations of the SGS algorithm.** SGS is able to locate facial features with guaranteed global optimum. However, the extraction accuracy of the algorithm (see Fig. 5.8 and Fig. 5.9) is still restricted by the following factors.

1. The algorithm represents facial features at a coarse granularity and fails to capture detailed structures for individual features, e.g. feature contours, which are required in applications demanding more accurate fea-
ture descriptions.

2. The node cost in SGS is based on the matching with averaged templates. For face images with varying appearances, this may not give an accurate estimate.

5.5.6 Short summary of SGS

In this section, we have proposed an efficient algorithm to coarsely locate facial feature. The algorithm searches for the best feature candidates with structural constraints. We have elucidated an efficient search strategy and demonstrated its effectiveness in our experiments. The experiments have also shown that the proposed model is robust w.r.t. varying feature appearances. However, in SGS, only locations of facial features are concerned during the extraction process and the achieved accuracy is limited. In the following, we propose a more flexible model for representing facial features, which incorporates additional parameters such as feature scale and rotation. We show that this model effectively improves the extraction results when appended to the SGS algorithm in the extraction cascade (see Fig. 5.3).

5.6 Component-based texture fitting (CTF)

In this section, we propose a more flexible feature model (CTF), which is used as the second step in the cascaded extraction (refer to Fig. 5.3). Each feature component defined in the previous section, e.g. an eye or mouth, is now represented by a shape parameterized by location, scale and rotation. The key objective of CTF is to find the optimal shape parameter based on the current texture enclosed by the shape. This is accomplished by using direct parameter prediction based on a set of training samples. To this end, we propose to use three techniques to approximate the prediction function, which are: (1) linear prediction, (2) neural-network regression and (3) Support Vector Regression (SVR). We demonstrate that the nonlinear approaches (neural network and SVR) clearly outperform the linear approach in extraction accuracy.

Fig. 5.11 portrays the structure of the algorithm CTF, which is summarized below with references to the corresponding subsections. Section 5.6.1 presents the definition of the feature model, which is followed by the definition of a cost function based on texture differences in Section 5.6.2. Furthermore, we propose in Section 5.6.3 to minimize the cost function by using direct parameter estimation. This can be solved by three training-based schemes, which are illustrated and compared to each other in Section 5.6.4. The experimental evaluation is presented in Section 5.6.5.
5.6. Component-based texture fitting (CTF)

5.6.1 Component-based feature model

In this subsection, we model each facial feature by a parameterized shape and its associated texture. Since the previous algorithm (SGS) already results in a close estimation of the facial feature positions, it suffices to utilize only local information to refine the extraction results for each feature. Moreover, the CTF model considers the scale and rotation of the shape in addition to its center locations, thereby providing a more accurate description for each feature.

Shape model

For each of the six facial features defined earlier in the previous section, we represent each feature shape by a vector \( s_j \) \((1 \leq j \leq 6)\) containing the concatenated coordinates of a set of key points marking the feature contour. This is similar to the shape definition in the statistical shape model (see Section 4.3.2). For brevity, in the following we omit the subscript \( j \) so that the results apply for any facial feature. Suppose \( \bar{s} \) is a mean shape vector for one of the features, e.g. the left eye, we can approximate the shape of any left eye \( s \) by the following

\[
s \approx T_{x,y,r,\phi}(\bar{s}).
\]  

(5.13)

In the above, \( T \) is a function that transforms the mean shape \( \bar{s} \) by a translation \( (x, y) \), a scaling \( r \) and a rotation \( \phi \). We denote \( p = (x, y, r, \phi)^T \) as a parameter vector for the shape model. Given \( p \) and pre-calculated \( \bar{s} \), any feature shape can be approximated by the model from Eqn. (5.13). For brevity, we use \( s_p \) to represent the shape instance that results from the specific model settings in parameter vector \( p \).
Texture model

For any given modeled shape \( s_p \) and a given face image, we now define an enclosed texture region covered by this shape. As shown in Fig. 5.12(a), we first build an extended shape by extending each feature point in \( s_p \) to an outwards-bounded neighboring point in the direction perpendicular to the contour curvature. The extended point is called the outer feature point of the original feature point and the extension range is proportional to the size of the facial feature. The outer feature points encapsulate a larger texture region and incorporate richer information to describe the feature (both the enclosed region by the original feature points and its surrounding texture). The texture region is then geometrically and photometrically normalized. For geometric normalization, we construct a triangle mesh based on the extended shape (see Fig. 5.12(a)), which is used to warp the texture to a standard frame by piecewise affine warping. The intensities of the resulting shape-free texture patch are then normalized to zero mean and unity standard deviation. Finally, the texture samples within this region are raster-scanned and ordered into one vector \( g_p \). Note that although the illustration example in Fig. 5.12(a) depicts a shape that fits exactly to the eye in the image, the derivation method above for texture \( g_p \) can be based on any shape instance which may be overlayed to any part of the image.

By applying the above texture-construction procedure to a set of annotated training images, we derive an average texture vector \( \bar{g} \). This is used in the next section as the basic reference for the texture fitting.

Having defined the shape and texture models used in CTF, we present in the following the cost function that defines the fitting error of the model to the true feature locations in an input face image. The aim of the model fitting is then to find the best model parameters to minimize this fitting error.

5.6.2 Cost-function definition for fitting

Given model parameter \( p \) and an input face image, we can derive its corresponding shape instance \( s_p \) and texture instance \( g_p \). A cost function for the fitting error is defined as

\[
C(p) = \|g_p - \bar{g}\|, \tag{5.14}
\]

which is the distance between the current texture model \( g_p \) and the mean texture \( \bar{g} \). We aim at finding the optimal \( p \), such that \( C(p) \) is minimized. In the remainder of this section, we approach this optimization problem by using direct parameter prediction.
5.6. Component-based texture fitting (CTF)

5.6.3 Model fitting by direct parameter prediction

Given an input face image \( f \) and an initial input shape instance \( s_{p_i} \) with model parameter vector \( p_i \), we seek shape vector \( s_{p_o} \), which minimizes the cost function as defined by Eqn. (5.14). The subscripts \( i \) and \( o \) refer to input and output, respectively. This minimization can be approached, for example, by adopting a generic optimization algorithm such as a gradient-descent technique [57]. However, the crude use of a generic optimization technique tends to lead to erroneous local minima and/or it can take too much computation cost. Considering the specific properties of facial feature appearances, we exploit here statistical relations between the parameter vector \( p \) and the facial feature texture.

More specifically, we seek a function \( \Phi \) that can directly predict the output shape parameter vector given the current shape and texture information only. The function takes as input the model-texture residual defined by \( \Delta g_i = g_{p_i} - \bar{g} \), and it generates output shape-parameter displacement \( \Delta p_i = p_i - p_o \) based on the vector \( \Delta g_i \). Hence, this means that

\[
\Phi(\Delta g_i) = \Delta p_i. \tag{5.15}
\]

Based on this, we can use the following procedure to fit an initial model to the input image, which is also visualized in Fig. 5.13. First, from the initial model parameter vector \( p_i \), we can derive the corresponding texture \( g_{p_i} \). By subtracting it by the average texture \( \bar{g} \), we derive \( \Delta g_i \). By applying Eqn. (5.15), a parameter correction \( \Delta p_i \) can be obtained. We then get the output shape parameter \( p_o \) from \( p_o = p_i - \Delta p_i \).

The key question now is the design of prediction function \( \Phi \), which is addressed in the next subsection.

Figure 5.12: Shape and texture models in CTF. (a): Extension of feature points to the outside regions and triangular mesh in the model. (b): Similar mesh construction for other facial features, where the hatched areas indicate the inner texture areas and the non-hatched areas indicate the outer extension parts.
5.6.4 Prediction functions

Since facial features of different people share certain common characteristics, intuitively speaking, there exist some intrinsic relations between the shape parameter deviation $\Delta p_i$ and the model-texture residual $\Delta g_i$. However, facial features also exhibit varying appearance under different conditions. Therefore, in order to more precisely capture the relation between $\Delta p_i$ and $\Delta g_i$, function $\Phi$ should be robust enough to accommodate for this feature variance. In order to capture this statistical property, we perform an offline estimation of function $\Phi$ based on a set of training samples. The learned function is used later to predict new instances.

Prediction functions are usually derived by applying regression techniques. In the following, three approaches are used to estimate $\Phi$: (1) linear estimation, (2) neural-network-based estimation and (3) SVR-based estimation. The first approach (linear estimation) is similar to the approach adopted by AAM, where a linear relation is assumed between the two variables $\Delta p_i$ and $\Delta g_i$. The latter two schemes, on the other hand, provide more flexible nonlinear function estimation.

For training-based estimation of $\Phi$, we first need to collect a set of representative training samples. Suppose we have a set of training face images and their ground-truth shape parameters. For each training image, we randomly displace (perturb) each vector element of the optimal $p_o$ (the ground-truth vector) to obtain a displaced shape $s_{p_i}$ and the corresponding $g_{p_i}$. We then use $(\Delta g_{p_i}, \Delta p_i)$ as a training sample for the function estimation.

---

$\Phi(\Delta g_{p_i}, \Delta p_i) = p_i - p_o$

$= (-0.33, -0.30, 0.05, -0.08)^T$

**Figure 5.13:** Illustration of the model-fitting procedure. The translation parameter $(x, y)$ takes relative coordinates to accommodate for various feature scales.
5.6. Component-based texture fitting (CTF)

For linear prediction of $\Phi$, it is assumed that a linear relationship exists between the model-texture residual $\Delta g_p$ and the shape-parameter displacement $\Delta p_i$. Here we leave out the subscript $i$ for brevity. The objective is then to obtain a prediction matrix $R$, such that

$$
\Delta p = R(\Delta g_p).
$$

(5.16)

Given the training set, we can solve Eqn. (5.16) by applying multivariate linear regression, which can achieve a minimum fitting error in the least-squares sense. A similar linear estimation approach has been adopted by AAM [10].

Linear estimation provides the simplest and most direct way for estimating $\Phi$. For a new face image, by applying Eqn. (5.16), we can quickly obtain the parameter deviation. Although it is fast, linear estimation only offers limited estimation accuracy. This is quantitatively analyzed in Section 5.6.5.

In order to achieve more flexible and robust parameter prediction, we resort to two nonlinear regression techniques to implicitly derive function $\Phi$, namely, neural-network regression and SVR.

(2) Neural-network-based prediction

In Appendix C, we have introduced the neural-network structure for classification problems (e.g. face detection in Chapter 2). In addition to classification, neural networks can also be applied to regression problems, where the target outputs are continuous variables instead of class labels.
In our case, we use a multi-layer fully-connected feedforward network with one hidden layer containing \( L \) nodes, as depicted in Fig. 5.14. The network receives \( K \) elements of the texture residual \( \mu = \Delta \mathbf{g} \) as inputs, and outputs the parameter deviation \( \tau = \Delta \mathbf{p} \). As mentioned in Appendix C, a nonlinear mapping can be constructed between the input and output vectors. Note that \( \tau_1...\tau_4 \) give normalized outputs between 0 and 1, and a linear scaling operation is required to convert them to (or from) the required range for each parameter.

The neural-network approximation of function \( \Phi \) requires the estimation of weight parameters \( w_{uv} \) and \( w_{vz} \) in the neural network as shown in Fig. 5.14. We use the backpropagation weight-learning rule as introduced in Appendix C, which minimizes the sum of squared prediction errors over the training samples, defined as

\[
\sum_{d} \sum_{1 \leq z \leq 4} (\Delta p_z^d - r_z^d)^2,
\]

where \( r_z^d \) is the target (ground-truth) value for the \( z \)-th (normalized) output parameter for the training sample \( d \). During the training, we use a separate verification set to prevent the overfitting problem (see Appendix C).

(3) SVR-based prediction

In Chapter 2, SVM (Support Vector Machine) has been used as a classification technique for face detection. As introduced in Appendix D, SVM seeks large margins between classes and offers good generalization performance. A similar reasoning can be also applied to SVR (Support Vector Regression), where real-valued functions are estimated (see Appendix E). In this section, we propose to apply one variation of SVR (\( \epsilon \)-SVR) for parameter prediction.

In our case, for simplicity, we estimate a prediction function for each individual parameter component \( \Delta p_z \) in \( \Delta \mathbf{p} \). In our implementation, we use the Radial Basis Function (RBF) \( e^{-\|u-v\|^2/\sigma^2} \) as the kernel function in the SVR formulation (see Appendix D and E).

The estimation of prediction function \( \Phi \) is the key step in CTF and its accuracy directly influences the performance of the feature extraction. In the following, we implement the three prediction schemes discussed previously and evaluate the feature-extraction performance by using the associated prediction function.

5.6.5 Implementation and evaluation of CTF

In this subsection, we first examine and compare the prediction behavior of the three prediction approaches using linear estimation, neural-network regression and \( \epsilon \)-SVR. The cascading performance of the best-performing algorithm in combination with the SGS algorithm (Section 5.5) is also presented.
5.6. Component-based texture fitting (CTF)

Figure 5.15: Convergence of the three prediction algorithms. Here the prediction of parameter $x$ of the right-eye component is used as an example. Left: accuracy estimation; Right: convergence diagram.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Perturbation ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center $x$</td>
<td>$\pm 40%$ of the width of the ground-truth feature shape</td>
</tr>
<tr>
<td>Center $y$</td>
<td>$\pm 40%$ of the height of the ground-truth feature shape</td>
</tr>
<tr>
<td>Scale $r$</td>
<td>$\pm 30%$ deviation w.r.t. the ground-truth scale</td>
</tr>
<tr>
<td>Rotation $\phi$</td>
<td>$\pm 0.1$ rad to the ground-truth rotation angle</td>
</tr>
</tbody>
</table>

Table 5.4: Perturbation ranges for training the prediction function.

In all of the following experiments, we use the same training and test image sets as specified in Section 5.5.5. For each facial feature, we randomly perturb each parameter element of $p$ and collect 25 samples for each image, so that altogether we collect around 10,000 training samples and 10,000 test samples for each evaluation. During the training (of the prediction function) and the testing, we apply the same perturbation ranges for the parameters (see Table 5.4). For a cascaded scheme, the choice of these ranges depends largely on the output behavior of its preceding algorithm. More specifically, the perturbation ranges should be at least larger than the converged output ranges of the preceding algorithm. For example, from the output diagram of SGS as shown in Fig. 5.9, the x-deviation of the output is roughly between -0.3 and 0.3. Therefore, we choose a slightly larger perturbation range for the x-deviation in CTF, e.g. between -0.4 and 0.4.

Capture ranges. In principle, the capture range of the CTF algorithm is determined by the following two factors: (1) the perturbation ranges adopted
Figure 5.16: Comparison of the parameter prediction accuracy of the three estimation approaches. Here the prediction of the right-eye component is used as an example.

Figure 5.17: Comparison of the extraction accuracy of the three estimation approaches (using right-eye component as an example). Left: single-step estimation; Right: iterative estimation.
during the training, and (2) the representation power of the estimation function $\Phi$, which can be linear or nonlinear. In Fig. 5.15, we show the convergence behaviors of the prediction functions trained with the three paradigms from Section 5.6.4. The three algorithms exhibit similar convergence behavior, with a slightly narrowed convergence range compared to the perturbation ranges for the training. It is worth noticing that the three algorithms have different accuracy levels which is evident in the left figure in Fig. 5.15. This is further examined now.

**Feature-extraction accuracy.** Before we come to the average pt-pt extraction error of CTF, we first examine the prediction accuracy for each parameter element $\Delta p_z$ and plot the prediction results in Fig. 5.16. The diagonal straight line shown in each plot provides the ground-truth for each parameter. The closer the predicted results are to this reference line, the more accurate the prediction function. The predicted parameter value along the $y$-axis in each plot is the averaged measure for all the test images. It can be seen from Fig. 5.16 that using the nonlinear prediction function based on neural-network regression and SVR clearly improve the prediction accuracy compared to the linear approach. The improvement is especially evident for large parameter deviations. Among the two nonlinear approaches, SVR-based prediction yields even a better performance than using neural-network regression.

In Fig. 5.17(a), we compare the extraction accuracy (pt-pt error) for the right-eye feature using the three prediction schemes. Similar conclusions can be drawn as above, where the SVR and neural-network prediction yield much better extraction accuracy than the linear approach.

