Gabor-Boosting Face Recognition

Mian Zhou

A Thesis submitted for the degree of Doctor of Philosophy

School of Systems Enigneering

University of Reading

September 2008

Declarations

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Mian Zhou

Acknowledgments

I would like to take this opportunity to express my deep gratitude to my supervisor Dr. Hong Wei for offering me the opportunity to conduct this research and providing invaluable guidance and support. I am also in debt to my second supervisor Dr. James Ferryman for all the helpful suggestions and support.

I am grateful to my colleagues who have been helping me a lot at Computational Vision Group. I would like say thanks to Dr. Ian Bland and Dr. Anthony Worrall who provide Grid Computing facilities so that I can finish my experiment in a limited time. I would like to thank Marc Bartel for encouraging me to continue my research. I would like to thank Dr. David Tweed and Longzhen Li for some technical discussion. I would like to thank Murray Evans and Dr. David Thirde for introducing STL and OOL in C++ to me. Also, I want to say thanks to Nicholas Carter, Saad Choudri, Anna Ellis, Dave Young, Mark Borg, Chen Qin, Hongwen Xu, Dr. Nils Siebel and Dr. Roberto Fraile for your kind help. I would like to thank all the individuals in computer vision and pattern recognition community for giving me help. I would like to thank Professor Mark Nixon for giving me an opportunity to attend *FG2006*. I would like to thank Dr. Xiaojun Wu for providing me the XM2VTS face database. Also I would like to give my thank to Professor Haizhou AI, Professor Weiming Hu, Professor Steve Maybank and Professor Yunhong Wang for useful discussion and suggestion. I also would like to thank George Scott for proof reading my thesis.

Finally, I would like to thank my wife Mrs. Jing Jin Zhu, my parents and all other family members for all your love, understanding, support and encouragement.

Dedicated to my parents

Abstract

In the past decade, automatic face recognition has received much attention by both the commercial and public sectors as an efficient and resilient recognition technique in biometrics. This thesis describes a highly accurate appearance-based algorithm for grey scale front-view face recognition - Gabor-Boosting face recognition by means of computer vision, pattern recognition, image processing, machine learning, etc.

The strong performance of the Gabor-Boosting face recognition algorithm is highlighted by combining three key leading edge techniques - the Gabor wavelet transform, AdaBoost, Support Vector Machine (SVM). The Gabor wavelet transform is used to extract features which describe texture variations of human faces. The AdaBoost algorithm is used to select most significant features which represent different individuals. The SVM constructs a classifier with high recognition accuracy.

Within the AdaBoost algorithm, a novel weak learner - Potsu is designed. The Potsu weak learner is fast due to the simple perceptron prototype, and is accurate due to large number of training examples available. More importantly, the Potsu weak learner is the only weak learner which satisfies the requirement of AdaBoost. The Potsu weak learners also demonstrate superior performance over other weak learners, such as FLD. The Gabor-Boosting face recognition algorithm is extended into multi-class classification domain, in which a multi-class weak learner called mPotsu is developed. The experiments show that performance is improved by applying loosely controlled face recognition in the multi-class classification.

The Gabor-Boosting face recognition algorithm is robust under conditions of small number of example images per subject and selection-bias in the training data. One potential application of the algorithm could be in highly secured face authentication systems dedicated to a small group of clients. These systems must give no false detections for impostors and an appropriate acceptance detection rate for clients.

Keywords: Face Recognition, Gabor Wavelets, AdaBoost, Boosting, SVM

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recently years, biometrics has become a very hot topic. No technology has been affected more than biometrics since the events of September 11. Biometrics is defined as automated methods of recognising a person based on a physiological or behavioural characteristic [1]. Biometrics involves identification using different characteristics of a human being, such as face, fingerprints, hand geometry, handwriting, iris, retinal patterns, vein patterns, voice, etc. For instance, at Frankfurt Rhein-Main international airport, an automated border control system via recognising faces of voluntary subscribers is deployed. Biometrics also attracts a great deal of commercial attention. In 2003, the annual income of global biometrics were $ 719 million [102]. Global biometric revenues are projected to grow from $ 3.01 billion in 2007 to $ 7.41 billion in 2012 [2], driven by government identity management programs and private-sector enterprise. The biometric technology is used in various markets, such as law enforcement, military, government, gaming and hospitality, health care, high-tech and telecom, industrial manufacturing, transportation, retail, etc.

1.1 Face recognition

Among many human characteristics, measuring facial characteristics is non intrusive [76]. Facial images are probably the most common biometric characteristic used by human to make personal recognition. Face recognition is emerging an active research area spanning several subjects such as image processing, pattern recognition, computer vision, machine learning, cognitive science, neuroscience,

psychology and physiology. Face recognition is not only a dedicated process, but also serves as general application of object recognition.

As one of the most successful applications in pattern recognition, image analysis and understanding, face recognition has received significant attentions over the past 20 years. There are a large number of commercial, security and forensic applications using face recognition technologies. These applications include automatic crowd surveillance, access control, face reconstruction, human computer interaction (HCI), multimedia communication, and content-based image retrieval. Many commercial face recognition systems [43, 17, 127, 57] adopt the technologies of video-based modelling, processing and recognition.

In general, an automatic face recognition systems are comprised of three key modules: face detection, feature extraction, and classification. The flowchart is shown in Figure 1.1. Face detection includes face localisation, segmentation and

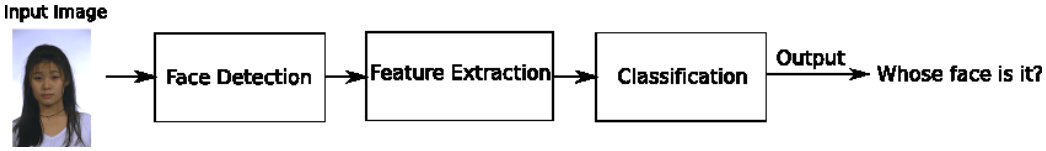


Figure 1.1: The flowchart of three modules of face recognition systems

normalisation, in which a pre-processed face image is obtained from an input scene, either simple or cluttered. Feature extraction derives a representative of the pre-processed face image using a set of features. There are various types of features possibly extracted from the process, such as facial features, statistical appearance-based features, transform coefficient features, algebraic features, etc. In some face recognition systems, there might be a feature selection module between feature extraction and classification. Due to high-dimensionality in the feature space or redundancy among features, such a process is used to select a smaller group of relevant features to represent faces and achieve better recognition performance. Classification is used to assign the above features into one of several categories, *i.e.*, subjects.

Depending on the application context, face recognition can be divided into two scenarios: face verification and face identification. In face verification, an individual who desires to be recognised claims an identity, usually through a personal identification number, an user name, or a smart card. The system conducts a

one-to-one comparison to determine whether the claim is true or not, *i.e.*, face verification is to ask a question - "Does the face belong to a specific person?". In face identification, the system conducts a one-to-many comparison to establish an individual's identity without the subject to claim an identity, *i.e.*, face identification is to answer the question - "Whose face is this?". Throughout this thesis, the generic term *recognition* is used, which does not make a distinction between verification and identification.

1.2 Gabor-Boosting algorithm

Recently, a breakthrough in face detection has been made by Viola and Jones [162]. Faces are quickly and accurately detected by cascade structured classifiers combined with simple Haar features. In the face detection system, the most significant contribution is the AdaBoost algorithm which selects 200 most important features from over 180000 features, and a complex classifier in a cascade structure is built on these important features. The face detector is the most rapid and accurate approach compared to other systems, and it pushes research and development of face detection into a new era. Now many real-time face detection applications, such as those used in digital camera, web cam and mobile phone camera adopt Viola and Jones AdaBoost approach. Facing the big success in face detection, two questions are spontaneously released - "Can the AdaBoost approach be transplanted into face recognition?" and if so "How can the AdaBoost be used in face recognition?".

This thesis gives an exploration into these two questions. The transplantation of the AdaBoost from face detection into face recognition introduces further subquestions in feature extraction, feature selection, and classification. These subquestions are

- In face detection, there are only two classes, *i.e.*, face and non-face, so Viola and Jones' work in face detection [162] is processed using two-class (binary) classification ideas. While, in face recognition, it is more likely there are more than two persons to be discriminated, so that face recognition deals with multiple classes. This prevents the direct transplantation from a two-class environment into a multi-class environment. The solution is to shift face recognition into two-class environments. Hence the sub-question is - "how can the two-class scenario be established in face recognition?".

- Viola and Jones' approach uses Haar features which are simple so that they can be quickly evaluated. Haar features are sufficient to discriminate differences between face and non-face. However, in face recognition, features are used to discriminate differences between individuals. Obviously, the appearance difference between face and non-face is much larger than the appearance between individuals. Hence, the sub-question is that "are the Haar features sufficient for face recognition?". If not, "what should the substitution be?".

- Within AdaBoost, weak learners (base classifiers)[1] are used to evaluate each feature, such that the type of weak learner is a very important factor which determines the performance of AdaBoost. Among the various types of weak learners, "which types of weak learners are beneficial to improve the performance of face recognition?".

- After the AdaBoost training, a complex classifier is built on weak learners with respect to selected features. The sub-question is: "is this classifier appropriate for face recognition?". If not, "what is the alternative which can be used for recognising human faces?".

- Beside the two-class AdaBoost, there are some multi-class variants of AdaBoost. "How do these variants fulfil the purpose of multi-class classification from face recognition?"

- The AdaBoost approach is very time-consuming [175]. Since images used for face recognition is larger than those images used for face detection in AdaBoost training, the AdaBoost training in face recognition consumes more time than the one in face detection. Hence, there is a sub-question - "how to reduce the computational time of the AdaBoost training?".

The algorithm described in this thesis is motivated by Viola and Jones' work in face detection [162]. The algorithm is named *Gabor-Boosting* algorithm, which is proposed, developed and tested. Gabor wavelet transform is used to extract Gabor wavelet features from face images. Due to very large number of features extracted, AdaBoost is used to select a small group of features representing individual's face. After training a classifier with existing faces, the recognition is done by outputting classification results. To evaluate the performance of the Gabor-Boosting face

[1]Weak learners are the elemental components in AdaBoost training.

recognition algorithm, the XM2VTS [108, 111] and FERET [125] face databases are used.

In face recognition systems, it is clear that evaluation and benchmarking of the algorithms is crucial. Previous work [111, 125, 126, 123, 133, 132, 75, 63, 54] on evaluation provides insights into how the evaluation of algorithms and systems can be performed efficiently. Although great amount of effort has been devoted to face recognition, there are still remains some challenges to be solved such as illumination, head pose, facial expression, occlusion (glasses, sunglasses, scarf, etc), facial hair (beard, mustache, etc.), and ageing. These difficulties make human face appearance in images having very large variations. Therefore, successful face recognition systems have been deployed only under constrained conditions. In this thesis, the face recognition is also deployed under constrained conditions

- Illumination is consistent across images
- No head pose, but only front-view;
- No facial hair and ageing variation, but allowing adequate occlusion (glasses) and facial expression variation.

In additional, all images are converted into grey scale.

1.3 Thesis Outline

This thesis is organised as follows: Chapter 2 presents a literature review of face recognition, in which major state-of-the-art techniques of face recognition and face databases are introduced. Chapter 3 gives the background and technical detail of Gabor wavelet transform and AdaBoost. Chapter 4 describes the initial Gabor-Boosting face recognition, in which face recognition is converted into two-class classification and Gabor wavelet transform is used to extract human facial features. Chapter 5 presents a novel weak learner used in AdaBoost training. Chapter 6 is an extension of the Gabor-Boosting face recognition from a binary domain to a multi-class domain, and also explains how to use grid computing technology to reduce the computational cost of feature selection. The final chapter concludes the work of the Gabor-Boosting algorithm in face recognition, and highlights directions for future work.

Chapter 2

Literature Review

Since 1970s, a great progress has been made in the computer based face recognition area. Face recognition has attracted the attention of researchers from various areas. In this chapter, a literature review on face recognition is present. Because the focus of this research is 2-D face recognition, 3-D face recognition is excluded from the chapter. Most recent approaches for face recognition can be divided into two categories: appearance-based face recognition and model-based face recognition. The appearance-based approaches process face images based on statistical analysis of pixel value distributions. The model-based approaches match a general 2-D face model on face images. Furthermore, a 2.5D approach - 3-D morphable face model is introduced, which adopts a 3-D face model to extract face representations from 2-D face images.

The chapter is organised as: Section 2.1 introduces appearance-based face recognition; Section 2.2 describes model-based face recognition; Section 2.3 gives a brief description of 3-D morphable face model; Section 2.4 briefly introduces rest major approaches. Some popularly used face databases are presented in Section 2.5.

2.1 Appearance-Based Face Recognition

Many approaches for object recognition are directly applied on 2-D images. A 2-D face image is treated as a high-dimensional vector. They are so called appearance based approaches. Image data can be represented as vectors which are points in a high dimensional vector space. For example, a 128×128 image can be represented

by a vector $x \in \mathcal{R}^{16384}$. In the high-dimensional space, every object is described if it can be depicted in the 128×128 image. If all vectors in the high dimensional space are the same type of objects, *e.g.*, faces, the natural constraints of the physical world indicate that the vectors will in fact lie in a lower dimensional linear subspace or non-linear manifold. Linear subspace analysis has significantly advanced facial recognition technology, but it does not have sufficient modelling capacity to preserve the variations of the face manifold and distinguish between different persons to achieve high accuracy in face recognition. Recent developments in nonlinear manifold analysis provide more flexibility and modelling capacity to analyse face manifolds. However, the generalisation of nonlinear methods is affected by the number of examples, *e.g.*, there is a small number of face example images in training set, but large variations of facial appearance during the testing. The phenomenon leads incorrect results and overfitting [179].

2.1.1 Linear Analysis

Three classical linear approaches are introduced here, which are Principal Component Analysis (PCA) [159], Independent Component Analysis (ICA) [13] and Linear Discriminant Analysis (LDA) [15, 153]. By projecting a face image to a subspace, the projection coefficients are used as the feature representation of each face image. The classification is performed between the test face image and the training prototype. All representations from these three linear approaches are considered as a linear transformation from the original image vector x to a projection feature vector y, *i.e.*,

$$y = W^T x \tag{2.1}$$

where W indicates the transformation.

PCA

Principal Component Analysis (PCA) is to find a subspace which accounts for the distribution of face images within the entire image space. The PCA is generated from the Karhunen-Loeve (KL) expansion, which has been widely used for face representation [84] and face recognition [159]. In [159], all face images in the training set are collected and composed into a covariance matrix. The eigenvectors and corresponding eigenvalues are generated from the covariance matrix. The

eigenvectors are visualised into a two-dimensional array and display ghost face like appearance. Hence, the eigenvectors are also named "eigenfaces" as shown in Figure 2.1. These eigenfaces span a small subspace in the image space. The

Figure 2.1: The seven eigenfaces calculated from [159]

subspace is called "face" subspace. Given the eigenfaces, every face in the training set can be represented as a vector with a sequence of weights. The weights are obtained by projecting the images into the face subspace by a inner product operation. The testing image is also represented by its vector. The recognition on the PCA approach is done by measuring the Euclidean distance between the testing vector and the existing face vectors in the face subspace. The method is illustrated in [159] using a large database of 2500 face images of 16 subjects with three head orientations, three scales, and three lighting conditions.

The PCA approach is enhanced in [122] by several extensions. The experiments are carried out in a large-scale face database, which contains 7562 face images of approximately 3000 people. The first twenty eigenfaces are introduced from a ran-

dom 128 images subset. Annotated information on gender, race, approximate age and facial expression is included in the extended approach. A view-based eigenfaces technique is introduced as modular eigenspace so that face recognition can be performed under various poses. The probabilistic analysis is integrated into the eigenface approach in [114]. The probability density is estimated in the eigenspace. Two types of density estimation are derived from modelling the eigenspaces: a multivariate Gaussian model and a Mixture Gaussian model. By applying Bayes' theorem, the maximum likelihood estimation is used for face detection and face recognition.

Some experiments are conducted to test the performance of the PCA approaches. It is reported that the approach is fair against the illumination change. In [15], it is mentioned that by discarding the three most significant eigenvectors, the variation due to lighting is reduced.

ICA

Like the PCA, Independent Component Analysis (ICA) is also a linear transformation. The PCA is to find the ranked principle components which describe the variation, however the ICA is to find the independent components by maximising the statistical independence between these estimated components. The independence of the components is measured by maximising Non-Gaussian distribution. The popular algorithms to computing ICA include infomax [16], FastICA [71, 72], and JADE [25].

The ICA is a generalised version of the PCA, and the PCA can be derived as a special case of ICA, as Gaussain component models are used. The difference betwen ICA and PCA is illustrated in Figure.2.2. There are 3-D data examples residing in a 3-D axes space. The Independent Component (IC) and Principal Component (PC) construct different coordination systems. The PC axes are orthogonal while the IC axes are not. If only two components are allowed, ICA chooses a different subspace than PCA. In the PCA projection, the data are sparsely distributed on the first PC, but closely clustered on the second PC. In ICA projection, the data are both sparsely distributed on the first and second ICs. Since the ICA axes are nonorthogonal, relative distance between points are different in PCA than in ICA. In [11], a comparison between ICA and PCA is given. It shows that the ICA outperforms PCA on a human face recognition task when both using same distance metric, *i.e.*, cosine angle. However, when the L1

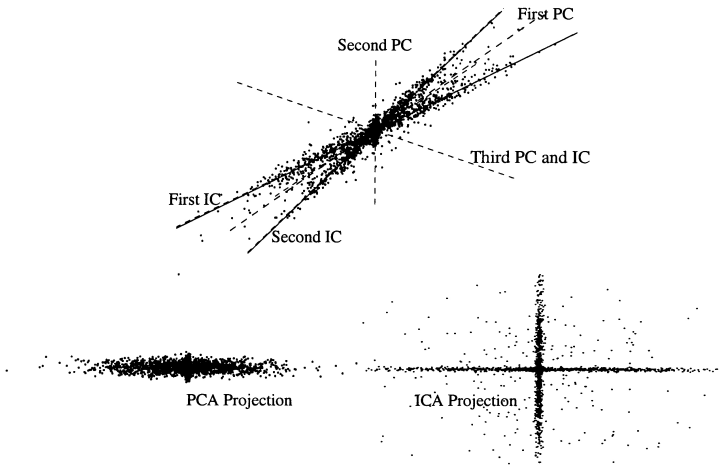


Figure 2.2: The difference between ICA and PCA [14].

norm is adapted, the PCA significantly outperforms ICA.

In [14], ICA was applied on face recognition in the FERET database under two architectures. In the first architecture, the images are treated as random variables and the pixels are treated as outcome. The spatially local basis images for the faces are found as shown in Figure 2.3. In the second architecture, the pixels are treated as random variables, and the images as outcome. A factorial face code is produced in the second architecture. The ICA separates the high-order moments of the input, while the second order moments are utilised in the PCA. The ICA factorial code representation is illustrated in Figure 2.4.

Liu and Wechsler [97] present an independent Gabor feature (IGF) method and its application to face recognition. The method derives independent Gabor features in the feature extraction stage, and an IGF feature-based probabilistic reasoning model (PRM) classification method is developed in the pattern recognition stage. The IGF method is firstly to acquire a Gabor feature vector from a set of Gabor wavelet representations of face images, then to reduce the dimensionality of the vector by means of principal component analysis, and finally to define the independent Gabor features based on the ICA. Experiments on face recognition are carried out by using the FERET and the ORL datasets. The images vary in illumination, expression, pose, and scale. The IGF method achieves 98.5% correct face recognition accuracy when using 180 features for the FERET dataset, and 100% accuracy for the ORL dataset using 88 features. The approach is extended

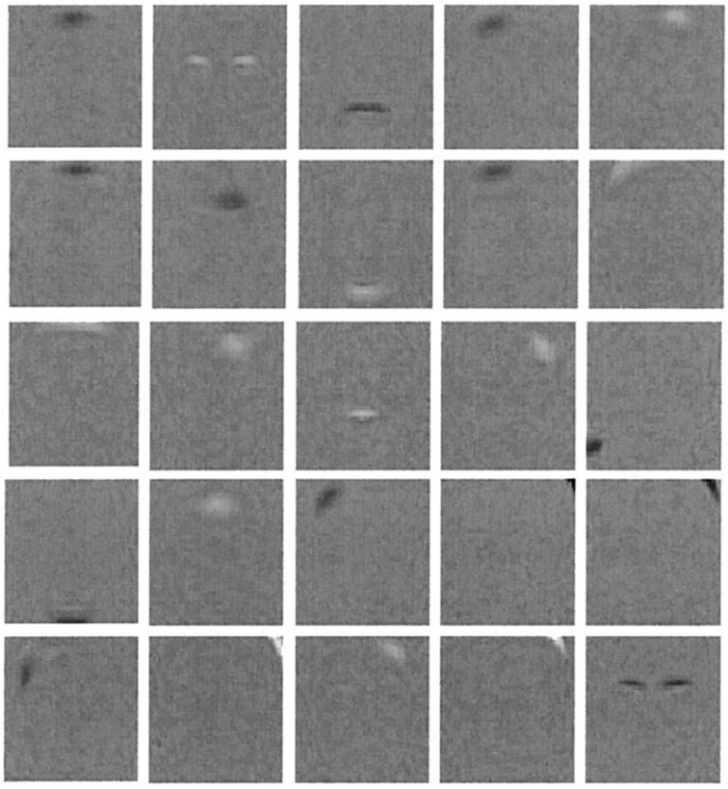


Figure 2.3: Twenty-five image sets obtained by Architecture I in [14]

with enhanced ICA [94].

LDA

Both PCA and ICA are unsupervised methods that yield a set of linear features of a particular dimension, but there is no consideration if this set of features is good for classification. In the PCA, the examples' cluster is maximised not only between different class clusters but also within the same class cluster. The PCA does not take classes into account when there are more than one class in a dataset. This leads to significant problems. For the data set on Figure 2.5(a), one class is indicated by circles and the other by stars.

The PCA would suggest projection onto a vertical axis, which captures the variance in the dataset, but cannot be used to discriminate it from the axes obtained by PCA, which are overlaid on the data set. The bottom of Figure 2.5 shows

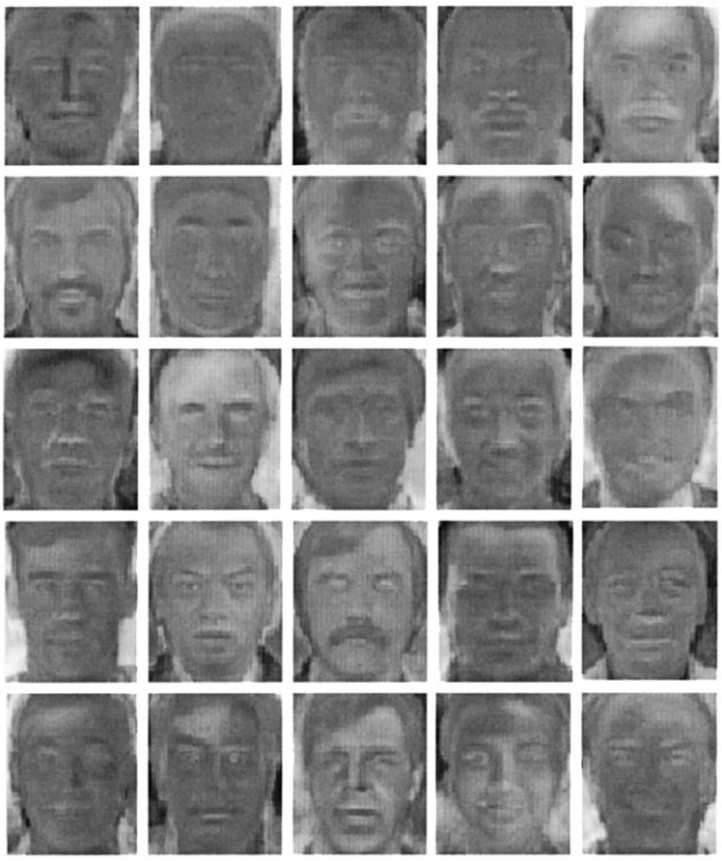
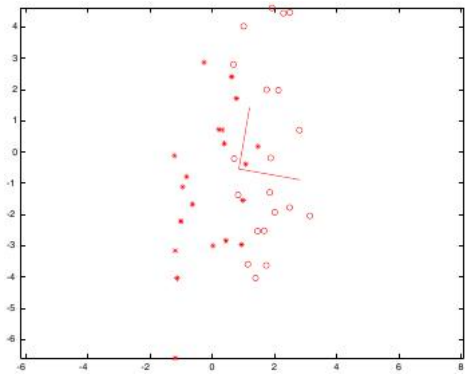


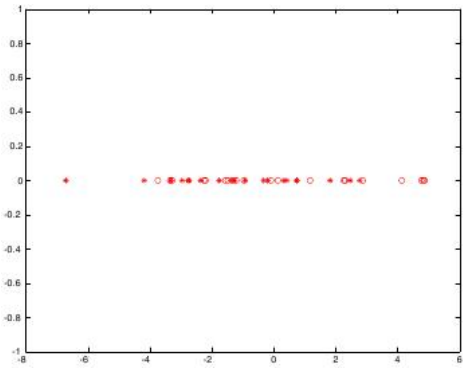
Figure 2.4: The basis images for the ICA-factorial representation obtained with Architecture II in [14]

the projections onto those two axes. Figure 2.5(b) is the projection onto the first PC, which has higher variance, but separates the classes poorly. Figure 2.5(c) shows the projection onto the second PC, which has significantly lower variance and gives better separation. Figure 2.5 shows the first PC would produce a bad classification, while the second PC would produce a good classification, despite the factor that the second PC does not scatter the data.

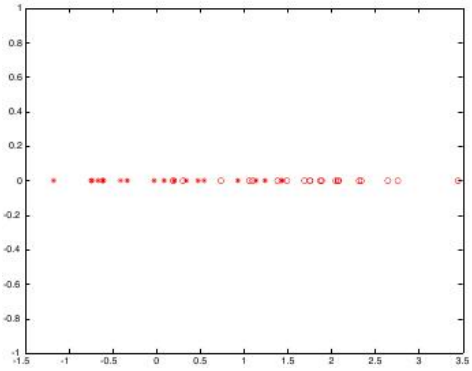
Linear Discriminant Analysis (LDA) is a statistical method to classify examples into different classes based on a set of measurements of examples. The LDA applied in face recognition is very successful, because LDA is originally for classification, *i.e.*, the LDA is a supervised learning approach. In LDA, the purpose is to maximise the discrimination between different classes, and recognition can



(a) The original 2D data set contains two classes



(b) The data are projected onto the first PC.



(c) The data are projected onto the second PC.

Figure 2.5: The PCA has no consideration for classification purpose. [45]

be apparently done based on this. The LDA also constructs a subspace that is constructed by the selected components. The LDA training is carried out by using scatter matrices. The method selects a set of features in such a way that the ratio of the between-class scatter and the within-class scatter is maximised. The between-class scatter matrix is defined as

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{2.2}$$

and the within-class scatter matrix is defined as

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T \tag{2.3}$$

where μ_i is the mean image of class X_i, and N_i is the number of examples in class X_i, and c is the number of classes. If the scatter matrix S_W is nonsingular, the optimal projection W_{opt} is chosen as the matrix with orthonormal columns which maximises the ratio of determinant of the between-class scatter matrix of the projected examples to the determinant of the within-class scatter matrix of the projected examples, *i.e.*

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|} \tag{2.4}$$

and the projection W_{opt} can expressed as

$$W_{opt} = [w_1 \quad w_2 \quad \dots \quad w_m] \tag{2.5}$$

where $\{w_i | i = 1, 2, \dots, m\}$ is a set of generalised eigenvectors of S_B and S_W corresponding to the m largest generalised eigenvalues. Because an upper bound on m is $c - 1$, there are $c - 1$ nonzero components.

However, in some real applications, the number of images in the training set N is normally much smaller than the number of pixels in each image n, so that the within-class scatter matrix S_W tends to be always singular. The phenomenon is called *small sample size* problem [51].

To overcome the small sample size problem, different methods have been proposed in face recognition literature.

Some methods reduce the dimension of the original sample space, which has been demonstrated to contain considerable discriminative information. In [15], a method called "Fisherfaces" is proposed. The method is achieved by using PCA to reduce the dimension of feature space to $N - c$, and then apply the LDA to reduce the dimension further to $c - 1$. In [178], the images are firstly preprocessed by photometrical and geometrical techniques, secondly projected by PCA, then projected by LDA, and finally using Euclidean metric and weight mechanism to make decision. These traditional solutions to the small sample size problem require the incorporation of a PCA step into the LDA framework. PCA is used as a

preprocessing step for dimensionality reduction so as to discard the null space of the within-class scatter matrix S_W of the training data set, and then LDA is performed in the lower dimensional PCA subspace.

The PCA+LDA methods discard a null space of the within-class scatter matrix S_W, and the null space may contain significant discriminative information. In [27], an LDA-based method that makes use of the null space of S_W is proposed. All the examples are firstly projected onto the null space of S_W, where the within-class scatter is zero, and then the optimal discriminant vectors of LDA are those vectors that can maximise the between-class scatter. Examples are processed directly in the original high-dimensional input space avoiding the loss of significant discriminatory information. Hence, the approach is called Direct LDA (D-LDA) or Null-space LDA. Various approaches [70, 100, 170, 26, 42] are developed based on D-LDA and are endeavoured to model and calculate the null space. The main drawback of these approaches is that the computational complexity of determining the null space S_W is very high due to the high dimension in S_W itself.

In [15], there is a comparative performance analysis carried out on four methods, which are a correlation-based method, a variant of the linear subspace method [144], a PCA method, and a LDA method. The four methods are tested on the Harvard database [62] and the Yale database [55]. The conclusions are drawn such as

1. All linear methods perform well if images in training set are similar to images in testing set.

2. The LDA method appears to be the best at extrapolating and interpolating over illuminant changing.

3. The largest three principal components does not improve the performance of the PCA method, so that removing these components can gain the performance.

4. The best number of selected principal components (PCs) is about 45.[1] The performance drops, when more than 45 components are used,

5. The LDA method appears to be the best on illuminance changing, but suffers when confronted with various facial expression.

[1]the conclusion is corresponding to the experiment in Appendix A where the best number is 49 very close to 45.

Recently, some progress has been made on linear subspace approaches. In two-dimensional PCA [171], an image is represented as a 2-D matrix instead of as a vector, and a two-dimensional PCA algorithm for face recognition is developed. Similar to two-dimensional PCA, Kong *et al.* [86] generalised the conventional LDA into 2-D Fisher discriminant analysis and applied it to face recognition.

Summary on linear approaches

In PCA, the eigenface coefficients in face representation are the most descriptive among the three major linear approaches. Due to many established algorithms or source codes, the PCA is easy to be implemented, and also servers as a baseline algorithm in face recognition. However, the PCA is not discriminative for class separation, since it does not take any class information into account. The ICA utilises higher-order statistics in the training data instead of only the second-order statistics in PCA, but there is no general closed-form solution in ICA. In addition, iterative methods, which are used to obtain the ICA representation, cause large amount of computational time. The LDA utilises the class information in the derivation of the representation for the face recognition task, and is a classification problem instead of subspace represetnation problem. The recognition rate in LDA is the highest among the three major linear approaches. However, due to the small sample size problem, the within-class scatter matrix becomes singular, which leads to poor performance on LDA. In Appendix A, the experiment of PCA and Canonical Variate is presented, which is also corresponding to the findings in the chapter.

2.1.2 Non-linear Analysis

In non-linear analysis, it is assumed that objects reside in non-linear structures. The non-linear structures are normally described as *manifold*. A manifold is an abstract mathematical space in which every point has a neighbourhood which resembles an Euclidean space, but in which the global structure may be more complicated. In a one-dimensional manifold (or one-manifold), every point has a neighbourhood that looks like a segment of a line. Examples of one-manifolds include a line, a circle, and two separate circles. In a two-dimensional manifold, every point has a neighbourhood that looks like a disk. Examples include a plane, the surface of a sphere, and the surface of a torus. Manifolds are important objects

in mathematics and physics because they allow more complicated structures to be expressed and understood in terms of the relatively well-understood properties of simpler spaces.

The face manifold is more complicated than that in linear subspace models. Linear subspace approaches approximate the non-linear manifold in a linear way. Therefore, non-linear manifold modelling is directly investigated to find the corresponding non-linear manifold. The comparison between linear approach and non-linear approach is given in Figure 2.6. In Figure 2.6(a), the principal com-

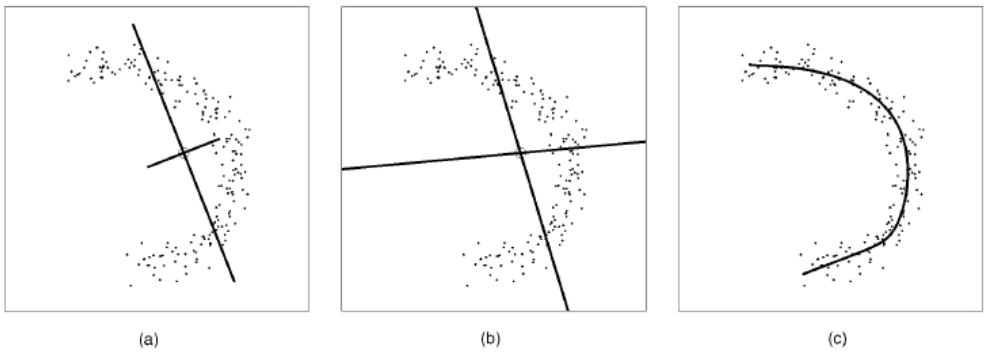


Figure 2.6: (a) PCA basis (linear, ordered, and orthogonal). (b) ICA basis (linear, unordered, and nonorthogonal). (c) Principal Curve (parameterised nonlinear manifold). [112]

ponent vectors are obtained with a toy data set corresponding to an essentially one-dimensional manifold. The first and second components are orthogonal to each other. In Figure 2.6(b), the ICA produces two unordered nonorthogonal components, one of which is roughly aligned with the first principal component. Figure 2.6(c) is an example of a principal curve, which is the simplest methods for principal manifold. The principal curve yields a compact and relatively accurate representation of the toy data. Also it displays that for capturing the non-linear manifold, non-linear approach is superior to linear subspace approach.