To further improve the parameter-prediction accuracy, an iterative prediction scheme can also be adopted, as originally introduced in AAM [10]. In this approach, Eqn. (5.15) can be iteratively applied to correct the parameter deviation until Eqn. (5.14) reaches the minimum. We show both the single-step prediction (Fig. 5.17(a)) and the iterative prediction results (Fig. 5.17(b)). For all three approaches, using the iterative optimization yields slightly better results. Considering the additional computation cost involved for the iterative approach, in the following we keep the single-step solution for the implementation efficiency.

Table 5.5 summarizes the average pt-pt error for all the facial features. Similar performance is observed as above, where neural-network regression and SVR improve the prediction results by 30% to 50%. The SVR approach achieves the best result with the only exception for the nose feature, where its performance is slightly worse than the neural-network approach. In the following, the SVR prediction is used as the default prediction scheme for CTF if not stated otherwise. A few examples of the visual results of the extraction using the SVR prediction are given in Fig. 5.18.
Component & Linear & Neural network regression & SVR \\ 
Right eye & 2.49 & 1.90 & 1.60 \\ 
Right eyebrow & 5.00 & 3.40 & 2.69 \\ 
Nose & 5.60 & 3.80 & 4.15 \\ 
Mouth & 6.77 & 5.00 & 4.42 \\ 

**Table 5.5:** Average pt-pt error (in pixels) for all the facial features. Due to the feature symmetry, the statistics related to the left eye (eyebrow) are skipped here.

![Figure 5.18: Example extraction results by CTF. (a) Initial feature model; (b) Fitted feature model.](image)

In the following, we concatenate the CTF algorithm to the SGS algorithm presented in the previous section and analyze the performance gain of the subcascade resulting from the concatenation.

**Evaluation of the subcascade: SGS and CTF.** We concatenate the CTF described in this section to the SGS algorithm described in Section 5.5 (abbreviated as SGS+CTF in the following). The CTF acts as a *refinement* to SGS in order to further reduce the extraction error. The corresponding cumulative distribution of the relative pt-pt error as defined in Eqn. (5.12) is depicted in Fig 5.19. For comparison, the error distribution of SGS+CTF is plotted together with SGS. By using the subcascade, the extraction accuracy is effectively improved by roughly 30%.

In comparison to Fig. 5.9, we depict the parameter-deviation distribution from SGS+CTF in Fig. 5.20. It can be seen that the predicted parameter has a narrower distribution centered at the origin as compared to Fig. 5.9. These parameters also give an important reference for the subsequent algorithm design in the complete cascade, which is presented in the next section.
5.6. Component-based texture fitting (CTF)

In this section, we have proposed an algorithm to obtain the basic shape descriptions of each facial feature. This is achieved by directly estimating the output shape parameters given the current shape and texture information. The estimation function is derived from a set of training samples. Here we propose to use nonlinear regression techniques such as neural-network implementation and SVR for a more accurate performance as compared to a linear approach [10].

In [93], we also proposed to use a subcascade of SVR functions for parameter prediction. By using different training ranges, the resulting SVR functions have different capture ranges and accuracy. As a result, they can be cascaded according to the principles in Section 5.3 to further improve the prediction performance.

It is worth pointing out that the CTF is carried out for individual facial features. Since the previous algorithm (SGS) already brings each feature close to the ground-truth position, it suffices to utilize local knowledge to correct the basic shape parameters for these individual features. Furthermore, by concentrating only on salient features, less texture variances are involved, resulting in more stable estimation results.

The prediction results achieved by CTF largely depend on the feature appearance in the training data. The a-priori knowledge about facial features are embedded in the training of the prediction function. However, for implemen-

---

**Figure 5.19:** Cumulative distribution of the relative pt-pt error of SGS+CTF as compared to using SGS alone.
tation efficiency and training feasibility, the feature model we have adopted only has limited flexibility (with location, scale and rotation parameters), so that the extraction results may not be optimal w.r.t individual feature deformations. In the following, as a final step, we increase the flexibility of the feature model and use direct optimization to refine the matching results.

Figure 5.20: *Parameter-estimation accuracy of CTF. The plots are depicted in the same scale as in Fig. 5.9.*
5.7 Component-based direct fitting (CDF)

In this section, we present the third-stage algorithm in the cascaded extraction. More specifically, we refine the feature-extraction results for each facial feature by using a more flexible appearance model, with the aim to achieve better adaptation for individual feature instances. In this appearance model, in addition to the shape parameter vector $p$ as used in CTF, we employ an additional appearance parameter vector $a$ to model the feature shape and texture deformations.

Differing from CTF in which a function is trained to direct predict the optimal model parameters, a generic optimization technique is adopted in this section for the model fitting. In order to successfully apply such a scheme, we need to consider the following two potential problems: (1) wrong local convergence occurring when the optimization function has a complex surface in the parameter space; and (2) high computation cost involved for complex optimization algorithms.

In our case, as the objective is to perform a refinement to the previous algorithms in the full cascade, we only need to perform the parameter search over a constrained parameter space. As a result, the problem of wrong local convergence is largely alleviated and the processing cost is also effectively reduced.

In the following, we introduce the appearance model used for each facial feature (Section 5.7.1) and explain how we solve the model-fitting problem by using a generic optimization solution (Section 5.7.2). The experimental evaluation is presented in Section 5.7.3.

5.7.1 Component-based appearance model

The applied appearance model is adopted from [10] and is partially based on the feature model developed in CTF. As mentioned before, for each facial feature, we use shape vector $s$ and its associated texture vector $g$ to model the shape and texture of the feature. Similar to the PCA analysis of shapes as in statistical shape model (Section 4.3.2), a PCA analysis of the texture sample can be also performed. Consequently, any normalized shape $s_n$ and texture $g_n$ (already normalized by definition) can be approximated by

$$s_n \approx \bar{s} + Q_s b_s, \quad g_n \approx \bar{g} + Q_g b_g,$$

where $\bar{s}$ is the mean shape vector, $Q_s$ contains the Eigenvectors derived from the PCA decomposition and $b_s$ is the shape-deformation parameter giving the projection of $s_n$ in space $Q_s$. Similar reasoning holds for the second equation about $g$. Fig. 5.21 depicts some examples of the first variation modes learned for the mouth model.
The appearance of each facial feature is therefore defined as a concatenated vector
\begin{equation}
\mathbf{c} = \begin{bmatrix} \mathbf{\Lambda} \mathbf{b}_s \\ \mathbf{b}_g \end{bmatrix},
\end{equation}
where \( \mathbf{\Lambda} \) is a diagonal scaling matrix with weights for \( \mathbf{b}_s \) to scale it to the range of \( \mathbf{b}_g \). To further remove correlation between \( \mathbf{b}_s \) and \( \mathbf{b}_g \), a third PCA is applied\(^6\) to \( \mathbf{c} \), and we obtain
\begin{equation}
\mathbf{c} = \mathbf{Q}_a \mathbf{a},
\end{equation}
where \( \mathbf{Q}_a \) stands for major combined appearance variations and \( \mathbf{a} \) is the appearance-deformation parameter giving the projections of \( \mathbf{c} \) in the space spanned by \( \mathbf{Q}_a \).

Therefore, the feature model used in this section incorporates the appearance parameter \( \mathbf{a} \) in addition to the shape parameter \( \mathbf{p} \) as used in CTF. The aim of the model fitting is then to find the optimal \( \mathbf{p} \) and \( \mathbf{a} \), such that the difference between the average texture \( \bar{\mathbf{g}} \) and the current texture \( \mathbf{g} \) with parameters \( \mathbf{p} \) and \( \mathbf{a} \) is minimized.

Since the shape parameter vector \( \mathbf{p} \) has already been addressed by the parameter prediction scheme in CTF, the additional gain in this section is the more customized feature extraction by adjusting the appearance parameter vector \( \mathbf{a} \). Nevertheless, \( \mathbf{p} \) is still allowed to vary within a small range for a fine-tuning purpose.

\(^6\)This PCA is a second recursion but now on the joint shape and texture data.
5.7. Component-based direct fitting (CDF)

5.7.2 Direct optimization of the model parameters

To achieve a refined feature extraction, we can directly solve the model-fitting problem by using a generic optimization algorithm. Based on the output statistics from the previous algorithm (Fig. 5.16), the optimization only needs to perform a constrained search over the parameter space. This not only reduces the possibility of finding wrong local minima, but also improves the search efficiency. More specifically, for model parameter $p$, we restrict the search range of each element of $p$ to half of the values as specified in Table 5.4. For appearance parameter $a$, we restrict the search range of each element to 2.5 times the standard deviation of the corresponding variation modes as found after the third PCA decomposition.

In our experiments, we have chosen the following three optimization techniques [57].

1. **Gradient descent.** Gradient descent is one of the most commonly-used optimization algorithms to quickly find a local minimum. This is achieved by taking steps proportional to the negative of the gradient (or the approximate gradient) of the function at the current point. We have experimented with two variants of the gradient-descent algorithm: Fletcher-Reeves (FR) conjugate gradient algorithm and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) conjugate gradient algorithm [22][57]. Both algorithms yield similar performance in terms of feature-fitting accuracy and efficiency. For chosen parameter search range, the gradient-based search often stops at wrong local minima.

2. **Simulated annealing.** Simulated annealing makes use of random sampling in a large search space, in order to locate a good approximation to the global optimum of a given function. We found in our experiments that the use of simulated annealing for extraction refinement is much more computationally expensive than the Simplex method (illustrated below) and yields no better results.

3. **Simplex algorithm.** The simplex algorithm uses the concept of a polytope of $N + 1$ vertices in $N$ dimensions (called a *Simplex*) to guide the search process. Although still a local optimization technique, the Simplex algorithm allows an occasional ‘jump’ out of local minima. In our experiments, it achieves more accurate results than the gradient-based algorithms while consuming only 10% of the computation time as compared to gradient descent. Therefore, we select the Simplex algorithm for direct optimization in this section.

In the following, we present the performance evaluation of the refinement algorithm.
5.7.3 Implementation and evaluation of CDF

In this section, we use the same training images and test images as used previously. The training set is employed to derive the appearance model. Similar to Section 5.6.5, for each test image, an average model is randomly initialized near the ground-truth position based on the output statistics from CTF. For each test image, we collect 100 of such cases and perform the feature-extraction refinement as introduced above. We measure the capture range and extraction accuracy of the algorithm and summarize the results in the following.

Capture range. As the final refinement step in the cascaded feature extraction, the CDF algorithm in this section has a relatively small capture range, which is constrained by the parameter search range and the computation cost. The parameter search-range of CDF needs to be larger than the output range of CTF in order to satisfy the cascading principle from Section 5.3. An example is given in Fig. 5.22 to show the difference of the capture ranges between CDF and CTF (its preceding algorithm in the extraction cascade), where CDF has a halved capture range of CTF.

Extraction accuracy. Fig. 5.23 portrays the cumulative distributions of the relative pt-pt errors for the following three algorithms: SGS, SGS+CTF, SGS+CTF+CDF. Using the CDF as a refinement step has effectively reduced the extraction error by 5%. In addition to quantitative measures, the improvement can be visually checked in Fig. 5.24. The CDF can be quite effective in
5.8 Performance summary

In this section, we give a brief summary of the characteristics and performance of the cascaded feature extraction proposed in this chapter. We also quantitatively analyze the performance gain by using the cascade. In Table 5.6, we summarize the design motivations for the three component algorithms in the cascade. These algorithms capture various characteristics of facial features and give different extraction performance in terms of robustness (convergence) and accuracy. This is summarized as follows.

1. From SGS to CDF, the algorithms use image inputs of increasing resolutions. Thus finer details are captured for succeeding processing stages.

2. Since the global relationships between facial features are already addressed by the first algorithm, the succeeding stages only need to exploit local model structures without interference from global deformations.

3. From SGS to CDF, the models have increasing flexibility, allowing for more deformations to accommodate for individual feature variations.

Figure 5.23: Cumulative distributions of the relative pt-pt error of the three-component extraction cascade as compared to previous results.

Reducing the extraction error by CTF for some specific facial features, such as the mouth in the second, third and fifth examples and the right eyebrow and nose in the fifth example.
Algorithm | SGS | CTF | CDF
---|---|---|---
Resolution | Low | Medium | High
Granularity | Global | Component-based | Component-based
Flexibility | Low | Medium | High

Table 5.6: Design considerations of the three component algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SGS</th>
<th>CTF</th>
<th>CDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture range (relative x)</td>
<td>Full range</td>
<td>$[-0.37, 0.37]$</td>
<td>$[-0.2, 0.2]$</td>
</tr>
<tr>
<td>Output range (relative x)</td>
<td>$[-0.3, 0.3]$</td>
<td>$[-0.18, 0.18]$</td>
<td>$[-0.1, 0.1]$</td>
</tr>
<tr>
<td>Relative/abs. mean accuracy</td>
<td>$0.175/3.5$</td>
<td>$0.08/1.60$</td>
<td>$0.055/1.1$</td>
</tr>
</tbody>
</table>

Table 5.7: Performance comparisons of the three component algorithms in the cascaded extraction. The right-eye component is used as an example for the measurements. The capture range and output range are based on the x-deviation only to exemplify the extraction performance. The mean accuracy is based on an average eye-width of 20 pixels.

4. The three algorithms follow a coarse-to-fine design paradigm and have increasing extraction accuracy and decreasing capture range.

Table 5.7 shows the performance comparisons between the three algorithms. The capture range of each algorithm fully covers the output range of its preceding algorithm and the extraction accuracy is effectively improved. This complies with the two cascading principles as presented at the beginning of this chapter. As a consequence, the overall capture range and the extraction accuracy are both enhanced. Fig. 5.24 portrays visual example results employing the cascaded algorithm.

Comparison with existing approaches

It can be seen from the previous discussion that by cascading our proposed three algorithms, we have achieved a significantly improved capture range and extraction accuracy which is clearly better than any single algorithm in the cascade. It is also interesting to compare our proposed extraction technique with statistical shape models (e.g. ASM and H-ASM), which are the mostly-used state-of-the-art extraction techniques with high performance.
Figure 5.24: Sample results achieved by the three-algorithm cascade. (a) face detector output; (b) output from SGS; (c) output from CTF; (d) output from CDF.

Since our approach starts with a full-search SGS algorithm, it is not sensitive to the model initialization. In contrast with this, the ASM (or AAM) has a limited capture range. For example, even for H-ASM which has a relatively good convergence capability, the capture range is limited to 0.37 in relative x-deviation (see Chapter 4).

We have also compared the extraction accuracy of the cascaded algorithm
to the ASM (H-ASM). For each test case, we perturb the initial model (or search range) with various deviations. The average extraction accuracy is 3.8 pixels for the cascaded approach, which is a 8% reduction compared to H-ASM and roughly 24% reduction compared to conventional ASM.

Although we use three algorithms in the cascade, the required processing cost is comparable to H-ASM (or ASM). This is because the first algorithm in the cascade operates on coarse granularity and can quickly locate globally correct feature structures. This reduces the complexity of the prediction function of the second algorithm. Similarly, the second algorithm further improves the accuracy and reduces the search space for the final algorithm. In this way, the total computation cost is kept at a minimum level, e.g. around 0.5 second on a PC platform of 3.0 GHz.

It is worth pointing out that in a related work [15], a multistage feature-extraction approach is proposed. The approach first applies a set of independent feature detectors for 17 feature points using the Adaboost classification technique. Following that, a conventional AAM is applied to refine the results. Different from [15], in our approach, the design of each component algorithm obeys the cascading principles as proposed in Section 5.3. For example, the output statistics of each algorithm are measured and the subsequent algorithms are optimized based on these statistics (e.g. training-parameter selection and search range selection). In this way, we ensure that each algorithm effectively improves the overall performance and adjacent algorithms are properly coupled together. Furthermore, the computation efficiency is another advantage since no sophisticated classifiers are required for local feature point detection.

5.9 Summary and conclusions

In this chapter, we have first presented a cascaded facial feature extraction framework. Within this framework, we have defined several metrics (capture range and average extraction accuracy) to measure the performance of a model-based algorithm. We have designed a new three-algorithm cascade for an incremental modeling of facial features. Fig. 5.25 portrays the algorithm steps for the facial feature extraction. Our approach uses an incremental modeling of facial feature structures, with additional parameters incorporated at each stage, such that the model has more flexibility and description accuracy. This enhanced flexibility is usually accompanied by a reduced convergence capability, due to the enlarged parameter search space and more complex cost functions. However, in a cascaded framework, the initial algorithm with better convergence capability can effectively reduce the search range (or complexity) of succeeding algorithms, keeping the overall computation cost at a low level.