The classical non-linear approaches such as Kernel PCA [143], Kernel LDA [172], Kernel ICA [10] and Isomap embedding [156] are introduced in this subsection.

Kernel PCA

The kernel PCA (KPCA) [143] is to apply an non-linear mapping from the input space $\mathcal{R}^N$ to the feature space $\mathcal{R}^L$ by $\Psi(x) : \mathcal{R}^N \rightarrow \mathcal{R}^L$, where L is larger than N and possibly infinite. The mapping $\Psi(x)$ is made by the use of kernel functions meeting the Mercer's theorem [161]

$$k(x_i, x_j) = (\Psi(x_i) \cdot \Psi(x_j)), \tag{2.6}$$

where the kernel function $k(x_i, x_j)$ in the input space corresponds to dot-products in the higher dimensional feature space. Because computing a covariance matrix is based on dot product, PCA in the feature space can be formulated without direct computation of $\Psi(x)$. Assuming that the mean of the projection of the examples in the feature space is equal to zero, the covariance is given by

$$\Sigma_K = \langle \Psi(x_i)\Psi(x_i)^T \rangle \tag{2.7}$$

with resulting eigenvector equation $\lambda V = \Sigma_K V$. The eigenvector solutions V exist with coefficients $\{\omega_i\}$ such that $V = \sum_{i=1}^{T} \omega_i \Psi(x_i)$, where T is the total number of training examples. The equivalent eigenvalue problem can be formulated as

$$T\lambda\omega = K\omega \tag{2.8}$$

where ω is a set of $\{\omega_i\}$, and K is a $T \times T$ matrix. The KPCA principal components of any input example can be computed with simple kernel computation against the data set. The n-th principal component y_n of the example x is given by

$$y_n = (V^n \cdot \Psi(x)) = \sum_{i=1}^{T} \omega_i^n k(x, x_i) \tag{2.9}$$

Moghaddam [112] extends the KPCA approach with a *maximum a posteriori* (MAP) matching rule using a Bayesian similarity measure derived from dual probabilistic subspaces. In [181], the KPCA application in face recognition is demonstrated and the recognition performance indicates its advantage over other traditional subspace approaches.

Kernel LDA

The Kernel Fisher Linear Discriminant (KFLD) [172] is similar to KPCA. The projected examples are centred in the feature space. The within-class and between-class scatter matrices are denoted by S_W^Ψ and S_B^Ψ, and applying FLD in kernel space. The eigenvalues λ and eigenvectors w^Ψ of

$$\lambda S_W^\Psi \omega^\Psi = S_B^\Psi w^\Psi \tag{2.10}$$

which can be obtained by

$$W_{opt}^\Psi = \arg\max_{W^\Psi} \frac{|(W^\Psi)^T S_B^\Psi W^\Psi|}{|(W^\Psi)^T S_W^\Psi W^\Psi|} \tag{2.11}$$

where $W_{opt}^\Psi = [w_1^\Psi \quad w_2^\Psi \quad \dots \quad w_m^\Psi]$ is the set of generalised eigenvectors corresponding to the m largest generalised eigenvalues.

For given classes t and u and their examples, the kernel function is defined by

$$(K_{rs})_{tu} = k(x_{tr}, x_{us}) = \Psi(x_{tr}) \cdot \Psi(x_{us}) = \Psi(x_{tr})^T \Psi(x_{us}) \tag{2.12}$$

Let K be an $m \times m$ matrix, where K_{tu} is a matrix composed of dot product in the feature space $\mathcal{R}^L$. From the theory of reproducing kernels, any solution $w^\Psi \in \mathcal{R}^L$ must lie in the span of all training examples in $\mathcal{R}^L$. The solution is obtained by solving the following equation

$$\lambda K K \alpha = K Z K \alpha \tag{2.13}$$

where Z is an $m \times m$ block diagonal matrix. The $\Psi(x)$ is projected into a lower dimensional space spanned by the eigenvectors w^Ψ in a way similar to KPCA.

In [172], KFLD achieves lower error rates than those in PCA, LDA, ICA, KPCA approaches in face recognition.

Kernel ICA

The kernel based ICA is proposed by Bach and Jordan [10], who use contrast functions based on canonical correlations in a reproducing kernel Hilbert space to develop a new class of ICA algorithm.

ISOMAP Embedding

Isometric Feature Mapping (ISOMAP) [156] is a global optimal, and asymptotic method in which convergence guarantees the flexibility to learn a broad class of nonlinear manifolds. The approach seeks to preserve the intrinsic geometry of the data, as captured in the geodesic manifold distances between all pairs of data points. The core is to estimate the geodesic distance between far away points, given only input-space distances. For neighboring points, input space distance provides a good approximation to geodesic distance. For faraway points, geodesic distance can be approximated by adding up between neighboring points. These approximations are computed efficiently by finding shortest path in a graph with edges connecting neighboring data points. Figure 2.7 describes how ISOMAP exploits geodesic paths for nonlinear dimensionality reduction on the "Swiss roll" data set. In Figure 2.7(A), there are two arbitrary points (circled) on a nonlinear

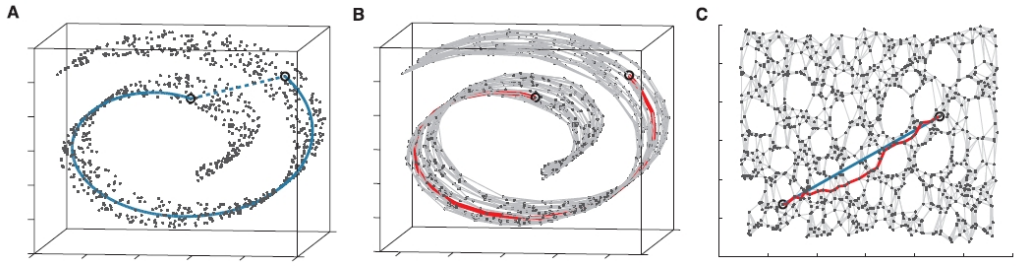


Figure 2.7: ISOMAP exploits geodesic paths on the "Swiss roll" data set.

manifold. The Euclidean distance between them in the high dimensional input space may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). In Figure 2.7(B), the neighbourhood graph G constructed in ISOMAP with k-near neighour with $k = 7$ on 1000 data points. It allows an approximation (red segments) to the true geodesic path to be computed. Figure 2.7(C) shows the two-dimensional embedding recovered by ISOMAP, which best preserves the shorest path distances in the neighbourhood graph. Straight lines in the embedding represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph path (red lines).

Yang [173] presents an extended ISOMAP method that utilise LDA for pattern classification, and shows promising results compared with other best classification

methods.

2.1.3 Bayesian Framework

In some face recognition systems, a probabilistic similarity between faces is measured based on Bayesian framework. In [114, 113], the eigenface method is proposed based on simple subspace norms with a probabilistic model. The similarity between faces on a standard Bayesian analysis of image is divided into two classes: *intra-personal difference* and *extra-personal difference*. The intra-personal difference measures the variations in the appearance of the same person with different expressions and illuminations, while the extra-personal difference measures the variations in the appearance between different persons. The high-dimensional probability density functions for each class are estimated from training data with either single modal density or multimodal densities. The similarity of faces is measured based on *maximise a posteriori* (MAP) probability to decide intra-personal difference or extra-personal difference. In [113], Moghaddam *et al.* also proposed an face alignment process (see Figure 2.8) used in face recognition systems.

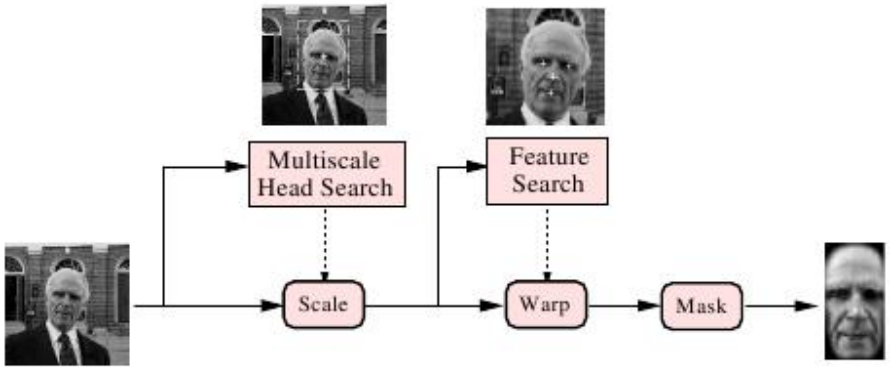


Figure 2.8: The face alignment process [113]

2.1.4 Summary on appearance-based approaches

Over the past 30 years, appearance-based face recognition has been explored and many efficient algorithms have been developed. However, current appearance-based approaches in face recognition systems encounter difficulties in practice due

to the small number of available training face images and complex facial variations in the test images. Human face appearance has a number of variations in different lighting conditions, head pose, and facial expressions. There are only small number of images avaliable for training. If a sufficient amount of representative data is not available, a switch [106] from non-discriminant techniques to discriminant techniques may lead to poor system performance. Some other techniques, such as face synthesis, can obtain additional training images from the available training data. These techniques [177] are helpful to enhance the performance of face recognition. Furthermore, the techniques such as classifier fusion [101] can help to enhance the performance of face recognition systems.

2.2 Model-Based Face Recognition

The model-based approaches in face recognition is to build a model of the human face, which is a flexible model capable to capture facial variations. The prior knowledge of a human face is applied in building up the model, such as the distance between relative feature positions among facial landmarks (i.e. eyes, nose, etc). Early approaches [53, 21, 82, 80] are developed based on localising positions of facial landmarks. A recent feature-based system, *i.e.*, Elastic Bunch Graph Matching [167], is to find the topological relationship and feature values. Active Shape/Appearance Model [29] uses shape and texture information. The model-based approaches usually contain three stages: model construction, model fitting/matching, and recognition.

2.2.1 Early Approaches

The original approach for face recognition can be traced to 1888 by Galton [53]. A framework for face recognition is proposed by collecting facial profiles. The facial profiles are drawn into curves as shown in Figure 2.9, and the norm of these curves are discovered. The facial profiles are classified by measuring their deviations from the norm.

Following the idea, many face recognition systems have been developed in 60s to 70s of 20th century due to the employment of computers. In [21], facial feature points are manually located by human operators. The positions of these points are feed to nearest neighbour or other classifiers for identifying the label

Figure 2.9: The curves depicted from facial profiles [53]

of the test image. A similar face recognition system without human intervention is developed by Kelly [82]. The recognition is determined by measurements of features, *e.g.*, width of head, distance between eyes, distance from top of head to eyes, distance between eyes to nose, and distance from eyes to mouth. In [80], facial feature points are located in two stages: coarse-grain stage and fine-grain stage. The coarse-grain stage simplifies the succeeding differential operation and feature finding algorithms. The fine-grain stage confines the processing to four smaller regions. The two stage processes are depicted in Figure 2.10.

Human face is characterised by geometrical parameterisation. These parameters may be the distance or angles between key feature points on face images. These methods are limited by storage capacity and computational speed, and are not popular now.

2.2.2 Elastic Bunch Graph Matching

Elastic Bunch Graph Matching (EBGM) is originated from Dynamic Link Architecture (DLA) framework [89]. The DLA started by computing Gabor wavelets, and then it performs a flexible template comparison between resulting image decompositions using graph-matching. Object recognition is formulated as elastic graph matching, which is operated by loose optimisation of a cost function. Figure 2.11 shows the example of DLA model and graph matching on human faces. The left part displays the example of a stored object - human face, represented by a rectangular grid form of model graph. The vertices are labeled with magnitude coefficients of Gabor wavelets. The matching begins with an undistorted copy of the object graph. The right part displays the results after grid graph matched

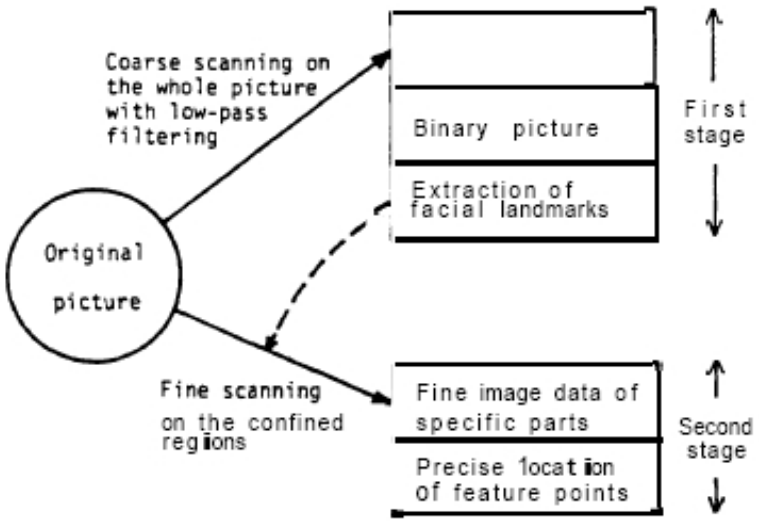


Figure 2.10: Two-stage process for computer measurement of features of human-face photographs. [80]

with face images. The graph is firstly positioned by "global moves", and it is then modified by individual Gabor wavelets diffusion. The grid demonstrates that the graph is accepted as the best match.

Graphs

The model of EBGM approach is called *labeled graph*. A labeled graph $\mathcal{G}$ contains N nodes connected by E edges. The nodes are called *fiducial points*, which are facial landmarks, *e.g.*, the pupils, the corners of the mouth, the tip of the nose, the top and bottom of the ears. The graph is object-adapted. If the object is represented by human faces, the geometrical structure is adapted to the structure of human faces as shown in Figure 2.12. There are face-adapted graphs for different poses. These nodes are positioned automatically by elastic bunch graph matching. Each node is also known as a jet. A jet is based on a wavelet transform, defined as a convolution of the image with a family of Gabor kernels. Hence, a jet J is defined as the set of 40 complex Gabor wavelet coefficients obtained for one image pixel. These 40 Gabor wavelets are varied with 5 different spatial frequencies and 8 orientations. The jet containing the imaginary and magnitude of the coefficients

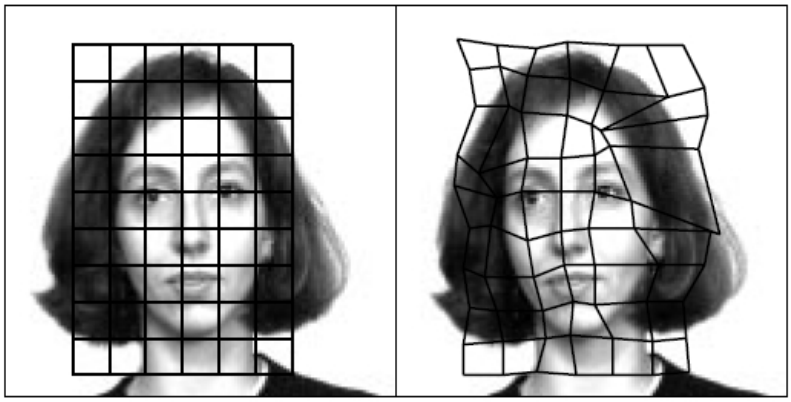


Figure 2.11: The example of DLA model and graph matching.[89]

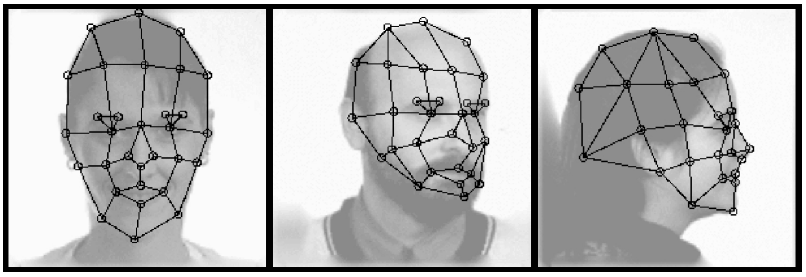


Figure 2.12: Multiview faces overlaid with labeled graphs. [167]

from the convolution is demonstrated in Figure 2.13. The graph representation of a face is based on the Gabor wavelet transform, *i.e.*, a convolution with a set of Gabor wavelet kernel. The phase coefficients vary approximately with wavelet frequencies.

To extract image graphs automatically, a general representation rather than individual models is needed for face recognition. The representation should cover a wide range of possible variations in the appearance of faces, such as different contoured eyes, mouth, or noses, beards, variations with different gender, age and race, etc. The general representation is in a stack-like structure, which combines a set of M individual model graphs. The combined graph is called a *face bunch graph* (FBG). Figure 2.14 shows the Face Bunch Graph. Each model graph has the same structure. Nodes are referred to identical fiducial points called bunch. An eye bunch containing a set of jets, may represent closed, open, female and male

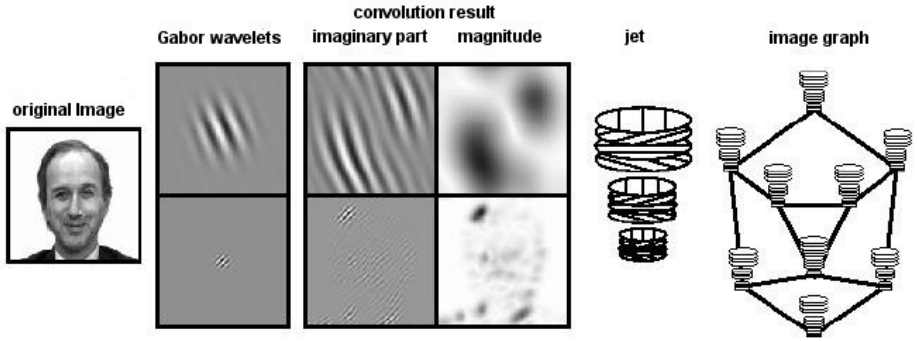


Figure 2.13: The graph representation of a face is based on the jets. [168]

eyes etc.

Graph Matching

To represent a new face, the nodes are positioned on the face image by elastic bunch graph matching. A first set of graph is generated manually by locating fiducial points and connecting edges between them. Once an FBG is well defined, graphs for new images can be automatically generated by elastic bunch graph matching. Matching a FBG with a new image is done by maximising a graph similarity between an image graph and the FBG of identical pose. The graph similarity depends on the similarities of the jets and the topographic structure. Since the FBG provides several jets for every fiducial point, the best jet can be selected and reviewed. A heuristic algorithm is used to find the image which maximises the graph similarity function. The algorithm is as follows:

1. The location of the face is found by a sparse scanning of the FBG over image.
2. The size and position of face are refined. The FBG is varied in size and aspect ratio to adapt the right format of the face.
3. All nodes are moved locally and relative to each other to optimise the graph similarity.

The coarse to fine approach is applied so that the graph is extracted from the face image.

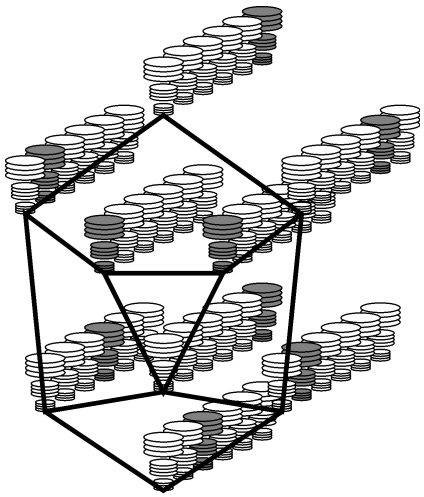


Figure 2.14: The Face Bunch Graph (FBG) serves as a general representation of faces. [167]

Recognition

After the graph has been generated from a new face image, the face is identified by comparing the similarity between an image graph and all model graphs. The graph with the highest similarity value is selected as the identity. The similarity function is an average over the similarities between pairs of corresponding jets. In [155], the recognition of EBGM is enhanced by an approach using Support Vector Machine (SVM) which addresses the derivation of optimal coefficients. The coefficients measuring the local similarity values are determined by the elastic graph matching procedure at each grid node.

2.2.3 Active Appearance Model

An Active Appearance Model (AAM) is a statistical model generated by combining a model of shape variation with a model of texture variation. In the AAM, "texture" means the pattern of grey-level values (intensities) or colours across an image patch. The AAM successfully generalises almost every valid examples, but it brings optimisation difficulties on parameter selection.

Shape Models

Building a shape model is based on a set of annotated face images. In these annotated face images, landmark points on faces are marked manually. These landmarks correspond to the key positions and outline of faces (as shown in Figure 2.15(a)). The shape is described in n landmark points in 2-D images, so that the shape is represented by a vector $\mathbf{x} = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T$. In [40], there are 400 labelled face images , *i.e.*, 400 different shapes in the face image dataset. To align these shapes into a common coordinate frame, *Procrustes analysis* [56] is used to minimise the sum of distances of each shape to the mean. Because the dimensionality of the shape vector $\mathbf{x}$ is high, PCA is used to reduce it into a more manageable manner. The shape vector $\mathbf{x}$ can be approximated through

$$\mathbf{x} \approx \bar{\mathrm{x}} + \mathbf{\Phi}\mathbf{b}_s \tag{2.14}$$

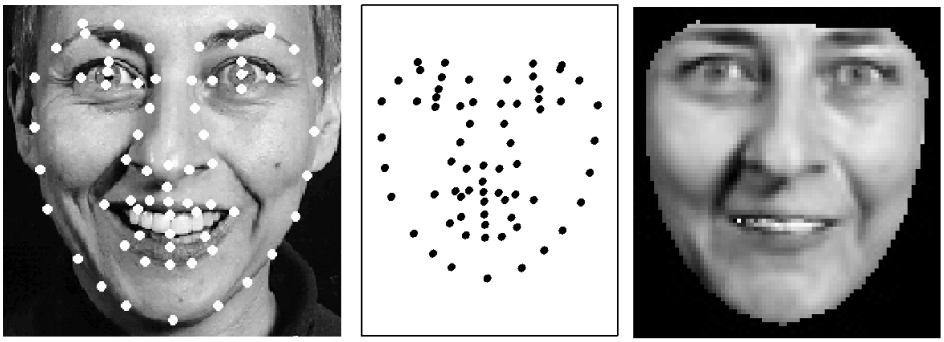
where $\mathbf{\Phi}$ contains t eigenvectors corresponding to the first t largest eigenvalues, and $\bar{\mathrm{x}}$ is the mean shape. The vector $\mathbf{b}_s$ defines a set of t parameters of a deformable model. Also Equation 2.14 can be formulated in a linear model

$$\mathbf{x} = \bar{\mathrm{x}} + \mathbf{P}_s\mathbf{b}_s \tag{2.15}$$

where $\mathbf{P}_s$ represents these t eigenvectors, also called a set of orthogonal modes of shape variation. The number t can be chosen so that the deformable model represents a 98% proportion of the total variance of the data. A model of an image is described by the shape parameter $\mathbf{b}_s$, combined with a transformation from the coordinate frame. The parameters for the shape model is $(X_t, Y_t, s, \theta, \mathbf{b}_s)$ where (X_t, Y_t) denotes the translation position, s is the scaling factor, and θ is the rotation.

Appearance Models

To build a texture model, *triangulation algorithm* [30] is used to warp each example image. The control points on each image are matched against the mean shape model, so that a "shape-free patch" is obtained (shown in Figure 2.15(c)). The shape-free patch is represented by a pixel-based grey-level vector $\mathbf{g}$. By applying PCA to the normalised patches generated from the training image set, a linear



(a) Labelled image (b) Points (c) Shape-free patch

Figure 2.15: A labelled training image gives a set of points and a shape-free patch.[28]

model is obtained as

$$\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g \tag{2.16}$$

where $\bar{\mathbf{g}}$ is the mean normalised grey-level vector, $\mathbf{P}_g$ is a set of orthogonal modes of grey-level variation, *i.e.*, a set of eigenvectors, and $\mathbf{b}_g$ is a set of texture parameters.

The shape and texture of any example can be combined by the parameter vectors $\mathbf{b}_s$ and $\mathbf{b}_g$. The combined appearance model is

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{P}_s^T (\mathbf{x} - \bar{\mathbf{x}}) \\ \mathbf{P}_g^T (\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix} \tag{2.17}$$

where $\mathbf{W}_s$ is a diagonal matrix of weights for each shape parameters. To explore correlations between the shape and texture variations, PCA is applied on the data, and gives a further model

$$\mathbf{b} = \mathbf{Q}\mathbf{c} \tag{2.18}$$

where $\mathbf{Q}$ are the eigenvectors and $\mathbf{c}$ is a vector of appearance parameters controlling both the shape and texture of the model.

The appearance model is built on 400 annotated face images [29]. On each annotated image, 122 landmark points are on the key positions. A shape model is generated with 23 parameters, and a shape-free grey-level model is generated with 114 parameters from the 10000-pixel patch. After the further PCA, the combined appearance model only contains 80 parameters out of 114, which cover

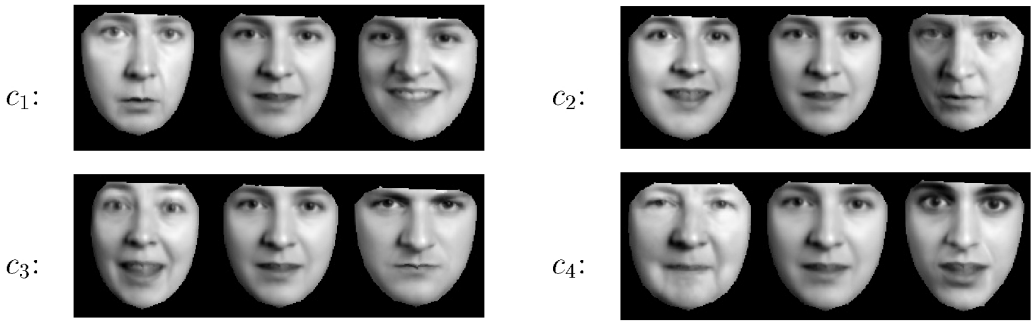


Figure 2.16: First four modes of appearance variation (±3 sd) [28]

98% of observed variation. Figure 2.16 shows the effect of varying the first four appearance model parameters, showing changing in identity, pose and expression.

Model Matching

Assuming a reasonable approximation, the model matching algorithm is to find the parameters of such a model which synthesises an image as close as to a target image. AAM matching is an optimisation problem in which the differences between a new image and a synthesised image by the appearance model is minimised. A different vector $\delta\mathbf{I}$ is defined as the subtracting between the gray-level values in the image and the gray-level values for the current model parameters. To find the best match between image and model, the magnitude of the difference vector $\Delta = |\delta\mathbf{I}|^2$ is minimised by varying with model parameters $\mathbf{c}$. The displacement of each model parameter from the correct value leads a special pattern in $\delta\mathbf{I}$ and the error in the model parameters $\delta\mathbf{c}$. In the training stage of AAM, a linear model is applied to learn the relationship between $\delta\mathbf{I}$ and $\delta\mathbf{c}$. In the matching stage of AAM, the knowledge of relationship is applied in an iterative algorithm for minimising Δ. The iterative AAM matching is shown in Figure 2.17.

Recognition

Once the appearance model has been matched to a new image, a full set of parameters can be computed. The classification system is used for recognising the individual appearing in an image. The classifier is trained by computing the ap-

Figure 2.17: The two examples of the AAM matching iterations. [28]

pearance parameters for all training images (10 images for each of the 30 individuals [91]), and establishing the distribution of appearance parameters for each individual. A discriminant analysis approach is used to enhance the effect of the inter-class variation. The metric is used as the Mahalanobis distance measure.

2.2.4 Summary on model-based approaches

The model-based approaches in face recognition exploit the intrinsic physical relationship with real faces. The physical relationship can be explained in geometric structure. Face variations due to different pose, illumination, and expression are modelled explicitly, which gives the possibility to handle these variations in practice. Prior knowledge on human faces is integrated into the approaches, so that not only statistic rules but also human knowledge are involved in face recognition. However, the model-based approaches suffer some drawbacks. 1) Constructing a model is very complicated and needs a lot of human intervention. Facial feature points are difficult to extract automatically with sufficient robustness. 2) Fitting or matching a model on an image is an optimisation process, which has highly computational costs since it is prone to be trapped into local minima. 3) The recognition results depend heavily on the results of fitting, so that there is a tradeoff between

accuracy and computational cost in the fitting process. 4) For matching a model, a high resolution and high quality image is required. The model-based approaches also need complicated initialisation.

2.3 3-D Morphable Face Model

The human faces can be treated as a manifold surface in a 3-D space. The 3-D morphable face model (3DMM) for face image synthesis and face recognition is developed by Blanz *et al.* [19, 20]. One advantage of the 3-D morphable face model is that it can easily handle variation on pose and illumination instead of 2-D models, *e.g.*, EBGM and AAM. The variance of pose and illumination is always obstacles for face recognition in 2-D space. Another advantage of the 3-D morphable face model is that a 3-D face surface is extracted from a single 2-D face image, which avoids expensive 3-D face/head scan. Face recognition uses the shape and texture parameters of the model, which represent intrinsic information of faces.

2.3.1 A Morphable Model of 3-D Faces

In [20], the morphable model is acquired from 3-D scans of 100 males and 100 females, aged between 18 and 45 years. These scans are recorded with a *Cyberwave* 3030PS laser scanner. The scans represent face shape in cylindrical coordinates relative to a vertical axis centred as for the head. There are 512 angular steps covering 360 and 512 vertical steps at a spacing of 0.615mm. After the raw scans are obtained, some preprocessing is needed.

1. Holes are filled and spikes are removed on the face surface.
2. 3-D data are aligned with a 3D-3D Absolute Orientation [64].
3. Heads are trimmed along the edge of a bathing cap.
4. Heads are cut vertically behind the ears to remove the back of the head.
5. Heads are cut horizontally at the neck to remove the shoulders.

After above preprocessing, a modified optic flow method [67, 18] is applied to establish dense point-to-point correspondence between a new face and a reference

face. The shape and texture vectors of the reference face are

$$\mathbf{S}_0 = (x_1, y_1, z_1, x_2, \ldots, x_n, y_n, z_n)^T \tag{2.19}$$

$$\mathbf{T}_0 = (R_1, G_1, B_1, R_2, \ldots, R_n, G_n, B_n)^T \tag{2.20}$$

where the reference face is a triangular mesh with $75,972$ vertices ($n = 75972$), (x_k, y_k, z_k) is Cartesian coordinate for each vertex, and (R_k, G_k, B_k) is the colour values from each vertex (and $k = 1, \ldots, n$).

The PCA is performed on the set of shape and texture vectors $\mathbf{S} + i$ and $\mathbf{T}_i$ of m example faces. There $m - 1$ eigenvectors $\mathbf{s}_1, \mathbf{s}_2, \ldots$ are computed in the shape vectors from PCA by a Singular Value Decomposition (SVD) [128]. The eigenvectors construct an orthogonal basis on the shape vectors

$$\mathbf{S} = \bar{\mathbf{s}} + \sum_{i=1}^{m-1} \alpha_i \cdot \mathbf{s}_i \tag{2.21}$$

where $\bar{\mathbf{s}}$ is the average from each shape vector, $\bar{\mathbf{s}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{S}_i$. So as to the texture vectors, an orthogonal basis is

$$\mathbf{T} = \bar{\mathbf{t}} + \sum_{i=1}^{m-1} \beta_i \cdot \mathbf{t}_i \tag{2.22}$$

The model coefficients α_i and β_i are used to represent a face in an image.

2.3.2 Morphable Model Fitting

Face image synthesis defines the positions of vertices of the 3-D model with illumination and colour. During the processing of fitting a model with a novel image, not only the shape and texture parameters α_i and β_i are optimised, but also the following rendering parameters are optimised. There are 22 rendering parameters concatenated into a vector ρ: pose angles ϕ, θ, and γ, 3-D translation $\mathbf{t}_w$, focal length f, ambient lighting intensities $L_{r,amb}, L_{g,amb}, L_{b,amb}$, directed light intensities $L_{r,dir}, L_{g,dir}, L_{b,dir}$, the angles θ_l and ϕ_l of the directed light, colour contrast c, and gains and offsets of colour channels $g_r, g_g, g_b, o_r, o_g, o_b$. In analysis-by-synthesis iterations, the fitting algorithm finds model parameters and rendering parameters, and produces an image as similar as possible to the input image $\mathbf{I}_{input}$ as shown in

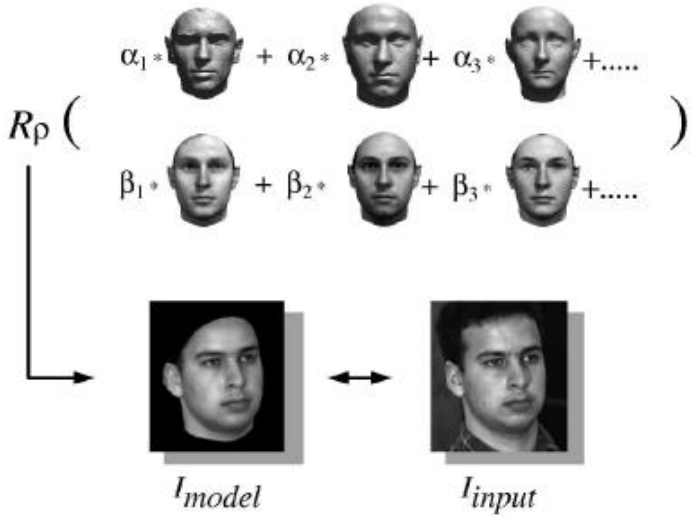


Figure 2.18: Fitting a morphable model: analysis by synthesis iterations.[20]

Figure 2.18. The goal of the fitting is to find shape and texture coefficients α and β such that rendering R_ρ produces an image $\mathbf{I}_{model}$ that is as similar as possible to $\mathbf{I}_{input}$.

The face reconstruction from a single image is shown in Figure 2.19. For initialisation, seven facial feature points, such as the corner of the eyes or the tip of the nose, are marked in image coordinates. On the morphable model, these 7 points are also defined as vertices of the mesh corresponding to the points in the image. The primary objective in analysing a face is to minimise the sum of square differences over all colour channels and all pixels in the input image and the symmetric reconstruction

$$E_I = \sum_{x,y} \|\mathbf{I}_{input}(x,y) - \mathbf{I}_{model}(x,y)\|^2 \tag{2.23}$$

A stochastic version of Newton's method is used to minimise the cost function in the fitting procedure. Because the face model is separated into four regions - eyes, nose, mouth and the surrounding face area, the optimisation is also separated by each region to obtain local parameters, *i.e.*, $\alpha_{r_1}, \beta_{r_1}, \ldots, \alpha_{r_4}$ and β_{r_4}.

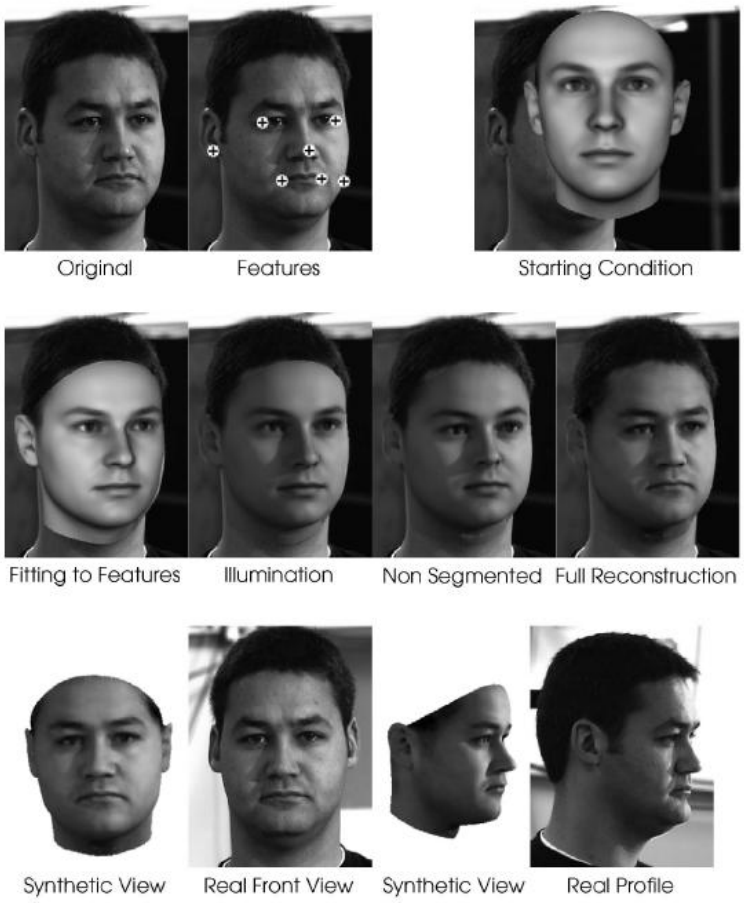


Figure 2.19: The Face reconstruction from a single image by the 3-D morphable model. [20]

2.3.3 Recognition

After fitting the model, recognition can be performed based on model coefficients α and β, which represent intrinsic shape and texture information of faces. For identification, all gallery images are analysed by the fitting algorithm, and the shape and texture coefficients are stored. Given a probe (testing) image, the fitting algorithm computes coefficients which are compared with all images in the database to find the nearest neighbour. Figure 2.20 shows the recognition scheme. Huang *et al.* [69] extended the 3-D morphable model with component-based recognition. Support Vector Machine (SVM) is used to decompose the face into a set

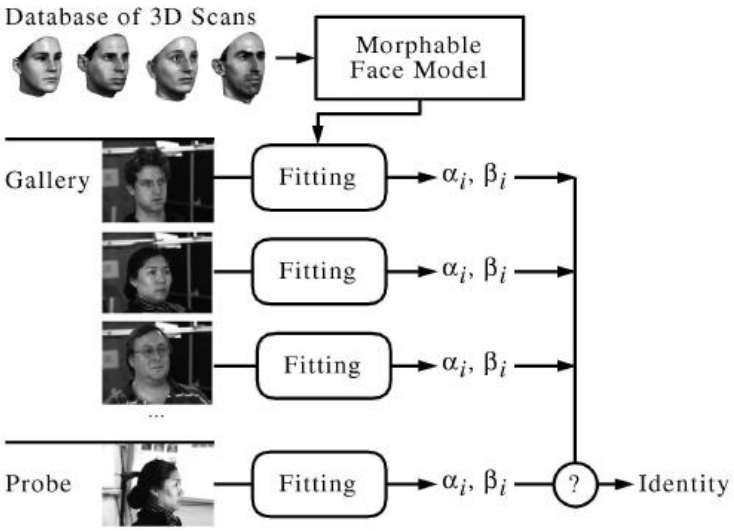


Figure 2.20: The recognition scheme in the 3-D morphable model. [20]

of components. The system achieved 90% recognition rate on a database of 1,200 real images.

2.4 Other Approaches

There is a number of other approaches which have been explored in face recognition from different aspects. The *trace transform* [79], a generalisation of the radon transform, is a new tool for image processing which can be used for recognising faces [152, 151] under transformations, *e.g.*, rotation, translation and scaling. Hidden Markov Model (HMM) [115] is an example of statistical model based approaches for face recognition. The pseudo two-dimensional HMM structure is proposed by Nefian and Hayes [116] for face detection and recognition.

2.5 Face Database

Along with the development of face recognition technology, a comparatively large number of face images in both training and testing sets have been collected. The face databases provide researchers a group of well-designed and organised face

images, which are used for training and testing. In addition, with the numerous theories and techniques in face recognition, it is clear that evaluation and benchmarking of these algorithms is crucial. The face databases also provide a platform that evaluation could be done.

While there are many databases in use currently, the choice of an appropriate database to be used should be made based on the task given (aging, expressions, lighting etc). Another way is to choose the data set specific to the property to be tested (*e.g.*, how algorithm behaves when given images with lighting changes or images with different facial expressions).

In this section, some face database popularly used by researchers are introduced. An comprehensive review on face databases on face detection, recognition and expression classification is conducted by Gross [60].

Colour FERET

The FERET [124] program set out to establish a large database of facial images that is gathered independently from the algorithm developers. The FERET database is collected in 15 sessions between August 1993 and July 1996. The database contains $1,564$ sets of images for a total of $14,126$ images that includes $1,199$ individuals and 365 duplicate sets of images. A serial of FERET evaluations attract many institutions and companies to participate.

XM2VTS

The XM2VTS database [111] contains four sessions of 295 subjects taken over a period of four months. Each recording contains a speaking head shot and a rotating head shot. Sets of data taken from this database are available including high quality colour images, $32KHz$ 16-bit sound files, video sequences and a 3-D model.

Yale

The Yale face database [55] contains $5,760$ single light source images of 10 subjects each seen under 576 viewing conditions (9 poses by 64 illumination conditions). For every subject in a particular pose, an image with ambient (background) illumination was also captured.

PIE

The PIE database [147] contains 41,368 images of 68 people, each person is under 13 different poses, 43 different illumination conditions, and with 4 different expressions.

ORL

The ORL face database [137] contains ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open or closed eyes, smiling or not smiling) and facial details (glasses or no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

MIT-CBCL

The MIT-CBCL face recognition database [165] contains face images of 10 subjects. It provides two training sets. The first training set contains high resolution pictures, including frontal, half-profile and profile view. The second training set are synthetic images (324 per subject) rendered from 3-D head models of the 10 subjects. The head models are generated by fitting a morphable model to the high-resolution training images, but the 3-D model is not included in the database. The testing set consists of 200 images per subject.

AR Face Database

The AR face database [134] contains 4,000 colour images corresponding to 126 individuals (70 men and 56 women). Images feature frontal view faces with different facial expressions, illumination conditions, and occlusions (sun glasses and scarf).

UMIST Face Database

The UMIST face database [59] consists of 564 images of 20 people. Each covering a range of poses from profile to frontal views. Subjects cover a range of race, sex, and appearance. Each subject exists in their own directory labelled and images are numbered consequently as they were taken. The files are all in PGM format, approximately 220×220 pixels in 256 shades of grey.

2.6 Summary

In this chapter, an extensive review on face recognition is conducted. Two types of state-of-the-art approaches for face recognition are introduced. A 2-D combined with 3-D approach is also reviewed. Face recognition is still a very challenging topic after 30 years of research [179]. Pose and illuminance are still two major problems that degrade the performance of face recognition systems. Appearance-based approaches focus on statistical distribution of facial texture. The texture is locally distributed on human faces. Also, appearance-based approaches abandon some prior information of human faces, *e.g.*, head, hair, ears and so on. Model-based approaches focus on the topologyl and shapes of faces, but ignore the statistical analysis of facial texture. It proves that appearance-based and model-based approaches can be combined, such that appearance-based approaches deal with local texture information, and model-based approaches deal with global shape information.

Chapter 3

Gabor Wavelet and AdaBoost

This chapter presents the fundamentals of the Gabor-Boosting algorithm: Gabor wavelets and AdaBoost. Section 3.1 discusses the Gabor wavelets and how face images are represented by Gabor wavelet features. In Section 3.2, the AdaBoost algorithm is presented, and the principle of constructing the "strong" AdaBoost ensemble classifier, is discussed.

3.1 Gabor Wavelets

A 2-D Gabor filter is a linear filter whose kernel is similar to the 2-D receptive profiles of the mammalian cortical simple cells [78]. A 2-D Gabor kernel is a 2-D Gaussian function multiplied by a 2-D harmonic function. Generally, a harmonic function is a Fourier basis function. Especially, in a 2-D Gabor kernel it is a sinusoidally modulated function, in a form of complex exponential function. The Gaussian function varies in dilation and the harmonic function varies in rotation and frequency, so that a group of 2-D Gabor filters can be formed into 2-D Gabor wavelets. Gabor wavelets capture local structure corresponding to spatial frequency, *i.e.*, scale, spatial localisation (coordinates), and orientation selectivity.

Gabor wavelets are used to extract facial information from face images. Section 3.1.1 gives the background study on Gabor wavelets. The definition of Gabor wavelet is given in Section 3.1.2. Section 3.1.3 illustrates the Gabor wavelet transform. Section 3.1.4 presents the representatives of face images - Gabor wavelet features.

3.1.1 Background of Gabor Wavelets

2-D Gabor wavelets are widely adopted as a feature extraction approach in texture segmentation [39, 38, 77, 157], iris recognition [33], face recognition [167, 168], face expression recognition [65, 103, 176], and image retrieval [105].

When Gabor filters are applied on computer vision or image processing tasks, the one biggest problem is how to select the appropriate Gabor filters, *i.e.*, finding appropriate parameters for Gabor filters. The parametric characterisation has been studied extensively and different types of schemes have emerged. In [39], 2-D Gabor filters transform different texture into detectable filter-output discontinuities at texture boundaries. Gabor filter output is modeled as Rician random variables [39]. A decision algorithm for selecting optimal filter parameters based on the Rician model is developed by Dunn and Higgins [38]. In [77], a bank of Gabor filters is characterised as uniformly covering the spatial-frequency domain, and a filter selection scheme is presented based on minimal "energy" loss in reconstructed images from the filtered images. Teuner *et al.* [157] choose parameters of Gabor filters based on the analysis of spectral feature contrasts obtained from iterations of pyramidal Gabor transforms. The work benefit from not needing prior knowledge of the texture image which means the segmentation processing is unsupervised. Since these parametrization solutions are all to choose optimal frequency or orientation of Gabor filter, an alternative wavelet scenario is proposed to bypass the optimisation.

A 2-D Gabor wavelet model with multi-scale and multi-orientation is originally proposed by Daugman [31, 33] for biometric research. In [33], by using a three-lay neural network, a nonorthogonal 2-D Gabor representation is generalised. Daugman also has applied his 2-D Gabor wavelet model on human iris recognition. In [32], the visible texture of the human iris is transformed as a sequence of multi-scale 2-D Gabor wavelet digits. Lades *et al.* [89] has applied Gabor wavelets for face recognition using the Dynamic Link Architecture (DLA) framework. The DLA starts by computing Gabor wavelets, and then it performs a flexible template comparison between resulting image decompositions using graph-matching. Wiskott *et al.* [167, 168] have extended on DLA by developing a Gabor wavelet based Elastic Bunch Graph Matching (EBGM) to label and recognise faces. Faces are represented by labelled graphs using Gabor wavelet transform. The labelled graphs consist of nodes and edges which are positioned by elastic bunch graph

matching processing for comparing similarities. The testing is done on the FERET database [123] showing a high recognition rate for frontal face images. Liu and Wechsler [96] applied the Enhanced Fisher linear discriminant Model (EFM) to an augmented Gabor feature vector [98] derived from the Gabor wavelet representation of face images. Liu [95] also presents a Gabor-based kernel Principal Component Analysis (PCA) method by integrating Gabor wavelet representation and the kernel PCA for recognition. Fan *et al.* [42] combined Gabor wavelet and Null space-based Linear Discriminate Analysis (LDA) simultaneously on each orientations for generating feature vectors.

In analysis of facial expression, the recognition is to analyse the relationship between the movements made by facial features, such as eyebrows, eyes and mouth. These facial features can be defined as point-based visual properties of facial expressions. Hong *et al.* [65] use Gabor wavelets of five frequencies and eight orientations to define a "big" General Face Knowledge (GFK) with 50 nodes[1] on a face, and a "small" 16-node GFK with three frequencies and four orientations. The method which fits these nodes with face image is the elastic graph matching proposed by Wiskott *et al.* [167, 168] in face recognition. Zhang *et al.* [176] use 34 facial points for which a set of Gabor wavelet coefficients, the Gabor wavelets with three frequencies and six orientations have been utilised. Lyons *et al.* [103] use a fiducial grid of 34 nodes and apply wavelets of five frequencies and six orientations.

3.1.2 The Definition of Gabor Wavelets

Gabor wavelets are introduced to image analysis due to their biological relevance and computational properties [78]. A Gabor wavelet [33] (sometimes, called Gabor Kernel or Gabor Elemental Function [39]) is defined as

$$\psi_{\mu,\nu}(z) = \frac{\|k_{\mu,\nu}\|^2}{\sigma^2} e^{-\frac{\|k_{\mu,\nu}\|^2 \|z\|^2}{2\sigma^2}} [e^{ik_{\mu,\nu} z} - e^{-\frac{\sigma^2}{2}}] \tag{3.1}$$

where $z = (x, y)$ indicates a point with x, the horizontal coordinate and y, the vertical coordinate. The parameters μ and ν define the angular orientation and the spatial frequency of the Gabor kernel. In Equation 3.1, the spatial frequency modulates the size of the 2-D discrete Gabor kernel, so that ν also determines the scale of kernel. The operator $\|\cdot\|$ denotes the norm operator. The parameter σ is

[1]The nodes are landmark points on human faces.

the standard deviation of Gaussian window in the kernel. The wave vector $k_{\mu,\nu}$ is defined as

$$k_{\mu,\nu} = k_\nu e^{i\phi_\mu} \tag{3.2}$$

where $k_\nu = \frac{k_{max}}{f^\nu}$ and $\phi_\mu = \frac{\pi\mu}{8}$ if eight different orientations have been chosen. k_{max} is the maximum frequency, and f is the spatial factor between kernels in the frequency domain.

Wavelets

Gabor kernels in Equation (3.1) are all self-similar since they are generated from one kernel (from one mother wavelet) by dilation and rotation via the wave vector $k_{\mu,\nu}$. Each kernel is a product of a Gaussian envelope formulated in the Equation

$$\frac{\|k_{\mu,\nu}\|^2}{\sigma^2} e^{-\frac{\|k_{\mu,\nu}\|^2 \|z\|^2}{2\sigma^2}} \tag{3.3}$$

and a complex plane wave $e^{ik_{\mu,\nu}z}$. The complex wave determines the oscillatory part of the kernel. The term $-e^{-\frac{\sigma^2}{2}}$ compensates for the Disparity Compensated (DC) value which makes the kernel DC-free. DC-free [88] is a wavelet terminology that ensures wavelets do not lose any generality such that there is no minimal energy loss when images are reconstructed by the wavelets. The effect of the DC term becomes negligible when the parameter σ, which determines the ratio of the Gaussian window width to wavelength, is a sufficiently large value.

Parametrization

In this thesis, five different scales and eight orientations of Gabor wavelets are used, i.e., $\nu \in \{-1, \ldots, 3\}$, and $\mu \in \{0, \ldots, 7\}$. Since images used in this thesis are smaller than the images with 128×128 size used in [167, 168], the scale range is from -1 to 3 rather than from 0 to 4. The eight orientations in radian are from 0 to $7\pi/8$ with a $\pi/8$ interval. Gabor wavelets are modulated by a Gaussian envelope function with relative width $\sigma = 2\pi$. The maximum frequency is $k_{max} = \frac{\pi}{2}$, and the factor $f = \sqrt{2}$. These parameters are chosen according to previous findings [168, 95]. The kernels exhibit desirable characteristics of spatial frequency, spatial locality, and orientation selectivity.

Complex Gabor

Gabor kernel is a product of a Gaussian and a complex plane wave with real and imaginary parts, also called even and odd. The Equation 3.1 is separated into real and imaginary parts, so that the real part is

$$\frac{k_\nu^2}{\sigma^2} e^{-\frac{\|z\|^2 k_\nu^2}{2\sigma^2}} \{\cos(k_\nu \cos(\phi_\mu)x + k_\nu \sin(\phi_\mu)y) - e^{-\frac{\sigma^2}{2}}\} \tag{3.4}$$

and the imaginary part becomes

$$\frac{k_\nu^2}{\sigma^2} e^{-\frac{\|z\|^2 k_\nu^2}{2\sigma^2}} \sin(k_\nu \cos(\phi_\mu)x + k_\nu \sin(\phi_\mu)y) \tag{3.5}$$

The real part and the imaginary part of 40 Gabor wavelets are shown in Figure 3.1 and Figure 3.2 respectively. These Gabor wavelets share 5 scales and 8

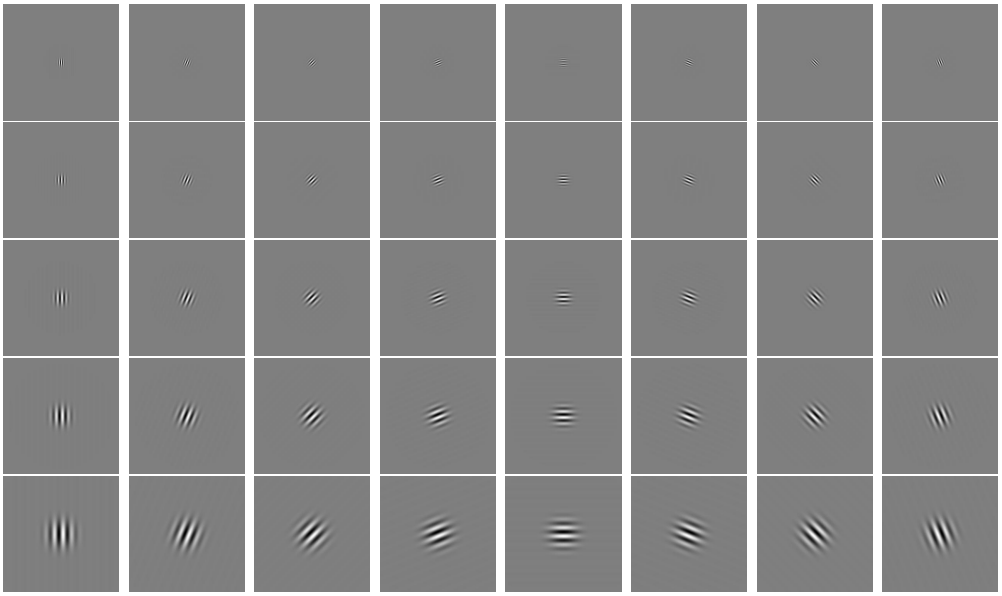


Figure 3.1: The real part of the 5×8 Gabor wavelets.

orientations. The orientations from left to right are $0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}, \frac{\pi}{2}, \frac{5\pi}{8}, \frac{3\pi}{4}$, and $\frac{7\pi}{8}$. The scales from top to bottom are -1, 0, 1, 2, 3.

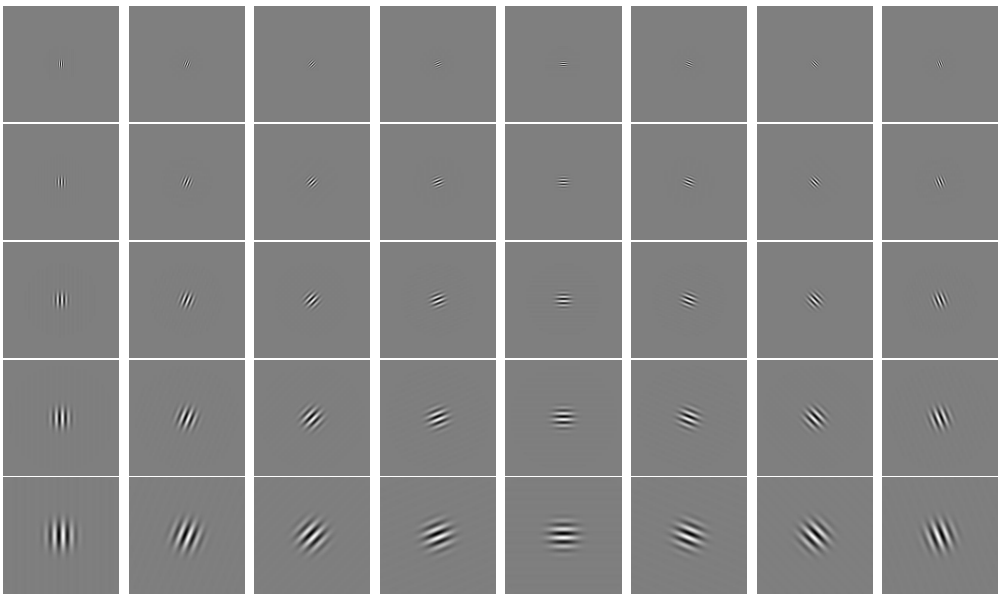


Figure 3.2: The imaginary part of the 5×8 Gabor wavelets.

3.1.3 Gabor Wavelet Transform

In computer vision, *features* stand as a piece of "interesting" information which is relevant for solving a specific vision application. In appearance based approaches, features refer to the result after a general neighbourhood operation applied on an image. The result contains local or global information which contributes to resolving a specific vision problem. An image can be represented by a group of features. A computer vision system works on these features rather than image pixels directly. Extraction of features is defined in terms of local neighbourhood operations. In this thesis, Gabor wavelet transform is the process to extract features which are relevant to face recognition. Since the selective schemes are available with a range of frequencies and orientation intervals, Gabor wavelets are ideal for face feature representation.

This section illustrates the Gabor wavelet transform by convolution. First of all, the concept of convolution is given. Secondly, the 2-D discrete convolution is presented. Thirdly, the size of mask for convolution is determined. Finally, the magnitude response is used as extracted features.

Convolution

The Gabor wavelet transform is the course of two-dimensional convolution of an image with a family of Gabor wavelet kernels defined by Equation 3.1. Two-dimensional (2-D) convolution [150] is a specific type of local neighbourhood operation which belongs to the linear approach in image analysis. The 2-D convolution g of two-dimensional functions f and h is denoted by $f * h$. The function of 2-D convolution is expressed as

$$\begin{aligned} g(x,y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a,b)h(x-a,y-b)\,\mathrm{d}a\,\mathrm{d}b \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a,y-b)h(a,b)\,\mathrm{d}a\,\mathrm{d}b \\ &= (f*h)(x,y) = (h*f)(x,y) \end{aligned} \tag{3.6}$$

where the operator $*$ symbolise the convolution operator. The 2-D convolution is an integral of the functions f and h. The convolution means a linear filtering process using the filter h on the image f. The linear filtering is often used in local image pre-processing.

2-D discrete convolution

In the 2-D discrete image domain, the linear filtering calculates the image pixel $g(x,y)$ as a linear combination of the pixel value in a local neighbourhood of the pixel $f(a,b)$. The 2-D discrete convolution is described as

$$g(x,y) = \sum_a \sum_b f(a,b)h(x-a,y-b) \tag{3.7}$$

where $f(a,b)$ is the 2-D discrete input image and $h(x,y)$ is the so-called **convolution mask** or convolution kernel. The convolution mask is often used with an odd number of pixels in rows and columns, such as 3×3, 5×5 and so on. An example [34] of a 3×3 convolution mask is

$$\begin{bmatrix} \mathrm{h}_4 & \mathrm{h}_3 & \mathrm{h}_2 \\ \mathrm{h}_5 & \mathrm{h}_0 & \mathrm{h}_1 \\ \mathrm{h}_6 & \mathrm{h}_7 & \mathrm{h}_8 \end{bmatrix}$$

where h_n is the coefficient of the convolution mask. The 2-D convolution result is the linear neighbourhood operation weighted by the corresponding coefficients in the mask h. In practice, fast convolution is used to increase the speed of the convolution. A fast convolution algorithm takes the fast Fourier transform (FFT) of the input image and the convolution mask, multiplies them together, and then performs the inverse fast Fourier transform (IFFT). In this thesis, the 2-D discrete convolution function is provided by OpenCV [90] in implementation.

The Size of Mask

In 2-D discrete convolution, the size of a Gabor filtering convolution mask is an important issue. It should be large enough to show the nature of Gabor wavelets. However, it should not be too large, otherwise computation cost will be increased. For instance, in Figure 3.1, the size of the Gabor convolution masks is 301×301, and the Gabor wavelets with the lowest frequency are shown on the top row. The span of these Gabor masks only possess a small part at the centre of the mask, and the rest of the area conveys no information. The size of a Gabor mask should be large enough to cover the shape of Gabor wavelet. The size of Gabor wavelet is determined by the spatial extent of the Gaussian envelope, which is then determined by the spatial frequency ν and the deviation of Gaussian function σ in Equation 3.1. According to Dunn *et al.* [38], the Gabor filter is truncated to six times the span of the Gaussian function. The span of Gaussian function is $\frac{\sigma}{\|k_\nu\|}$ and $k_\nu = \frac{k_{max}}{f^\nu}$, so that the Gabor mask is truncated to a width W

$$W = \frac{6\sigma}{\|k_\nu\|} + 1 = 6f^\nu \frac{\sigma}{k_{max}} + 1 \tag{3.8}$$

Taking $k_{max} = \frac{\pi}{2}$, the factor $f = \sqrt{2}$ and the standard deviation of the Gaussian $\sigma = 2\pi$, the width is

$$W = 24 \cdot 2^{\frac{\nu}{2}} + 1 \tag{3.9}$$

For five different spatial frequencies $\nu \in \{-1, \ldots, 3\}$, the corresponding size of Gabor filtering masks are 19×19, 25×25, 35×35, 49×49 and 69×69. Figure 3.3 with the $\frac{3\pi}{8}$ orientation, the corresponding real masks with the different spatial frequencies.

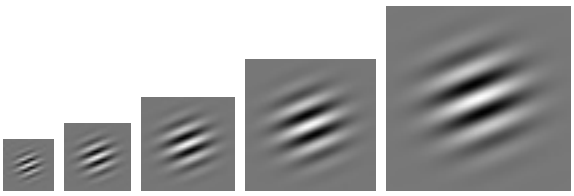


Figure 3.3: When orientation $\mu = 3$, *i.e.*, $\frac{3\pi}{8}$,the corresponding Gabor filtering convolution masks with the five different spatial frequencies $\nu \in \{-1, \ldots, 3\}$.

Magnitude Response

To get the Gabor wavelet transform of an image, the 40 Gabor wavelets are convolved with the image. Let $I(z)$, where $z = (x, y)$ defines the position in the image, the convolution of an image I and a Gabor kernel $\psi_{\mu,\nu}$ is defined as

$$O_{\mu,\nu}(z) = I(z) * \psi_{\mu,\nu}(z) \tag{3.10}$$

where $*$ denotes the convolution operator, and $O_{\mu,\nu}$ is the convolution result corresponding to the Gabor kernel at the orientation μ and the spatial frequency ν.[2] Since Gabor wavelets are of the complex form, the convolution results contain the real response and imaginary response as follow

$$O_{\mu,\nu}(z) = \Re\{O_{\mu,\nu}(z)\} + i\,\Im\{O_{\mu,\nu}(z)\}$$

where $\Re$ represents the real response and $\Im$ represents the imaginary response. The real response of Gabor filtering is an image $I(z)$ convolved with the real unit of Gabor kernel described as (3.4). The real response of Gabor filtering is

$$\Re\{O_{\mu,\nu}(z)\} = I(z) * \Re\{\psi_{\mu,\nu}\} \tag{3.11}$$

The imaginary response is the image convolved with the imaginary part described as Equation 3.5. It is expressed as

$$\Im\{O_{\mu,\nu}(z)\} = I(z) * \Im\{\psi_{\mu,\nu}\} \tag{3.12}$$

[2]In some cases, the spatial frequency ν is also called scale.

Figure 3.4: A face image selected from the **FERET** database.

Given a face image shown in Figure 3.4 in the **FERET** database [123], the 40 Gabor real responses and imaginary responses are displayed in Figure 3.5 and Figure 3.6 respectively. From Figures 3.5 and 3.6, it is hard to see a human face

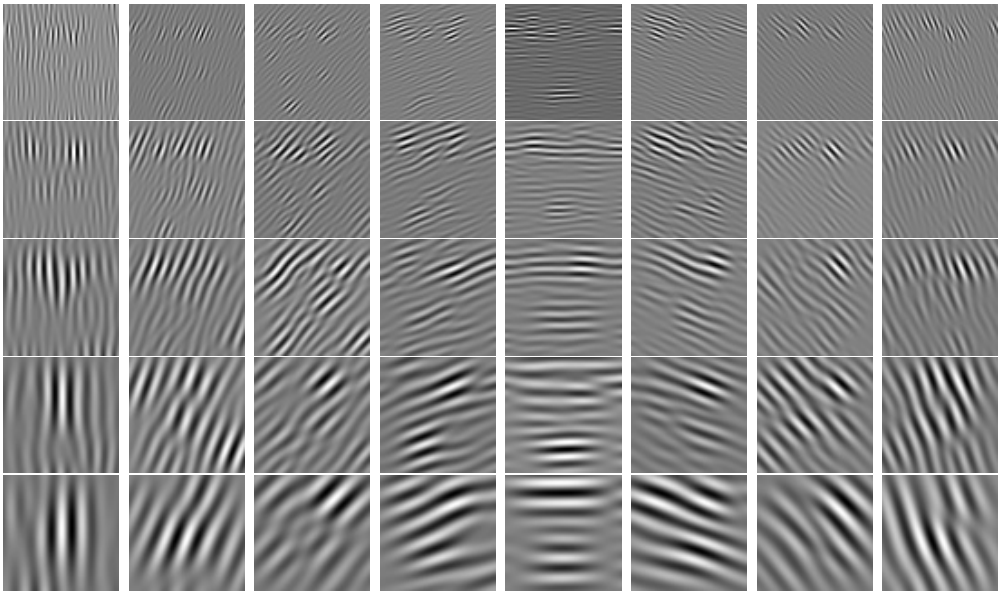


Figure 3.5: The 40 real response images.

rather than the stripes across the images.