The proposed cascaded approach has achieved 10-20% error reduction than
contemporary extraction techniques, such as ASM and H-ASM. Furthermore, it is highly robust for input variations. When used as the second step in face recognition, it generates reliable extraction outputs for face normalization and final feature-vector generation. In our real-time tests with live camera input, the feature extraction is able to cope with most frontal-view faces except for some cases when people wear dark-frame glasses and have a dark-colored moustache, which naturally influence the detection reliability of eyes and mouth.

The proposed cascaded feature extraction also has a complexity-scalable nature. In addition to the full cascade employing all three algorithms, the subcascade SGS and SGS+CTF can be also used for feature extraction in application scenarios where less accuracy is required and/or the computation resources are more restricted.

**Figure 5.25:** *Summary of the three algorithms in the cascaded feature extraction.*

---

**Diagram:**

- **Face estimation**
  - **Sparse graph model**
    - a. Exhaustive pattern search & confidence map generation
    - b. Tree-based node pruning & prob. shape evaluation
  - **Component-based texture model**
    - a. Shape + texture modeling
    - b. Shape parameter prediction based on texture information
  - **Component-based appearance model**
    - a. Shape (with additional deformable parameter) + texture
    - b. Direct parameter optimization using Simplex algorithm
  - **Feature-point locations**
Chapter 5. Cascaded Model-Based Facial Feature Extraction
Chapter 6

Face Identification

6.1 Introduction

In previous chapters, we have discussed various techniques for face detection and facial feature extraction. Based on the extracted facial feature points, the face region can be properly cropped and scaled for the final processing stage in a face-recognition system, i.e. face identification. As shown in Fig. 6.1, this and the next chapter is primarily devoted to techniques associated with face identification.

The aim of face identification is to determine the identity of the input face by searching from a database of known individuals. The best match from the database is then returned as the identified face. For some applications, the returned identity is further verified based on e.g. a threshold criterion to determine whether it should be rejected as an ‘unknown’ identity. In this thesis, we only consider the strategy to locate the correct best-match. Note that in literature, face identification and face recognition are usually used interchangeably. In this thesis, we explicitly use face identification to refer to the last stage of face recognition, excluding the preprocessing steps such as face detection and facial feature extraction.

Another closely-related concept is face verification, which compares a given input face image with a claimed identity, and confirms or rejects the claim. Different from face identification, which performs a one-to-many matching, the face verification only performs a one-to-one matching and forms as such a sub-problem of face identification. In this thesis, we concentrate mainly on the problem of face identification.
In this and the following chapter, we use the term *gallery faces* to refer to face images stored in the database, and *probe faces* to refer to the input face images that need to be identified. When the correct identity of the probe face is located from the face gallery, we call it a *correct match*.

The performance of the identification process is usually characterized by the *identification rate* or the *cumulative rank curve*. The identification rate refers to the ratio between the number of correct matches and the total number of probe faces used. In some cases, the *cumulative rank curve* is also used to measure the performance of an identification algorithm. More specifically, in order to obtain the cumulative rank curve of an identification algorithm, for each given probe face, the gallery faces are first ranked according to their similarity to the probe face. A correct rank-$i$ match is assigned when the ground-truth face is among the first $i$ sorted gallery faces. Consequently, a *cumulative rank curve* $R(i)$ can be generated, where $R(i)$ is the ratio between the number of correct rank-$i$ matches and the total number of probe faces. In addition to the identification rate and the cumulative rank curve, which are both used to measure the probability of an algorithm to find the right identity, the *False Acceptance Rate (FAR)* is used as a complementary measure to indicate the probability that an ‘unknown’ probe face is incorrectly assigned an identity in the database.

In recent years, face identification has aroused great interest in both in-
Within-class variances.  Between-class variances.

(a) Within-class variances.  (b) Between-class variances.

Figure 6.2: Within-class and between-class variances in face images (picture source: Yale Face Database B [21]).

dustry and academia. Although significant progress has been achieved, face identification in unconstrained situations typically occurring in embedded applications, still remains a challenging problem. The major reason causing this difficulty is the relatively large within-class (or intra-class) variances existing between images of the same individual as compared to the between-class (or inter-class) variances existing between images of different individuals (see Fig. 6.2 demonstrating these variances). The large within-class variances are mainly caused by the highly variable lighting condition, facial expression, face poses and aging. It has been shown in [4] that different human faces, when captured under the same condition, are often more similar to each other than images of the same face captured under very different conditions. Various techniques have been proposed in literature to address this problem, with an aim to separate different individuals as much as possible for a better discrimination.

The aim of this chapter is to give a brief overview of existing identification techniques in literature. Some of the existing techniques are used as a reference to benchmark our new proposal in the next chapter.

The remainder of this chapter is organized as follows. Section 6.2 first introduces the basic constituents of a face-identification algorithm: feature representation and classification function. In Section 6.3, we review holistic and local structural features, which are commonly used in literature. In Section 6.4, we discuss several representative techniques for classification-function design. Note that generally the feature representation and the classification function can be designed independently from each other. Section 6.5 concludes this chapter.
6.2 Key elements of face-identification techniques

In recent years, various identification algorithms have been evaluated, and most of them can be characterized by two key elements: feature representation and classification function.

**Feature representation.** Usually a feature vector \( \mathbf{x} \) is used to represent a face image. This feature vector can have different physical meanings. The simplest way to extract a feature vector is to take the raw image intensities of the preprocessed face image. An alternative is to use local texture information at some specific facial feature points. In the design of an identification technique, it is desirable to choose a feature representation that creates a good separation between different individuals. More specifically, in the feature space where feature vector \( \mathbf{x} \) is located, feature vectors from the same individual person should form a condensed cluster, while feature vectors from different individuals should be as distant as possible.

**Classification function.** A classification function \( f_c \) takes feature vector \( \mathbf{x} \) as input, and assigns it to one of the identities from the gallery faces. Suppose a gallery contains samples of a finite number of identities, then \( f_c(\mathbf{x}) \) returns one of the identity labels, or it outputs an ‘unknown’ identity when it fails to locate a suitable identity. Note that different from face detection, where only two classes (face/non-face) are concerned, face identification deals with multiple identities, thereby leading to a multi-class classification problem.

In practice, the choice of a specific feature representation and the classification function depends largely on the type of application and its performance goals. In the following, we first review several commonly-used feature representations in face identification. Following this, in Section 6.4, we evaluate a set of classification functions that can be applied to these features to perform the identification.

6.3 Feature representation

This section deals with two classes of commonly-adopted features: **holistic features** and **structural local features**. Holistic features use global image intensities as basic features, while structural local features are usually derived from the geometry or appearance of local facial features (or feature points). In some cases, a combination of holistic features and structural local features can be used to obtain an enriched representation [78].
6.3. Feature representation

6.3.1 Holistic features

Holistic features use the raw raster-scanned image intensities from a preprocessed face image as the basic representation. It should be noted that in addition to the basic preprocessing of face images such as cropping and scaling, some additional processing may be necessary to further reduce the interferences from factors such as illumination and pose variations. These factors can contribute to undesired large within-class variances. For example, illumination correction may be necessary when the gallery and/or probe face images have different lighting conditions [24]. Pose normalization may be necessary to align each face image to a standard reference view (e.g., front view) for varying poses. This can be achieved by building correspondences between the extracted facial features of an input face and those of a reference face [98]. This additional preprocessing reduces the large within-class variances introduced by illumination and poses, resulting in a more compact within-class distribution in the feature space.

Holistic features can be easily obtained from a face image and this approach has been widely adopted in face-identification systems, such as [70], [4], [79] and [46]. Alternative approaches advocate to exploit facial structure knowledge and incorporate it into the feature generation. This leads to the structural local feature representation, which is discussed in the following subsection.

6.3.2 Structural local features

Typically, structural local features rely on an accurate extraction of facial features. The locations and/or local appearances of these features are used to...
Earlier work using structural local features are based on pure geometry measures, e.g. the distance between the eyes, etc [31]. Later techniques incorporate local texture information to characterize individual faces. A popular approach is to use Gabor wavelet transforms to represent the local appearance of a set of extracted facial feature points. This technique was first proposed for face identification in [77] and was later adopted in other algorithms such as [3].

More formally, we use \( I(p) \) to denote the pixel intensity value of point \( p = (x, y) \) in a local image patch centered at a feature point \( p_0 = (x_0, y_0) \). The local appearance of this feature point is then obtained by convolving the local image patch with a set of Gabor wavelet kernels, given by

\[
J_m(p_0) = \int I(p') \phi_m(p_0 - p') dp'.
\]

(6.1)

In the above, \( \phi_m \) is a Gabor wavelet kernel defined as

\[
\phi_m(p) = \frac{\|k_m\|^2}{\sigma^2} \exp\left(-\frac{\|k_m\|^2\|p\|^2}{2\sigma^2}\right) \left[\exp(ik_m \cdot p) - \exp\left(-\frac{\sigma^2}{2}\right)\right].
\]

(6.2)

A Gabor wavelet kernel has the shape of a plane wave with wave vector \( k_m \), restricted by a Gaussian envelope function (see Fig. 6.3). From a biological viewpoint, this waveform imitates the human visual perception system. Usually, a discrete set of Gabor kernels is used to characterize each local feature point. These kernels (\( \phi_m \)) have different wave vectors \( k_m \) with various scales and orientations. Assuming that 5 different scales and 8 different orientations are used, indexed by \( v = 0, 1, ..., 4 \) and \( \mu = 0, 1, ..., 7 \), respectively, we have

\[
k_m = \begin{pmatrix}
    k_v \cos \varphi_{\mu} \\
    k_v \sin \varphi_{\mu}
\end{pmatrix},
\]

(6.3)

where \( k_v = 2^{-\frac{v+2}{2}} \pi \), \( \varphi_{\mu} = \frac{\pi \mu}{8} \), and \( m = 8v + \mu \). Consequently, the texture of a local facial feature is characterized by a set of wavelet coefficients \( J_m \) (\( 0 \leq m \leq 39 \), usually called a jet [77]), derived from convolutions with \( 5 \times 8 \) Gabor convolution kernels \( \phi_m \) (\( 0 \leq m \leq 39 \)). For a face image with \( L \) facial feature points (see Fig. 6.3), a total of \( L \) jets are generated and concatenated into a structural local feature vector. It is reported in [77] that these jets are relatively robust to illumination changes, translation, distortion and scaling.

In summary, the holistic features capture global characteristics of a face image and do not discriminate between the different parts of the face region. The structural local features capture the geometry and the local appearance of facial features and give a structural description of the face. This structural description usually requires an accurate extraction of facial features (feature points), as was discussed in Chapters 4 and 5.
6.4 Classification function

In the previous section, we described the generation of feature vectors for face identification using holistic and/or structural approaches. In this section, we explain the definition of a classification function based on these features.

Recall that in Chapter 2, we have introduced discriminant functions to characterize the decision surfaces that separate two classes. In a two-class problem (class $\omega_1$ and class $\omega_2$), discriminant functions $g_{\omega_i}$ define the following classification rule:

$$f_c(x) = \begin{cases} 
\omega_1, & \text{if } g_{\omega_1}(x) > g_{\omega_2}(x), \\
\omega_2, & \text{if } g_{\omega_1}(x) < g_{\omega_2}(x).
\end{cases}$$

(6.4)

Different from the above, face identification is a multi-class classification problem. Suppose we have $C$ individuals (classes) in the gallery ($C > 2$): $\omega_1, \omega_2, ..., \omega_C$. Altogether $C$ discriminant functions $g_{\omega_i}$ can be defined, leading to the following classification function:

$$f_c(x) = \omega_i, \text{ if } g_{\omega_i}(x) > g_{\omega_j}(x), \forall j, 1 \leq j \leq C \text{ and } j \neq i.$$  

(6.5)

In the following, we first review two categories of techniques used in the above-mentioned multi-class framework: (1) distance-based approaches, with special emphasis on subspace-based classification, and (2) learning-based approaches such as neural networks.

6.4.1 Multi-class formulation for classification-function design

In this subsection, we review two categories of techniques using multi-class discriminant functions: distance-based and learning-based approaches. Distance-based techniques define certain distance metrics, based on which the input feature vector can be compared to the gallery feature vectors. The gallery vector having the minimum distance to the input vector is a correct match. In distance-based approaches, the discriminant functions can be easily defined on the basis of the distance metrics applied. In addition to distance-based approaches, learning algorithms such as neural network can be also used to derive multi-class discriminant functions [33].

**Distance-based classification**

Given a probe feature vector $x$, the distance-based classification computes the distance between $x$ and each of the gallery vectors $x_i$, denoted as $d(q(x), q(x_i))$, where $q$ indicates a transform function of $x$. The classification function is then defined as

$$f_c(x) = \arg\min_i d(q(x), q(x_i)),$$  

(6.6)
where $d(\cdot)$ defines a distance metric (e.g. Euclidean distance). The distance can be defined based on the original feature representation (when $q(x) = x$), or it can be defined in a transformed feature space, where $q$ implies the transform function (see Subspace-based distance measures in the following). We can easily see from the above that the discriminant function for the class $\omega_i$ can be defined as e.g. $g_{\omega_i}(x) = 1/d(q(x), q(x_i))$.

Simple distance measures. The simplest distance measure can be directly defined on the original feature vector $x$. An example is the distance metric used in [77] for comparing two Gabor feature vectors, which is given by
\begin{equation}
    d_s(x, x_i) = 1 - \frac{(x \cdot x_i)}{\|x\| \cdot \|x_i\|}.
\end{equation}

However, for holistic features, the use of a simple distance metric may not be effective. To this end, an important class of approaches adopts a subspace-based comparison. These approaches employ a transformed space in which feature vectors from different individuals can be more easily separated. In the following, we review two well-known subspace algorithms based on PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis).

Subspace-based distance measures. Eigenface (PCA) is one of the first subspace-based face-identification approaches. By applying PCA to a set of training feature vectors of $M$ dimensions (see Appendix B), a new feature space with basis $\Phi_p$ is generated, spanned by the first $K$ Eigenvectors (Eigenfaces), with typically $K \ll M$. Some examples of these Eigenvectors depicted in image form are shown in Fig. 6.4(a). The basis matrix $\Phi_p$ is a low-dimensional and rotated space compared to the original feature space, which accounts for the major variations of the training feature vectors. Based on $\Phi_p$, any feature vector $x$ can be transformed to a new feature vector $y$ in the new space by $y = \Phi_p^T(x - \bar{x})$, where $\bar{x}$ is the mean of all training vectors. The classification function used by PCA is then defined by the (Euclidean) distance between the transformed vectors of a given feature vector $y$ and that of a gallery feature vector $y_i$, given by
\begin{equation}
    d_p(x, x_i) = \|y - y_i\|,
\end{equation}
where $y = \Phi_p^T(x - \bar{x})$ and $y_i = \Phi_p^T(x_i - \bar{x})$, with $1 \leq i \leq C$.

It is worth noticing that the transformed space resulting from PCA is based on the sample variance characterizing the scatter of the entire sample set, irrespective of the class membership of each sample. Although PCA provides good dimensionality reduction for the original samples, the transformed directions may not be optimal from a discriminative point of view. This is
briefly illustrated in Fig. 6.4(b), where the transformed direction found by PCA cannot provide a proper separation between two classes ‘1’ and ‘2’. In the following, we review another classification technique based on LDA. The aim of LDA is to identify projection directions along which different classes are best separated. It performs a dimensionality reduction while preserving as much as possible the class discrimination.