Texture detection can be operated based on the magnitude of the output of the Gabor filtering [22]. The magnitude response of Gabor filtering is widely used in texture detection. Since the human face contains various textures, the magnitude response of Gabor filtering will enhance the recognition of a face. It is the square

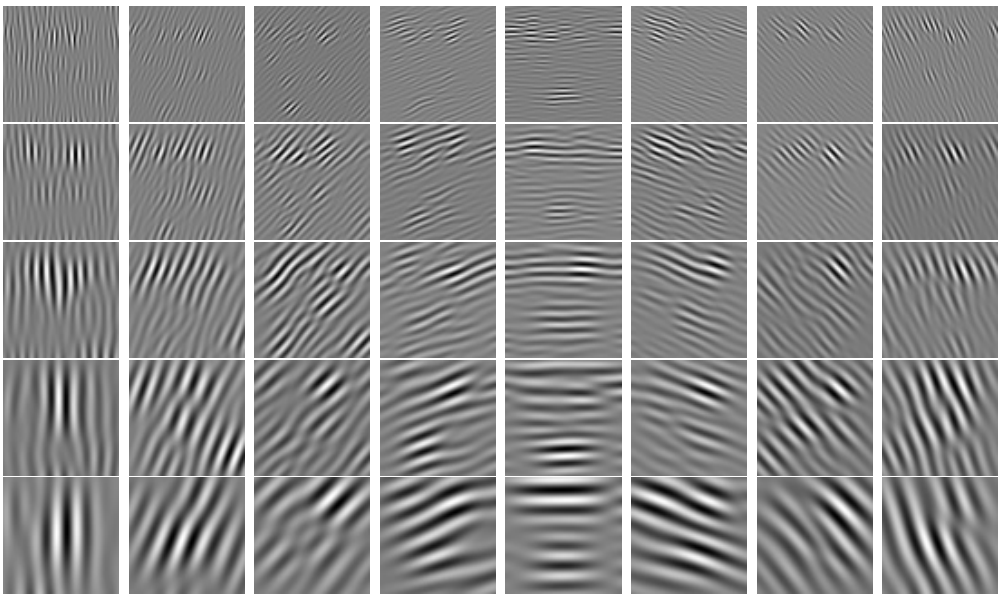


Figure 3.6: The 40 imaginary responses

root of the sum of the squared real response and imaginary response, such as

$$\|O_{\mu,\nu}(z)\| = \sqrt{\Re^2\{O_{\mu,\nu}(z)\} + \Im^2\{O_{\mu,\nu}(z)\}} \tag{3.13}$$

It can be seen that the magnitude response is modulus. These 40 Gabor magnitude responses are shown in Figure 3.7 relating to the image shown in Figure 3.4. The magnitude responses demonstrate local variance within low spatial-frequency and global variance within high spatial-frequency. In the top rows, more precise dissimilarity across the face are shown, while in the bottom rows, more high-level scale dissimilarity across the face are shown. In this thesis, the magnitude response of Gabor filtering is adopted to extract Gabor Wavelet Features.

3.1.4 Gabor Wavelet Feature

As stated before, a Gabor wavelet feature j is configured by three key parameters: the position $z = (x, y)$, the orientation ν, and the spatial frequency μ. The value of a Gabor wavelet feature is the corresponding magnitude response of a Gabor wavelet transform

$$j(z, \nu, \mu) = \|O_{\mu,\nu}(z)\| \tag{3.14}$$

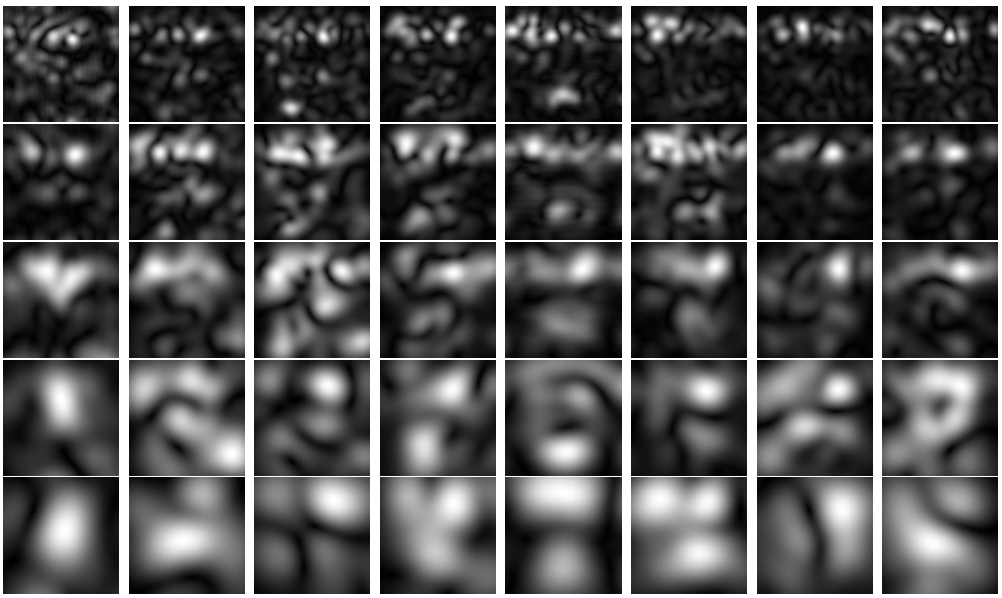


Figure 3.7: The 40 magnitude responses

Gabor wavelet features vary in orientation, frequency and position. There are eight different orientations from 0 to $\frac{7\pi}{8}$ with the interval $\frac{\pi}{8}$, and five different scales from -1 to 3. These 40 Gabor wavelets are applied on every position on a face image, so that the total number of Gabor wavelet features is determined by the number of orientations, the number of scales, and the resolution (size) of the image applied. For example, for an image with 64×64 pixels, the total number of Gabor wavelet features is $64 \times 64 \times 5 \times 8 = 163,840$. If the number of Gabor wavelets is fixed, the only factor which changes the total number of features is the size of the image.

3.2 AdaBoost

AdaBoost, standing for Adaptive Boosting, is a machine learning algorithm, formulated by Freund and Schapire [47, 48, 142]. AdaBoost can be used in conjunction with many other learning algorithms to improve their performance and learning accuracy.

This section gives a description of AdaBoost. In Section 3.2.1, a brief introduction of AdaBoost with a horse-racing analogy is given. Section 3.2.2 presents the

AdaBoost algorithm, and Section 3.2.3 describes how a "strong" ensemble classifier is constructed by a number of "weak" learners. Finally, an experiment of AdaBoost training and classification is given in Section 3.2.4 in the Iris dataset.

3.2.1 Introduction to AdaBoost

Boosting is a general method which is used to "boost" the accuracy of any given learning algorithm. Boosting is originated in a theoretical framework for machine learning called "PAC" (Probably Approximately Correct) learning model [160]. After the PAC releasing, there is a discussion whether a "weak" learning algorithm performing just slightly better than random guessing can be boosted into an arbitrarily accurate "strong" learning algorithm. Kearns and Valiant [81] are first to respond to the discussion with the positive answer. In 1989, Schapire proposed the first provable polynomial-time boosting algorithm [140]. Boosting is firstly applied to real-world application for an Optical Character Recognition (OCR) task, relying on neural networks as base learners in [37]. Although Freund and Schapire [47, 48, 142] claims that AdaBoost often tends not to overfit when running for a large number of iterations, other simulations [61] on data sets with higher noise content could clearly show overfitting side-effects. Due to the accuracy of Boosting, the algorithm has been extended for regression [49], multi-class classification [182], and unsupervised learning [130]. Recently, Boosting has been successfully implemented in various applications. OCR [37] is implemented, which used boosted classifier of neural network. In human face detection, AdaBoost was used to train a "strong" classifier to detect faces in an image [162, 163].

Horse-racing analogy An analogy [47, 48, 142] between the AdaBoost algorithm and horse-racing gambling is drawn for a better understanding on AdaBoost. A horse-racing gambler, whose purpose is to maximise his winning, wants to create a computer program helping his betting. The program would accurately predict the winner of a horse race. To create such a program, he consults a successful horse-racing gambling expert for the optimal rule to bet. However, the expert fails to answer, because the expert can not articulate a grand set of rules for choosing a winning horse. On the other hand, when the gambler provides the specific data of races to the expert, the expert can easily come up with a "rule of thumb" for the set of races. The possible "rule of thumb" is like "betting on the horse that

has recently won the most races" or "betting on the horse with the most favoured odds". The rule of thumb is very rough and moderately inaccurate, but the rule is quite reasonable. The rule can be easily concluded from the specific data rather than from a random guess. If the gambler keeps on asking the expert on different collections of races, the expert would draw many rules of thumb. To get the maximum advantage from these rules of thumb, two problems must be resolved.

- How could the gambler choose the collections of races which are given to the expert for extracting the most useful rules of thumb?
- When many rules of thumb are extracted, how are these rules combined into one single accurate rule?

3.2.2 AdaBoost algorithm

AdaBoost is an efficient method of producing a highly accurate predication rule by combining a set of rough and moderately inaccurate rules of thumb. In general, the so-called "rule of thumb" is simply designed rules which give low accuracy over long-term observation. For computer vision and pattern recognition, the word "rule" is replaced by the word "classifier", "learner", "hypothesis", etc. In this thesis, the term "learner" is used. For the term "rule of thumb", it is called *weak learner*. AdaBoost refers to a method of combining a set of "weak" learner into a "strong" classifier which gives a high accuracy for prediction. In order to design the "strong" classifier, two issues need to be resolved

- How is a set of "weak" learners organised into a "strong" classifier?
- How is a "weak" learner selected so that it contributes to a "strong" classifier?

AdaBoost resolves these two problems and gives the better performance of classification.

The AdaBoost algorithm is introduced to solve the difficulties of the earlier boosting algorithm. AdaBoost is an *adaptive* boosting algorithm, because AdaBoost adapts to the error rates of individual "weak" classifier. Pseudocode for AdaBoost is given in Table 3.1.

Table 3.1: The AdaBoost algorithm

1: Given the training set $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the data of the ith example, and $y_i \in Y = \{0, 1\}$.
2: Initialise the weights $\omega_{1,i} = \frac{1}{n}$ for each example (x_i, y_i).
3: **for** $t = 1, \ldots, T$ **do**
4: Train a weak learner h_t using the weights $\omega_{t,i}$.
5: Get the weak learner with error

$$\varepsilon_t = \sum_{i=0}^{n} \omega_{t,i} \|h_t(x_i) - y_i\|^2 \tag{3.15}$$

6: Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$
7: Update the weights

$$\omega_{t,i} = \omega_{t,i} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \tag{3.16}$$

8: Normalise the weights

$$\omega_{t+1,i} = \frac{\omega_{t,i}}{\sum_{i=1}^{n} \omega_{t,i}} \tag{3.17}$$

where the normalisation can be expressed by a normalisation factor Z_t so that all $\omega_{t+1,i}$ will keep a probability distribution.
9: **end for**
10: The final "strong" (ensemble) classifier:

$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

Training Data There are n examples in the training set $(x_1, y_1), \ldots, (x_n, y_n)$. x is the data of the example which can be integer or real, positive or negative. y is the label of the example, and it is assumed that $y \in \{0, 1\}$. In this chapter, only the two-class classification is discussed[3]. AdaBoost maintains a collection of weights on each example. All the weights ω_t are kept as a probability distribution D_t.

$$D_t(i) = \omega_{t,i}$$

At the beginning, the distribution D_t is uniform such that all weights are equal to $1/n$.

Weak Learner AdaBoost processes "weak" classifiers repeatedly in $t = 1, \ldots, T$ iteration. In each iteration, a "weak" learner h_t is trained by using the distribution D_t. To train a weak learner, there are many approaches, *e.g.*, Naive Bayes [36], Perceptron [135], Linear Discriminant Analysis (LDA) [106], Neural network [131], etc. When h_t is trained, the calculation of the error ε_t is shown in Equation 3.15. When an example (x_i, y_i) is misclassified by the weak learner h_t, *i.e.*, $h_t(x_i) \neq y_i$, the weight $\omega_{t,i}$ is added in error ε_t. When an example is classified, the weight will not be counted in ε_t. The smaller the ε_t is, the stronger is the weak learner h_t. Normally, the error ε_t is relative large, but does not exceed 0.5. In practice, a weak learner is a learning algorithm that can use the weights $\omega_{t,i}$ on the training data.

Importance Once a weak learner has been trained, AdaBoost will determine a parameter α_t, which indicates how important the corresponding weak learner h_t is in the final classifier $H(x)$. When α_t is larger, the corresponding weak learner h_t contributes more in $H(x)$. Note that $\alpha_t \geq 0$ if $\varepsilon_t \leq \frac{1}{2}$, such that α_t gets larger as ε_t gets smaller.

Updating The weight $\omega_{t,i}$ is updated using the rule shown in Equation 3.16. The effect of the updating weight is to increase the weight of examples misclassified, and to decrease the weight of correctly classified examples. In this way, AdaBoost is modulated to focus on those examples which are "hard" to classify.

[3]Chapter 6 gives the description of multi-class classification.

Normalisation The updated weights are normalised by a normalisation operator Z_t. After updating the weights, all the weights may not constitute a probability distribution, *i.e.*, the sum of all weights may not be equal to 1. To forge the weights into a probability distribution, all the weights for the next iteration will be normalised according to Equation 3.17. $\omega_{t+1,i}$ is increased to the example x_i over other examples, when x_i is misclassified and other examples are classified correctly in the tth iteration. In general, the weights only increase or decrease in a relative manner.

Final classifier The final "strong" classifier $H(x)$ is a weighted majority vote of the T weak learners where α_t is the weight assigned to h_t. The "strong" classifier is called **ensemble**, which refers to a collection of statistical classifiers. An ensemble classifier consists of a set of individually trained classifiers, i.e., weak learners, whose predictions are combined. Previous research [118] has shown that an ensemble classifier is often more accurate than any of the single classifier in the ensemble. Bagging [23] and Boosting [46] are two relatively new but popular methods for producing ensemble classifiers. Bagging is a "bootstrap" [41] ensemble method that creates individual classifiers for its ensemble by training each classifier on a random redistribution of the training set. However, Boosting is focusing on producing a series of classifiers. The training dataset used for each member of the series is chosen based on the performance of the earlier classifier(s) in the series. The AdaBoost is a particular type of Boosting in ensemble.

Training error AdaBoost is concerned to reduce the training error in the most basic theoretical properties. The error ε_t of h_t can be replaced by $(\frac{1}{2} - \gamma_t)$. γ_t measures how much h_t is better than random guessing. The training error of the final classifier H is at most

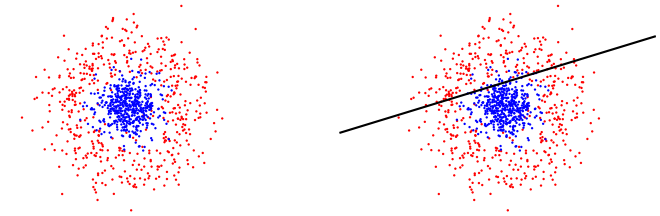
$$\prod_t [2\sqrt{\varepsilon_t(1-\varepsilon_t)}] = \prod_t \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2} \tag{3.19}$$

From Equation 3.19, if each weak learner is slightly better than random guessing, i.e. $\gamma_t > \gamma$ for some $0 < \gamma < \frac{1}{2}$, then the training error drops exponentially fast [47].

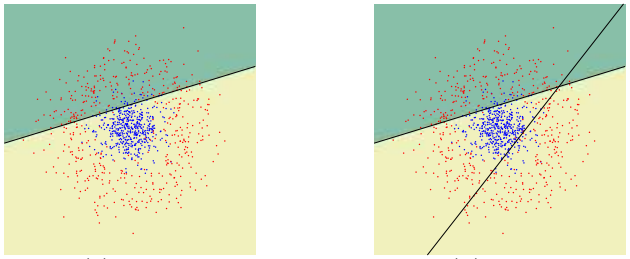
3.2.3 Construction of Ensemble Classifier

[109] gives a description of how a "strong" ensemble classifier is constructed. Figure 3.8 and 3.9 display this with an example. The original training data is shown in Figure 3.8(a). There are many points which represent the training examples: blue examples are labelled as the positive data and red examples are labelled as the negative data. The positive examples form as a random Gaussian distribution $N(0,1)$ with the centre at the origin 0 and standard deviation 1. Outside of the positive examples, the negative examples encircle the positive examples, which foumaleted as $\frac{1}{r\sqrt{8\pi^3}}e^{-1/2(r-4)^2}$ with r indicating the radius of the negative data. The original training data is not linearly separated, but non-linearly separable.

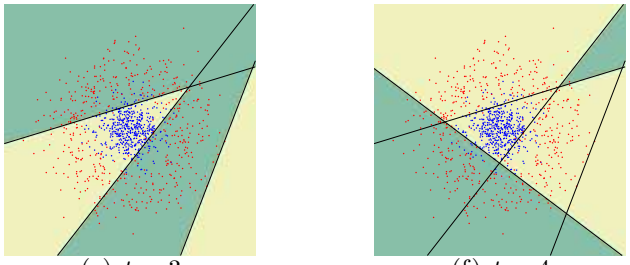
In AdaBoost, when $t = 1$, the algorithm is to find a weak learner h_1. As there are many possible solution for the weak learner, an optimal weak learner is found by minimising the weighted training error ε_1. The nature of perceptron makes the weak learner h_1 exhibit as a line in Figure 3.8(b). The line separates the training examples into two categories: plausible positive and plausible negative. In Figure 3.8(c), the examples are classified by the first weak learner, the green area reflects the plausible negative, and the yellow area reflects the plausible positive examples. In the iteration $t = 2$, the weak learner h_2 is shown in Figure 3.8(d). Because the importance α_2 on the iteration is not significant, the classification result still stays the same as the first iteration, *i.e.*, the plausible positive and the plausible negative area still stay the same. From Figure 3.9, it shows that the classification errors on the first and the second iterations are the same. When the algorithm comes to the third iteration, the third weak learner h_3 is found and displayed in Figure 3.8(e). The weak learner h_3 outlines the right-bottom boundary of the negative examples. The positive examples are constrained between the weak learner h_1 and h_2. The plausible positive area covers most positive examples, and the plausible negative area covers most negative examples, so that the classification error in Figure 3.9 drops very significantly in the third iteration. When AdaBoost comes into the iteration $t = 4$ as shown Figure 3.8(f), the 4th weak learner is found. The weak learner h_4 defines the left boundary of the negative examples, so that many negative examples are misclassified as positive examples. In Figure 3.9, the classification error grows rapidly. When the algorithm comes into the iteration $t = 5$ as shown in Figure 3.8(g), the positive examples are constrained in a triangle area, and the classification error drops again. Finally, AdaBoost goes into the



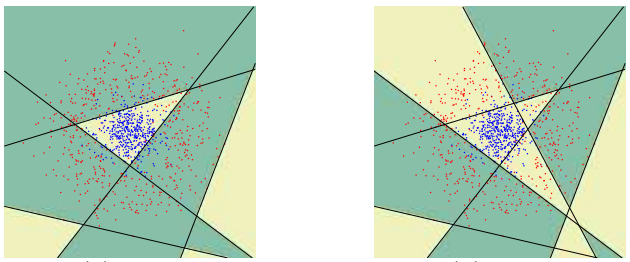
(a) The original data (b) The first weak learner



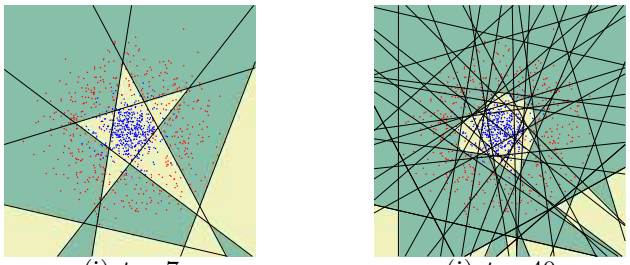
(c) $t = 1$ (d) $t = 2$



(e) $t = 3$ (f) $t = 4$



(g) $t = 5$ (h) $t = 6$



(i) $t = 7$ (j) $t = 40$

Figure 3.8: Construction of the "strong" ensemble classifier by weak learners

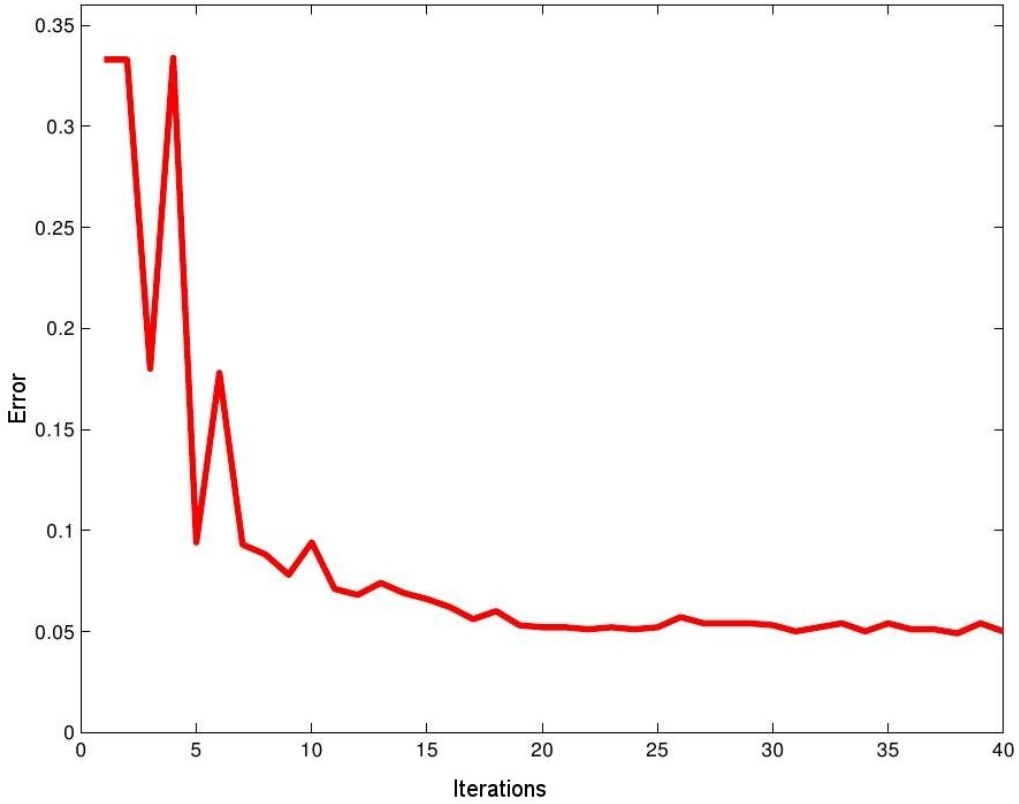


Figure 3.9: Classification error converge

iteration $t = 40$, the positive examples are outlined by many weak learners, and the classification error converge to 0.05.

From Figure 3.8, it is seen that AdaBoost uses many weak learners to fit the training data. The weak learners outline the boundary of the positive examples.

3.2.4 Iris dataset Experiment

To see how the performance can be boosted, Iris dataset is applied in AdaBoost training and classification. Two weak learners are used in the AdaBoost training to observe the boosted performance. These two weak learners are Artificial Neural Network (AAN) weak learner and Naive Bayes weak learner. Also, the two types of weak learners are compared according to the accuracy and the speed of training.

This section is organised as follow: It discusses two methods for weak learners using weights in the AdaBoost training. Then it introduces the ANN weak learner,

and introduces the Naive Bayes weak learner. The Iris data set is described and finally, the results are discussed.

Weak Learners in AdaBoost

Weak learners deal with training examples and the distribution D_t, rather than only training examples. In practice, there are two ways to process the examples and the weights.

- The weak learner may be an algorithm that uses the distribution D_t on the training examples.
- The training examples are re-sampled according to the distribution D_t, and these (unweighted) re-sampled examples are used to train the weak learner.

The most straightforward way to train a weak learner with training examples with respect to the weight is RPROP[131] Artifical Neural Network algorithm. In OpenCV, RPROP algorithm has been implemented as a class with an interface for inputting examples and corresponding weights. The alternative way is to adapt a Naive Bayes classifier as the weak learner, and resample the examples according to the weights.

Weak Learner: ANN:RPROP

An ANN is a mathematical model or computational model based on biological neural networks. It contains an interconnected group of artificial neurons. These are essentially simple mathematical models defining a function $f : X \rightarrow Y$. Each type of ANN model corresponds to a class of such functions.

In this chapter, OpenCV Machine Learning (ML) library is used for creating weak learners. The ML library implements feedforward ANN. More particularly, the ANN uses multi-layer perceptions (MLP). MLP consists of the input layer, output layer and one or more hidden layers. Each layer of MLP contains one or more neurons that are directionally linked with the neurons from the previous and the next layer.

All the neurons in MLP are similar. Each neuron has several inputs, *i.e.*, it takes the output values from several neurons in the previous layer as inputs. Each neuron has several outputs, *i.e.*, it passes the response to several neurons in the next layer. The values computed from the previous layer are summed with certain

weights, individual for each neuron, plus a bias term, and the sum is transformed using an activation function.

The larger the network size (the number of hidden layers and their sizes), the more is the potential network flexibility, and the error on the training set could be made arbitrarily small. But at the same time the learned network will also "learn" the noise present in the training set, so the error on the test set usually starts increasing after the network size reaches some limit. Besides, a larger network takes a longer time for training than a smaller one. To build an ANN weak learner, it is not necessary to design a large network, but a simple network. For an ANN:RPROP weak learner, the input layer includes the number of neurons equal to the number of elements in the examples, *e.g.*, an example containing m digits (an m length vector), has m neurons in the input layer. There is one hidden layer including two neurons to keep the network structure as simple as possible. The output layer includes one neuron which gives positive or negative results. Figure 3.10 shows the ANN structure.

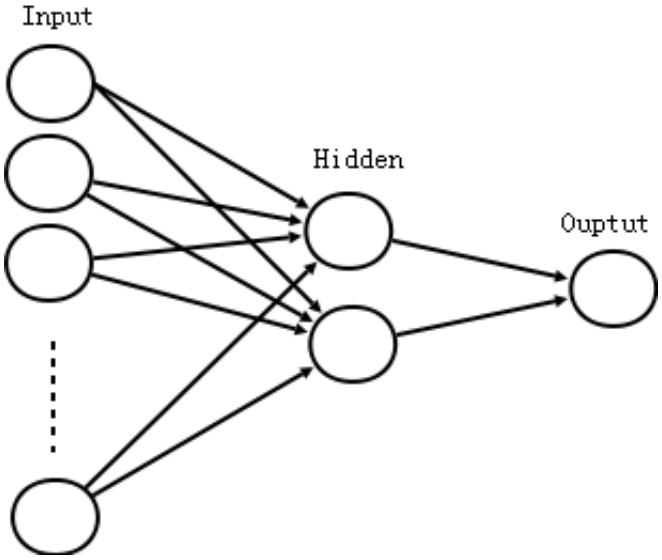


Figure 3.10: The ANN layers and nodes structure.

The RPROP (Resilient backpropagation) [131] algorithm has been shown to perform well. The ML library in OpenCV implements a batch RPROP algorithm for training MLPs.

Weak Learner: Naive Bayes

A naive Bayes classifier is a simple probabilistic classifier in which Bayes' theorem is applied with strong independence assumptions. Depending on the precise nature of a probability model, naive Bayes classifiers can be trained efficiently in a supervised learning setting. In spite of their naive design and apparently oversimplified assumptions, naive Bayes classifiers often work much better in many complex real-world situations than one might expect. An advantage of the Naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix. This simple classification model assumes that examples with the same label are normally distributed (though, not necessarily independently distributed). In two-class classification, the naive Bayes model supposes that positive examples form a normal distribution, as do negative examples.

In statistics, there are four definitions to explain the resampling process. The first one is to estimate the precision of examples' statistical properties.

To put the weights into a naive Bayes weak learner, the training examples should be resampled, *i.e.*, resampling to generate a new training set according to the given distribution. Some new training examples are generated in the process of resampling. The number of new examples generated depends on the probability. If the probability is high, more new examples are generated, and vice versa. The new examples share the same values and labels with the examples associated with the probability. The number of newly generated examples can be calculated by

$$n' = \lfloor \omega_{t,i} \times n \times fa \rfloor \tag{3.20}$$

where $\omega_{t,i}$, is the weight of corresponding example (x_i, y_i), also represents the probability of the example. n is the number of examples in the original training set. fa is an amplified scale factor which decides how the original set is expanded. $\lfloor x \rfloor$ denotes a function that returns the highest integer less than or equal to x. The amplified scale factor fa should not be small, otherwise extra new examples can not be generated. Also, the factor fa should not be too large, otherwise the total number of examples in the new set will be too large, which leads to high computational cost in training the weak learners. In this section, the factor is set

as $fa = 50$, such that the new training set is not only a moderate size, which does not cause high computational cost, but also the examples reflect the probability distribution.

In the first iteration of AdaBoost training, there is no need to resample the training set. After the first iteration, the weight for each example is updated. The distribution D_t will not stay a uniform distribution. The weights for some examples might be greater than those for other examples. Hence, the resampling process is required after AdaBoost training starts at the second iteration, and so on.

Iris flower data set

The Iris flower data set (also called Fisher's Iris data set) [8, 44] is a multivariate data set as an example of discriminant analysis. It is sometimes called Anderson's Iris data set as Edgar Anderson collected the data to quantify the geographic variation of Iris flowers in the Gaspe peninsula. The dataset consists of 50 examples from each of three species of Iris flowers, which are setosa, virginica and versicolour. Four features were measured from each example, they are the length and the width of sepal and petal. Based on four features, the species of the flower can be determined. The plot of the Iris flower data set is shown in Figure 3.11

Experiment results

The Iris flower data set is tested by the AdaBoost algorithm with implementing both the ANN weak learner and the Naive Bayes weak learner. There are three classes in the iris flower data set, *i.e.*, setosa, versicolor and virginica. Table 3.1 describes the AdaBoost algorithm in a two-class classification mechanism, so that to test the Iris flower data set according to Table 3.1 the data set is divided into three scenarios: setosa versus virginica, setosa versus versicolour, and virginica versus versicolour. In each example of the data set, all four features are used in AdaBoost training.

The classification error of the ensemble classifier is expected to be small enough, *i.e.*, close to 0. In the setosa versus virginica scenario and setosa versus versicolour scenario, AdaBoost training with both ANN and Naive Bayes weak learners are finished at the first iteration. The first weak learner (both ANN and Naive Bayes) achieves zero error rate, so that the data set is well trained and there is no need

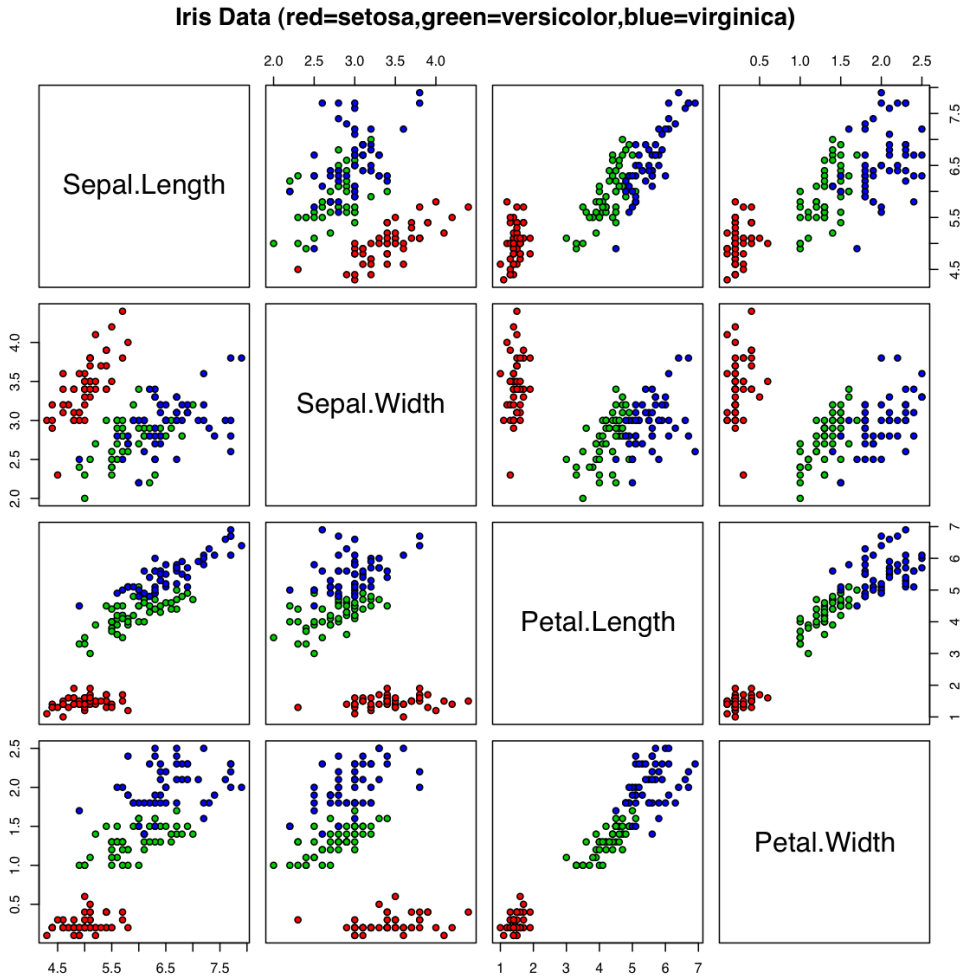


Figure 3.11: The plot of Iris flower data set

for further training. From Figure 3.11, it can be observed that the cluster of the setosa examples are distant from the cluster of the virginica and the versicolour. The setosa examples are perfectly separated from other examples. Therefore, AdaBoost may not be the most efficient method for well separated data..