More specifically, suppose we have $C$ classes $\omega_1 \sim \omega_C$, and each class $\omega_i$ has $C_i$ sample vectors $x_i^{(k)}$ with $1 \leq k \leq C_i$. LDA first defines a within-class scatter matrix $S_w$, which characterizes the sample variation within one individual, given by

$$S_w = \sum_{i=1}^{C} \sum_{k=1}^{C_i} (x_i^{(k)} - \bar{x}_i) (x_i^{(k)} - \bar{x}_i)^T.$$  \hspace{1cm} (6.9)

In Eqn. (6.9), $\bar{x}_i$ is the mean of samples from class $\omega_i$. In addition, the between-class scatter matrix $S_b$ is defined to characterize the sample variation between different individuals, which is specified by

$$S_b = \sum_{i=1}^{C} (\bar{x}_i - \bar{x}) (\bar{x}_i - \bar{x})^T.$$  \hspace{1cm} (6.10)

In the above, $\bar{x}$ is the mean of all sample vectors. The optimal transformation $\Phi_l$ defined by LDA maximizes the following function:

$$J(\Phi_l) = \frac{\det(\Phi_l^T S_b \Phi_l)}{\det(\Phi_l^T S_w \Phi_l)},$$  \hspace{1cm} (6.11)
This can be further formulated as a generalized Eigenvalue problem and solved in closed form if $S_w$ is nonsingular. Interested readers are referred to [17] for more detailed information. In practice, however, $S_w$ is often singular because feature vector $x$ is usually of high dimensionality while the number of available training samples is rather limited. Therefore, usually PCA is first applied to the data samples as a preprocessing step to reduce the sample dimensionality. Afterwards, the LDA is applied to the samples of reduced dimensionality to maximize the class separability.

Having derived the transformation $\Phi_l$, we can define the classification function of LDA in a similar way as in PCA (Eqn. (6.8)), which is based on the Euclidean distance between the transformed vectors of a probe feature vector and a gallery feature vector. The gallery feature vector yielding the minimum distance is the best match.

In addition to the above, a number of extensions of PCA and LDA have also been proposed in literature, such as view-based PCA [52], regularized LDA [42], etc. In addition, nonlinear PCA and LDA are also proposed to extend the linear subspaces by using kernel techniques [79]. Distances defined in these subspaces are claimed to provide a better separation between different individuals than using the original feature space.

**Learning-based classification**

In addition to the distance-based approaches, discriminant functions for face identification can be also derived by traditional learning-based techniques such as neural networks. The neural network concepts presented earlier in this thesis can also be applied to generate multiple discriminant functions for face identification.

As mentioned in Chapter 2, a neural network is capable of realizing a nonlinear function from a set of training samples. Suppose we need to generate a classification function for discriminating $C$ individuals, we can use a network with $M$ inputs and $C$ outputs, where $M$ is the dimension of input feature vector $x$. The $i$-th ($1 \leq i \leq C$) output unit of the network can be trained with the ground-truth value $t$ by

$$
t_i(x) = \begin{cases} 1, & \text{if } x \text{ belongs to class } \omega_i, \\ 0, & \text{otherwise}. \end{cases} \quad (6.12)
$$

The network can be designed with the standard multi-layer feedforward structure, and trained using the backpropagation algorithm (Appendix C) to minimize the output error on the training set. Given input feature vector $x$, a trained network generates $C$ outputs $g_i(x)$ for $1 \leq i \leq C$. The best match thus corresponds to the output yielding the maximum value. Therefore, function $g_i$ can be seen as the discriminant function implied by the multi-output neural network.
In addition to neural network, Bayes decision theory can also be used for the multi-class face identification. In this case, the discriminant function is defined by \( g_i(x) = P(\omega_i | x) \) as \textit{a-posteriori} probabilities.

A common problem for applying a general pattern-classification technique (e.g. neural network, Bayes decision theory) to face identification is the so-called \textit{small-sample-size} problem. For these techniques, usually a large number of samples are required for each class to reliably estimate the decision boundaries. However, in practice, only a few samples are available for each individual, which causes great difficulties in deriving an accurate estimation of discriminant functions. In the following, we review a category of techniques that converts the face-identification problem to a two-class classification framework, which to a certain extent relieves the estimation problem as mentioned above.

6.4.2 Two-class formulation for classification-function design

As an alternative to the multi-class formulation, the face-identification problem can be also interpreted within a two-class framework, which was first proposed in [46]. Instead of seeking discriminations between all individuals, in a two-class formulation, an input feature vector is compared to a gallery vector to determine whether the two vectors originate from the same individual or different individuals. More specifically, given input feature vector \( x \) and a gallery feature vector \( x_i \), we determine whether the difference vector \( \Delta_i = x - x_i \) belongs to the within-class difference, denoted as class \( \Omega_w \), or the between-class difference, specified by class \( \Omega_b \). This also complies with our original problem statement for face identification, which is to best differentiate within-class and between-class variations of faces.

By using the difference vector for classification, the multi-class identification problem is converted to a traditional two-class classification problem. This offers the advantage that more samples can be collected for \( \Omega_w \) and \( \Omega_b \), even when there are only limited numbers of training samples for each individual. Suppose we have \( C \) individuals in the gallery, each of which has \( C_i \) training samples. We can have a total of \( 2C C_i (C_i - 1) \) training samples for class \( \Omega_w \) and \( 2C_i (C - C_i) \) training samples for class \( \Omega_b \). If \( C = 10 \) and \( C_i = 4 \), then we already have 120 and 1440 samples for \( \Omega_w \) and \( \Omega_b \), respectively.

In [46], the authors make use of Bayes decision theory for the classification between \( \Omega_w \) and \( \Omega_b \). Denote \( \Delta \) as a difference vector between a probe feature vector and a gallery feature vector, a similarity measure \( s(\Delta) \) is defined by the \textit{a-posteriori} probability, such that

\[
s(\Delta) = P(\Omega_w | \Delta) = \frac{P(\Delta | \Omega_w) P(\Omega_w)}{P(\Delta | \Omega_w) P(\Omega_w) + P(\Delta | \Omega_b) P(\Omega_b)},
\]

(6.13)
Employing the Maximum A-Posteriori (MAP) rule (see Appendix A), we can derive a classification function based on the following specification

\[
\begin{cases} 
\Delta \in \Omega_w, & \text{if } P(\Delta|\Omega_w)P(\Omega_w) > P(\Delta|\Omega_b)P(\Omega_b), \\
\Delta \in \Omega_b, & \text{otherwise.} 
\end{cases} 
\] (6.15)

During the identification, the input vector \(x\) is compared to each gallery vector \(x_i\) by evaluating the difference vector \(\Delta_i = x - x_i\). If \(x\) falls into the decision regions of more than one individual, then the similarity metric in Eqn. (6.14) is used to select the individual yielding the maximum \(s(\Delta_i)\). In [46], it is mentioned that an alternative for defining the similarity measure is formed by the Maximum Likelihood (ML) rule, given by

\[
s'(\Delta) = P(\Delta|\Omega_w). 
\] (6.16)

The key issue in performing the above-mentioned identification procedure concerns a reliable estimation of \(P(\Delta|\Omega_w)\) (and \(P(\Delta|\Omega_b)\) in the MAP case). This can be achieved by the subspace density-estimation technique introduced in Chapter 2, where a similar estimation problem is addressed for face detection.

In addition to the Bayesian classification, the two-class classification scheme can be also realized using alternative classification techniques, such as SVM [39].

### 6.5 Summary and conclusions

In this chapter, we have introduced the problem of face identification, which is the last principal processing stage in a face-recognition system. We have summarized the widely-used feature representations and classification functions in literature. The feature representation can be based on holistic features such as normalized image intensities or structural local features, such as transformation by Gabor wavelets. Based on a given feature representation, the actual face identification can be solved by a direct multi-class or two-class classification of difference vectors. In the former case, we have discussed several well-known techniques including Eigenface (PCA), Fisherface (LDA) and neural-network-based identification. In the latter case, we have discussed mainly Bayesian identification.

In the past years, holistic feature representation has received broad interest and many transform-based techniques have been developed such as Eigenface, Fisherface, Bayesian recognition and their various extensions. The structural local feature representation, represented by EBGM [77], gives promising results in several FERET tests and receives also much attention. Generally speaking, for face images containing sufficient feature details, structural local features give better between-class discrimination than holistic features [55][54].
This is because local features capture more important facial information and are less sensitive to environmental changes such as global illumination. However, structural local representation requires reliable facial feature extraction, which can be complicated with images of very low resolution or when the feature details are difficult to perceive (e.g. partial occlusion). In these cases, the holistic feature representation can be more effective.

In the next chapter, we present two cascaded classification techniques, one based on LDA with holistic features for small databases and another based on Gabor-wavelet-based local features for large databases.
7.1 Introduction

In the previous chapter, we have reviewed a number of well-known face-identification techniques based on various feature representations and classification functions. Although tremendous efforts have been spent on this topic, face identification in unconstrained real-life situations still remains a challenging problem, which is caused by the following arguments.

1. Complex face distributions. Due to the large variations in illumination, expression and poses, faces images from one person form a highly complex cluster in the image space, which is mingled with clusters from other persons, thereby causing significant complications in discrimination.

2. Small sample size problem. In many real-life scenarios, too few training images exist to characterize high-dimensional feature spaces. This causes instability of many popular classification algorithms, which require a good approximation of the sample distribution w.r.t. the true class distribution.

A considerable part of existing work on face identification employs a fixed classification function to discriminate all individuals (e.g. PCA, LDA, Bayesian or neural network) with a fixed feature representation (e.g. holistic or structural local features). However, using a fixed classification function and a fixed
feature representation is not optimal for completely discriminating all individuals, especially for large face databases. In this chapter, we propose a cascaded face-identification technique which aims at higher identification accuracy. At each stage in the identification cascade, we select a subset of the most promising candidates and dynamically adapt classification for these selected candidates. Based on the first stage of the identification, the algorithm selects e.g. 50% best-ranked face candidates, and the ground-truth face has a very high probability to be among these selected faces. At succeeding stages, the classification criterion is modified to discriminate the remaining classes for higher performance, based on the assumption that better discrimination can be achieved for a small population than a large population. In this way, the true match can be expected to ‘float’ up to the best-ranked position, enhancing the overall identification accuracy. Within this classification framework, we propose the following two realizations.

1. Cascaded LDA based on holistic features, where different between-class statistics are derived at each stage for refined classification.

2. Cascaded feature selection and matching based on Gabor features. At each stage, a new feature-selection procedure is used to choose a subset of feature components for each person, which is based on its discriminating capability within a given candidate set. An efficient matching function is defined as an accumulation of the differences among these selected components only. Due to the efficiency of the feature selection and matching procedure, this strategy is particularly suitable for large face databases.

The remainder of this chapter is organized as follows. In Section 7.2, we first present an overview of the proposed cascaded identification scheme. Section 7.3 and Section 7.4 describe two realizations of the scheme by adaptive LDA and adaptive feature selection and classification, respectively. The experimental results of both realizations confirm the effectiveness of the proposed scheme. Section 7.5 concludes this chapter.

7.2 Cascaded face-identification scheme

The cascaded face identification aims at enhanced identification by adopting a coarse-to-fine multistage identification procedure, where a series of classifiers are used to gradually reject unlikely candidates in the gallery database until the best match is found. During each stage, according to the outputs of the current classification, a similarity ranking of the face candidates is derived and promising candidates with high rankings are passed on to the next stage for refined classification (see Fig. 7.1).
The above scheme assumes that the identification accuracy can be improved by optimizing the discrimination between a constrained set of candidate faces. In other words, by gradually constraining possible candidates, we can optimize the discrimination between the most similar faces to the probe face, thereby enhancing the classification performance. Note that except for the first stage in the identification cascade, the classification at later stages is carried out by dynamically adapting to the remaining face candidates.

The key question in achieving an effective cascaded identification is how to dynamically perform an optimized discrimination within a specific candidate set. In practice, the following two strategies can be used (either separately or in combination):

1. Dynamically adapt the feature representation (the feature space).

2. Dynamically adapt the classification function.

Since the feature representation (or classification function) is customized for a different (reduced) candidate set for each stage, the identification accuracy can be expected to improve by each stage. On the other hand, since the dynamic adaptation of feature representation (or classification function) has to be performed online during the identification, the identification algorithm for each stage should have high training and identification efficiency. Classification schemes involving complex optimization procedures (e.g. SVM, neural networks) require high computation cost and would severely degrade the efficiency of the algorithm, if applied in a cascaded manner.

In the following, we present two implementations of the cascaded face-identification scheme. In Section 7.3, we apply the traditional LDA (Linear Discriminant Analysis) from Chapter 6 to the cascade scheme using holistic features. It derives different transformed spaces at each stage (adaptive classification) to better separate the most similar faces in the database. Afterwards, a more efficient implementation is presented in Section 7.4, which makes use of
Algorithm CascadedLDA

Input: A gallery set $C_0$ and probe face $u$.

Output: Identity of $t$.

1. Compute within-class scatter matrix $S_w$ base on $C_0$.
2. Let $C = C_0$.
3. for $i = 1$ to STAGES
4. Compute between-class scatter matrix $S_b^{(i)}$ based on $C$.
5. Find transform matrix $W$ to maximize $\frac{\det(W^T S_b^{(i)} W)}{\det(W^T S_w W)}$ as in the traditional LDA formulation.
6. Select candidates with the smallest distance to $u$ under $W$, which form a new candidate set $C'$.
7. Let $C = C'$.
8. return the best match found from the last stage.

Table 7.1: Description of the cascaded LDA algorithm.

Adaptive feature selection and matching based on the Gabor-wavelet features. Besides a state-of-the-art performance, the approach is attractive because it is extremely efficient even for large databases.

Related work. It is worth mentioning that a closely-related scheme to the coarse-to-fine principle adopted in our cascaded identification is to use divide-and-conquer strategy. In [39], this is exploited in a two-stage SVM-based identification framework. In this work, faces in the gallery database are pre-clustered into several groups during an offline training stage, and the classification can be optimized within each group. However, this is a static method with no adaptation to the probe face, and can be problematic with cases located near the cluster boundary.

With respect to the adaptive feature selection as proposed in Section 7.4, a closely-related feature-selection approach (called the reliable component scheme) is proposed in [71] to select a set of reliable binary features for privacy-protected biometric authentication. This scheme can be seen as a special case of the so-called CMMD-based (Class-specific Maximum Marginal Diversity) feature selection (see Table 7.2).

Our feature-selection approach in Section 7.4 is closely related with the Kullback-Leibler (KL) distance. In [74], the authors have proposed a so-called Maximum Discrimination Analysis (MDA) that is also derived from the KL distance. The MDA outperform the conventional linear discriminant analysis and is shown to be a good indicator of class discrimination.
7.3 Cascaded LDA

In this section, we apply the LDA (Linear Discriminant Analysis) to the cascaded framework. Recall from Chapter 6 that the transformed space induced by LDA relies on the between-class variances of the candidate sets. This motivates us to use LDA in the cascaded framework for refined discrimination.

More specifically, a single-stage LDA emphasizes the global between-class variances which may cause overlapping of locally neighboring classes. By ranking the similarity of candidate classes and picking a subset of best-ranked (locally closely located) classes, we can perform LDA on these selected classes, which can lead to refined classification results. In Table 7.1, we briefly summarize the cascaded LDA scheme. We first calculate the within-class scatter matrix $S_w$ based on all candidates $C_0$, which is reused for later stages. Here we assume that $S_w$ is constant for all classes, independent of the chosen training candidates. This improves the estimation stability for succeeding stages when fewer samples are available for estimating $S_w$. The adaptivity of the algorithm resides in the calculation of the between-class scatter matrix $S_b$, which is dependent on the candidate set used for the training. At the succeeding stages, a transformed space $W$ is derived to maximize the between-class discrimination between a set of adjacent classes in the previous transformed space, thereby enhancing the discrimination between these selected candidates.

Experimental results. We have employed the ORL face database [62] as an example to demonstrate the effectiveness of the cascaded approach. The ORL database contains 400 face images of 40 people with $92 \times 112$ pixel resolution. Some example images from the database are shown in Fig. 7.2. We have scaled the images to $23 \times 28$ pixels for efficient computation without losing important visual information. Afterwards, we have randomly selected three images per person for the training and the remaining images are used for testing. Twenty independent tests are performed with different partitions. During each test, we have employed a three-stage classification scheme, where
the second and the third stage selects the top 30% and 15% candidates, respectively\(^1\). The resulting identification error is shown at the left of Fig. 7.3. By using the cascaded scheme, the average identification error is reduced from 13% to 10% (23% reduction). We have also varied the number of training images and plotted the average identification results at the right of Fig. 7.3. The identification error is effectively reduced by using the cascaded scheme, although the reduction is less significant when using more training images.

The cascaded LDA can be effective for a small database such as ORL. For larger databases, the dynamic LDA training involved during the identification stage can be too costly. In the following, we propose a more efficient algorithm in the cascaded identification framework. The algorithm makes use of adaptive feature selection and classification and is extremely efficient even for large databases.

### 7.4 Cascaded feature selection and classification

In this section, we propose a cascaded approach with efficient dynamic classification. At each stage in the cascade, we select a subset of person-specific (class-specific) discriminative features w.r.t. the specific population. Based on the selected features, a simple feature-matching function is defined to locate the most similar faces in the candidate set. The key advantage of this method is the dynamic feature selection, which is adaptive to specific candidate sets.

---

\(^1\) The number of stages and the candidate set size at each stage are chosen to give both high identification performance and computation efficiency.
and can be carried out efficiently in a cascaded framework.

7.4.1 Motivation for adaptive feature selection

The intuition behind the adaptive feature selection is that features can have various importance when used for identifying different people. For example, the skin color may be a useful feature in discriminating a group of people of varying ethnics. However, for a group of people of the same ethnics, this feature may be less effective, and we may have to resort to other features for more effective identification, such as appearances of the eyes, etc. The adaptive feature selection chooses the most discriminative features to distinguish a selected group of candidates. In this way, the between-class variances within this specific population are enhanced, leading to better class separability.

Furthermore, the proposed feature-selection scheme facilitates efficient matching for the following reasons:

1. The feature selection is based on a so-called class-specific maximum marginal diversity criterion, which can be efficiently computed for each feature component;

2. Due to the feature selection, only a subset of the total features is selected, resulting in reduced processing cost;

3. The classification function only involves the accumulation of simple distances between selected feature components (Section 7.4.3), which further improves the identification efficiency.

In the following, we give a detailed illustration of the proposed approach. In Section 7.4.2, we first introduce the class-specific feature-selection algorithm. Following this, Section 7.4.3 provides the corresponding classification function based on selected features. Section 7.4.4 summarizes the complete algorithm and Section 7.4.5 presents the implementation and experimental results.

7.4.2 Class-specific feature selection

It has been advocated previously in literature that in a classification task, not all components $x_i$ ($1 \leq i \leq |x|$) of feature vector $x$ are equally important from a discrimination point of view. It was shown in [23] and [83] that by applying a careful feature selection, similar or better identification accuracy can be achieved with a much smaller number of feature components. In addition to high identification accuracy, we aim at designing a feature-selection algorithm that is also computationally efficient. In the following, we first outline the principles of our proposed feature-selection technique.
Principles of class-specific feature selection

In order to find the most important features, we rank the feature components according to their individual discrimination power, and select the ones having the best discrimination capability\(^2\). In this chapter, we choose a criterion based on Maximum Marginal Diversity (MMD) [73] to evaluate the discrimination capability of each feature component. The marginal diversity refers to the average distance between the class-conditional density of each feature component and its mean, which has been shown to be a very effective criterion in the context of visual recognition [73]. More specifically, suppose we have \(C\) persons (classes) in the gallery \((\omega_1, \ldots, \omega_C)\), given feature vector \(x\) with \(M\) components \(x_i\) \((1 \leq i \leq M)\), the Marginal Diversity (MD) of feature \(x_i\) is defined as

\[
MD(x_i) = \frac{1}{C} \sum_{k=1}^{C} \int p(x_i|\omega_k) \log \frac{p(x_i|\omega_k)}{p(x_i)} \, dx_i, \quad (7.1)
\]

where \(p(x_i)\) is the probability density function (pdf) of \(x_i\) for all samples and \(p(x_i|\omega_k)\) is the pdf of the component \(x_i\) for class \(\omega_k\). In Eqn.(7.1), the ratio between \(p(x_i|\omega_k)\) and \(p(x_i)\) measures the difference between \(p(x_i|\omega_k)\) and \(p(x_i)\), which conveys important information concerning the discriminating power of feature component \(x_i\).

According to the MMD principle, features with large MD values are selected as good features. Therefore, the MMD principle clearly favors those components where all the class-conditional densities are as different as possible. This is in accordance with our concept to search for large between-class separability.

Motivated by the aforementioned MMD specification, we extend this into a new scheme called Class-specific Maximum Marginal Diversity (CMMD). The difference between CMMD and MMD is that MMD selects feature components based on maximum discrimination between all classes, whereas CMMD is based on maximum discrimination between class \(\omega_k\) currently considered and all other classes. These components are the most distinct features that discriminate class \(\omega_k\) from the other classes. Our experimental results have shown that the application of the CMMD outperforms the original MMD criterion from [73] in the cascaded identification framework. In the following, we summarize the CMMD criterion for selecting the best feature set for person \(\omega_k\).

CMMD discrimination criterion. The best feature set characterizing class \(\omega_k\) should contain components with large class-specific maximum marginal di-

\(^2\)Sometimes, a subset of feature components is selected which maximizes a certain discrimination criterion [68]. Since the feature-subset selection usually involves high computation cost, it is not considered in this thesis.
Select components for class $\omega_k$

| $x_i$ | $x_j$ |

![Density plots](image)

**Figure 7.4:** Selection of discriminating components for class $\omega_k$. Component $x_i$ is more discriminative than $x_j$ because the difference between the pdf of class $\omega_k$ and that of the total average is larger for $x_i$ than for $x_j$.

versities (CMMD), defined as

$$MD_{\omega_k}(x_i) = \int p(x_i | \omega_k) \log \frac{p(x_i | \omega_k)}{p(x_i)} dx_i. \quad (7.2)$$

For class $\omega_k$, we rank the CMMD of each individual feature component and select the highest-ranked components to constitute the best feature set.

According to the above criterion, the best feature components for class $\omega_k$ should have a class-conditional pdf that is distant from the pdf of all classes. These are the most discriminating and informative features that distinguish class $\omega_k$ from other classes. This selection criterion is also visually illustrated in Fig. 7.4. In this example, we evaluate the discriminating power of components $x_i$ and $x_j$ for $\omega_k$. It can be noticed that the distance between $p(x_i)$ and $p(x_i | \omega_k)$ is much larger than that between $p(x_j)$ and $p(x_j | \omega_k)$. According to the CMMD principle, $x_i$ is a better feature than $x_j$ in terms of discriminating capability.

It is important to notice that the selected feature set for a specific class $\omega_k$ is closely related with the pdf $p(x_i)$ over all classes. For different sets of classes, we have different pdfs $p(x_i)$, which in turn influence the CMMD of class $\omega_k$. This is also in accordance with the following intuition: features that distinguish mostly person $\omega_k$ from a group of persons can be less useful in distinguishing the same person from another group of persons. This property enables adaptive feature selection based on a specific population and is crucial to the successful application of the adaptive feature selection within the cascaded identification framework.
Algorithm Estimation procedure for CMMD.
1. Use an N-bin histogram $h_i$ to estimate $p(x_i)$, where each bin is equally probable (see Fig. 7.6).
2. Calculate the span $N_k$ of samples of person $\omega_k$ w.r.t $h_i$, i.e. the number of bins of $h_i$ between the smallest sample value and the largest sample value of $\omega_k$.
3. Use $h^k_i$ consisting these $N_k$ bins to characterize $p(x_i|\omega_k)$.
4. Derive $MD_{\omega_k}(x_i) = \log(N/N_k)$ (according to Eqn. (7.3)).

Table 7.2: Practical estimation algorithm of feature importance based on the CMMD criterion.

Implementation issues of class-specific feature selection

The direct application of Eqn. (7.2) for selecting class-specific discriminative features requires the estimation of the pdf for both $p(x_i)$ and $p(x_i|\omega_k)$. Given sufficient face samples, $p(x_i)$ can be fairly reliably estimated using standard histogram-estimation techniques. However, the estimation of the class-conditional pdf $p(x_i|\omega_k)$ is often hampered by the limited sample size, when normally only a few (e.g. 2-5) sample images are available for each person. Instead of performing a brute-force histogram estimation, we propose an alternative estimation procedure as shown in Table 7.2.

In the estimation procedure from Table 7.2, we first estimate $p(x_i)$ by using EP (Equi-Probability) histograms. An EP histogram is a histogram where each bar represents the same probability fraction of the data. Fig. 7.5 illustrates the construction of an EP histogram with a simple example. Suppose we have a set of sample values for variable $x$, its corresponding traditional five-bin histogram is shown at the left of Fig. 7.5. Alternatively, we can also calculate the six quantiles ($0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \text{maximum}$) of the sample values and draw a histogram with five bars, each representing one-fifth of the total unity probability.

Fig. 7.6 shows an example of using EP histograms for the probability estimation. Assume we have estimated the EP histogram for $p(x_i)$, we then check the samples of $x_i$ from class $\omega_k$ as compared to the estimated histogram for $p(x_i)$. In the example from Fig. 7.6, the samples with the minimum and maximum values from $\omega_k$ fall in the 8-th and 10-th bins of the histogram for $p(x_i)$. In this case, we say that the samples from $\omega_k$ span $N_k = 3$ bins. Due to the limited number of samples for $\omega_k$, we assume that samples from person $\omega_k$ have an equal probability to fall into one of the $N_k$ bins spanned by the chosen samples. Accordingly, we use the range of the sample span to characterize $p(x_i|\omega_k)$. By applying the above estimation to Eqn. (7.2), it
follows that

\[ MD_{\omega_k}(x_i) = \sum_j h^k_i(j) \log \frac{h^k_i(j)}{h_i(j)} \simeq N_k \times \frac{1}{N_k} \log \frac{1/N_k}{1/N} = \log \frac{N}{N_k}, \]  

(7.3)

where \( j \) is the index of the histogram bins. Our experimental results have shown that the estimation procedure gives more reliable performance than the bruteforce histogram estimation of \( p(x_i) \) and \( p(x_i|\omega_k) \).

Let us assume that we have selected a subset of features for each class based on its importance computed by Eqn. (7.3). For each class \( \omega_k \), the selection results are stored in a binary vector \( w_k \), where each vector element \( w_{ki} \) is unity if the \( i \)-th component is selected and zero otherwise. In the following, we present the matching function used to measure the similarity between a probe face \( u \) and all the candidate faces in a gallery set.

### 7.4.3 Similarity function

Given probe face vector \( u \) containing a number of components indexed by \( i \), we define a simple similarity function between \( u \) and class \( \omega_k \) as follows:

\[ S(u, \omega_k) = \frac{\sum_i w_{ki}M(u_i, \omega_k)}{\sum_i w_{ki}}, \]  

(7.4)

where \( M(u_i, \omega_k) \) is a ternary distance indicator based on the \( i \)-th feature component. More specifically, \( M(u_i, \omega_k) \) is defined as follows:

\[ M(u_i, \omega_k) = \begin{cases} 
1, & \text{if } u_i \text{ is within the range covered by } h^k_i, \\
0, & \text{if the difference between } u_i \text{ and } h^k_i \text{ is smaller } p, \\
-1, & \text{otherwise}.
\]  

(7.5)

In the above definition, \( h^k_i \) is the EP histogram for the \( i \)-th component of class \( \omega_k \) (see Section 7.4.2) and \( p \) is a threshold. The matching function \( M \)
bases its output on the proximity between \( u_i \) and \( h^k_i \). Since the region close to the histogram boundary is usually ambiguous, we define a threshold \( p \), and the matching function outputs zero to indicate uncertainty if the difference between \( u_i \) and \( h^k_i \) is below \( p \) (see Fig. 7.6). The ternary indication simplifies the computation of function \( S \) in Eqn. (7.4). As a result, the similarity function \( S \) resembles a correlation parameter that ranges from \(-1\) to \(1\).

Note that the similarity measure defined in Eqn. (7.4) is computed on the selected components only (when \( w_{ki} = 1 \)). Based on Eqn. (7.4), we can rank the candidates based on their similarity to the probe face \( u \). To find the best match of \( u \) from a set of \( C \) person candidates, we only need to find person \( \omega_m \), such that \( S(u, \omega_m) = \max\{S(u, \omega_k) | 1 \leq k \leq C\} \).

Up to now, we have discussed the identification algorithm based on adaptive feature selection and matching. In the following, we apply the class-specific feature selection based on CMMD and the similarity function \( S \) in the cascaded identification framework as introduced in Section 7.2.

### 7.4.4 Complete cascaded algorithm

In Table 7.3, we present the cascaded face-recognition algorithm based on the CMMD feature selection from Section 7.4.2 and the corresponding matching function \( S \) from Section 7.4.3. The algorithm is a direct application of adaptive feature selection and classification to the cascaded identification framework.

---

3In practice, a suitable value for the threshold \( p \) is found empirically.
Cascaded feature selection and classification

**Algorithm** 
*CascadeFeatureSel*

**Input:** A gallery set $C_0$ and probe face $u$.

**Output:** Identity of $u$.

1. Let $C = C_0$.
2. for $j = 1$ to STAGES
3. for all class $\omega_k$ in $C$
4. Select a set of discriminative features for $\omega_k$ with the largest CMMD based on statistics of $C$.
5. Compute the similarity $S$ between $u$ and $\omega_k$ using Eqn. (7.4).
6. Let $C'$ contain a subset of candidates that are most similar to $u$.
7. Let $C = C'$.
8. return the best-match found from the last stage.

---

**Table 7.3:** Description of the cascaded feature selection and classification algorithm for identification of probe face $u$.

The feature-selection criterion is *depending* on the distribution of the candidate set $C$. Therefore, it is possible to adaptively select the most discriminating features w.r.t. $C$, so that the identification accuracy can be improved. Let us now briefly discuss some performance and technical aspects related to the cascaded identification algorithm in Table 7.3.

With respect to the efficiency, the proposed approach treats each feature component independently, which simplifies the multidimensional classification problem. Furthermore, in our experiments, normally only 30%-50% of all available components are selected and used, which further reduces the computation cost during the matching. In case that only two bins are used to estimate $h_i$, the matching function can be implemented with simple bit-based operations to further accelerate the processing [32][72].

Another aspect is about the feature representation. It should be noticed that the feature selection and matching we have discussed in this chapter is performed on individual components of the feature vector, in order to facilitate efficient operation. However, this strategy is only effective when the feature components are relatively stable for each person. For example, the holistic feature representation as used by PCA (LDA) is not a good representation in this context, since the pixel values at a given image position can be too easily influenced by factors such as image mis-alignment, illumination and pose.

In the following section, we demonstrate the effectiveness of the proposed approach by applying it to face-identification tasks.
7.4.5 Implementation and experimental results

This section presents the experimental results on the cascaded face identification based on adaptive feature selection and matching. The dataset we use is a subset of the FERET face database [54], which contains all individuals in the FERET database that have at least four sample images. Altogether 1167 face images of 237 persons are collected, which are preprocessed by the cascaded face detector from Chapter 3 and the cascaded feature extractor from Chapter 5. For each image sample, we derive the Gabor-wavelet-based local feature representation as discussed in Chapter 6. In our experiments, a total of 51 facial feature points are automatically extracted for each face. By applying a Gabor convolution, for each face, we obtain a feature vector with $51 \times 40 = 2040$ feature components.

In our experiments, we have randomly selected two sample pictures per person for training, and the remaining images are used as probe samples. The training images are selected such that they have a different capture time than the probe samples. This simulates a typical real-life face-identification scenario, where we need to predict face identities given only training images captured in a different situation. Fig. 7.7 portrays some examples of the training and probe images. Note that this is a more difficult scenario than commonly-adopted partitioning strategies from literature based on the FERET dataset.
We evaluate now several algorithms. For each algorithm, we perform 10 random tests, each of which adopts a different partition of the training and probe images.

As a first step, we evaluate the performance gain by using the cascaded face identification. To this end, we implement the single-stage class-specific classification based on adaptive feature selection (denoted as S-PSC) and the cascaded version (denoted as C-PSC). For C-PSC, we use three stages where the adaptive feature selection and classification are performed sequentially on three sets: a full candidate set, the top 50% candidates and the top 10% candidates. For example, after the first stage, only around 82% rank-1 identification rate is achieved, whereas in all test cases the true-match is among the highest 50% ranked candidates. After applying the second-stage classification, the rank-1 identification rate is 85%, whereas in 95% test cases the true-match is among the highest 10% candidates. The final stage further improves the rank-1 identification rate to around 89%. A typical rank performance curve is depicted in Fig. 7.8 to illustrate the performance gain by C-PSC. In summary, using the cascaded identification scheme effectively reduces the identification error by up to 33% for the rank-1 performance.