In Figure 3.11, the virginica and versicolour examples are adjacent. In the feature space projected by the length and the width of the sepal, the clusters of these two classes overlap. In the iris data set, classification on virginica versus versicolour scenario is difficult. With the ANN weak learner, AdaBoost takes nine iterations to achieve zero error rate. The ensemble classifier H is made of 9 ANN weak learners $\{h_1, \ldots, h_9\}$ with the corresponding importance. The blue line in Figure 3.12 shows changes of the overall error rate of ensemble H at each iteration. Looking at the error change globally, the error rate is eventually decreasing,

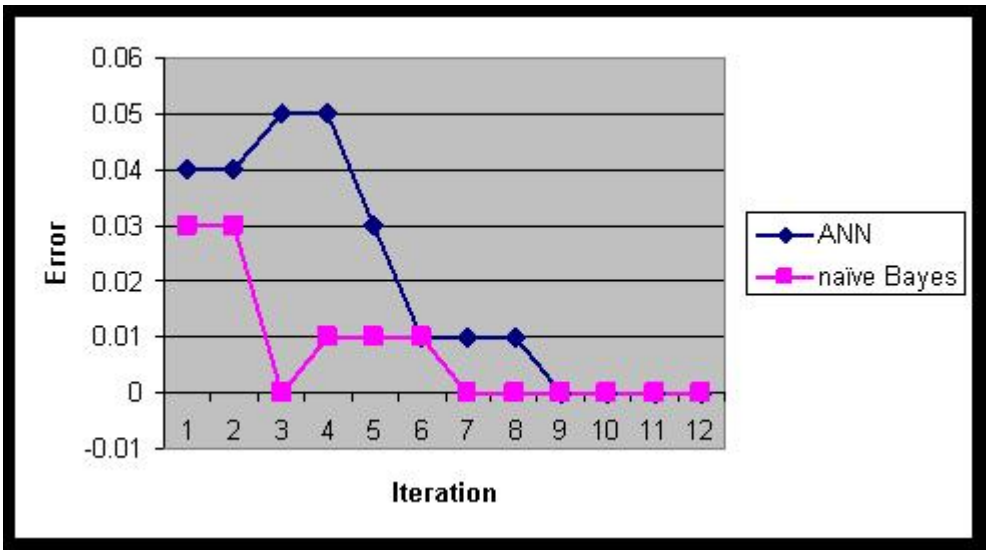


Figure 3.12: The error rates on different iterations with the ANN weak learner and the Naive Bayes weak learner.

although in some iteration, the error rate has increased.

With the Naive Bayes weak learner, AdaBoost takes seven iterations to form an ensemble H and its error rate reaches zero as the pink line shown in Figure 3.12. Although on the third iteration, the error has already reached 0.0, the training would not stop at that iteration. This is due to small example size (50 examples for each class) and inaccuracy in the training set. The error rate is eventually

decreasing, although in some iterations, errors have increased. The examples in the new training set are added by applying the resampling technique in each iteration. The number of new examples is decided by the weight of original examples and the amplified scale factor. The number of new examples in the iterations 1 to 7 is 100, 2596, 3969, 3343, 4247, 2911 and 3787.

From the result, AdaBoost is capable of performing classification on the Iris flower data set. The results from two different weak learners, the ANN and the Naive Bayes, are compared. From the perspective of accuracy, the Naive Bayes weak learner is slightly better than the ANN weak learner with less iteration to construct a strong classifier. However, due to the time consumed on resampling and training of a larger training size, the Naive Bayes weak learner is more computationally expensive than the ANN weak learner. Hence, both weak learners are appropriate for AdaBoost learning in this case.

The test explores the functions of AdaBoost classification, which displays the performance of any classification rules can be boosted from it.

3.3 Summary

In this chapter, two state-of-the-art algorithms used in computer vision and machine learning approaches, *i.e.*, Gabor wavelets and AdaBoost, are introduced. The Gabor wavelets are of similar shape as the receptive fields of simple cells in the primary visual cortex, and demonstrate significant potential in biometrics such as iris recognition and face recognition. AdaBoost is one of the most successful and popular learning algorithms. The boosting process is to construct a "strong" ensemble classifier from "weak" learning algorithms. By combining these two approaches in face recognition, the performance of the Gabor-Boosting algorithm is boosted.

Chapter 4

FLD based Gabor-Boosting

This chapter presents the Fish Linear Discriminant (FLD) weak learner based Gabor-Boosting facial feature selection algorithm. The scenario for face recognition in this chapter is face verification. In Section 4.1, face verification is generalised as a two-class classification task. The motivation of proposing the algorithm is given in Section 4.2. In Section 4.3, three schemes for feature pre-selection are introduced. In Section 4.4, FLD weak learner based AdaBoost training is developed for feature selection. Support Vector Machine (SVM) training with respect to selected features, is discussed in Section 4.5.

4.1 Face Verification

Face verification has its potential applications such as a door entry system associated with a smart card, and airport security check associated with a passport photograph, etc. In some sense, face verification could be considered as a special scenario of face recognition. However, face verification is different from face identification. In a face identification scenario, given a face image, the system decides whose face is the encountered person by comparing all faces in the database, and identifies whether the face belongs to the person. In a face verification system, the system is already given the possible identity, and only checks whether the identity is true or false. For example, in a smart card based authentication system, a chip on the card contains all information about the holder, such as name, date of birth, registration number, etc. Only the smart card holder can possess his or her smart card. When the smart card is read, the system obtains the identity of the holder.

Meanwhile a camera takes a snapshot of the holder's face to be compared with face images with the corresponding identity in the database. In general, a face verification system only needs to verify the identity rather than to tell who the identity is. From a technical point of view, face verification avoids searching a face database with all candidates. When there is a vast number of users, it could be time consuming.

From the point of view of classification, face identification can be treated as multi-class classification. Each person can be defined as one unique class in the system. All face images of such a person will be recognised as one class in the system. Most face recognition systems are able to recognise a number (≥ 3) of persons, so that face identification deals with multi-class classification. The classifier in face identification is capable of discriminating multiple faces with different criteria. Some approaches such as Principal Component Analysis (PCA) assume that there is a high-dimensional space from the data vector. The high-dimensional space is called face space, in which it is assumed that there exist many subspaces. Each subspace represents a person. Given a new face image, the identification is to find the subspace where the new face image lies. By finding the subspace of the new image, the subject (or the person) to whom the new image belongs is found.

Face verification can follow the same process as face recognition does. In face verification, there are two classes. One is called client, who is the right person, and the others called impostors. As the identity is already known, the verification is to confirm whether the new image lies in the specified subspace. If yes, it is accepted as a client who has already registered, otherwise it is rejected as an impostor. Consequently, there are only two results, *i.e.*, client or impostor. Therefore, face verification is categorised as a two-class classification. For example, there are four people registered in a face verification system as *client A*, *client B*, *client C* and *client D*. For *client A*, the system accepts face images which represent *client A*, and rejects all other face images (all face images of *client B*, *client C* and *client D*). For each client, a new classifier must be designed. The two-class classification in face verification is more efficient, because the structure of the whole system is simplified to only two subjects. Nevertheless, such a face verification system could become a high computational cost when the number of clients is large. This is because a unique classifier has to be built for each client.

4.2 Motivation

Since face verification can be treated as a two-class classification process, some two-class classification approaches can be adapted. Face detection as a well-known two-class classification approach has made rapid progress in performance. In face detection, there are two classes, *i.e.*, face and non-face. The rapid face detection algorithm was proposed by Viola and Jones [162] in 2001. A machine learning approach has been used for face detection. The approach is capable of processing images extremely rapidly and achieving a high detection rate. The work is distinguished by three key contributions, 1) a new image representation called Haar features; 2) a learning algorithm based on AdaBoost, which selects a small number of critical visual features from a larger set; 3) the detector which is made of cascade structure classifiers, that runs at 15 frames per second [162]. In this thesis, some contributions on face detection are adapted in face verification. Face images are represented by some features, like Haar features or Gabor Wavelet Features. AdaBoost selects a small number of features which discriminate clients and impostors in face verification.

4.2.1 Face Verification v.s. Face Detection

Although the AdaBoost algorithm for selecting features is described as object detection in [162], the feature selection approach is very general and can be applied for other objection detection and recognition purposes. Hence, face verification also uses the AdaBoost algorithm to select a small set of significant features representing client, and uses the selected features to construct a classifier. In face detection, the final classifier is to distinguish face and non-face, but the final classifier in face verification is to distinguish whether a face belongs to a particular individual. The difference between face and non-face is more obvious to human vision perception than face verification. In a machine vision system, when scaling a face image into a small size, the face image resembles a horizontal dark bar across the two eyes and a vertical light column across the nose. By sensing the bar and the column, plausible faces can be detected in images, and non-face images are rejected. In general, decision making in face detection is determined by global variation across the face. However, in face verification, the local variation in faces is very important to contribute to the classification. If face detection and face verification are projected into a face feature space, the distance between a face and

a non-face object is larger than the distance between individual faces. Therefore, face verification needs more precise description and representation for face images than face detection.

4.2.2 Gabor wavelet feature v.s. Haar feature

Haar features in [162] are used to extract face features, and provide a rich image representation which supports effective learning. Four examples of Haar features are used in [163] as shown in Figure 4.1. The sum of the pixels which lie within

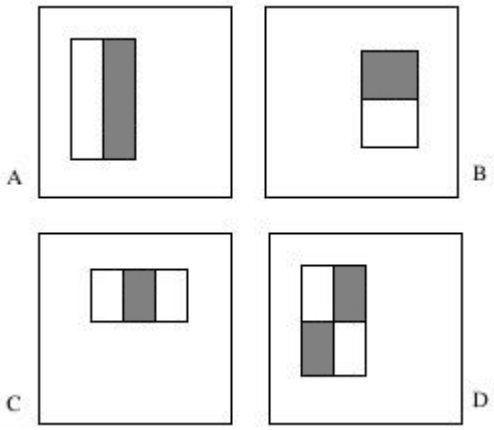


Figure 4.1: The four examples of Haar features. [162]

the white rectangles are subtracted from the sum of pixels in the grey rectangles. These four Haar features are primitives of all Haar features. All Haar features vary with positions, size of rectangle, and the types. The position determines where the Haar feature is in an image. The size of rectangle determines how large the Haar feature is. It also can be considered as the scale of the Haar feature, because with a larger size of rectangles, the Haar feature detects more global variation, whilst with a smaller size of rectangles, the Haar features detects more local precise variation. Among these four examples, there are three primitives of Haar features. For (A), the primitive represents features which detect the horizontal variation of pixels in images. For (B), the primitive represents features which detect the vertical variation of pixels in images. (C) can be considered as an alternative of (A), which also detects the horizontal changes. For (D), the primitive is more

complicated than others. There are four rectangles in the primitive, and they are in a skew-symmetric structure. The primitive represents features which detect the diagonal variation of pixels in images. Therefore, Haar features contain three kinds of variations: horizontal variation, vertical variation and skew-symmetric variation.

Haar features are similar to Gabor wavelet features. They both have the position variable. The size of rectangle in Haar features is equivalent to spatial frequency ν in Gabor wavelet features, which decides the size of Gabor wavelet kernel. The primitives are equivalent to the orientations in the Gabor wavelet features. As there are three primitives, there are three orientations in Haar feature: horizontal (0°), vertical (90°) and skew-symmetric (45°). However, Gabor wavelet features can contain any orientation. To detect the salient change of image in different orientations, Gabor wavelet features are more appropriate than Haar features. In face verification, facial texture is complicated and the salient change could be in different orientations. Gabor wavelet features will provide more precise representation than Haar features do on face verification. In [169], both Gabor wavelet features and Haar features are used on glass detection. The results show that the detection algorithm using Gabor wavelet features have better performance than the detection algorithm using Haar features. Therefore, Gabor wavelet features are used to represent faces in this thesis.

In [145, 174], two similar algorithms of Gabor wavelet features and AdaBoost algorithm are also presented. In these two algorithms, face recognition is not based on classifying subjects, but on classifying *intra-personal difference* and *extra-personal difference* [114] with difference images between two face images. However, human face image appearance has potentially very large intra-personal difference due to facial expression, occlusion (*e.g.* glasses), lighting, pose, and other issues. On the other hand, the extra-personal difference may be small due to the similarity of individual appearance. Figure 4.2 illustrates examples of appearance difference of different subjects. In Figure 4.2, (a) and (b) are images from different people, but their appearance difference in the input space is smaller than images of the same person but with different poses and lighting changes as shown in (c) and (d). Adini *et al.* [3] demonstrate that the differences between images of the same face due to lighting, viewpoint, and facial expression changes could be larger than those between different face images. Therefore, the intra-personal and extra-personal recognition mechanism may not be appropriate for face verification.

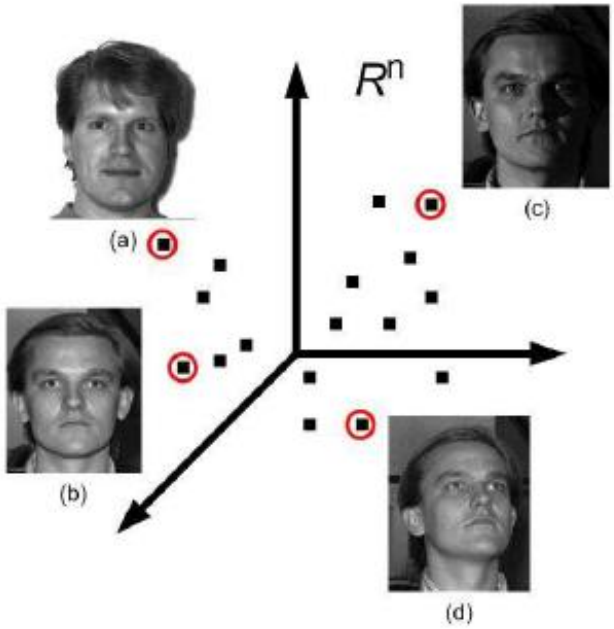


Figure 4.2: The Intra-personal difference versus the Extra-personal difference. [102]

4.3 Feature pre-selection schemes

As mentioned in Section 3.1.4, a Gabor wavelet feature j is configured by three key parameters, which are the position $z = (x, y)$, the orientation μ, and the spatial frequency ν (also called scale), so that the number of possible Gabor wavelet features is determined by the number of positions, orientations, and scales. It is common to use the parameters as $\nu \in \{-1, \ldots, 3\}$, and $\mu \in \{0, \ldots, 7\}$, and the number of position is due to the size of images processed by the Gabor wavelet transform. If all features are used in feature selection, the process will take a very long time to select significant features as face representatives. Three schemes are developed to define the number of positions.

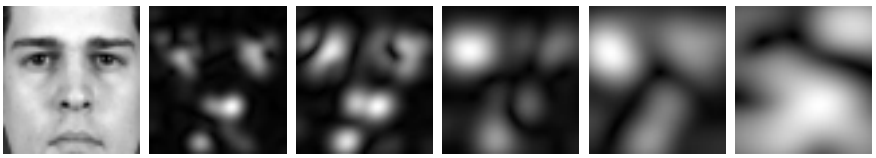
- **Full size** There is no reduction on the number of total features. The number of positions is equal to the original number of pixels;
- **Interlace** The features are sampled column by column and row by row. The number of positions is roughly one quarter of the total number of pixels;

- **Scaling** The response images are scaled down according to the spatial frequency, and the number of positions is reduced.

In **Full size**, the number of positions is the same as the number of pixels in the image, *e.g.*, if there is an image with 64 pixels in the width and 64 pixels in the height, the number of features is $64 \times 64 \times 5 \times 8 = 163,840$. If the full size scheme is adapted, unless the size of image is comparatively small, the number of features will be very large. Large numbers of features introduces difficulties to the algorithm, *e.g.*, time-consuming and complexity of design. Hence, it is better to reduce the number of features if possible. The other two schemes are grounded on this concern.

The **Interlace** scheme is similar to the Interlace technique [12] for improving the picture quality of a video signal in the Television industry. After the Gabor wavelet transform on a single image, there are 40 response images left with the same size as the original image. The scheme is to remove even rows and columns such that only odd rows and columns remain. This makes the number of pixels remaining in the response images roughly one quarter of that in the original image.

The **Scaling** scheme is used to reduce the number of features by scaling down these response images with higher spatial frequencies ν. There is a phenomenon that in higher frequency response images, the response images appear blurred. In Figure 4.3(a) is a 55×51 XM2VTS face image. Images in Figures 4.3(b), 4.3(c), 4.3(d), 4.3(e) and 4.3(f) are the magnitude of response images at frequency $\nu = \{-1, 0, 1, 2, 3\}$, respectively, and the orientation is $\frac{\pi}{4}$, *i.e.*, $\mu = 2$. As the spatial frequency increases, the response images become more blurred. This indicates that in higher frequency response images, the neighour features are highly correlated. In Figures 4.3(e) and 4.3(f), human vision perception is unable to distinguish the facial characteristics. In addition, in 2-D discrete convolution, convolving images



(a) (b) (c) (d) (e) (f)

Figure 4.3: The response images from low spatial frequency to higher frequency.

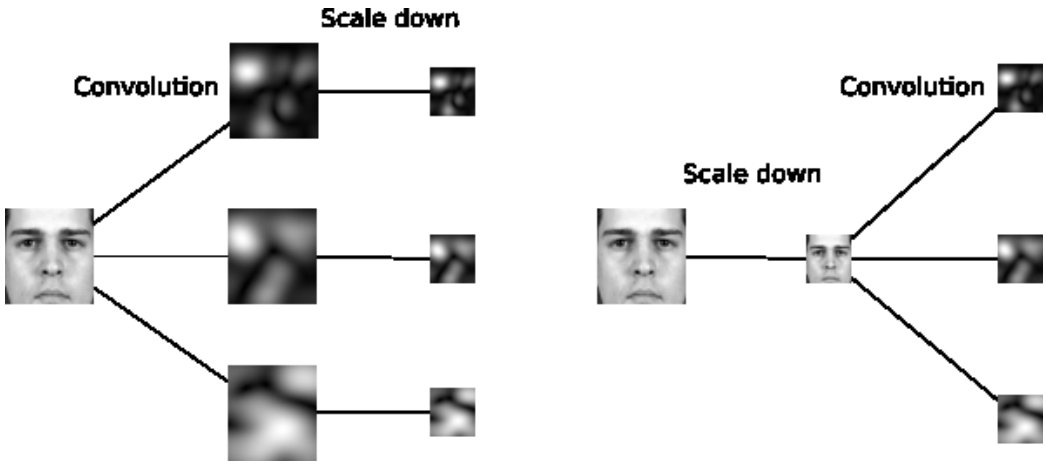


Figure 4.4: The two processes are equivalent in terms of results.

with high frequency Gabor wavelets is equivalent to scaling down images and convolving with lower frequency Gabor wavelet as shown in Figure 4.4. In the left of Figure 4.4, the original face image is firstly convolved by three Gabor wavelets with $\nu = \{1, 2, 3\}$ to obtain three magnitude responses, then these responses are reduced into the $\frac{1}{4}$ of the original size. On the right of Figure 4.4, the original face image is firstly reduced to $\frac{1}{4}$ of the original size, then convolved by three Gabor wavelets with $\nu = \{-1, 0, 1\}$. These magnitude responses are the same as those reduced magnitude responses on the left. The results from Figure 4.4 are magnified in Figure 4.5. Images in Figures 4.5(a), 4.5(b) and 4.5(c) are the results from the left part, while images in Figure 4.5(d), 4.5(e) and 4.5(f) are the results from the right part. The top row and the bottom row in Figure 4.5 show that the responses are identical. The processing in which convolution is first done and then scale down, is equivalent to the processing in which scale down is first and then convolution. Figure 4.5 also indicates that higher frequency response images can be scaled down into smaller sizes without significant information loss

Therefore, in the Scaling scheme, the number of features is reduced by scaling down the response images with higher frequencies. For an image with the size $W \times H$, the response image with the lowest frequency is kept to size of $W \times H$. For the second lowest frequency, the size of response images is $\frac{W \times H}{4}$; for the third lowest frequency, the size of response image is $\frac{W \times H}{16}$, and so on. The corresponding number of features are $W \times H \times 8$, $\frac{W \times H}{2^2} \times 8$, $\frac{W \times H}{2^4} \times 8$ and so on. If the width

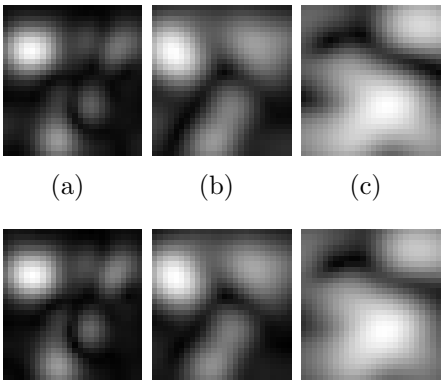


Figure 4.5: Comparison between response images after convolving with high frequency Gabor wavelet and after convolving with low frequency Gabor wavelet.

$W = 64$ and the height $H = 64$ with five frequencies $\nu = \{-1, \ldots, 3\}$ and eight orientations, the total number of features is

$$(W \times H + \frac{W \times H}{2^2} + \frac{W \times H}{2^4} + \frac{W \times H}{2^6} + \frac{W \times H}{2^8}) \times 8 = 43648$$

With **Full size**, the total number of features is $163,840$. In the **Scaling** scheme, the size deduction ratio is $43648/163840 \approx 0.27$.

4.4 Feature Selection

After applying the pre-selection scheme to reduce the feature space, the AdaBoost algorithm is applied for feature selection. This section presents the algorithm firstly, and then an elementary part of AdaBoost - FLD weak learner is introduced.

Feature selection is a technique, commonly used in machine learning and pattern recognition for reducing feature space. It selects a subset of relevant features from a large number of features to construct robust learning models. By removing irrelevant and redundant features from the training data, feature selection helps in improving the performance of learning models. Feature selection also helps to distinguish features which are important for learning. From a theoretical perspective, feature selection for classification or recognition is to obtain an optimal subset

from all features against the exhaustive search of all possible subsets.

4.4.1 AdaBoost algorithm for Feature Selection

A variant of AdaBoost is adapted for selecting most important features for face verification. The feature selection algorithm is with respect to individual clients. If the face verification system is required being able to verify all clients, the algorithm will be applied to select the most significant features for each client, such that a small number of features corresponding to each client are collected and constructed into a classifier to verify that client. A feature group is unique for a client. The algorithm for feature selection is listed in Table 4.1. In the learning algorithm, the

Table 4.1: The variant algorithm of AdaBoost for feature selection

---

Given example $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the data of the ith example, which are contributed by k features $\{j_1, \ldots, j_k\}$, and $y_i \in Y = \{0, 1\}$ for impostor and client, respectively.
Initialise the weights $\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of impostors and clients, respectively.
for $t = 1, \ldots, T$ **do**
    Normalise the weights, $\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{i=1}^{n} \omega_{t,i}}$ so that ω_t form a probability distribution.
    for all $\{j_1, \ldots, j_k\}$ **do**
        Train a weak learner h_j which is restricted to use a single feature j.
        The error is calculated as $\varepsilon_j = \sum_{i=1}^{n} \omega_{t,i} |h_j(x_i) - y_i|^2$ with respect to $\omega_{t,i}$
    end for
    Choose the optimal weak learner h_t with the lowest error ε_t from all h_j.
    Select the corresponding feature j_t from the weak learner h_t as a significant feature.
    Remove the feature j_t from the feature set $\{j_1, \ldots, j_k\}$.
    Update the weights $\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i}$, where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
end for

---

learning process contains n training examples in the training set. Each example

(x_i, y_i) is defined such that x_i is the data and y_i is the label. The data of each example contains k features $\{j_1, \ldots, j_k\}$ which construct a vector with k elements. The label y_i, also called class of the example, is 0 or 1 for impostor or client respectively.

The example is associated with a weight ω to indicate how difficult the example for a weak learner is recognised. The allocation of weights on each example is called initialisation. Different from the initialisation in Table 3.1 of Chapter 3, in this algorithm, client (positive examples) and impostor (negative examples) are given different weights with respect to the number of corresponding examples in the training set. This is due to the fact that the number of client face images may not be equal to the number of impostor face images in the training set. The weight of an example also indicates prior probability which is referred to prior knowledge of the example in the whole population of examples. When positive and negative examples are well balanced, *i.e.*, the numbers are equal, each example will share the same prior probability $\frac{1}{n}$ if there are n examples. When positive and negative examples are not balanced, the prior probability for a particular class is evaluated by the number of examples in the class dividing the total number of examples. It is assumed that there are l positive examples and m negative examples in the training set, *i.e.*, the whole population $l + m = n$, and both accumulative prior probabilities of positive and negative examples are equal to 0.5. In the perspective of the vector space, the subspace spanned by positive examples takes half capacity of the vector space, and the subspace spanned by negative examples takes the other half. Within positive examples, the prior probability of each example should be the same. That is because the examples belonging to the same class are of equal importance and no example in the same class will be more important than others at the beginning of AdaBoost learning. Hence, if an example is positive, the weight is $\frac{1}{2l}$, otherwise the weight is $\frac{1}{2m}$ at the initialisation.

After initialising the weights, AdaBoost training is going to the stage of iteration running. The number of iterations in the training is determined by the number of expected selected features. If T features are selected from the whole set of features, the number of iterations will be set to T. At the beginning of each iteration, the weights need to be normalised so that all weights form a probability distribution, *i.e.*, the weights are divided by the sum of all weights.

AdaBoost performs an exhaustive search on all features to find the optimal feature. For each feature j, the algorithm trains a weak learner h_j based on the

training examples associated with weights. The weak learner h_j is restricted to use one single feature j rather than all features $\{j_1, \ldots, j_k\}$, such that the data of each example is represented by only one numeral in the weak learner. This weak learner is called a single feature based weak learner. After the weak learner h_j is trained, the error is calculated with respect to the weights. From Table 4.1, it can be seen that the error ε is the sum of the weights whose corresponding examples are misclassified. It is obviously that the range of the error is $0 \leq \varepsilon \leq 1.0$.

After the training is done on all single feature based weak learners, the minimum error ε_t among weak learners is obtained by sorting algorithm. The corresponding weak learner h and feature j are retained. The feature j is labelled as j_t, which is the selected feature from the iteration t. The weak learner h_t is the weak learner generated from the same iteration. Also, it is necessary to remove the feature j_t from the whole feature set $\{j_1, \ldots, j_k\}$, so that the feature is excluded in the next iteration.

At the end of each iteration, the weights need to be updated according to the performance of the weak learner h_t. The weights are updated in order to emphasise those examples which are misclassified by the weak learner h_t. If an example is classified correctly by h_t, the weight $\omega_{t+1,i}$ will be decreased. If an example is misclassified, the weight will be keep the same.

4.4.2 Weak Learner: Fisher Linear Discriminant

The choice of weak learner is crucial for the process of AdaBoost training. Some considerations of weak learners are firstly given, then the FLD (Fisher's Linear Discriminant) weak learner is presented.

Weak Learner

A weak learner using a single feature is very important for the AdaBoost feature selection. There are many possible choices for weak learners, such as Artificial Neural Network (ANN), Naive Bayes, Fisher's Linear Discriminant, Support Vector Machine (SVM) and so on. The choice of weak learner is based on two factors:

- The computational cost of the weak learner training on a given training set should be low.

- The classification accuracy of the weak learner is slightly better than random guess.

Firstly, the training time for weak learner should be fast enough, otherwise the overall training of AdaBoost will take an extremely long time. For example, training a weak learner requires 1 second on a certain computer, and there are $40,000$ features in the training set, so each iteration will take roughly 11 hours. If the aim is to select 20 features, the total computational time will be about 9 days. Hence, it is important to choose a fast weak learner. Secondly, a learner is defined as "weak" so it's classification ability is not expected to be strong. The classification accuracy of a weak learner should normally be better than random guessing, which is 0.5. Hence, an ideal weak learner should be simple and fast. Among those base classifiers mentioned above, a Fisher Linear Discriminant (FLD) weak learner is an appropriate solution in selecting Gabor wavelet features [73, 87, 9]. The FLD weak learner is fast because it is a linear classifier. Non-linearity requires complex structures like SVM and ANN to produce higher accuracy, but is penalised by more computational cost. The FLD weak learner does not make the assumption on the distribution of each class, such as Naive Bayes does.

Fisher Linear Discriminant

Fisher's Linear Discriminant (FLD) comprises methods used in statistics and machine learning to find a linear boundary which best separates two or more classes of objects. FLD was first proposed by R.A. Fisher in 1936 [44]. He defined the separation between two class examples to be the ratio of *the variance between the classes* to *the variance within the classes*. For a better classification performance, the ratio should be as small as possible. By maximising the variance between classes and minimising the variance within classes, the classifier can reach the optimal performance. When the ratio is at its minimum, the optimal classifier is found.

The input data for weak learners is one-dimensional examples $x_1, \ldots, x_n$, and l is the number of positive examples, and m is the number of negative examples. Each example is labelled with $y = 1$ or $y = 0$ as positive or negative respectively. The purpose is to find a good predictor for the class y of any example. The predictor is defined as

$$y = \mathbf{w}x + b \tag{4.1}$$

w is the weight coefficient of the predictor, and b is the bias of the predictor. The problem is how to find **w** and b. The mean of positive examples and negative examples are

$$\mu_{y=1} = \frac{1}{l} \sum_{y_i=1} x_i \tag{4.2}$$

$$\mu_{y=0} = \frac{1}{m} \sum_{y_i=0} x_i \tag{4.3}$$

so that the variance between classes is $S_b = \mu_{y=1} - \mu_{y=0}$, and the variance within the positive class is

$$S_{w_{y=1}} = \sum_{y_i=1} (x_i - \mu_{y=1})^2 \tag{4.4}$$

and the variance within the negative class is

$$S_{w_{y=0}} = \sum_{y_i=0} (x_i - \mu_{y=0})^2 \tag{4.5}$$

The overall variance within classes is $S_w = S_{w_{y=1}} + S_{w_{y=0}}$. The solution for the coefficient **w** optimising the ratio is

$$\mathbf{w} = S_w^{-1}(\mu_{y=1} - \mu_{y=0}) \tag{4.6}$$

The means of projected points are

$$\widehat{\mu}_{y=1} = \mathbf{w} \cdot \mu_{y=1} \tag{4.7}$$

$$\widehat{\mu}_{y=0} = \mathbf{w} \cdot \mu_{y=0} \tag{4.8}$$

The bias b is calculated as

$$b = -\frac{\widehat{\mu}_{y=1} + \widehat{\mu}_{y=0}}{2} \tag{4.9}$$

Given a set of examples $x_1, \ldots, x_n$ with corresponding label $y = 0, 1$, the training of the FLD weak learner is to find **w** and b to form a predictor. From the training example, it is easy to compute the means and variances.

4.4.3 Reduction of Computational Time

In AdaBoost, the error of the final strong classifier is bounded by Equation 3.19 in Chapter 3. Thus, there is one hypothesis for AdaBoost.

> *If weak learners that are slightly better than random guessing can be consistently found, then the error of the final strong classifier drops exponentially fast.* [47]

If a weak learner is equivalent to random guessing, its classification error will be 0.5. If a weak learner has its error rate more than 0.5, this indicates that given a number of labelled examples, more than half the examples are misclassified. Therefore 0.5 is the limit bounding the error of a weak learner. To have high accuracy in the final classifier, the error of the weak learner should be less than 0.5. Because in the original AdaBoost (in Table 3.1), the weak learner generated from iterations are combined into an ensemble. Any weak learner with low discrimination power will distort the overall performance of the ensemble. Only a weak learner with performance better than random guessing is allowed to be combined into the final classifier. Although, for each weak learner, the error is expected to be as small as possible, with a simple structure and fast running speed, the weak learners can not be guaranteed to achieve very high classification accuracy.

In AdaBoost based feature selection, low recognition accuracy in a weak learner also indicates that the corresponding Gabor wavelet feature is irrelevant or redundant for face verification. When a weak learner has the error equal or greater than 0.5, AdaBoost training on the weak learner will stop, because the performance of the classifier can not be boosted.

For example, in the experiment which will be described in Section 4.5.2, there are $29,120$ features trained in the first iteration. The error distribution of the corresponding $29,120$ weak learners are illustrated in Figure 4.6, in which the horizontal axis shows the error and the vertical axis displays the number of weak learners. It can be seen clearly from Figure 4.6, the distribution of the errors from all weak learners looks like a normal distribution with the mean error at 0.35. Most errors of the weak learners are in the range 0.2 to 0.5. There are some weak learners whose errors are equal or greater than 0.5, *i.e.*, error of random guessing. All weak learners having performance worse than random guessing will stop in training at the next iteration. The corresponding feature will be discarded at the next iteration to build a weak learner.