As a next step, we compare C-PSC with a number of well-known face-identification algorithms from literature. The results are portrayed by Fig. 7.9. The algorithms that we have used include simple distance-based measure,

---

4^Note that the feature selection result for the first stage is the same for any given probe face. Therefore it can be pre-computed to reduce the identification cost.
Figure 7.9: Rank performance comparison between C-PSC and other face identification algorithms. Top: rank performance on FERET dataset with Gabor feature representation. Bottom: rank performance on the same dataset with holistic feature representation.

PCA, LDA and Bayesian classification (all discussed in Chapter 6).

The rank performance of these algorithms as compared to our proposed ap-

\footnote{The PCA is performed on the Gabor vectors, with a reduction of the dimensions down to 100.}
7.5. Conclusions and discussions

Figure 7.10: *Summary of the cascaded feature selection and matching scheme.*

A novel approach is depicted at the top of Fig. 7.9. From Fig. 7.9, we can see that in this experimental scenario, PCA gives the worst performance. The simple cosine distance achieves far better results than PCA. The Bayesian and LDA techniques make use of inter-class and intra-class differences and achieve improved recognition performance, where the absolute identification rate increased with 3% (rank-1 performance) as compared to the basic cosine measure. Note that the C-PSC approach clearly outperforms all other approaches, adding around 2-3% to the absolute identification rate compared to the Bayesian or LDA approaches.

Due to the highly efficient training and matching procedure of PSC, the multistage C-PSC scheme is very efficient and can be readily applied in practice for online identification. The identification cost involved is mainly affected by the size of the gallery size, the number of stages and the candidate selection rate during each stage. The latter two parameters can be tuned to give the best trade-off between the identification accuracy and efficiency.

For comparison purposes, the performance of several algorithms on the holistic feature representation is depicted at the bottom of Fig. 7.9. It can be seen that using a Gabor feature representation gives much better classification results (over 20% improvement) for all algorithms, which also justifies our decision to adopt the Gabor feature representation in our cascaded identification system.
7.5 Conclusions and discussions

In this chapter, we have proposed a cascaded face-identification framework using a set of adaptive classifiers. During each stage in the identification cascade, a refined classification is performed on a selective set of promising candidates, achieving an optimized discrimination for this specific group of candidates. We have shown that the cascaded approach can be applied beneficially to both small and large databases. We have evaluated two new cascaded techniques based on LDA and the adaptive feature selection.

With respect to small databases, we have demonstrated the effectiveness of the proposed cascaded LDA. The experimental results on the ORL database show that by using a three-stage cascaded LDA, the rank-1 identification error can be reduced by 23%. The cascaded LDA requires dynamic training during the identification and the training procedure of LDA can be costly for a large gallery.

In order to apply the proposed scheme to large-scale databases, we have presented a cascaded identification scheme which has inherent adaptivity and even more important, it can be efficiently implemented (see Fig. 7.10). The scheme is based on adaptive feature selection and matching, where for each person a set of most representative features is chosen based on the class-specific maximum marginal diversity. The matching of a given face to a gallery face is based on measuring the difference between them on the selected features. To this end, we have introduced a simple similarity function for fast matching. For a difficult test scenario on the FERET dataset, the identification error is reduced by at least 18% as compared to the state-of-the-art techniques in literature.
Conclusions, Applications and Outlook

8.1 Thesis summary and conclusions

In this thesis, we have addressed various topics in face recognition. These topics are divided into three key parts: face detection, facial feature extraction and face identification. For each part, we have reviewed the state-of-the-art techniques in literature and proposed a set of novel techniques based on cascaded structures. During the design of each technique, we aim not only at improving the individual algorithm performance, but also at optimizing the algorithm architecture as a way to enhance the overall system performance w.r.t. accuracy, efficiency and robustness. More specifically, we have proposed three cascaded frameworks for the principal stages in face recognition.

1. For face detection, we have contributed with a cascaded detection structure that is based on neural networks. We have proposed a new training technique to form a neural-network ensemble, which is used as the basic building block in a detector cascade. Consequently, a series of neural-network ensembles are cascaded with scalable structural complexity. The cascade gradually rejects non-face candidates using different ensembles, enabling fast and reliable face detection.

2. For facial feature extraction, we have defined two basic principles for constructing a cascade of model-based algorithms. These principles are: (1) the capture range of each algorithm should cover the output range of
its preceding algorithm, and (2) the algorithms should have a decreasing extraction error. We have also proposed three model-based algorithms to form a cascade, namely, sparse graph search, component-based texture fitting and component-based direct fitting. The performance of each algorithm is tuned based on the output statistics of its predecessor, which preserves the above design principles.

3. For face identification, we have proposed a new online cascaded identification, which adopts multiple steps to gradually constrain the candidate sets for improved identification accuracy. We have applied LDA and a new adaptive feature selection algorithm to this cascaded framework.

Let us now briefly discuss the major results of the key chapters in this thesis.

In Chapter 2 and 3, we mainly focused on the algorithm design for face detection. Two cascaded detectors were proposed, i.e. a heterogeneous detector based on the analysis of different image features (e.g. color, geometry) and a homogeneous detector based on neural networks with similar structures. In the latter approach, we have constructed a family of neural networks of similar topology and arranged them in two layers. The first layer is a novel parallel ensemble structure for improved detection accuracy, whereas the second layer is a cascaded structure of various ensembles for improved detection speed. The structures of the networks are chosen such that they attempt to capture the prominent facial feature structures. With respect to performance, the cascaded-ensemble technique was compared to a number of state-of-the-art detectors from literature, and achieved a high detection accuracy (94.4% on the CMU+MIT dataset) with a significantly reduced computation cost. It can process 4-5 image frames per second, which is a significantly reduced computation cost as compared to using the most complex subdetector alone (several minutes per image). Furthermore, the proposed ensemble structure is most suitable to be implemented in parallelized hardware architectures, either in multiprocessor layout or with reconfigurable hardware cells.

The proposed two-layer algorithm architecture for the face detector (ensemble and cascade) is not restricted to neural networks. It can be generalized to other embedded classifiers with classification diversity and complexity scalability. Some possible classifier choices other than neural network are SVM and Bayesian classifiers.

In Chapter 4 and 5, we have discussed the algorithm design for facial feature extraction. The first proposal we have developed is called H-ASM, which extends the Active Shape Model by using Haar-based local feature modeling. For each feature point, we represent the context of this point by capturing a local sample block, which is transformed to the Haar domain to obtain an efficient representation. The H-ASM achieved a doubled capture range and a 17%
accuracy improvement compared to conventional ASM. Secondly, in Chapter 5, we have presented a cascaded framework for feature extraction employing a sparse graph model, a component-based texture model and a component-based appearance model. The motivation for this concept is that we tend to perceive objects first globally and then focus more on details. From a technical point of view, we can use a set of cascaded algorithms, where the algorithms of the first steps focus on more global and rigid feature properties, while the succeeding algorithms concentrate on more local and detailed properties. In this way, the feature locations can be globally estimated with high robustness and the more detailed descriptions can be obtained in a step-wise manner. We have obtained an 8% accuracy improvement compared to H-ASM and a 24% improvement compared to ASM. Another advantage is the guaranteed global optimum at the first stage, such that the extraction is not sensitive to initial model deviations.

It should be noted that each component algorithm in the extraction cascade has to satisfy the design requirements on **capture range** and **extraction accuracy**. For example, a component algorithm has to take into account the output range of its previous algorithm and adapt its own performance by e.g. adjusting training parameters. In this way, the overall performance can be optimized. This is fundamentally different from a simple multistage approach, where a set of **independent** algorithms are heuristically combined.

The proposed approach is also relatively robust with varying facial appearances w.r.t. ethnic group and facial expression. However, since we have not addressed the varying face poses and partial feature occlusions, it is worth studying these aspects in future work.

In Chapter 6 and 7, we have explored the algorithm design for face identification. More specifically, we proposed a selective cascade to gradually reduce the candidate size and derive a customized classification for each candidate set. We have adopted LDA for this framework and demonstrated the effectiveness of the approach (23% reduction of identification error rate). The main contribution in Chapter 7 is a novel classification scheme which is particularly suitable for large databases. We have proposed a new criterion called class-specific maximum marginal diversity (CMMD) for adaptive feature selection. For each feature vector generated (e.g. Gabor-based features), we select important features according to their discriminating power (CMMD) and the matching is carried out on those features based on a computationally-efficient similarity function. When used in the overall framework, our cascaded identification outperforms several state-of-the-art identification algorithms (e.g. 18% reduction of identification error).

The proposed feature selection approach also demonstrates the importance of selecting the most relevant features for face identification. It would be interesting to further exploit the possibilities to improve the face-recognition
performance via refining the feature representation. This potentially gives additional gains besides pursuing a more powerful classifier for identification.

The content of this thesis has shown that a cascaded approach is attractive for all key stages in face recognition. With respect to efficiency, a cascade can be used to be selective in the initial processing, so that the succeeding processing stages are focusing on more detailed properties. With respect to accuracy, a cascade can be employed to gradually refine the processing, so that the final results are of sufficiently high quality.

Let us now briefly comment on the applicability of our work for embedded systems in comparison with state-of-the-art literature. First, the algorithms presented in this thesis are efficient enough and of sufficiently high quality to initiate experimental embedded system implementations. Second, when looking to literature, it can be concluded that despite the many advances in face-recognition research, most of these advances are not suitable for consumer applications, because of the inherent complexity of the algorithms. For our work, we have taken efficiency and accuracy as a leading design principle from the beginning, and applied it to all stages of the face recognition. We have been able to make significant progress in improving the efficiency while maintaining the high performance. In this aspect, the work of this thesis is certainly unique and can form an important basis for further embedded application design for face recognition.

8.2 Evaluated applications

The research work described in this thesis was executed within several projects. As a result, the proposed face-recognition techniques were embedded into several application environments and evaluated. Since this provides useful experimental experience and interesting aspects about system integration, we briefly discuss these applications below.

8.2.1 Face recognition for smart homes

One of the first applications of our proposed face-recognition system was aiming at a smart-home environment, which was carried out in the HomeNet2Run project [59]. In a smart home, it is desirable that the electronic devices can consciously sense and understand their surroundings and adapt their services according to the contexts (e.g. environments). One important aspect in this context awareness is user identification, in which machines can actively identify the users present in their environments and customize services accordingly. Face recognition provides a natural visual interface for such context-aware applications, endowing devices’ perceptual abilities to communicate with people, e.g. to recognize people and remember their preferences and behavior [51].
Fig. 8.1 gives an abstract description of the perceptual interface of home devices, which has been realized in the HomeNet2Run project. In this scenario, a face-recognition module is embedded into consumer devices, such as televisions, video and music players, where the recognition module continuously monitors its surroundings and captures user presence. By the automatic identification of the user’s face, the home devices can tailor services (e.g. selection of favorite TV or music programs) for each user. In addition, the face-recognition module is further connected to a home network using heterogeneous network connections (e.g. Ethernet, IEEE-1394). Consequently, the face-recognition results can be shared and reused by various devices connected to the network.

The embedded face-recognition system consists of a kernel component performing the face recognition and an interfacing component providing a uniform interface to different hosting devices. Fig 8.2 portrays the operation flow of the complete recognition system. The system operates on live-feed video from a low-cost consumer camera and performs the identification in near real-time. For each frame, the system detects existing faces and keeps track of the located faces (Fig. 8.2, stage A). From all detected faces, those faces with large rotations (in-plane or out-of-plane rotation) or low-resolutions are discarded. The remaining face images are selected as key samples for further feature extraction, face normalization and face identification (Fig. 8.2, stages B, C, and D).

Note that in the smart-home scenario, multiple users exist and the face-recognition system needs to keep track of an active face at a certain moment. Since motions of faces from the same user form a smooth trajectory, a new user can be easily identified by detecting a ‘break’ of this trajectory. For a user entering the scene for the first time (Fig. 8.2, stage E), a number of key samples need to be processed prior to making an identification decision. For
users already present in the scene, the identification result for the current frame only contributes to the accumulated recognition confidence (Fig. 8.2, stage F). In this way, a more stable identification can be achieved for a sequence of face images.

The above face-recognition system has been embedded in home devices to actively monitor the user actions and the results are integrated in a home server application which maps the user identity to the user profiles. Accordingly, customized services can be invoked on a personalized basis. At the left of Fig. 8.3, we show some typical identification results and at the right of Fig. 8.3, we show an example web interface for smart TV-program selection. A successful demonstration of this system was presented at the IFA consumer electronics exhibition in 2003 [8]. We used a database of 25 people and performed tests under a variety of environments. The system achieves an average recognition rate of 97% with a 3-4 frames/second processing speed.
8.2.2 Face recognition for secure biometric identification

In [32] and [72], a privacy-preserving face-verification scheme was proposed that generates privacy-protected feature templates (feature vectors). The feature vectors are first generated using techniques from Chapter 3 and Chapter 5. These unprotected vectors are then transformed into binary vectors based on their statistical properties. The scheme for extracting the binary vectors is combined with a so-called helper data scheme [71], which leads to renewable and privacy-preserving facial templates. A block diagram illustrating the template protection scheme is portrayed in Fig. 8.4.

8.2.3 Face recognition for video surveillance

In the CANDELA (Content Analysis and Network DELivery Architectures) project [58], a distributed surveillance system was constructed to facilitate smart retrieval of video data. In the corresponding system architecture, a number of VCA (Video Content Analysis) algorithms are integrated into one framework, which communicate with the video database via distributed processing middleware [76]. Our proposed face-recognition system is embedded as one VCA module and generates in real-time MPEG-7 compliant metadata, which includes important information such as the location of a face and its corresponding timestamp, etc. The metadata facilitates efficient database storage and retrieval. A demo of the system with our face-recognition module was successfully demonstrated at the ICME (International Conference on Multimedia & Expo) exhibition in 2005.
8.3 Future work

Despite the successful applications of our work as described in the previous sections, we identify in this section several interesting aspects that need further investigation.

1. **Robustness for various poses.** In this thesis, we have mainly focused on the frontal-view face recognition. In some applications, where users may not be cooperative with the system (e.g. video surveillance), it is crucial for the system to deal with faces having varying poses. In principle, the current frontal-view-based techniques can be extended to deal with multiple poses by modeling different views separately. Another solution to the pose problem is to use 3-D face modeling [6]. The 3-D modeling has shown good performance in dealing with head poses in FERET tests [53].

2. **Identification accuracy with time elapse.** A major difficulty for face identification is formed by the potential large within-class variances caused by a number of factors such as illumination and expression. Among these factors is the time elapse (also called aging effect), which has shown to severely deteriorate the performance of most existing algorithms. In [36], the authors elaborated on building a face-aging model using a similar reasoning as in the statistical shape model. By using a
set of training samples with varying ages, the variation direction of face appearances for aging can be derived. The model showed an improvement of the face recognition performance from 50-60\% to 60-70\%, which is however still insufficient for practical use.

3. **New hardware and software implementations.** In this thesis, computation efficiency is an important performance goal in the algorithm design. It is interesting to further exploit implementation issues to deploy the algorithms on various embedded platforms. For example, for mobile applications, it would be interesting to exploit very efficient mapping of algorithms on DSP media processors [33]. It would also be useful to optimize algorithms for embedded processing in cameras to facilitate intelligent video surveillance.
Bayes Decision Rule

Bayes decision theory defines a classification rule under the assumption that the underlying probabilities of class samples are known. More specifically, suppose we have two classes $\omega_1$ and $\omega_2$, and we use feature vector $x$ to represent each sample. In the following, we define several important concepts.

1. $P(\omega_i)$ ($i = 1, 2$): a-priori probability of class $\omega_i$.

2. $p(x|\omega_i)$ ($i = 1, 2$): class-conditional probability density function of class $\omega_i$, describing the distribution of the feature vector in class $\omega_i$. $p(x|\omega_i)$ is also called the likelihood function of $\omega_i$ w.r.t. $x$.

3. $P(\omega_i|x)$ ($i = 1, 2$): The conditional probability that the feature vector belongs to class $\omega_i$ given measurement $x$.