The AdaBoost algorithm for feature selection is modified in Table 4.2. In this improved algorithm, a feature set J is defined, which contains all features $\{j_1, \ldots, j_k\}$ at the beginning of AdaBoost training. In each iteration, only one significant feature is selected, and some features are removed from the set J. In

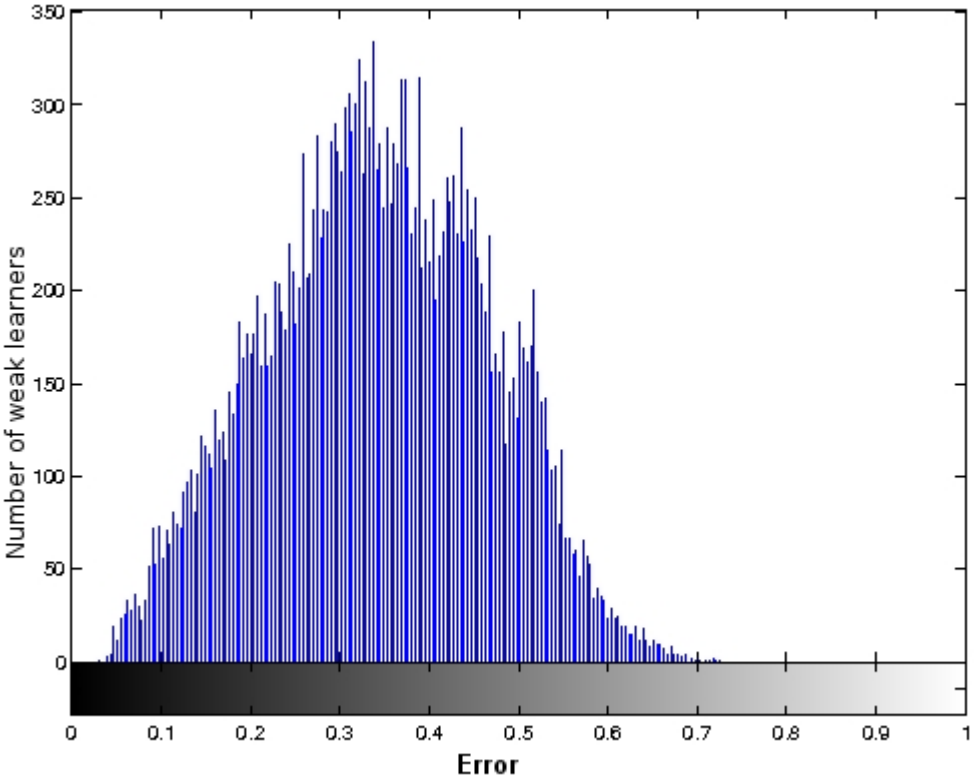


Figure 4.6: The distribution of the error of all weak learners

each iteration, the exhaustive search is on the set J rather than on all features $\{j_1, \ldots, j_k\}$.

4.5 Experiments

The eXtended Multi Modal Verification for Teleservices and Security (XM2VTS) face database is used in the experiments. A small group with 200 face images and a large group with 800 face images of experiments are performed on feature selection. Based on the selected features from the large group of experiments, an SVM classifier is trained for face verification.

Table 4.2: The improved algorithm of AdaBoost for feature selection

Given example $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the data of the ith example, which are contributed by a set J including k features $\{j_1, \ldots, j_k\}$, and $y_i \in Y = \{0, 1\}$ for impostor and client respectively..
Initialise the weights $\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of impostors and clients respectively.
for $t = 1, \ldots, T$ **do**
    Normalise the weights, $\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{i=1}^{n} \omega_{t,i}}$ so that ω_t form a probability distribution.
    for all j in the set J **do**
        Train a weak learner h_j which is restricted to use a single feature j.
        The error is calculated as $\varepsilon_j = \sum_{i=1}^{n} \omega_{t,i} |h_j(x_i) - y_i|^2$ with respect to $\omega_{t,i}$
        if $\varepsilon \geq 0.5$ **then**
            Remove the feature j from the set J.
        end if
    end for
    Choose the optimal weak learner h_t with the lowest error ε_t from all h_j.
    Select the corresponding feature j_t of the weak learner h_t as a significant feature.
    Select the feature j_t out of the set J.
    Update the weights $\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i}$, where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
end for

4.5.1 XM2VTS Face Database

The developed feature selection algorithm was tested in XM2VTS [111]. The XM2VTS database is a multi-modal face database which is captured onto high quality digital video. The XM2VTS includes colour images, sound files, video sequences and 3D Models. The XM2VTS contains four sessions of 295 subjects recorded over a period of four months at approximately one month intervals.

In XM2VTS, there are 295 subjects with 200 clients and 95 impostors. Since only the frontal face images are concerned, for each subject 8 up-right and frontal view images have been used. The database was divided into three sets: training set, evaluation set, and testing set. The training set is used to build face models.

The evaluation set is for selection of a threshold that determines if a person is accepted or rejected. The testing set is to test the performance of the algorithm. In face verification, a person is accepted if he is a legal client; a person is rejected if he is an impostor. The features are extracted and selected in face images from the training set which includes 200 clients with the first 4 images of each client. In the evaluation set, two face images across 200 clients and 8 images across 20 impostors are collected. In the testing set, two images across 200 clients and 8 images from 75 new impostors are used. The images are selected to take account of moderating differences in illumination, expressions and facial details.

Each subject has 8 frontal images. The images are stored in colour PPM format [166] and at resolution 720×576. To perform feature selection and face verification, the images need to be adjusted and segmented, because these large images not only contain inner face area but also the background, the hair, the necks, and so on. The redundant areas such as background, hair, etc, are irrelevant for the face verification application. The segmentation to obtain only the inner face area is performed. Due to the appearance based approach adopted, for segmenting a face image, some reference points are needed to correspond with all face images. In this thesis, the two pupils on the face are the reference points to segment inner face images. The following steps are taken in the processing.

1. If two pupils on a face image are not on the same horizontal line, the image will be rotated to make the two pupils horizontal.

2. The distance between the pupils is measured. The face image is scaled so that the distance between the pupils is 26 pixels.

3. The image is segmented by using a window 55 pixels high and 51 pixels wide. The left pupil is matched with position $(13, 19)$, and the right pupil is matched with position $(39, 19)$ in the window.

4. The segmented image is converted into gray-scale.

Therefore, all images are properly rotated, scaled, segmented and converted to fit a grid size of 55×51 as illustrated in Fig 4.7.

4.5.2 Feature Selection Results

The experiment includes feature extraction and feature selection. The images used are all those in the training set. In the training set, there are 4 images per client

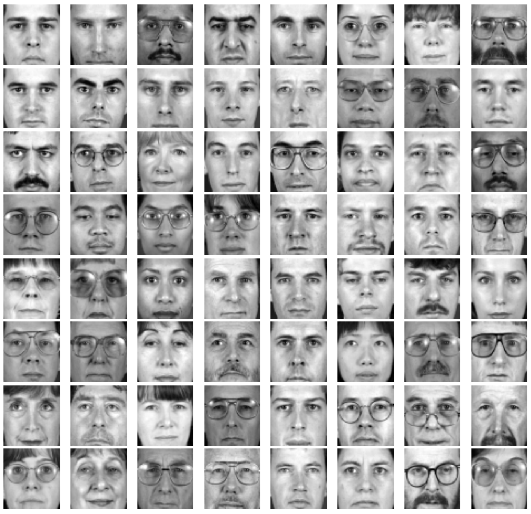


Figure 4.7: The XM2VTS images after rotation, segmentation and scaling

across 200 clients. 40 Gabor wavelets are generated according to Equation (3.1) with $\nu \in \{0, \ldots, 4\}$, and $\mu \in \{0, \ldots, 7\}$. Each image is convolved with these 40 wavelets. The outputs are 40 magnitude responses for each image. Each point in a magnitude response image corresponds to a feature. The features $O_{\mu,\nu}(z)$ vary with three parameters: orientation μ, scale ν, and position z. For each client, there are $8 \times 5 \times 55 \times 51 = 112,200$ features generated as input for feature selection in each iteration of AdaBoost. The edge of image is discarded due to edge effects caused by convolution. The second method is to apply the **Interlace** scheme to reduce the number of features. The actual size of each image is reduced to 24×25. Consequently the number of features for each image is reduced to $8 \times 5 \times 24 \times 25 = 24,000$ instead of $112,200$ features.

For each image, there are $24,000$ features generated by Gabor wavelet transform. Two groups of experiments are carried out.

- In the first group, a small training set is used. The training set only contains the first 50 subjects of the XM2VTS with 4 images per subject. For each client, 4 images are taken as positive examples, and the other 196 images are taken as negative examples.

- In the second group, a large training set is used, which includes all 200 clients in the XM2VTS database. For each client, 4 images are taken as positive

examples, and the other 796 images are taken as negative examples.

This approach gives a large number of training data which are required by the AdaBoost algorithm. However it leaves the ratio between positive and negative examples unbalanced, such as 4/196 and 4/796.

Dealing with more than one minimum errors

The feature is selected with the lowest error of the corresponding weak learners. However, in some situations (especially with a small training set) there are more than one weak learners sharing the same lowest error. This presents the problem on how to select features and how to update weights. In this thesis, if the situation occurs, in the current iteration, all features which share the same lowest error are selected as the significant features. The weights are updated according to each selected feature associated with a weak learner separately, and the updated weights are the average weight for all features with the same minimum error. To avoid the problem of multiple lowest errors, the solution is to adopt a larger training set.

Result on the small training set

For the small training set, the top 20 features are shown in Figure 4.8. The weak

(a) The 1st client (b) The 4th client

Figure 4.8: The Gabor wavelet features selected in the first group experiment.

learners associated with these features have the same classification errors, which are equal to zero during AdaBoost feature selection. For the first client, the top 20 features are mostly located on the person's right temple and beside the left eyebrow. Figure 4.8(a) shows these features on the first client face. The locations of the first 20 features (marked with red crosses) selected for the first client by

AdaBoost are shown. These features indicate that the left eyebrow and right-upper temple are significant features for the first client which discriminate that client from others. Most features distributed into two local regions are highly correlated and redundant, as explained in [121]. In the fourth client, the top 20 features are located on the person's cheek near the nostril which appears as a deep valley and a nevus on his left face as shown in Figure 4.8(b). These are important signs for the fourth client which discriminate that client from others. These features are also highly correlated.

Result on the large training set

For the large training set, all 800 images from the 200 clients were used. The AdaBoost algorithm selected the top 20 features from the first client to the 8th client as shown in Figure 4.9. Some features overlap because they share the same

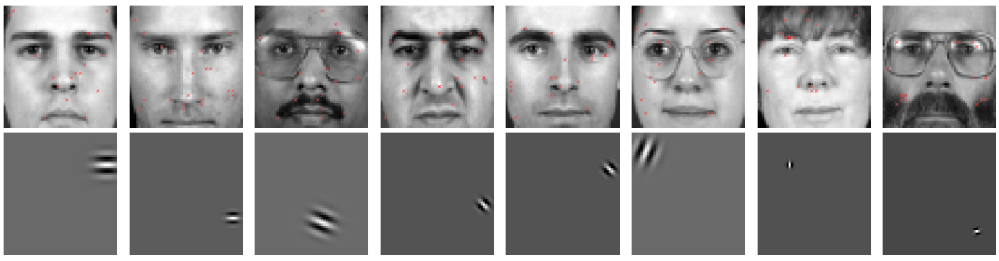


Figure 4.9: The Gabor wavelet features selected after AdaBoost training in the second group experiment

position, but with different orientation or scale. The first feature selected for each client is shown in the bottom row of Figure 4.9. The results shows that the selected features are randomly distributed in the face area rather than concentrated on particular regions of the faces as are the results from the small training set. This indicates that these features are less correlated and less redundant than the features selected in the small training set. Most features are located on eyes, eyebrows, nose, lips, glasses frames or beard.

Compared to the high correlation among the features in the small training set, the features selected from the large training set can be treated as representatives of local regions. The experiments in the large example size separate the features sparsely, and lead to decorrelation and redundancy reduction. From Figure 4.9, the

top 20 features for different individuals are located in different positions. Although most features correspond to the common key characteristics on a face, *e.g.*, eyes, eyebrows, nostril, lips and so on, the top selected features for each individual are varied. As concluded in [121], different subjects can be classified depending on what objects they contain, the faces can be classified depending on the unique characteristics in which an individual is different from others, *e.g.*, the nevus on the left face of the fourth client is a unique characteristics for this person. There are some common features selected in both the large and small training sets as shown in Figure 4.10. In Figure 4.10(a), the first selected feature in the large training set is in the eyebrow region, where this is also selected in the small training set. For the fourth client in Figure 4.10(b), the first selected feature corresponds to the nevus on the left face in the small training set. The selected features for the first

(a) The 1st client (b) The 4th client

Figure 4.10: Common features selected in both the large and small training set

client and the fourth client in the small training set are shown in Table 4.3 and Table 4.4, respectively. For the first client, most features are distributed into two clusters: one consists of 8 features with the scale and the orientation fitted to 2 and 4 separately.

Time reduce

Both algorithms in Table 4.1 and Table 4.2 have been implemented. The algorithm that excludes the big error weak learners displays a significant computational improvement over the original one. The comparison of the two algorithms on time cost is shown in Figure 4.11. The horizontal coordinate indicates which iteration the AdaBoost training is on, and the vertical coordinate shows the number of features trained in the corresponding iteration. The colour lilac shows the number of

Table 4.3: The top 20 features selected by the AdaBoost for the first client in XM2VTS.

Scale ν	Orientation μ	Position (x, y)	Scale ν	Orientation μ	Position (x, y)
1	7	(5,3)	2	4	(49,15)
1	4	(45,13)	2	4	(49,11)
1	4	(47,13)	2	4	(47,15)
1	6	(5,3)	2	4	(45,15)
2	4	(45,11)	0	7	(5,3)
2	4	(45,13)	0	6	(5,3)
2	4	(47,13)	0	6	(3,5)
2	4	(47,11)	1	0	(5,5)
2	4	(49,13)	0	7	(3,3)
1	0	(5,3)	1	6	(39,13)

Table 4.4: The top 20 features selected by the AdaBoost for the fourth client in XM2VTS.

Scale ν	Orientation μ	Position (x, y)	Scale ν	Orientation μ	Position (x, y)
0	0	(34,35)	1	0	(35,36)
0	0	(35,35)	1	0	(36,36)
0	0	(36,35)	1	6	(46,33)
0	7	(36,36)	1	6	(46,34)
1	0	(34,34)	1	6	(47,34)
1	0	(35,34)	1	7	(45,32)
1	0	(34,35)	1	7	(34,35)
1	0	(35,35)	1	7	(35,35)
1	0	(36,35)	1	7	(36,35)
1	0	(34,36)	1	7	(36,36)

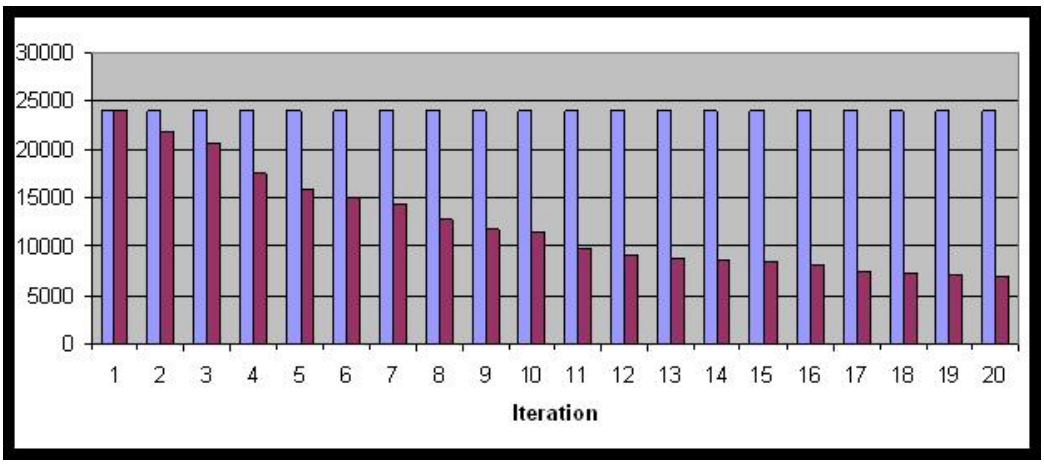


Figure 4.11: The comparison of the improved algorithm and the original algorithm on computational time

features without using the computational time reduction technique, and the colour purple shows the number of features using the technique. In the first iteration, there are 24,000 Gabor wavelet features ready for AdaBoost training. After the first iteration, 2,177 weak learners have an error greater than 0.5, and the corresponding features are excluded from the set J, such that there are 21,822 features remain for the second iteration. In each iteration, some features are removed. In the 20th iteration, only 6,194 features remain. In the total 20 iterations, computational time is reduced to 54.23% of the original time. If there are more than 20 iterations in the training, the computation time will be reduced to much less than 54.23% of the original time. Therefore, the approach which excludes some big error weak learners can reduce the computational cost significantly.

4.5.3 Classification Results

With the 20 Gabor wavelet features selected for each client, a face image is represented by a vector of 20 features. Verification is performed on the training set and the testing set by using a Support Vector Machine (SVM) [119] on each client (subject).

SVM

SVMs are a collection of learning approaches used for generalised linear classification. A special property of SVMs is that they minimise the classification error and maximise the geometric margin, so that SVMs are also known as maximum margin classifiers. For two-class classification, it is assumed that there is a hyperplane (also called a *decision boundary*) separating the clusters of two classified data. SVMs project input vectors to a higher dimensional space where a maximal separating hyperplane is created. The separating hyperplane maximises the distance between the two classes. An assumption is made that the larger the margin or distance between two classes, the lower the generalisation error of the classifier will be. SVMs are used to maximise the margin between classes and minimise a quantity proportional to the number of mis-classification errors. The original optimal hyperplane algorithm is a linear classifier. However, non-linear SVMs can be created by applying the kernel method [154] to maximum-margin hyperplanes. The algorithm is allowed to fit the maximum-margin hyperplane in the modified feature space. The classifier is a hyperplane in the high-dimensional feature space, which may be non-linear in the original input space. The kernels can be adopted as a polynomial kernel, a radial based kernel, or a Gaussian radial basis kernel. In this thesis, a Matlab toolbox named OSU Support Vector Machines (SVMs) Toolbox [104] is used to perform the task of classification.

Results

Experiments were carried out on eight clients from the XM2VTS. A non-linear SVM classifier with a polynomial kernel of degree 3 was constructed from the 800 training examples, which are the first four images across all clients. The testing set is composed from the other four images across all clients and the eight images across all impostors in XM2VTS. The test results of 8 clients with $1,560$ testing examples (among them, 4 positive examples and $1,556$ negative examples from 199 clients with 4 images and 95 impostors with 8 images per subject) are shown in Table 4.5 with the false negative rate $\mathcal{FN}$ and the false positive rate $\mathcal{FP}$. By adjusting the bias of each SVM classifier for each client, the false negative rate $\mathcal{FN}$ is set to 0%, 25%, 50%, 75% or 100%, and the corresponding classification error (Error rate) and false positive rates are shown in Table 4.6. The optimal boundary for a SVM classifier gives a low error rate but a high false negative

Table 4.5: The classification results from client 1 to client 8 (C1 to C8) with 20 features in XM2VTS.

	Training Set (%)		Testing Set (%)	
Client	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	0.0	0.0	0.0
2	0.0	0.0	50.0	0.0
3	0.0	0.0	50.0	0.0
4	0.0	0.0	75.0	0.0
5	0.0	0.0	50.0	0.0
6	0.0	0.0	50.0	0.0
7	0.0	0.0	100.0	0.38
8	0.0	0.0	100.0	0.0

rate because the ratio between the positive examples and the negative examples remains unbalanced [1]. Therefore, the decision surface gets close towards the actual boundary of the negative class, but far away from the place where the positive class resides in the feature space. Moreover, SVMs are only concerned with the trade off between margin and mis-classification error, but not with the false negative rate. This leads to the trained SVM classifiers having strong ability to recognise the negative examples, but relatively weak ability to recognise the positive examples. It also makes the false positive rate much closer to the classification error rate. By adjusting the bias of the SVM classifier, the false rate can be reduced, while the classification error rate is increased, or *vice versa*, *e.g.* for client 4, the optimal boundary of the SVM classifier gives the false negative rate equal to 25%, and the error rate is 0.26%. However, when the bias of the classifier is adjusted, no false negative is detected, and the error rate is increased to 2.95%. Table 4.6 shows that the classification of clients 1, 4 and 5 has better performance than for other clients. It indicates that the 20 selected features contribute well enough for some clients to verification, but they are not sufficient to verify some other clients *e.g.* clients 6 and 7. In these cases, it may need more features selected by AdaBoost for client representation.

[1] It is also called *selection-bias* or *example-bias*. Section 5.4 gives the detail.

Table 4.6: The classification results from client 1 to client 8 (C1 to C8) in XM2VTS with shifting bias

False(%) Negative	Error rate(%)/False Positive rate(%)			
	C1	C2	C3	C4
0	4.94/4.95	13.72/13.75	8.53/8.55	2.95/2.96
25	1.73/1.67	12.12/12.08	11.10/8.29	0.26/0.20
50	0.83/0.71	8.72/8.61	9.58/7.13	0.32/0.19
75	0.38/0.19	1.41/1.22	8.30/6.10	0.32/0.13
100	0.26/0.0	0.58/0.32	2.12/1.35	0.26/0.0
	C5	C6	C7	C8
0	2.31/2.31	81.92/82.13	53.08/53.21	15.64/15.68
25	0.45/0.39	14.81/14.78	46.47/46.53	8.59/8.55
50	0.38/0.26	1.86/1.74	44.49/44.47	0.51/0.39
75	0.38/0.19	0.51/0.32	6.47/36.38	0.38/0.19
100	0.32/0.06	0.26/0.0	0.64/0.39	0.32/0.06

4.5.4 Discussion on SVMs in Face Verification

The ratio between the number of positive and negative examples is unbalanced. Whether in the small or large training set, the number of positive examples is much less than the number of negative examples. Hence, in the feature space, the subspace which all positive examples dominate is much smaller than the subspace which all negative examples dominate. The negative set contains sufficient data which could describe the generality of all negative examples, however the positive set only has 4 examples, which contains insufficient data to describe itself.

Each example is hypothesised as a point in a two-dimensional feature space as in Figure 4.12. The blue stars and blue circles represent the positive training examples and testing examples, respectively. The red star and red circles represent the negative training and testing examples. The blue solid line denotes the actual boundary for the positive examples, while the blue dash line denotes the predicted boundary for the positive examples. The predicted boundary means that the boundary is predicted by the SVM classifier, rather than the actual boundary of the examples. For the negative examples, the red solid line and dash line indicate the actual boundary and predicted boundary respectively. Most negative training examples and testing examples are concentrated in the right part of Figure

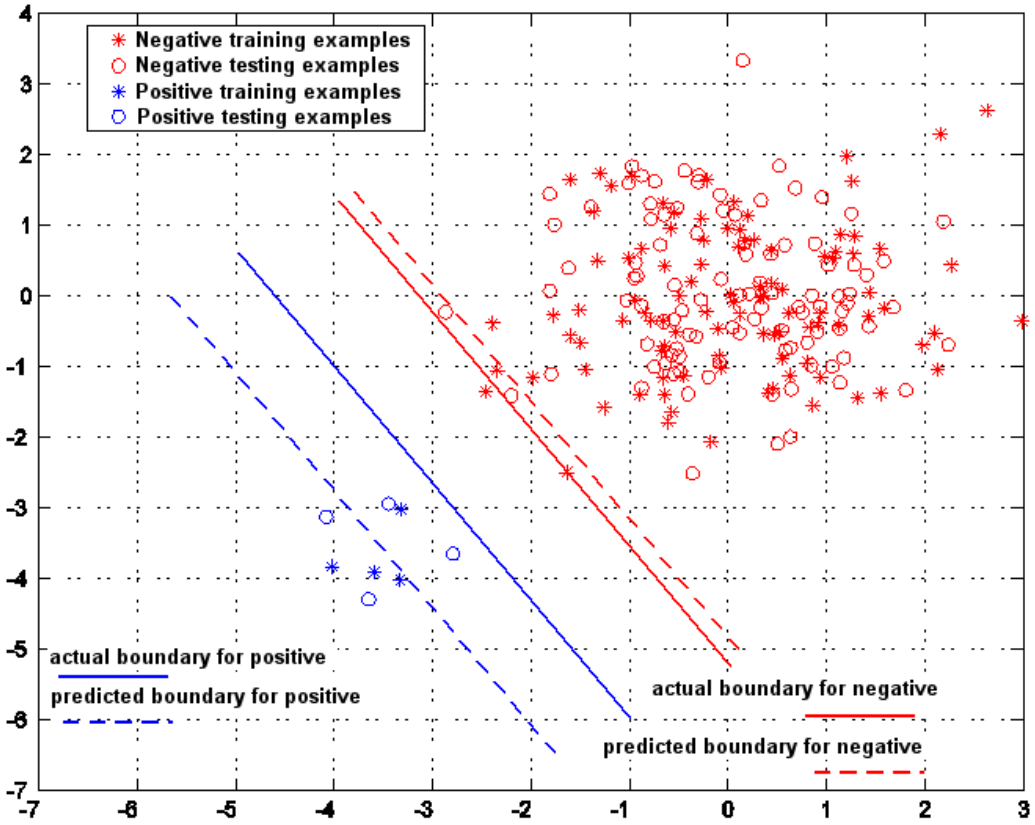


Figure 4.12: The hypothesis on the SVM classification results

4.12, while positive examples amass in the left part. The actual boundary for the negative is close to the predicted boundary, which indicates the SVM classifier has achieved a good performance on recognising the negative examples. However, the predicted boundary for the positive is far from the actual boundary, which indicates the bad performance on positive examples. The reason is, the positive training examples do not overlap well with the positive testing examples. Some positive examples are recognised as non-positive, because they are out of the predicted boundary. To avoid imbalance on positive and negative, an increase in the number of new positive examples could be useful.

The size of training data From the perspective of machine learning, to train "ability" on a machine, a data set must be given to the machine to extract information and to learn the relevant decision rules. To build a data set for training (or

learning), there is a trade-off between **representatives** of the training set, **conciseness** of the training set and **prior probability** of classes. The representatives of data indicate how much the data covers all the possible variants. The more data in the data set, the higher the representatives are. The conciseness of the data refers to the size of the data set. If the size is big, the conciseness of the data set is poor. As long as the examples are sufficiently representative, the number of training examples should be as small as possible. The conciseness of the data is related to computational cost. More data normally leads to more computation time. In supervised learning, the number of examples in a certain class should be in accordance with the prior probability of the class. For example, to recognise a face as belonging to male or female, a training set is built with the number of male face images equal to the number of female face images, because it is assumed that the prior probability of male and female are 0.5. In face detection, there are two classes to detect, face and non-face. However, the prior probability of face is far less than the prior probability of non-face, so that the number of face images should be far less than the number of non-face images. Due to the representatives of the data set, the number of face images should not be too few, but must contain enough face images to describe all variants of the face class. Therefore, to build an appropriate training data set, these three factors, **representatives**, **conciseness** and **prior probability**, should be taken into account, and these three factors should be well balanced.

4.6 Summary

This chapter presents a FLD based Gabor-Boosting feature selection for face verification. The face images in face verification are defined as either clients or impostors, and face verification is treated as a two-class classification task for each client (subject). After Gabor wavelet transform, faces reside in a very high-dimensional feature space. Feature pre-selection is performed to reduce the feature space to a relatively low-dimensional feature space, and then AdaBoost training is further applied for the purpose. In the AdaBoost training, FLD is used as weak learners, and the strategy is applied for reducing training time. The experiments show the selected features for some clients, and classification performance is tested in the XM2VTS face database. The SVM classifiers are briefly introduced with consideration of balance of examples size. Although the FLD based weak learner displays

an appropriate contribution to face verification, the weights for each example are not applied in the training of each FLD weak learner. The next chapter is dedicated to solving this problem.

Chapter 5

Potsu based Gabor-Boosting

This Chapter presents a new weak learner named Potsu weak learner in the AdaBoost training for feature selection. Both the XM2VTS and the FERET face databases are tested with the Potsu weak learner based Gabor-Boosting algorithm. Section 5.1 introduces the Potsu weak learner. In Section 5.2, the feature selection of Gabor-Boosting algorithm with the Potsu weak learners is proposed, and optimisation approaches are applied to reduce the computational cost. In Section 5.3, the performance made by the AdaBoost classifier is presented. In the results of classification, the overfitting phenomenon is observed. Section 5.4 explores the reasons that cause overfitting. In Section 5.5, two methods are applied on classification to address the overfitting problem. In Section 5.6, the Potsu weak learner is compared with the FLD weak learner with respect to the performance of classification. In Section 5.7, the FERET face database is tested with AdaBoost feature selection of Potsu based binary Gabor-Boosting algorithm and SVM classification.

5.1 POTSU Weak Learner

A novel weak learner for AdaBoost training is presented in this section. Firstly, the construction of an ensemble classifier in AdaBoost is explained. Secondly, the requirement of the training on weak learners in AdaBoost is discussed. Thirdly, the novel weak learner - Potsu weak learner is proposed. The Potsu weak learner benefits from the perceptron prototype which is rapid on training and Otsu's thresholding algorithm which is accurate when the number of examples is large.

5.1.1 Construction of an ensemble classifier

As stated in Section 3.2 in AdaBoost training, corresponding weak learners h_t are collected and built into an ensemble classifier in association with a group of selected significant features. In the original AdaBoost, the ensemble classifier is

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where, α_t is the importance factor of the weak learner h_t, which indicates how importantly the weak learner h_t contributes to an ensemble classifier, and T is the number of iterations. The importance of a weak learner is evaluated by

$$\alpha_t = \log \frac{1}{\beta_t} \tag{5.1}$$

The parameter β_t is defined as $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$, where the parameter ε_t is the error of each iteration. The importance is also expressed as

$$\alpha_t = \log \frac{1-\varepsilon_t}{\varepsilon_t}$$

Because the error ε_t of the weak learner from each iteration should be less than $1/2$, the importance α_t will be greater than zero. The smaller the error ε_t is, the larger is the importance α_t. This indicates that weak learners which have slightly "stronger" discriminating power, play more important roles in the ensemble classifier to make a decision.

5.1.2 Requirements of weak learners

As discussed in Section 3.2.4, there are two ways to process weak learners. In the first way, the Naive Bayes weak learner is adopted for training and the training data is re-sampled. The size of the new training set is increased from 25 to 42 times the original size after re-sampling. This requires more computational time. Since AdaBoost training has already inevitably resulted in higher computational cost, increasing the training set size will increase the difficulty on AdaBoost training. Hence the re-sampling approach is not appropriate in the practice of AdaBoost. A better way to implement weak learners is to use the weights directly on the examples, which is the Artificial Neural Network (ANN) applied in Section 3.2.4.

The training of weak learners is to achieved the minimum error ε upon the training data. The error is evaluated by

$$\varepsilon = \sum_{i=1}^{n} \omega_{t,i} |h(x_i) - y_i|^2 \tag{5.2}$$

In machine learning, training a weak learner is equivalent to finding a criterion which can predict the labels of the training examples correctly. Different criteria could induce different results on labelling the examples, *i.e.*, the number of correct predictions. Normally, the learners are required to achieve as many correct predictions as possible. However, the aim of training a weak learner is to find an optimal criterion with a desirable requirement, which is not the maximum prediction, but is a weighted prediction.

Different weak learners, *i.e.*, classifiers, are needed to follow different requirements. Linear Discriminant Analysis (LDA) classifiers require maximising between-class variance and minimising within-class variance. Naive Bayes classifiers use the method of maximum likelihood or maximise a posterior probability. Support Vector Machine (SVM) classifiers are required to maximise the margin between classes and minimise a quantity proportional to the number of mis-classification errors. k-nearest neighbour (k-NN) classifiers are based on the nearest training examples in the feature space. ANN classifiers are required to minimise the ratio between misclassified examples and total examples. For instance, if an ANN classifier is chosen as a weak learner, the error is evaluated by $\frac{1}{n}\sum_{i=1}^{n} |h(x_i) - y_i|^2$ instead of Equation 5.2 without taking weights into account. Therefore the requirement of the ANNs is different from the requirement of weak learners in AdaBoost. In this thesis, weak learners are required to minimise the error ε which is a weighted ratio between misclassified examples and total examples.

It is assumed that training examples for a weak learner reside in a 2-D feature space[1], where there are many lines across data clusters as shown in Figure 5.1. There are two classes: positive examples represented by blue dots and negative examples represented by red dots. Many lines are across the positive data and the negative data, and each line represents a different criterion to separate the positive and the negative examples. Among these criteria, there must be an optimal one which minimises the error ε. If an ANN classifier is adopted as a weak learner, the

[1]Actually, the training examples are in a 1D feature space used for face recognition in this thesis.

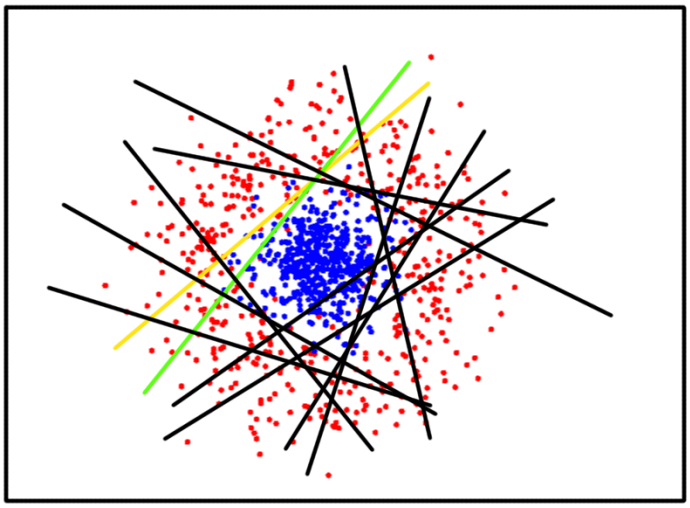


Figure 5.1: Example: a 2-D feature space explains the optimal criterion of weak learner in AdaBoost.

optimal criterion is chosen as the yellow line in Figure 5.1. That is because the optimal criterion in the ANN classifier minimises the ratio between the number of misclassified examples and the number of total examples. Although the yellow line is the best one among other lines for separating the examples, the criterion does not achieve the minimal ε. The green line in Figure 5.1 is the optimal criterion actually for the weak learner to minimise ε, although there are more examples misclassified by the green line. When those examples are misclassified, the accumulated error ε does not increase significantly since their weights are low. Some examples are misclassified by the yellow line and correctly classified by the green line, since they have higher weights. The accumulated error ε will increase significantly.

The requirement for weak learner in AdaBoost training is that the training criterion must be satisfied with minimal ε. The classifiers mentioned above, such as LDA, Naive Bayes, SVM, k-NN, and ANN, do not satisfy with the requirement. None of these classifiers can be directly introduced as the weak learner in the training, so a new weak learner needs to be developed.

5.1.3 Potsu weak learners

A new weak learner for AdaBoost training is proposed in this section. The new weak learner combines the concept of perceptron with Otsu's algorithm, so that

the proposed weak learner is named **Potsu** weak learner. The Potsu weak learners use an example based heuristic approach to threshold the training data.

Perceptron

To train a weak learner rapidly, it is necessary to keep the design of weak learner as simple as possible. Among various prototypes of weak learner, a design of *perceptron* [52] is chosen, as the perceptron is the simplest type of linear classifier.

The perceptron is a type of two-class classifier that maps its input to an output value. A perceptron based weak learner consists of observation f_j on a feature j, a threshold θ_j, and a parity p_j as below

$$h_j(x) = \begin{cases} 1 & \text{if} \quad p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

The observation f_j on a feature j are numerical values, and the perceptron is dealing with data in a one-dimensional space. The parity p_j indicates the direction of the inequality sign, *e.g.*, positive examples may reside above the threshold θ_j in some cases, or in other cases reside below the threshold θ_j. There are only two options for the parity p_j - either 1 or -1, so that finding an appropriate parity is not a difficult task. After the parity is determined, the task in learning a perceptron is to find the optimal threshold θ_j. A novel algorithm is proposed to find the optimal threshold similar to *Otsu's Thresholding* method.

Otsu's algorithm

The Otsu thresholding method [120] has been widely used in image thresholding in a wide range of applications, such as noise reduction for human behaviour recognition, medical image processing, real-time segmentation of images, document segmentation and so on. Sahoo *et al.* [136]'s study concluded that the Otsu thresholding method was one of the best threshold selection methods for general real world images with respect to uniformity and shape measures. The Otsu thresholding method is based on a very simple idea: finding the threshold that maximises the between-class variance. The Otsu method is optimal for thresholding large objects from the background. It indicates that the Otsu method is fine for thresholding a histogram with bimodal or multi-media distribution. In gray scale image thresholding, the task is to separate the foreground and background. The

foreground may be objects needing to be tracked or processed, and the background is anything other than the specified objects. For example, in Optical Character Recognition (OCR), binary operation is required for document segmentation. The thresholding method is to separate some pixels belonging to characters and other pixels belonging to white-space. In gray scale images, pixel values range from 0 to 255 in 256 integers. Each integer is taken as a possible threshold to segment the foreground and the background. For each possible threshold, the between-class variance is computed. From numerous possible thresholds, the optimal threshold is the one which has the maximum variance. The detailed algorithm for the Otsu thresholding method can be found in [93].

Thresholding algorithm

In the Potsu weak learner, the method is to select a threshold which minimises the error ε_j of a weak learner on the given training examples. Similar to the Otsu method, some possible thresholds are collected beforehand. These possible thresholds are called *threshold candidates*. If there are n non repetitive features, the number of threshold candidates will be $n - 1$. The error ε_j is calculated according to each threshold candidate, finally an optimal threshold is found with the minimum error among the candidates. The algorithm for training a Potsu weak learner is given in Table 5.1.

Parity The parity of Potsu weak learner is determined by comparing the means of all positive examples between all negative examples. If the mean of positive examples is less than the mean of all negative examples, feature values of all positive examples will be below the threshold. It is assumed that the positive examples and the negative examples are separated and normally distributed. The parity is approximately predicted in the training of Potsu weak learners. Since weak learners only have "weak" discriminating power, the parity is not required to be highly accurate. In brief, it is not necessary to implant a "strong" discriminating power in weak learners.

Threshold The training of a Potsu weak learner is to label examples. Once the parity is determined, *e.g.*, $p = 1$, the value $f_j(x)$ of the feature j below the threshold θ are classified as positive, otherwise classified as negative. The threshold θ should be between the minimum and the maximum of $f_j(x)$. It is in the range

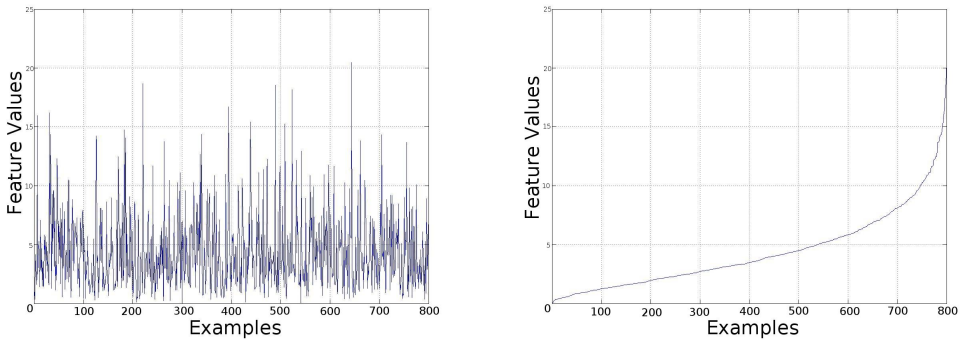
Table 5.1: The algorithm for training Potsu weak learners

1: Given a feature j and example $(x_1, y_1), \ldots, (x_n, y_n)$ and $y_i \in Y = \{0, 1\}$ for negative and positive respectively, $f_j(x)$ is the value of the feature j on the example data x.
2: Construct a Potsu weak learner according to Equation 5.3.
3: Compute the mean feature value $\bar{\mu}_1$ on all positive examples and the mean feature value$\bar{\mu}_0$ on all negative examples
4: **if** $\bar{\mu}_1 < \bar{\mu}_0$ **then**
5: the parity $p = 1$
6: **else**
7: the parity $p = -1$
8: **end if**
9: Sort all $f_j(x_i) \quad i \in \{1, \ldots, n\}$ into $f'_j(x_i), \ldots, f'_j(x_n)$
10: **for** Every two adjacent sorted examples $f'_j(x_i)$ and $f'_j(x_{i+1})$ **do**
11: Compute the ith threshold candidates $\theta_i = \frac{f'_j(x_i)+f'_j(x_{i+1})}{2}$.
12: **end for**
13: There are $n - 1$ threshold candidates $\theta_1, \ldots, \theta_{n-1}$.
14: **for all** Threshold candidates $\theta_1, \ldots, \theta_{n-1}$ **do**
15: Suppose θ_i as the threshold θ_j in Equation 5.3.
16: Calculate the error $\varepsilon = \sum_{i=1}^{n} \omega_i |h_j(x_i) - y_i|^2$ with respect to ω_i.
17: **if** $\varepsilon > 0.5$ **then**
18: reject the threshold candidate θ_i.
19: **end if**
20: **end for**
21: Select the corresponding threshold candidate θ with the lowest error ε over all candidates θ.
22: A trained Potsu weak learner is $h_j(x) = \begin{cases} 1 & \text{if} \quad pf_j(x) < p_j\theta \\ 0 & \text{otherwise} \end{cases}$

of

$$\text{Min}\{f_j(x)\} < \theta < \text{Max}\{f_j(x)\} \tag{5.4}$$

The threshold θ should be a quantitative value within this range. For instance, there are some examples below a threshold θ' and some examples above. If the threshold θ' has been shifted higher, more examples will be recognised as positive, and vice versa. If the threshold θ' is close to the optimal threshold θ, the error ε' with respect to θ' is also close to the optimal error ε with respect to θ. Since weak learners do not require the "strong" discriminating power, it is necessary to collect a set of possible thresholds, *i.e.*, *threshold candidates* under the framework of Potsu to find a threshold. The performance of weak learners is evaluated on each candidate. Finally, the candidate with the best performance will be selected as the optimal threshold. The number of candidates is required to be large, otherwise the candidates could not cover the whole range of features' value. On the other hand, the number should not be too large to compute quickly. The feature values on 800 examples in XM2VTS database are shown in Figure 5.2(a). They are sorted, to be put in order as shown in Figure 5.2(b). It is hypothesised that each candidate



(a) (b)

Figure 5.2: Feature values on 800 examples: original data and sorted data.

θ_j exists between two adjacent values. The candidate θ_j is approximated to the middle of these two adjacent values, *i.e.* the mean of the two values. There are $n-1$ candidates $\theta_1, \ldots, \theta_{n-1}$, when there are n training examples existing in the training set. The threshold candidates are displayed in Figure 5.3, in which, the blue stars indicate feature values, and the short vertical lines indicate the threshold candidates. As shown in Figure 5.3, the threshold candidates are in the middle

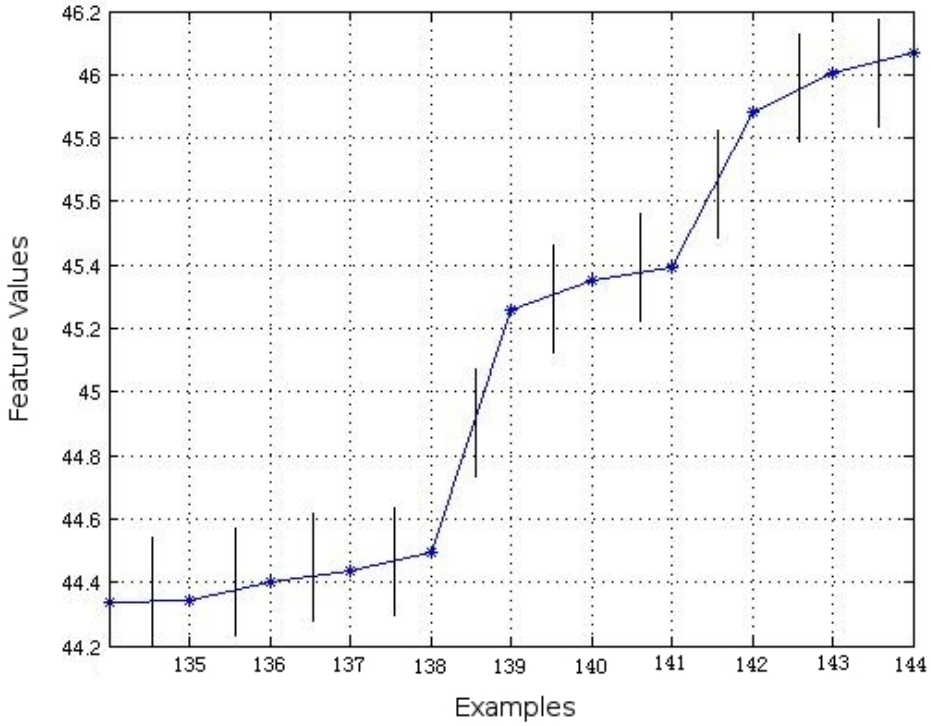


Figure 5.3: Threshold candidates and feature values

of the two adjacent feature values. For each candidate, the accumulated error of Potsu weak learner h with the threshold θ_k is calculated by

$$\varepsilon_k = \sum_{i=1}^{n} \omega_i |h(x_i) - y_i|^2 \tag{5.5}$$

Once ε_k is greater than 0.5, the calculation on the accumulated error is stopped, and the candidate is no longer taken into account in the following selection. This is because, the weak learner should perform better than a random guess [47] for preventing the loss of any generality.

5.1.4 Summary on Potsu Weak Learner

The Potsu weak learner adopts a heuristic approach to find an optimal threshold reasonably close to the best possible solution. With a large number of examples,

the accuracy on predicting the threshold is very high, due to a large number of candidates in selection. In addition, finding an optimal threshold in the Potsu weak learner is very fast due to the perceptron prototype, because perceptrons are especially suited for simple problems in pattern classification. The classification in perceptron is based on thresholding, which is one of the fastest algorithms for segmentation and recognition in pattern classification. The Potsu weak learners adopt a threshold selection technique. In this technique, the algorithm firstly is to collect a set of threshold candidates, and then evaluate the training performance on each candidate, and finally the candidate with the minimal error is selected as the optimal threshold. In [50, 92, 141], weak learners are modelled as the log likelihood ratio (LLR) (see Appendix B), which requires to predict probability density functions. Obviously, a Potsu weak learner is faster than those LLR based weak learners which demand the complicated prediction on the probability density function.

The Potsu weak learners are fast in processing one-dimensional data, but it could be exceedingly computationally expensive in higher dimensional data. In one-dimensional data, the number of threshold candidates is $n-1$ from n examples. However, if the data is expanded into a 2-D space, the number of candidates will be increased to $(n-1)^2$. If the data is in an m-D space, the number will be $(n-1)^m$. It is concluded that the number of candidates increases exponentially with the number of dimensions of the data. More candidates in training a Potsu weak learner lead to higher computational cost. Therefore, the Potsu weak learners are most suitable in low-dimensional data.

5.2 Feature Selection by Potsu

The AdaBoost training is used to select significant features from a large feature set. Since AdaBoost training is notoriously time consuming, optimisation techniques are proposed to speed up the process. The optimisation techniques and the Potsu weak learners are integrated into AdaBoost, and a new variant of AdaBoost - Binary Gabor-Boosting algorithm is presented. In the experiments, the results from selected features are discussed with three different feature pre-selection schemes, *i.e.*, the **Full** scheme, the **Interlace** scheme, and the **Scaling** scheme.

5.2.1 Optimisation on AdaBoost

As discussed in Section 4.4.3, each iteration in Binary Gabor-Boosting training rejects some features which have higher error rate. This is used to reduce the computational cost meaningfully during the training. The reason is that weak learners with high error rates can not be integrated into a strong classifier. The following considerations are taken.

- There are more negative examples than positive examples.
- Positive examples are more important than negative examples.
- The performance on positive and negative examples can be evaluated separately.

In the XM2VTS experiment, the ratio between the number of positive examples and negative examples is highly unbalanced due to the nature of the problem and database. If a strong classifier is trained based on these unbalanced data, the ability to recognise negative examples will be much stronger than the ability to recognise positive examples. However, in a face verification system, the ability to recognise a client should be the most important concern, because the purpose of such a system is not to recognise impostors. The positive examples are more important than the negative examples. At the beginning of the training, the positive and negative examples are given different weights according to the number of examples in training. Hence, the performance of positive and negative examples can be evaluated separately. It indicates that rejection or acceptance of a weak learner will be not only based on its overall classification error, but also on negative examples' error with respect to the weights and false negative rate.

Based on these considerations, the Binary Gabor-Boosting algorithm rejects some "unqualified" features in each iteration. In the XM2VTS experiment, the rejection on a Potsu weak learner is based on negative examples' error with respect to the weights $\varepsilon_{y_i=0}$ and false positive rate $\mathcal{FP}$. The error $\varepsilon_{y_i=0}$ should not exceed a half of the sum of weights belonging to negative examples as

$$\varepsilon_{y_i=0} \leq \frac{1}{2}\sum_{i=1}^{n}\omega_i(1-y_i) \tag{5.6}$$

For any $\varepsilon_{y_i=0}$ disobeying Equation 5.6, the corresponding feature will be removed from the training set. The term $\sum_{i=1}^{n}\omega_i(1-y_i)$ is the sum of weights of all negative

examples. If all negative examples are misclassified and all positive examples are classified correctly, the error ε will be equal to $\sum_{i=1}^{n} \omega_i(1-y_i)$. The half of the sum means that the weak learner has just achieved classification performance equivalent to a random guess. At the beginning of AdaBoost training, $\frac{1}{2}\sum_{i=1}^{n} \omega_i(1-y_i)$ is equal to 0.25. After several iterations, the number is changed, due to updating the weight on each example. To build a strong AdaBoost classifier, weak learners should have the classification performance better than a random guess over all negative examples.

The false negative rate $\mathcal{FN}$ is a term to describe a statistical error of misclassification on positive examples. Due to there being only four positive examples in the data set, the possible $\mathcal{FN}$ could be 0.0, 0.25, 0.5, 0.75 and 1.0 corresponding to 0, 1, 2, 3, 4 positive examples misclassified. In the Binary Gabor-Boosting algorithm, the $\mathcal{FN}$ of each weak learner should not exceed 0.5, *i.e.*,

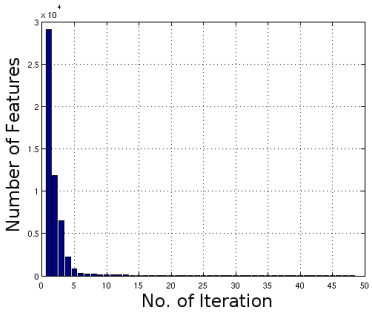
$$\mathcal{FN} < 0.5 \tag{5.7}$$

which means it is not allowed more than 2 positive examples mis-classified.

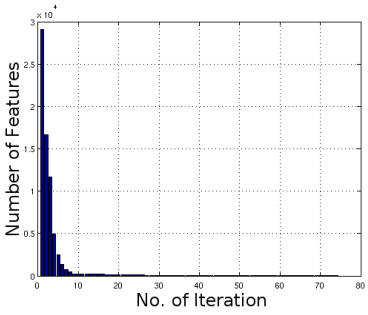
By controlling $\varepsilon_{y_i=0}$ and $\mathcal{FN}$, the number of features is dramatically reduced at each iteration during the training. This greatly reduces the computational time. The number of features trained in each iteration from the first client to the eighth client is displayed in Figure 5.4. The pre-selection scheme is the **Interlace**, so that there are 29120 features at the beginning of the training. From Figure 5.4, after the first iteration, roughly a half of the features are removed from the whole set. In the eight clients, six clients have less than 15000 features at the second iteration, and two clients (the second and the eighth) have slightly more than 15000 features. The number of features drops significantly when the iteration goes further.

From Figure 5.4, from the first client to the seventh client, the number of features in the seventh iteration drops below 1000. It was planned in the training to select 200 features. However, when the computational optimisation is applied, the training is able to select from 30 to 70 features.

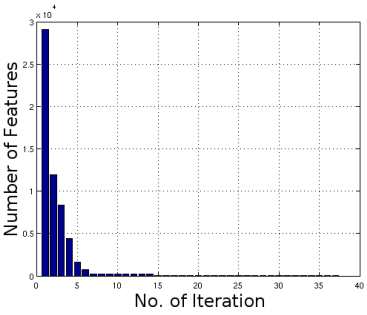
Table 5.2 shows the difference on the total number of weak learners trained with and without the optimisation, and the ratio on the time cost. Consequently, the time ratio between the optimisation and no optimisation varies from 1 : 18 to 1 : 30, which proves that the optimisation reduces the computational cost of training significantly. For example, training a feature on an *Intel Pentium 4*



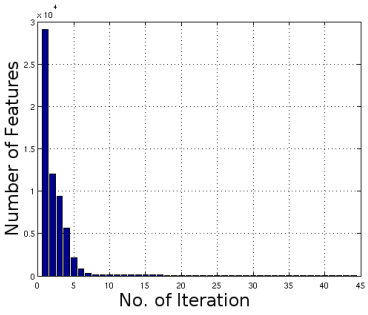
(a) The 1st client



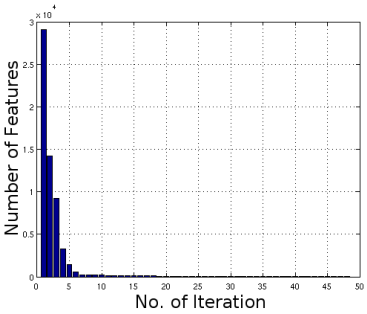
(b) The 2nd client



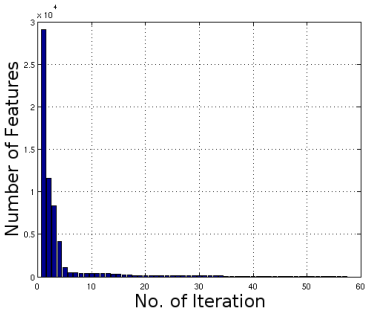
(c) The 3rd client



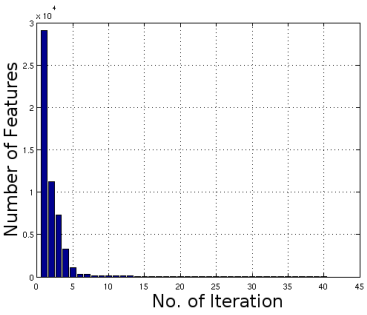
(d) The 4th client



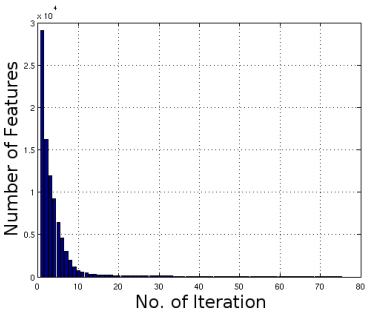
(e) The 5th client



(f) The 6th client



(g) The 7th client



(h) The 8th client

Figure 5.4: The number of features in each iteration for 8 clients

2.8 GHz machine takes 70 milliseconds. For the first client, the training without optimisation will take $1,396,632 \times 70\text{ms} \approx 97,764\text{s}$, *i.e.*, roughly 27 hours; however, the training with the optimisation will take $53,302 \times 70\text{ms} \approx 3,731\text{s}$, *i.e.*, roughly one hour.

Table 5.2: The difference between the number of iterations with and without the optimisation

Client	no. of Iterations	no. of weak learners (with optimis.)	no. of weak learners (without optimis.)	Ratio (with:without)
C1	48	53302	1396632	1:26
C2	74	72088	2152105	1:29
C3	37	58663	1076774	1:18
C4	44	61451	1280334	1:20
C5	48	60920	1396632	1:22
C6	57	61592	1658244	1:26
C7	40	54249	1164020	1:21
C8	75	90460	2181225	1:24

Another time-reducing optimisation technique, as described in Section 4.4.3, is to reject any positive or negative features which have an error greater than 0.5. This optimisation method is named *optimisation 1*, and the one evaluating positive and negative examples separately is called *optimisation 2*. The *optimisation 1* can save the computational cost around 50% [180]. The comparison chart between the training without any optimisation, with *optimisation 1*, and with *optimisation 2* is given in Figure 5.5. The Figure shows the number of remaining features in each iteration for 20 iterations. Because there are different specifications on the training images, the beginning number of features in *optimisation 2* is more than the number in *optimisation 1*. After the training of the first iteration, the number of remaining features in *optimisation 2* quickly drops to half of the original number, but in *optimisation 1*, there is only a slight drop on the number of features. After the fifth iteration, the remaining features in *optimisation 2* can be ignored because it is very small compared with the number in *optimisation 1*. Therefore, the optimisation method - *optimisation 2* is the most efficient for reducing the computational cost.

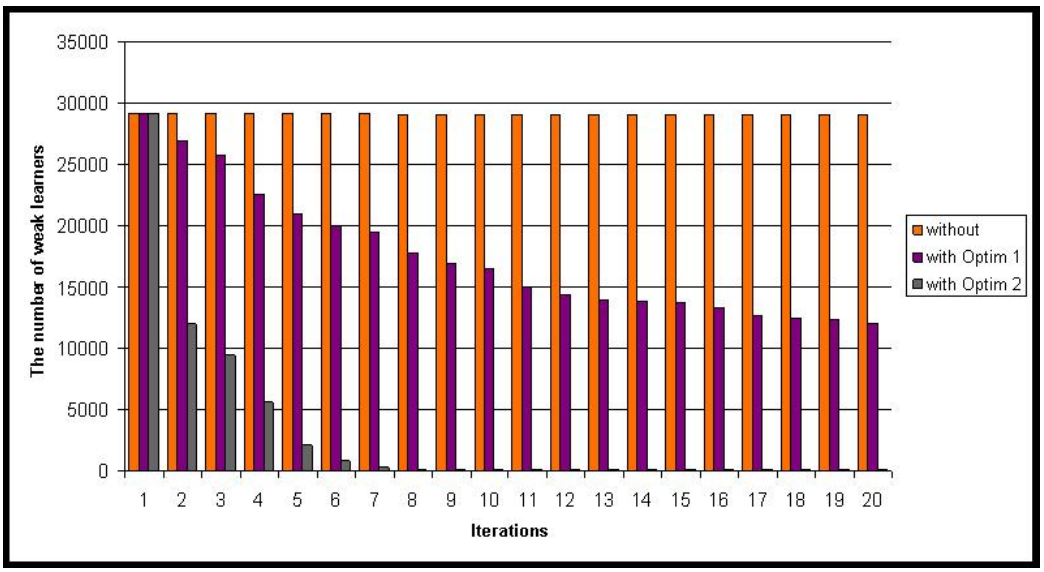


Figure 5.5: The training time comparison between no optimisation, optimisation 1, and optimisation 2

5.2.2 Binary Gabor-Boosting algorithm

A new AdaBoost algorithm, named as Binary Gabor-Boosting algorithm, is proposed with the Potsu weak learners and the *optimisation 2*. The new AdaBoost algorithm is designed not only to select significant features, but also to construct the strong AdaBoost classifier. The strong classifier is composed of several Potsu weak learners which are built on the corresponding Gabor wavelet features. It is described in Table 5.3 -(impostor: negative examples; client:positive examples).

The experiment is operated on two notable face databases - XM2VTS and FERET. In this section, the results from XM2VTS are discussed, while the results from FERET will be discussed in Section 5.7.

5.2.3 Result discussions

The Binary Gabor-Boosting algorithm is trained and tested with the XM2VTS face database. The algorithm is based on each particular client, *i.e.*, subject, so that the images belonging to a client are treated as positive examples, and others are negative examples. In the first scenario, due to there being 8 images for each client and 200 clients available, the first four images (named as No.1, No.2, No.3 and No.4) of each client are collected into the training set. There are 800 examples

Table 5.3: Binary Gabor-Boosting algorithm for feature selection and classification

1: Given example $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the data of the ith example, which are contributed by k features $\{j_1, \ldots, j_k\}$, and $y_i \in Y = \{0, 1\}$ for impostor and client respectively..
2: Initialise the weights $\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of impostors and clients respectively.
3: **for** $t = 1, \ldots, T$ **do**
4: Normalise the weights, $\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{i=1}^{n} \omega_{t,i}}$ so that ω_t form a probability distribution.
5: **for all** $\{j_1, \ldots, j_k\}$ **do**
6: Train a Potsu weak learner h_j built with one single feature j with the weights $\omega_{t,i}$.
7: The error is calculated as $\varepsilon_j = \sum_{i=1}^{n} \omega_{t,i} |h_j(x_i) - y_i|^2$ with respect to $\omega_{t,i}$
8: **end for**
9: Choose the optimal weak learner h_t with the lowest error ε_t from all h_j.
10: Select the corresponding feature j_t of the weak learner h_t as a significant feature.
11: Remove the feature j_t from the feature set $\{j_1, \ldots, j_k\}$.
12: Update the weights $\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i}$, where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
13: Choose the importance $\alpha_t = \log \frac{1}{\beta_t}$, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
14: **end for**
15: The final "strong" (ensemble) classifier

$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

in the training set, and 800 in the testing set. Given a particular client to verify, the positive examples are the first four images of the client, and the negative examples are the other 796 images. Feature pre-selection schemes also are applied to remove non-useful features for reducing the computational cost before running the feature selection algorithm. The comparison of the **Full size** scheme, the **Interlace** scheme and the **Scaling** scheme is given here.

Features with the full-size pre-selection scheme

After the Gabor-Boosting training, a small set of features are selected as significant features for each client. The number of selected features from each client is different. From the 1st client to the 8th client, there are 75, 103, 54, 65, 60, 110, 52, 109 features, respectively. As the Full pre-selection scheme does not reduce the original number of features, the original feature set is larger than the ones in the Interlace and Scaling schemes.

Figure 5.6 displays the selected features projected on the corresponding client face image in the top row and the first Gabor wavelet features for each client in the bottom row. For most clients, these selected features are randomly distributed

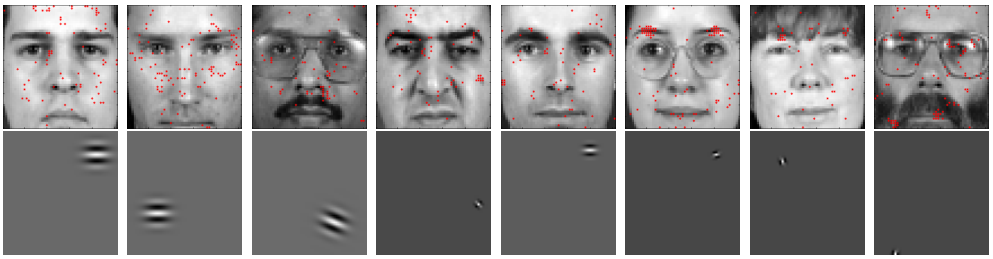


Figure 5.6: The selected features on face images and the first selected Gabor wavelet feature with the **Full** feature pre-selection scheme.

on faces. These features are also distributed on common facial feature areas, such as eyes, eyebrows, nose, and mouth. Obviously, there are features located on the mole of the 4th client's face. For the 6th client, two clusters of features are located around the two eyebrows. Similarly with the 7th client, some features also cluster around eyebrows. Feature clusters also appear in the 8th client face image.

Features with Interlace pre-selection scheme

Each client has different number of features selected, varying from 40 to 70. Figure 5.7 displays the selected features projected on the corresponding client face image in the top row and the first Gabor wavelet feature for each client in the bottom row. These selected features are randomly distributed across faces. It is observed that

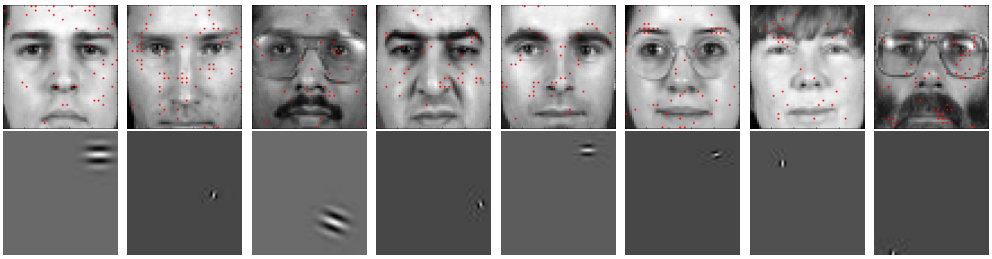


Figure 5.7: The selected features on face images and the first selected Gabor wavelet feature with the **Interlace** feature pre-selection scheme.

feature clusters disappear. Common facial feature areas, such as eyes, eyebrows, nose, and mouth, attract related features. Interestingly, there are some features located on nostrils both in the first client and fifth client. It may indicate that their nostrils could be an important feature for recognition. For the second client, more features are located around the eyes and the nose. For the third and eighth clients who wear glasses and have mustaches, there are more features around those two areas. The fourth client shows more features around the eyes, the eyebrows, and the gap between cheek and nose. The sixth client who also wears glasses, however has more features located along the eyebrows rather than the glasses. In the seventh client, the person is different from others, because she has hair on the forehead covering the eyebrows. Therefore, some features are laid on the plausible eyebrows area.

Features with the Scaling pre-selection scheme

The Scaling pre-selection scheme scales the response images down in higher spatial frequency. For a feature with higher spatial frequency, it is meaningless to project the feature onto the original face image, since the coordinate has been changed. To display the feature correctly, it is necessary to project the feature onto the scaled face image. Figure 5.8 shows the projected features and the first Gabor wavelet feature on each client. The first row shows the original images, and the red dots

Figure 5.8: The selected features on face images and the first selected Gabor wavelet feature with the **Scaling** feature pre-selection scheme.

represent the features with the spatial frequency $\nu = -1$. The second, third, fourth, and fifth rows are the face images which have been scaled down, and those red dots on images represent the selected features of the spatial frequency $\nu = 0, 1, 2, 3$, respectively. Most selected features are with the lowest spatial frequency $\nu = -1$. There are a few features with higher spatial frequency $\nu = 0, 1, 2$, as shown in Figure 5.8. No features at the spatial frequency $\nu = 3$ is selected in all eight clients. The number of selected features from each client is different. From the 1st client to the 8th client, there are 60, 45, 39, 30, 52, 83, 30, 62 selected features, respectively. It is observed that the number of selected features is approximately the same as the number selected by the **Interlace** scheme, and smaller than that with the **Full** pre-selection scheme.

5.2.4 Summary on Feature Selection

In this section, the feature selection by using Potsu weak learners is investigated. Similar to the FLD weak learner based AdaBoost feature selection, some optimisation techniques have been applied to reduce the computational time. The *Optimisation* 2 is different from the *Optimisation* 1 in Chapter 4, and the training with the *Optimisation* 2 is much faster than with the *Optimisation* 1. With the Potsu weak learner and the *Optimisation* 2, the Binary Gabor-Boosting algorithm is proposed. The results of selected features are displayed and discussed. With three feature pre-selection schemes, there are some observations from selected features. The number of selected features is different with different schemes. The features are randomly distributed on the faces. Some selected features are located on common facial landmarks.