Based on the above notations, Bayes decision rule states that

If $P(\omega_1|x) > P(\omega_2|x)$, $x$ is classified to $\omega_1$.
If $P(\omega_1|x) < P(\omega_2|x)$, $x$ is classified to $\omega_2$. \hspace{1cm} (A.1)

The above classification rule is also called maximum a posteriori (MAP) decision rule. It can be proved that the Bayes decision rule minimizes the classification error probability \cite{17}\cite{68}, which is defined as

$$P_e = P(x \text{ is classified to } \omega_2, \omega_1) + P(x \text{ is classified to } \omega_1, \omega_2). \hspace{1cm} (A.2)$$

According to Bayes rule in probability theory, we have

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)}, \hspace{1cm} (A.3)$$
where \( p(x) \) is the probability density function of \( x \). Consequently, Eqn. (A.1) can be rewritten as

\[
\begin{align*}
\text{If } p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2), & \quad x \text{ is classified to } \omega_1. \\
\text{If } p(x|\omega_1)P(\omega_1) < p(x|\omega_2)P(\omega_2), & \quad x \text{ is classified to } \omega_2.
\end{align*}
\]

Therefore, the estimation of \textit{a-posteriori} probability \( P(\omega_i|x) \) is then converted to the problem of estimating probability density function \( p(x|\omega_i) \) for each class. In practice, the classification can be also checked by the likelihood ratio

\[
\frac{p(x|\omega_1)}{p(x|\omega_2)} \leq \lambda, \tag{A.5}
\]

where \( \lambda \) is a threshold.

If we further assume that the two classes have equal \textit{a-priori} probabilities \( (P(\omega_1) = P(\omega_2)) \), then the classification rule only involves the comparison between \( p(x|\omega_1) \) and \( p(x|\omega_2) \). This is also called \textit{maximum likelihood} (ML) decision rule in literature.

The generalization of the above reasoning to the multiclass case is straightforward. Suppose we have \( C \) classes \( \omega_1, \omega_2, \ldots, \omega_C \), feature vector \( x \) is assigned to class \( \omega_i \) if

\[
P(\omega_i|x) > P(\omega_j|x), \forall j \neq i. \tag{A.6}
\]
Appendix B

Principal Component Analysis

Problems tend to arise in classification problems involving high-dimensional feature vectors (the so-called *curse of dimensionality*), which usually cause estimation difficulty and increase processing complexity. Principal Component Analysis (PCA) is a dimension reduction technique which *linearly* maps original high-dimensional feature vectors to low-dimensional vectors while retaining as much as possible the variation present in the original data set.

Suppose we have original feature vector \( \mathbf{x} \) of dimension \( M \), and denote \( C \) is the covariance matrix of a set of samples for \( \mathbf{x} \). The aim of PCA is to find a transformation matrix \( \Phi \) such that an \( N \)-dimensional feature vector \( \mathbf{y} \) can be generated by

\[
\mathbf{y} = \Phi^T (\mathbf{x} - \bar{x}).
\]  

(B.1)

Here \( \Phi \) is of dimensions \( M \times N \) (\( N \ll M \)), and \( \bar{x} \) is the mean vector. In order to derive \( \Phi \), PCA first defines \( \Psi \), which contains the Eigenvectors of the covariance matrix \( C \), given by

\[
\Psi^T C \Psi = \Delta,
\]  

(B.2)

where \( \Delta \) is a diagonal matrix containing the Eigenvalues resulting from the Eigen-decomposition. We then select the \( N \) largest Eigenvalues \( \lambda_1, \lambda_2, \ldots, \lambda_N \) and their corresponding Eigenvectors \( \Psi_1, \Psi_2, \ldots, \Psi_N \). These Eigenvectors are then used as the basis vectors of \( \Phi \) (also called *Principal Components*), thereby

\[
\Phi = \{ \Psi_1, \Psi_2, \ldots, \Psi_N \}.
\]  

(B.3)
Consequently, any given feature vector \( \mathbf{x} \) can be mapped to a low dimensional feature vector \( \mathbf{y} \) by applying Eqn. (B.1). Conversely, a new vector \( \hat{\mathbf{x}} \) can be reconstructed by the following:

\[
\hat{\mathbf{x}} = \bar{\mathbf{x}} + \sum_{i=1}^{N} y_i \Phi_i, \tag{B.4}
\]

where \( y_i \) is the \( i \)-th component of \( \mathbf{y} \). It can be proved [68] that PCA minimizes the mean square reconstruction error, denoted by \( \| \mathbf{x} - \hat{\mathbf{x}} \| \).
Multi-Layer Feedforward Neural Network

Neural network has been used widely as a learning tool to approximate arbitrary functions based on a set of training samples. Various architectures and training methodologies have been proposed in literature to solve different problems. In the following, we briefly review one of the most commonly-used architectures: the multi-layer feedforward neural network.

C.1 Network structure

The basic building block in a multi-layer feedforward network is a sigmoid neuron, as depicted in Fig. C.1(a). The neuron computes a linear combination of its inputs and then applies a threshold by a nonlinear sigmoid function. Given inputs \( \{x_1, x_2, ..., x_n\} \), the output of the neuron is

\[
o = \sigma(y) = \frac{1}{1 + e^{-y}}, \quad \text{where} \quad y = \sum_{i=0}^{n} w_i x_i \quad \text{and} \quad x_0 = 1 \quad \text{(bias parameter)}.
\]

(C.1)

Multiple sigmoid neurons can be constructed in a multi-layer structure, which contains the following elements:

1. A set of inputs \( x_i \);

2. One or more hidden layers: each hidden layer consists of a number of neurons, each of which receives inputs from its previous adjacent layer;
3. An output layer: the output layer consists of a number of output neurons, each of which receives inputs from its adjacent hidden layer.

In Fig. C.1(b), we depict an example of two-layer neural network with \(d\) inputs and \(p\) outputs.

**C.2 Feedforward calculation**

Given a network architecture as above, we derive in the following the network output values \(o_j\) (\(1 \leq j \leq p\)) based on a given set of inputs \(x_i\) (\(1 \leq i \leq d\)) and the connection weights between the network units (inputs or neurons). For illustration brevity, we only give the derivation for networks with one hidden layer (see Fig. C.1(b)). However, a similar line of reasoning can be also applied to networks with multiple hidden layers. For the network depicted in Fig. C.1(b), each output \(o_k\) is derived using the following calculation

\[
o_k = \sigma \left( \sum_j w_{jk} o_j + w_{0k} \right), \quad \text{where} \quad o_j = \sigma \left( \sum_i w_{ij} x_i + w_{0j} \right). \quad (C.2)
\]

In the above, \(w_{ij}\) represents the connection weight between the \(i\)-th input and the \(j\)-th neuron in the hidden layer, and \(w_{jk}\) represents the connection weight between the \(j\)-th neuron in the hidden layer and the \(k\)-th neuron in the output layer. \(w_{0k}\) and \(w_{0j}\) are bias parameters. Since each neuron does not have to be connected to every unit (neuron or input) in its previous layer, the calculation in Eqn. (C.2) only needs to be performed on those valid connections. Note that Eqn. (C.2) gives a feedforward calculation, where the output values are derived by propagating the inputs forward through the network. The choice of the parameters for a network (the connection weights) is determined by
Algorithm Backpropagation algorithm for training a multi-layer NN.

Input: A set of training samples \((x, t)\), where \(x\) gives the input values for the NN and \(t\) gives the target output values. A neural network with \(d\) inputs, one hidden layer with \(m\) hidden neurons, and \(p\) outputs. Learning rate \(\eta\) and momentum \(\alpha\).

1. Initialize all weights \(w_{ij}\) and \(w_{jk}\) to small random numbers.
2. for each training sample \((x, t)\)
   3. Do a feedforward calculation of output \(o_k\) as in Eqn. (C.2).
   4. for each output neuron \(k\)
      5. \(\delta_k \leftarrow o_k(1 - o_k) (t_k - o_k)\)
   6. for each hidden neuron \(j\)
      7. \(\delta_j \leftarrow o_j(1 - o_j) \sum_k w_{jk} \delta_k\)
   8. for each weight \(w_{ij}\) and \(w_{jk}\)
      9. \(w_{ij} \leftarrow w_{ij} + \Delta_{ij}\), and \(w_{jk} \leftarrow w_{jk} + \Delta_{jk}\),
      where \(
      \Delta_{ij} = \eta x_i + \alpha \Delta_{ij}^p
      \) and \(
      \Delta_{jk} = \eta o_j + \alpha \Delta_{jk}^p
      \).
   10. \(\Delta_{ij}^p = \Delta_{ij}, \Delta_{jk}^p = \Delta_{jk}.\)

Table C.1: Backpropagation algorithm for training a three-layer feedforward neural network. Here \(\Delta_{ij}^p (\Delta_{jk}^p)\) records the value of \(\Delta_{ij} (\Delta_{jk})\) in the previous iteration. Momentum \(\alpha\) is used to relate the previous update information with the current update.

A training procedure based on a set of input and output samples (see next Section C.3). It is worth pointing out that due to the nonlinearity implied by the sigmoid neurons, a multi-layer feedforward network can approximate arbitrary nonlinear functions implied by the training set [45].

C.3 Network training

The parameters of a multi-layer feedforward network (thereby the classification function implied) can be derived by minimizing a pre-defined error function over a set of training samples. This can be achieved by an iterative backpropagation learning scheme. In this scheme, the training samples are sequentially fed to the network. For each sample, the current output error is computed with respect to the target output value. Accordingly, the neuron connections are updated to correct this error. In Table C.1, we give the backpropagation algorithm to traverse the training set in one iteration (also called one epoch in literature). This procedure is usually iteratively carried out until the network reaches a stable status.

Since the backpropagation algorithm aims at minimizing the classification
error over the training samples, the results may not generalize well to unseen test samples, which is usually referred as the *generalization performance* of a classification technique. Suppose we use a validation data set that is independent from the training data to measure the classification performance of the neural network for unseen samples. In Fig. C.2, we depict the relation between the classification error on the training set (and the validation set) and the iteration epochs. The validation error initially decreases with the training error, but at some later point it starts to increase. This is because the network becomes *overly tuned* to the specific training data, thus affecting the generalization performance. Therefore, in practice, a separate validation set is usually adopted to determine the point when the training iterations need to terminate.

**Figure C.2:** *Classification error on the training set and the validation set versus the training iteration epochs.*
Support Vector Machines

Support Vector Machine (SVM) is a technique to generate a hypersurface to optimally separate two given classes $\omega_1$ and $\omega_2$. Informally speaking, the generated hypersurface splits the two classes such that the distance between the hypersurface and the closest class samples (called margin) is maximized. In the following, we first introduce SVM solution to separate two classes by a linear boundary (hyperplane). Following that, we extend this solution to the nonlinear case by applying the so-called kernel mapping.

D.1 SVM: linear classification

In this section, we first introduce the basic motivation for SVM to optimally discriminate two linearly separable classes. This is later extended to cope with linearly non-separable cases.

D.1.1 Case 1: linearly separable classes

The original SVM formulation starts with a two-class classification by linear decision surfaces (hyperplanes). For two linearly separable classes as shown in Fig. D.1(a), there may exist multiple hyperplanes that can correctly separate the two classes. Intuitively speaking, a reasonable choice is to select the hyperplane that has the largest distance to the samples from both classes.

More specifically, as shown in Fig. D.1(b), given two classes $\omega_1$ and $\omega_2$, we would like to find hyperplane $\mathbf{w}^T \mathbf{x} + w_o = 0$, which has the largest distance to the nearest points in $\omega_1$ and $\omega_2$ (called margin). Suppose we have no preference
to either of the two classes, it is then reasonable to select a hyperplane which has the same distance to the respective nearest points in both classes. For notational consistence, we scale \( w \) and \( w_0 \) so that the distance between a hyperplane and its nearest points in both classes is 1. Therefore, the samples at the margin will satisfy \( w^T x + w_0 = \pm 1 \), where +1 holds for \( \omega_1 \) and -1 holds for \( \omega_2 \). Suppose for each sample \( x_i \), we denote its class by \( y_i \) (\( y_i = 1 \) for \( \omega_1 \) and \( y_i = -1 \) for \( \omega_2 \)), then we aim at finding \( w \) and \( w_0 \) such that

\[
J(w) = \frac{1}{2} \|w\|^2 \quad \text{is minimized,} \\
\text{subject to:} \quad y_i(w^T x_i + w_0) \geq 1.
\]

### D.1.2 Case 2: linearly non-separable classes

In many cases, however, the samples from the two classes are not linearly separable. In this case, the optimization goal in Eqn. (D.1) is extended to the following:

\[
J(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum \xi_i \quad \text{is minimized,} \\
\text{subject to:} \quad y_i(w^T x_i + w_0) \geq 1 - \xi_i.
\]

In the above, \( C \) is a weighting factor and \( \xi_i \) is called a slack variable. If \( \xi_i = 0 \), the sample falls outside the band enclosed by two hyperplanes defining the margin and is correctly classified (see Fig. D.1(c)). If \( 0 < \xi_i \leq 1 \), the sample falls inside the band and is correctly classified. If \( \xi_i > 1 \), the sample falls inside the band and is misclassified. The optimization goal in formula (D.2) is to maximize the margin while keeping the misclassified samples as few as possible.
possible. According to [68], the above optimization problem can be converted to an equivalent problem as follows\(^1\):

\[
D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \quad \text{is maximized.} \tag{D.3}
\]

subject to: \(0 \leq \alpha_i \leq C\) and \(\sum_i \alpha_i y_i = 0\).

Suppose we have obtained solutions \(\alpha^*_i\) to the above problem (see [68] for more detailed explanation), then the optimized decision hyperplane satisfying Eqn (D.2) has the following parameters: \(w^* = \sum_i \alpha^*_i y_i x_i\) and \(w^*_0 = y_i - w^* \cdot x_i\) (for \(\alpha^*_i > 0\)). This leads to the final form of the optimal hyperplane as follows:

\[
g(x) = \sum_i \alpha^*_i y_i (x^T_i \cdot x) + w^*_0. \tag{D.4}
\]

Here \(\alpha^*_i\) is only non-zero when \(x_i\) lies on \(w^{*T} x + w^*_0 = \pm 1\), which is called a support vector. It should be noted that the optimal hyperplane is only determined by the support vectors lying at the border between the two classes, which usually constitute only a small subset of the training samples.

### D.2 Extension to nonlinear classification

In addition to linear decision hyperplanes, the SVM classification can be also extended to generate nonlinear decision surfaces. This can be achieved by projecting the original feature vector \(x\) to a higher dimensional vector \(z(x)\),

\(^1\)For detailed derivation, interested readers are referred to [13][68].
where the linear classification technique mentioned above can be applied. This can be illustrated by a simple example as shown in Fig. D.2. Accordingly, the solution decision surface is given by

\[ g(x) = \sum_i \alpha_i^* y_i (z^T(x_i) \cdot z(x)) + w_0^*. \]  

\hspace{0.5cm} \text{(D.5)}

A drawback of this nonlinear mapping is that the computation complexity is significantly increased since the dot products \( z^T(x_i) \cdot z(x) \) in Eqn. (D.5) have to be computed in a much higher dimensional space. In order to cope with this problem, the kernel functions are introduced, which can express the inner product of new vectors (e.g. \( z^T(x_i) \cdot z(x) \)) as a function of the inner product of the corresponding vectors in the original space (e.g. \( x_i^T \cdot x \)). Suppose \( K \) is such a kernel function, then Eqn. (D.5) can be rewritten as:

\[ g(x) = \sum_i \alpha_i^* y_i K(x_i, x) + w_0^*. \]  

\hspace{0.5cm} \text{(D.6)}

Commonly-used kernel functions in literature are:

1. Polynomials: \( K(u, v) = (u^T v + 1)^q, q > 0 \).

2. Radial Basis Functions (RBF): \( K(u, v) = \exp(-\frac{||u - v||^2}{\sigma^2}). \)

3. Hyperbolic Tangent: \( K(u, v) = \tanh(\beta u^T v) + \gamma. \)

In this way, the computation complexity is greatly reduced by using the kernel mapping, which avoids the direct computation of the decision surface in a higher dimensional space.
Appendix E

Support Vector Regression

In this section, we extend the SVM solution to the classification problem as presented in Appendix D to regression problems. We use one variant of SVR (Support Vector Regression) $\epsilon$-SVR as an illustrating example.

Similar to SVM, first let us consider the linear regression problem. We seek a linear function

$$y = w \cdot x + w_0,$$

which is used to approximate a set of training samples $(x_i, y_i)$. In $\epsilon$-SVR, this function should have at most $\epsilon$ deviation from $y_i$ for all the training data. At the same time, the function should be as flat as possible [65]. In other words, no error is counted within a tube of radius $\epsilon$ around the regression (see Fig. E.1). Therefore, the aim is to find $w$ and $w_0$, such that

$$J(w) = \frac{1}{2} \|w\|^2 \quad \text{is minimized,}$$

subject to:

$$\begin{cases} y_i - w \cdot x_i - w_0 \leq \epsilon, \quad \text{and} \\ w \cdot x_i + w_0 - y_i \leq \epsilon. \end{cases}$$

(E.2)

However, sometimes not all training data fall inside the $\epsilon$-tube and some errors need to be allowed. In analogy to the soft margin in SVM, slack variables $\xi$ and $\xi^*$ are introduced to cope with this problem (see Fig. E.1). Hence, the
optimization problem is formulated as

\[
J(w) = \frac{1}{2} \|w\|^2 + C \sum_i (\xi_i + \xi_i^*) \quad \text{is minimized,} \tag{E.3}
\]

subject to:

\[
\begin{align*}
& y_i - w \cdot x_i - w_0 \leq \epsilon + \xi_i \\
& w \cdot x_i + w_0 - y_i \leq \epsilon + \xi_i^* \\
& \xi_i, \xi_i^* \geq 0.
\end{align*} \tag{E.4}
\]

The constant \( C > 0 \) determines the trade-off between the flatness of the function and the amount up to which deviations larger than \( \epsilon \) are tolerated. According to [65], the above optimization problem can be converted to an equivalent problem as follows:

maximize \[ -\frac{1}{2} \sum_{i,j} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x_i \cdot x_j) \]

\[ - \epsilon \sum_i (\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*) \]

subject to: \[ \sum_i (\alpha_i - \alpha_i^*) = 0, \text{for } 0 \leq \alpha_i, \alpha_i^* \leq C. \tag{E.5} \]

Here \( \alpha_i \) and \( \alpha_i^* \) are auxiliary variables used to solve the optimization problem. Similar to the SVM formulation, the solution regression function takes the following form:

\[ f(x) = \sum_i (\alpha_i - \alpha_i^*)(x_i \cdot x) + w_0^*. \tag{E.6} \]
By applying the kernel mapping as used in SVM (see Appendix D), the above formulation can be extended to the nonlinear case. Correspondingly, we can obtain a nonlinear regression function as follows:

\[ f(x) = \sum_i (\alpha_i - \alpha_i^*) K(x_i, x) + w_0^*, \]  

(E.7)

where \( K \) is a kernel function as defined in Appendix D.
References


Summary

During the past few years, there is an increasing demand for smart devices in consumer electronics. These smart devices should be capable of consciously sensing their surroundings and adapting their services according to their environments. Face recognition provides a natural visual interface for such applications and can be embedded into corresponding devices to facilitate context awareness. The continuous growth of computing power brings face recognition within the reach of consumer devices and embedded applications.

Compared to traditional face recognition in professional applications, face recognition in embedded/consumer applications is characterized by a large variability of operating environments and limited computation power and image quality. We aim at designing a face-recognition system which has a performance that is competitive to a professional system but has a significantly higher efficiency in terms of computation.

In this thesis, we aim at employing multiple algorithms that coordinate with each other for enhanced face-recognition performance while managing the overall complexity. More specifically, we propose new techniques for three major processing stages in face recognition, namely, face detection, facial feature extraction and face identification. At each stage, our major contribution is the design of a number of novel algorithms that are further combined into a cascaded structure. In this cascaded framework, we focus mainly on the following two aspects: (1) design of individual algorithms to meet system requirements, and (2) optimization of algorithm ordering and interfacing to improve the overall system performance.

For face detection, we have proposed two pruning detection cascades, where we use fast detectors to quickly discard large non-face background areas and more accurate detectors at succeeding stages to refine the detection results. In this way, both the high detection accuracy and the processing efficiency can be achieved simultaneously. The first cascade is based on various feature detectors, namely, a color-based detector, a feature-geometry-based detector and a neural-network-based detector. The second cascade is based on a set of neural-network ensembles. First, for improved detection accuracy, we propose a novel training technique to form a coordinated ensemble of neu-
ral networks. Second, for improved detection efficiency, we build a cascade of neural-network ensembles with scalable structures. The approach achieves one of the highest detection accuracies in literature with a significantly reduced computation cost. The proposed structure is also suitable to be implemented in parallelized hardware architectures.

For facial feature extraction, we have first developed an improved algorithm of the Active Shape Model (ASM), which extends ASM by using Haar-based local feature modeling. The enhanced modeling enriches the representation power of ASM and leads to doubled convergence capability and a 17% improvement in accuracy. Afterwards, we have developed a cascaded extraction framework for a set of model-based extraction algorithms. We have defined a set of principles to guide the construction of such a framework, which examines the performance relations between adjacent algorithms. As an implementation, we propose a three-algorithm cascade for facial feature extraction, which consists of a sparse graph model, a component-based texture model and a component-based appearance model. These algorithms capture different characteristics of facial features, giving an increasing extraction accuracy coupled with a decreasing convergence. By tuning the performance of each algorithm based on the output statistics of its preceding algorithm, a feature model can be progressively ‘pulled’ to the correct position. The experiments show that our approach is not sensitive to large model deviations and achieves a high extraction accuracy (24% gain compared to ASM).

For face identification, we have explored a selective cascade for improved identification performance. The selection cascade gradually reduces the candidate size and derives a customized classification function for each candidate set. We have applied Linear Discriminant Analysis in this framework and illustrated the effectiveness of the approach (23% reduction of identification error). Furthermore, we have investigated a new adaptive feature selection as a more efficient implementation of the cascaded identification. This approach selects a set of the most discriminating features for each person based on the so-called class-specific maximum marginal diversity. According to the selected features, an efficient matching function is defined. Our cascaded algorithm effectively improves the identification of single algorithms and outperforms several well-known face identification algorithms (e.g. by a reduction of identification error by 18%).

We have successfully applied selections of our developed face-recognition techniques in several applications, such as smart user identification in a connected home environment, face recognition for secure biometric identification and face recognition for video surveillance and database retrieval. In several extensive tests, the system has demonstrated competitive performance with respect to accuracy, efficiency and robustness.
Samenvatting

Er is een groeiende behoefte aan intelligente apparaten in de consumentenelektronica. Deze intelligente apparaten zouden in staat moeten zijn om hun omgeving bewust te verkennen en zichzelf en hun diensten aan te passen aan het betreffende milieu. Gezichtsherkenning is een belangrijk visueel hulpmiddel voor zulke systemen om hun gebruiker te identificeren en deze informatie te gebruiken voor het vergroten van het ‘omgevingsbewustzijn’ van het systeem. Door de voortdurende groei in rekenkracht komt gezichtsherkenning langzamerhand binnen bereik van toepassingen in ‘embedded systems’ en de consumentenelektronica.

Vergeleken met gezichtsherkenning in professionele toepassingen zoals bij openbare veiligheid, wordt gezichtsherkenning als ingebouwd systeem gekenmerkt door sterke variaties in de omgevingscondities en een door kosten begrense rekenkracht en beeldkwaliteit. Het onderzoek in dit proefschrift streeft naar het ontwerpen van een gezichtsherkenningssysteem met een performance die vergelijkbaar is met een professioneel systeem maar met een hogere efficiëntie in de uitgevoerde berekeningen.

De gekozen benadering in dit proefschrift heeft als doel om verschillende algoritmen in een systeem toe te passen, die gezamenlijk in een gecoördineerde vorm voor een verbeterde prestatie zorgen, terwijl de complexiteit beheersbaar blijft. Het gezichtsherkenningssysteem is verdeeld in drie signaalbewerkingsstappen: gezichtsdetectie, extractie van de gezichtseigenschappen en gezichtsidentificatie. In elk van deze drie stappen is een aantal nieuwe algoritmen toegepast die achtereenvolgens in een cascade structuur worden ondergebracht. Daarbij concentreert het onderzoek zich hoofdzakelijk op de volgende twee aspecten: (1) ontwerp van onafhankelijke algoritmen om bepaalde systeemeisen te bereiken en (2) optimalisatie van algoritmevolgorde en data-overdracht om de algemene systeemprestaties te verbeteren.

Voor gezichtsdetectie worden twee ‘pruning’ gecascadeerde detectors voorgesteld, met het doel om grote beeldgebieden die geen gezicht bevatten snel te kunnen uitsluiten en meer nauwkeurige detectors in daaropvolgende bewerkingstappen ter verbetering van detectieresultaten. Op deze wijze kunnen gelijktijdig een hoge nauwkeurigheidsgraad in detectie en een efficiënt gebruik
van rekenkracht worden bereikt. De eerst voorgestelde cascadestructuur is gebaseerd op diverse detectors van gezichtskennmeren, zoals een op huidskleur gebaseerde detector, een detector voor geometrische gezichtseigenschappen en een detector die is geconstrueerd met neurale netwerken. Een tweede cascade-structuur die is onderzocht is gebaseerd op een reeks ensembles van neurale netwerken met gelijkwaardige structuren. Teneinde de detectienauwkeurigheid te verbeteren, wordt een nieuwe trainingstechniek gepresenteerd die tot doel heeft een gecoördineerd ensemble van neurale netwerken te vormen. Daarnaast wordt een verbeterde efficiëntie in rekenkracht verkregen door de toepassing van een cascade van neuraal-netwerk ensembles met schaalbare structuren. Deze benadering bereikt één van de hoogste detectienauwkeurigheden uit de literatuur, maar met een beduidend lager benodigde rekenkracht. De hierbij voorgestelde structuur is tevens geschikt om in parallele hardware architectu ren te worden uitgevoerd.

Voor de extractie van gezichtseigenschappen is eerst een uitbreiding van het Active Shape Model (ASM) algoritme ontwikkeld, met een op Haar-Wavelets gebaseerde lokale eigenschapsmodellering. Deze uitbreiding verbetert ASM en leidt tot een verdubbeld convergentiegebied en een verbetering van de nauwkeurigheid met 17%. Vervolgens is een drievoudig extractie-algoritme ontworpen voor een verzameling van modelgebaseerde extractie-algoritmen. We hebben een verzameling van richtlijnen gedefinieerd voor de constructie van het gehele systeem, dat gebruik maakt van de performancerelaties tussen naburige algoritmen. Voor de implementatie stellen we een cascade voor van drie algoritmen voor de extractie van gezichtseigenschappen, bestaande uit een (‘sparse’) graafmodel, een op componenten gebaseerd structuurmodel en een op componenten gebaseerd gezichtsuitdrukkingsmodel. Deze deelalgoritmen herkennen ieder verschillende kenmerken van het menselijk gezicht, hetgeen leidt tot een toenemende extractienauwkeurigheid gekoppeld aan een afnemende convergentie. Door het juist instellen van de prestatie van elk deelalgoritme welke gebaseerd is op de statistiek van het uitgangssignaal van het voorgaande algoritme, wordt het eigenschapsmodel progressief naar de juiste positie in het beeld geleid. Experimenten hebben aangetoond dat onze benadering niet gevoelig is voor grote variaties in het model en een hoge extractienauwkeurigheid bereikt (24% winst vergeleken met ASM).

Bij gezichtsidentificatie is een selectieve cascade onderzocht voor verbetering van de prestatie in identificatie. De selectiestap in het begin vermindert geleidelijk het aantal kandidaten en leidt tevens tot een populatiespecifieke classificatiefunctie voor elke set van kandidaten. In dit kader is de Lineaire Discriminant Analyse (LDA) toegepast en de doeltreffendheid van deze benadering aangetoond (23% vermindering van de identificatiefout). Voorts hebben we een nieuwe selectie van eigenschappen onderzocht die een heel efficiënte implementatie is van een gecascadeerde identificatie. Deze benadering selecteert een verzameling van de meest onderscheidende eigenschappen
van elk persoon, waarbij de zogenaamde klassespecifieke marginale diversiteit wordt gebruikt. Gebruikmakend van de geselecteerde eigenschappen is een efficiënte bijbehorende matchingsfunctie gedefinieerd. De cascade van algoritmen verbetert hiermee de identificatie bij elk van de algoritmen afzonderlijk en overtreft hierbij verscheidene bekende algoritmen voor gezichtsidentificatie uit de literatuur (bijv. een vermindering van identificatiefout van 18%).

De technieken voor gezichtsherkennings zijn met succes toegepast in verschillende applicaties waaronder identificatie van gebruikers in een huis van de toekomst met genetwerkte omgeving, gezichtsherkenning voor biometrische identificatie en gezichtsherkenning in een videobewakingssysteem met databanken. In uitgebreide testsessies is het systeem met competitieve prestaties gedemonstreerd met betrekking tot robuustheid, nauwkeurigheid en efficiënt gebruik van rekenkracht.
Acknowledgements

Five years ago, I was very delighted to get the offer from Prof.dr. Peter de With to work as a Ph.D student at the Eindhoven University of Technology. I thank Peter, Prof.dr. Han Vinck from the University of Essen and Fred Snijders from Philips Research Eindhoven for their efforts in initializing the project within which I performed my Ph.D research. This thesis work would have never been accomplished without their support. Today I am proud to present this thesis as a final product of my four years’ work at the TU/e. I would like very much to thank the following people for all the help they have offered me during all these years.

First of all, I would like to express my sincere appreciation to my promoter Prof.dr. Peter de With for his endless support, encouragement and enthusiasm during all these years. I will always remember our joyful conversations on things from scientific research to life experience. Peter always showed firm confidence in me when I doubted myself and encouraged me when I was uncertain about difficulties.

I would also like to extend my appreciation to my second promoter Prof.dr. Jan Biemond. He has read the thesis thoroughly and provided very valuable comments. I also thank Dr. Raymond Veldhuis for his careful review of this thesis and fruitful discussions.

The research work presented in this thesis has been carried out in cooperation with Philips Research Labs. in Eindhoven within the ITEA project HomeNet2Run. I would like to thank Frank van Tuijl and Peter van Grinsven from Philips Research for their support in this project. Later I also cooperated with the Information Systems and Security group at Philips Research on privacy-protected biometrics. I would like to thank Dr. Michiel van der Veen, Dr. Tom Kevenaar and Ton Akkermans for their close cooperation.

Next I would like to thank my former colleague Stephan Mietens for all his help whenever I had difficulties in work and life. I would like to express my gratitude to Dirk Farin for his wonderful Linux source codes and inspiring discussions. Thanks to Yannick Morvan, Milan Pastrnak, Goran Petrovic, Bahman Zafarifar, Egor Bondarev and Rob Wijnhoven. I will always remember the nice jokes we had during the lunch time. I thank Jungong Han, Weilun
Lao and Ping Li for the enjoyable time we have spent together, with whom I feel not far away from home.

Special thanks to Rob Albers for his help in translating the thesis summary into Dutch.

I also would like to thank (Eindhoven) Embedded Systems Institute ((E)ESI), with which I have a formal working contract for my Ph.D research. Special thanks to Marloes van Lierop, who was always ready to help for all the managerial issues. Thank my old (E)ESI colleagues Giovanni Russello, Ana Sokolova, Mohammad Mousavi, Kalok Man and Richard Verhoeven for the little, friendly international community we had.

Thanks to all my current colleagues at Philips Research Lab. Eindhoven for their patience during the final period of this thesis work.

Last but not the least, a big hug to my family. I thank my father and mother for their love and support in all aspects of my life. Finally, loving thanks to my husband for his encouragement and support whenever I had difficulties.
Curriculum Vitae

Fei Zuo was born on the 2nd of April 1977 in Xi’an, China. She received her B.Sc. in computer science from Xi’an Jiaotong University, China, in 1998, and her M.Sc. in computer science from the same university in 2001. Her master’s thesis addressed the design of a visual interaction platform by using image processing techniques. Subsequently, she joined the Eindhoven University of Technology, The Netherlands, where she worked towards a Ph.D degree on face detection and recognition. Since 2005, she became a research scientist at Philips Research Laboratory in Eindhoven, where she is involved in developing new video coding techniques.