5.3 Ensemble classification by AdaBoost

In the XM2VTS face database, the features are selected and the strong classifier is built for each client. The experiment is carried out on eight clients separately, and a small group of features are selected for each client. For each client, the first four images are taken for training, and the other four images are for testing. For 200 clients, the training set has 800 face images, and the testing set has 1560 face images[2]. Given a specific client in the training set, four images belong to the client, and the other 796 images are impostors, so the extremely unbalanced ratio between the number of positive and negative examples is $4 : 796$.

The classification results of the ensemble classifier H from the first client to the eighth client are shown in Table 5.4. All ensemble classifiers have the best performance on the training set, where all classifiers have achieved zero error, *i.e.*, no mis-classification. Both false positive $\mathcal{FP}$ and the false negative $\mathcal{FN}$ are equal to zero so that the ensemble classifiers have achieved the perfect results in the training set. However, in the testing set, the ensemble classifiers reveal poor classification performance on positive examples. All positive examples are misclassified except the 5th client, but all negative examples are classified correctly.

[2]200 clients with 4 images and 95 impostors with 8 images

Table 5.4: The classification performance of AdaBoost Classifier

Client No	Number of Features	Training Set (%)			Testing Set (%)		
		Error	$\mathcal{FN}$	$\mathcal{FP}$	Error	$\mathcal{FN}$	$\mathcal{FP}$
1	48	0.0	0.0	0.0	0.5	100	0.0
2	74	0.0	0.0	0.0	0.5	100	0.0
3	37	0.0	0.0	0.0	0.5	100	0.0
4	44	0.0	0.0	0.0	0.5	100	0.0
5	48	0.0	0.0	0.0	0.5	75	0.0
6	57	0.0	0.0	0.0	0.5	100	0.0
7	40	0.0	0.0	0.0	0.5	100	0.0
8	75	0.0	0.0	0.0	0.5	100	0.0

5.4 Overfitting

Experimental results in Section 5.3 suggest the performance of the ensemble classifier on the training set does not represent its overall performance. The ensemble classifier is trained to perform perfectly on the training set, but reveals poor performance on other data sets, *e.g.*, the testing set. This phenomenon is so called *overfitting*. The overfitting is a wide research topic in pattern recognition, machine learning, data mining, etc. There are many issues that lead to overfitting. In this case, the issues causing the overfitting are categorised into two main sources: 1) inferior training data, and 2) classification algorithm. An investigation of the overfitting phenomenon in the case of ensemble by AdaBoost is presented in Appendix C.

5.4.1 Overfitting caused by the training data

The inferior training data may lead to overfitting, which refers to the problem that classifiers are too specific, or too sensitive to the particulars of the training dataset used to build the classifier. Inside this source, there are two issues that lead to model overfitting, and they are **curse of dimensionality** and **selection bias**.

Curse of dimensionality

The overfitting in this case is partly due to the small number of positive examples in the training set. Generally, the number of training examples is related to the

number of features used in classification. In [75], it is stated that the number of training examples per class should be at least ten times the number of features. If the number of training examples does not satisfy this requirement, the performance of classification will be degraded. This phenomenon is called "curse of dimensionality". The number of features used in the classification is around 30 to 70. According to the requirement in [75], the number of training examples per class should be around 400 to 700. In this case, the number of training examples of positive class is only four, which is far less than the required number. The performance in the testing set on positive examples is very poor, since the discriminating power on positive examples suffers from the curse of dimensionality. However, there are 796 negative examples given in the training set, which are more than ten times the number of selected features. The curse of dimensionality is avoided on the negative examples, so that the discriminating power on the negative examples is very strong, *i.e.*, zero false positive rate. In this case, the reason of overfitting is the number of positive examples is far less than the number of selected features, which causes the curse of dimensionality, and the curse of dimensionality leads to the overfitting phenomenon.

Selection bias

Since the standard face database is used in the experiment, the total number of training examples is fixed. With a small number of positive examples and a large number of negative examples, the biased training data implies *selection bias* [164]. The selection bias possibly happens under some circumstance, in which one kind of example (*e.g.* negative) is more likely to be included, but other kinds of example (*e.g.* positive) is more likely to be excluded in the training set. This causes the measurement of statistical significance, *i.e.*, the discriminating power, to appear much stronger or weaker than other kinds of examples. More critically, it may possibly cause completely deceptive results. In this case, the ratio between positive and negative examples are $4 : 796$, so that the discriminating power on negative examples is very strong ($\mathcal{FP} = 0$), but the discriminating power on positive examples is very weak ($\mathcal{FN} = 100\%$). Hence, the selection bias in the training dataset affects the results of classification.

5.4.2 Overfitting caused by classification algorithm

A phenomenon that AdaBoost is prone to overfit has been observed in [61, 107]. In [118], it concludes that ensemble classifiers in AdaBoost may suffer from overfitting in the presence of noise which may explain some of the decreases in performance. Inside the source, there are two issues pertinent to overfitting. The first issue is that the AdaBoost training over-emphasises noisy examples, and the second issue is due to the weighted voting used by ensemble classifiers.

Over-emphasis on noisy examples

In terms of measurement, a training dataset contains some level of noise. Freund and Schapire [46] suggested that the poor performance of AdaBoost ensemble classifier results from overfitting the training dataset since later training may be over emphasising examples which are actually noise. In AdaBoost, an example with a higher weight is more emphasised in the training than other examples with lower weights. After each iteration of AdaBoost, the weights of some misclassified examples are increased. In the next iteration, the training will place more emphasise on these examples which are often referred to as "hard" examples. Because the AdaBoost training always focuses on "hard" examples, a "strong" ensemble classifier is constructed at the end. However, if these "hard" examples include high level noise (or actually noise), the AdaBoost training learns the noise rather than the nature of the data itself. For a problem with noise, focusing on misclassified examples may cause the ensemble classifier to focus on boosting the margins of noise, but not on boosting the margins of examples. In fact, the weight updating misleads the AdaBoost training in overall classification. In [118], noises from different levels are added into data sets, and the results demonstrate the error rate of the AdaBoost may indeed increase as the number of weak learners increases. [138] suggests that later weak learners focus on increasing the margins between examples. Early stop is useful in AdaBoost training for the purpose of increasing performance. In general, AdaBoost ensemble's poor performance may be partially explained by overfitting noise.

Weighted voting

Within the ensemble classifier, each weak learner is associated with an importance α_t, which is also considered as a weight for that weak learner. Hence, the output

of ensemble classifier is the combining weighted voting from all weak learners. Previous work [149] has shown how different weight voting schemes can generally be resilient or lead to overfitting. In the work, optimising the weights is the method of choice for achieving minimal generalisation error. In general, uniformly distributed weights may cause the overfitting, while with optimised weights the ensemble classification is insensitive to overfitting.

5.5 Solutions to Overfitting

Two main sources leading to overfitting in AdaBoost ensemble classifiers have already been discussed . This section uses some techniques to suppress each source respectively, and to examine the performance of classification. For the inferior training data, the training data is reorganised by applying cross validation, and the real performance of algorithm is revealed. To investigate whether AdaBoost is prone to overfitting in this case, an SVM classifier is adopted with respect to the selected features, and the overfitting effects are suppressed, and a better generalisation on performance is demonstrated.

5.5.1 Cross Validation

The overfitting caused by inferior training data is due to two reasons - curse of dimensionality and selection bias. Both issues arise from the very small number of positive examples. To reduce the level of curse of dimensionality and selection bias, more positive examples should be used in the training dataset. However the number of positive examples is fixed in the database. Hence, cross-validation technique is used to rotate the positive training and testing examples, and help reveal the true performance of algorithm.

Cross-validation [85, 35], also called rotation estimation, is the statistical approach to dividing a dataset into two subsets. The analysis is initially performed on one subset, while another subset is retained for subsequent use, such as confirming and validating the initial analysis. The initial subset is the training set, and the other subset is the testing set, also called validation set. The performance can be estimated by averaging over all possible divisions of subsets.

In this section, a *leave-one-out* cross-validation has been used. As the name suggests, the leave-one-out cross-validation uses a single example of each class

from the original dataset as the testing dataset, and the remaining examples for training. It is repeated in turn such that each example of the class in the dataset is used once as the testing data. Errors are estimated by simply averaging over the turns. The leave-one-out cross-validation is an excellent tool to test the robustness of a classifier, which is sensitive to small changes in the training set. If a classifier performs well under the leave-one-out cross validation, it suggests that the classifier is satisfactory.

There are eight face images for each client in the XM2VTS database. By using the leave-one-out cross-validation, the training set and the testing set have been changed to seven images for training and one image for testing. For a particular client, the number of examples in the training set has been increased to 803, which involves 7 positive examples and 796 negative examples. The leave-one-out cross validation is carried out 8 times. Table 5.5 shows the performance on the 4th client with the **Scaling** pre-selection scheme. As indicated in Table 5.5, validations work

Table 5.5: Performance of leave-one-out cross-validation on the 4th client

Leave-one-out	Number of Features	Training Set (%)		Testing Set (%)	
		$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	38	0.0	0.0	0.0	0.0
2	28	0.0	0.0	0.0	0.0
3	19	0.0	0.0	0.0	0.0
4	39	0.0	0.0	100	0.0
5	18	0.0	0.0	0.0	0.0
6	37	0.0	0.0	0.0	0.0
7	40	0.0	0.0	0.0	0.0
8	36	0.0	0.0	0.0	0.0

very well except of that the cross-validation leaves the 4th image out. In the cross-validation, the only one positive example in the testing set is mis-classified, but all other negative examples are classified correctly.

By applying *leave-one-out* cross-validation, the number of positive examples is increased, so that the degree of curse of dimensionality and selection bias are reduced in the case. In the original training set, the number of positive examples is four, while in these cross-validation sets, the number becomes seven. Although the number of positive examples is still much less than the number of features (around

20 to 40) and the selection still bias to negative class, the overall performance of classification is significantly increased. In general training, to suppress overfitting, the training data should have the number of examples per class ten times more than the desirable number of features, and the number of examples in each class should be roughly equal. However, in this case, with small increase in the number of positive examples, the performance has improved greatly.

5.5.2 SVM

Since AdaBoost is prone to the overfitting, an alternative classifier - SVM is used to classify the examples. In this section, an SVM classifier is trained with the given selected features on the training set.

Results from SVM classification

With eight clients, eight SVM classifiers are trained with a polynomial kernel with degree of three. From the 1st client to the 8th client, there are 48, 74, 37, 44, 48, 57, 40 and 75 features used to construct the SVM classifiers. The training of the SVM actually is to adjust infrastructure between each selected feature and maximise the margin between the positive and the negative. After the SVM classifiers are well trained, the training set and the testing set are given for the testing purpose. Table 5.6 shows the performance of SVM classification belonging to eight clients. On the 1st, 2nd, 4th, 5th and 6th clients, these SVM classifications have achieved

Table 5.6: The performance of SVM classification on eight clients with all selected features

Client	Number of Features	Training Set (%)		Testing Set (%)	
		$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	48	0.0	0.0	0.0	0.0
2	74	0.0	0.0	0.0	0.06
3	37	0.0	0.0	50	0.0
4	44	0.0	0.0	0.0	0.0
5	48	0.0	0.0	0.0	0.0
6	57	0.0	0.0	0.0	0.0
7	40	0.0	0.0	100	0.13
8	75	0.0	0.0	100	0.0

perfect performance, in which all positive examples in both the training set and the testing set are classified correctly. For the 2nd client, only one negative example is recognised as positive, and the false positive rate $\mathcal{FP}$ is 0.06% which is very small and can be neglected. Hence, the SVM can achieve low error rate. However, when the SVM encounters some "hard" clients, the poor performance apparently exists. The 8th client still gets overfitting with the training set and has a 100% false negative rate in the testing set. Even more, the 7th client presents a slightly worse performance in which the classifier not only overfits the training set, but also causes exiguously false positive in the testing set.

Suppressing Overfitting

In the previous section, two issues for ensemble classifiers in AdaBoost to be overfitted are presented: the issue of over-emphasising on noisy examples and the issue of optimising weights. From the perspective of these two issues, the SVM is used to suppress the overfitting effects.

Support Vectors In [46], AdaBoost training emphasises on noisy examples at the later iterations, so that the overfitted result is acquired at the end of training. The training of SVM is the optimisation process which is to find the primal form of the Lagrange formalism. By optimising the Lagrange function, a small group of support vectors (SVs) are obtained, which defines the hyperplanes between classes. The SVs are some examples with non-zero Lagrange multipliers. In this case, there are 5 to 11 examples taken as SVs from a total of 800 training examples. The numbers of SVs in the SVMs based on these eight clients are shown in Table 5.7. Hence, SVMs use a small number of examples to construct classifiers instead of the whole set of examples (including the noisy examples). This optimisation process is equivalent to finding some examples, *i.e.*, SVs, which contain the lowest level of noise. By only using SVs, noisy examples are excluded in the construction of the classifier. The training and classification only emphasise the examples with the lowest level of noise, so that the overfitting caused by noisy data is reduced to the minimum level. In addition, when SVMs are designed originally, the overfitting issues are taken into account. SVMs avoid overfitting by choosing a specific hyperplane among the many that can separate the data in the feature space. SVMs find the maximum margin hyperplane, which maximises the minimum distance from the hyperplane to the closest SVs

Table 5.7: The numbers of SVs in the SVMs based on eight clients

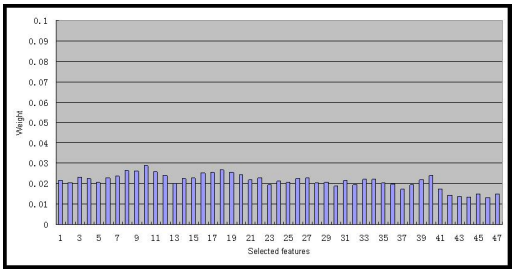
Client	No of SVs	No of Positive SVs	No of Negative SVs
1	7	1	6
2	11	2	9
3	5	1	4
4	10	1	9
5	9	1	8
6	7	1	6
7	7	2	5
8	5	2	3

Optimisation on weights Sollich and Krogh [149] displayed that uniformly distributed weights normally lead to overfitting, while with optimised weights the classification is insensitive to overfitting in ensemble classifiers. The ensemble classifier in AdaBoost contains a set of weights for weak learners which vary in a small range. Each selected feature j_t is used to construct a weak learner h_t, and each weak learner is associated with a weight α_t. The weight indicates how importantly the corresponding feature plays a role in classification. When these weights are scaled into a probability distribution, *i.e.*,$\sum \alpha_t = 1$, the scaled weights show some similarity to a uniform distribution. The probability distributions of weights from eight clients are shown in Figure 5.9 and 5.10. In the probability distribution, the variances between weights are small, which convey all selected features have roughly equalled importance in classification.

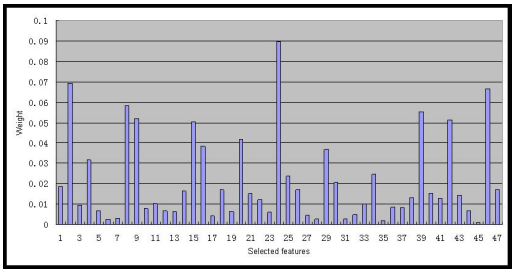
The SVM classifier also contains a set of weights for selected features which are randomly distributed. The training process of SVM is to find a linear discriminant function

$$g(x) = \omega^T \mathbf{x} + b \tag{5.8}$$

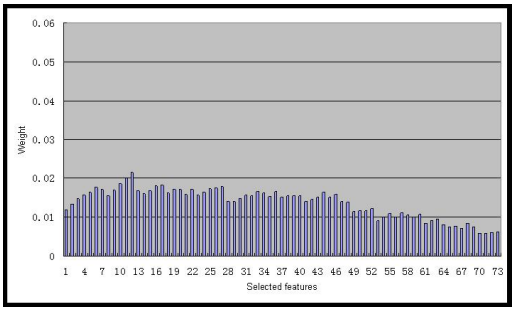
where $\mathbf{x}$ is the vector $\{x_1, x_2, \ldots, x_n\}$ formed by n selected features, ω is the weight vector $\{\omega_1, \omega_2, \ldots, \omega_n\}$ for corresponding n features, and b is a bias. The weights also imply how importantly the corresponding features play roles in classification. When these weights are scaled into a probability distribution, the weights vary dramatically. The probability distributions of weights from eight clients are shown in Figure 5.9 and 5.10. For SVM, a few features have their weights much higher than other features, which indicates that only a small number of features play important



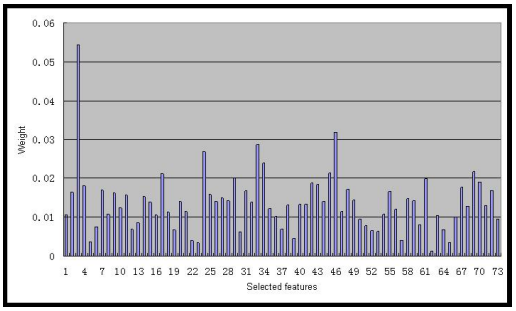
(a) AdaBoost on the 1st client



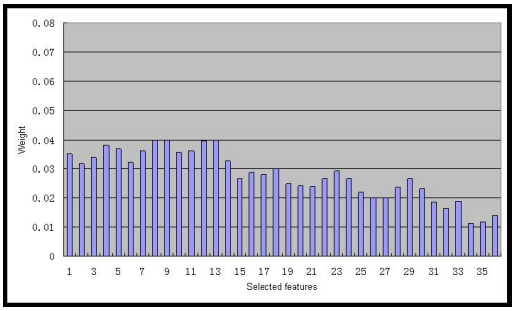
(b) SVM on the 1st client



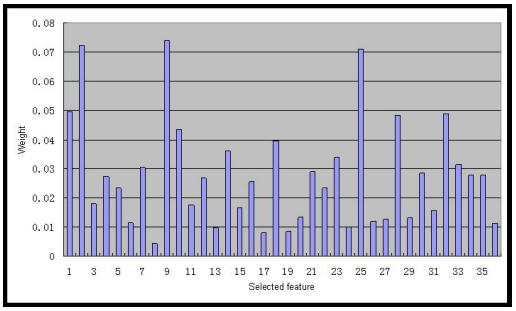
(c) AdaBoost on the 2nd client



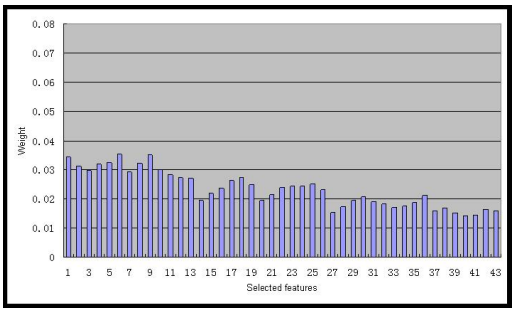
(d) SVM on the 2nd client



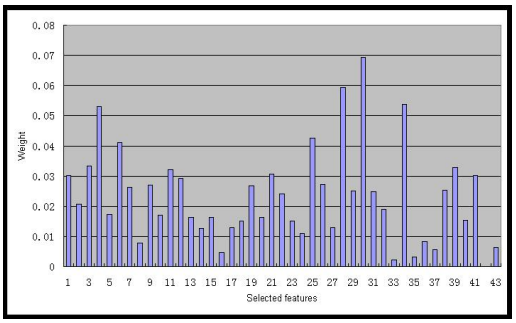
(e) AdaBoost on the 3rd client



(f) SVM on the 3rd client

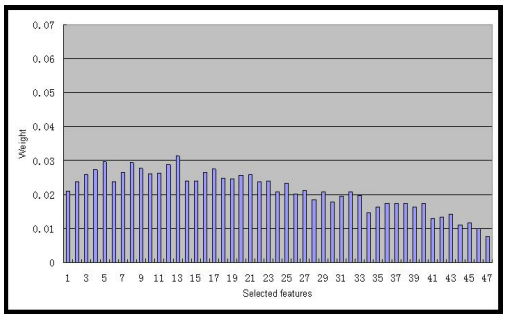


(g) AdaBoost on the 4th client

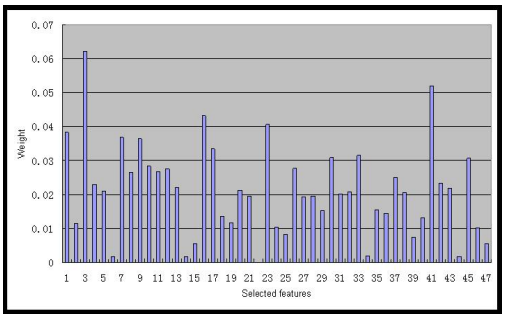


(h) SVM on the 4th client

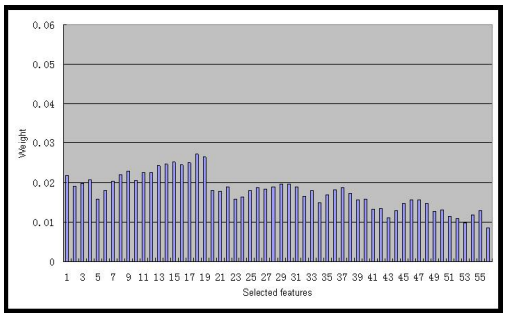
Figure 5.9: The weights' distribution from the 1st client to the 4th client



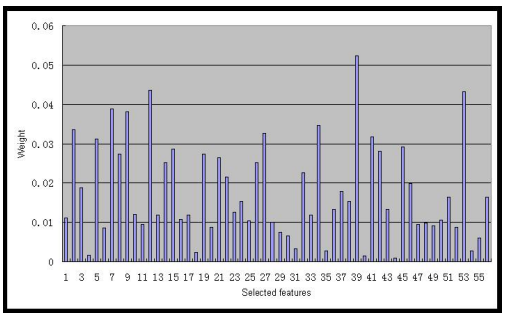
(a) AdaBoost on the 5th client



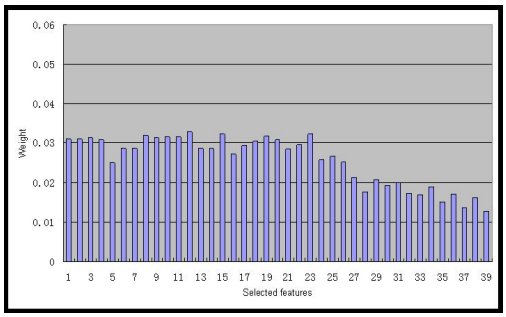
(b) SVM on the 5th client



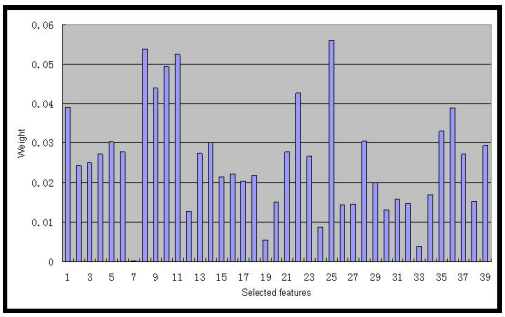
(c) AdaBoost on the 6th client



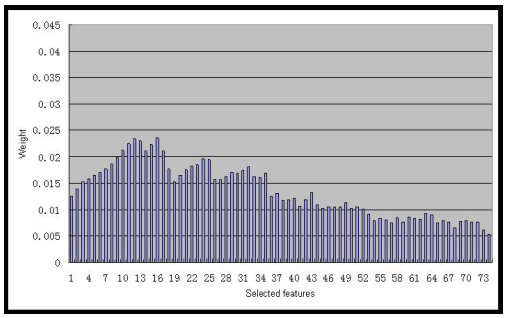
(d) SVM on the 6th client



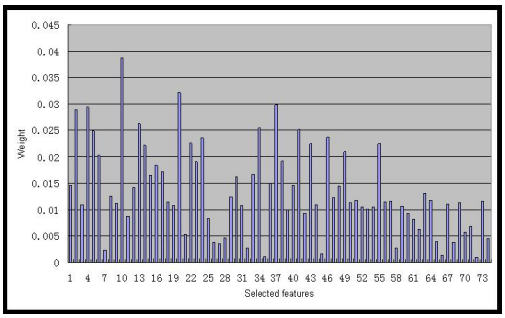
(e) AdaBoost on the 7th client



(f) SVM on the 7th client



(g) AdaBoost on the 8th client



(h) SVM on the 8th client

Figure 5.10: The weights' distribution from the 5th client to the 8th client

roles in classification. In this case, the ensemble classifier shows a nearly uniform distribution on weights for selected features, while the SVM classifier shows optimised weights. Hence, when the weights for selected features are optimised, the overfitting effect is reduced in SVMs.

The overfitting is suppressed by using SVMs, and the overall performance is improved. Compared with the results from SVM classification, the ensemble classifier in AdaBoost shows some drawbacks from two issues. Firstly, the ensemble classifiers are prone to fit some noisy examples at later iterations, while the SVMs only focus on some important (non-noisy) examples - SVs. In addition, the ensemble classifiers have a set of nearly uniformly distributed weights, but the SVMs demonstrate some optimisation on weights for improving the performance of classification. Since the SVM classifiers have better overall performance than the ensemble classifier in AdaBoost, the overfitting effects can be partly explained due to the ensemble classifier in AdaBoost being prone to overfitting. The results proves that the AdaBoost is indeed susceptible to overfitting in this case, so that it is better to use additional classifiers, such as SVMs, in the classification.

5.5.3 Summary on Overfitting

Both inferior training dataset and ensemble classifier in AdaBoost lead to overfitting. By improving the training dataset or applying alternative classifiers, the overfitting effect is reduced, and even avoided in some cases. When the training dataset is improved by applying cross-validation, the performance is 87.5% hit rate on positive examples and 100% hit rate on negative examples. When the classifier is changed to SVM, the performance of classification improves for some clients, which demonstrates using an SVM classifier is an appropriate choice for classification. In the real implementation under a restricted face database, it is impossible to add more new face images into the database, which makes improving the training dataset very difficult. Hence, the only way for suppressing the overfitting effect is to use SVMs as the classifiers. In the proposed algorithm, significant features are selected by AdaBoost algorithm, while the verification is performed by SVM.

5.6 Potsu weak learner v.s. FLD weak learner

Instead of using all selected features, an additional experiment is carried out with only the top 20 features. The evaluation set is fixed in a 20-dimensional feature space for each client. An SVM is trained, and the performance for each client is shown in Table 5.8. For the 1st, 4th, 5th and 6th clients, the performance is

Table 5.8: The performance of SVM evaluation on eight clients with 20 selected features by Potsu

Client	Number of Features	Training Set (%)		Testing Set (%)	
		$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	20	0.0	0.0	0.0	0.0
2	20	0.0	0.0	25	0.0
3	20	0.0	0.0	50	0.0
4	20	0.0	0.0	0.0	0.0
5	20	0.0	0.0	0.0	0.0
6	20	0.0	0.0	0.0	0.0
7	20	0.0	0.0	50	0.0
8	20	0.0	0.0	75	0.13

very good with 20 features. For some clients, like the 2nd and 3rd clients, the performance could be improved by adding more features. For the 3rd client, the $\mathcal{FN}$ rate is fixed at 50%, no matter whether 37 or 20 features are used. For those "hard" clients, such as the 7th and 8th clients, the performance is varied. Adding more features into the evaluation set does not guarantee the improvement, but leads to overfitting.

Table 5.9 demonstrates the performance comparison of face verification algorithm with Potsu and FLD weak learners. The AdaBoost algorithm is used for feature selection and SVMs for classification. Both SVM classifications use 20 features. The AdaBoost algorithm with the Potsu weak learner has a better result than the one with FLD weak learners. This indicates that the Potsu weak learner in AdaBoost training contributes better to the classification performance compared to the FLD weak learner. In general, the Potsu weak learner is much superior than FLD weak learners in AdaBoost training.

Table 5.9: Performance of the Potsu weak learner and the FLD weak learner with only 20 features.

Client No	$\mathcal{FP}$		$\mathcal{FN}$	
	Potsu	FLD	Potsu	FLD
1	0.0	0.0	0.0	0.0
2	0.125	0.25	0.0	0.0
3	0.25	0.25	0.0	0.0
4	0.0	0.375	0.0	0.0
5	0.0	0.25	0.0	0.0
6	0.0	0.25	0.0	0.0
7	0.25	0.5	0.0	0.002
8	0.375	0.5	0.001	0.0

5.7 FERET Testing

The proposed Gabor-Boosting algorithm is also tested in the FERET face database. In this section, the FERET face database is introduced. To accomplish face recognition, pre-processing is applied to FERET face images. By using the Potsu weak learner based AdaBoost, the significant features are selected. In classification, the ensemble classifier and the SVM classifier are performed.

5.7.1 FERET Database

The Facial Recognition Technology (FERET) program and development of the FERET database is sponsored by the DOD (Department of Defence) Counterdrug Technology Development Program in USA. In addition, the National Institute of Standards and Technology (NIST) serves as a technical agent for distribution of the FERET database. As a part of the FERET program, the database of face images was collected in 15 sessions between December 1993 and August 1996. The database is used to develop, test and evaluate face recognition algorithms. In order to display the state-of-art in face recognition technology, the FERET database has been made available to scientists in the research area. The goal of the FERET program is to develop new technology for the automatic recognition on human faces. The program contains three objectives [124].

- To develop algorithms required for a face recognition system.

- To create a large database of facial images for face recognition.
- To test and evaluate the algorithm based on the developed database.

The FERET database contains 1,564 sets of shots for a total of 14,126 images with 1,199 individuals. The whole sets are divided into two portions, the development portion and the sequestered portion. The development portion of 503 sets is released to the public for scientific research, while the sequestered portion is reserved by the US government for independent evaluation. The original FERET database was released in 2001 and consisted of 14,051 gray scale images. The original FERET database is also called the Grey FERET database. In 2003, a colour version of the FERET database was released, which contains 11,338 facial images from 994 individuals. The new FERET database is called Colour FERET database. In the thesis, without mentioning either Colour or Grey, the FERET database refers to the Colour version.

Images from the FERET database are 512×768 pixels, and are stored in PPM (Portable Pixel Map) image format. The images have not undergone any lossy compression or processing. The filenames of these images are in the form of *nnnnn_yymmdd_xx_c*. The filename is divided into four parts. *nnnnn* is a five digit integer that identifies the subject (person). These five digit integers are called the ID of a subject. *yymmdd* represents the date when the face image was actually captured. *xx* shows a pose of the face. There are 13 different poses shown in Table 5.10. The set *fa* represents the regular frontal image. The set *fb* represents alternative frontal image, which was taken shortly after the corresponding *fa* image was taken. A regular frontal image was captured for every subject at every shot session. In the FERET database, each subject includes 5 to 11 images. In each subject, there are at least two front views (*fa* and *fb*). A different facial expression is requested for the second frontal image.

The database also contains ground truth files for the subjects and the images. Ground truths are recorded in two formats - XML file and plain text file. These files give the detailed ground truth of corresponding images, such as gender, with or without glasses, beard, etc. The information of these ground truth files are required in the pre-processing of face images.

Table 5.10: 13 poses in FERET Database with angles, number of images and the number of subjects

Pose	Angle	Images (#)	Subjects (#)
fa	0	1364	994
fb	0	1358	993
pl	+90	1312	960
hl	+67.5	1267	917
ql	+22.5	761	501
pr	−90	1363	994
hr	−67.5	1320	953
qr	−22.5	761	501
ra	+45	321	261
rb	+15	321	261
rc	−15	610	423
rd	−45	290	236
re	−75	290	236

5.7.2 Pre-processing on the FERET

In this thesis, only frontal view face images are considered, such that only the *fa* and the *fb* sets are used. As for the XM2VTS database, segmentation and pre-processing are also needed to remove useless substances, *e.g.*, the background with shadow, the subject's hair, face, ears, neck, clothes and so on. The original image is segmented into a smaller size of 128×128 pixels, which only contains the inner face part. The segmentation requirements are:

- The subimage with 128×128 pixels
- The distance between the centres of two pupils is 64 pixels
- Two eyes must be horizontal
- The height from the centre of pupil to the bottom of the image is 96 pixels

The requirements are as visualised in Figure 5.11. The steps of the segmentation are as follows

1. Calculate the angle θ of the two eyes' pupils in the original image I.
2. Rotate the image I with the angle θ, to make the two eyes horizontal and generate a new image I'.

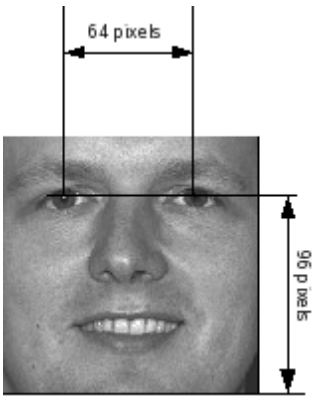


Figure 5.11: The visualisation of the requirements of the subimages.

3. Calculate the distance d between the two pupils (two pupils should be horizontal now).
4. Get a scaling ratio r by $r = 64/d$.
5. Scale the image I' with the scaling ratio r and generate a new image I''.
6. Crop the image I'' based on the eyes' coordinates, and get a 128×128 subimage.

By applying the above steps to the original face image, two pupils' coordinates on the original image must be acquired. The coordinates are obtained from the ground truth files. Some images do not have pupils' coordinates provided and a manual detecting program has to be applied. In image rotation, affine transformation [74, 66] is applied on the original image with the rotation matrix to rotate the original image.

Subimages of 128×128 pixels are further reduced to 64×64 pixels with consideration of the computational cost. To prevent information loss, the resize algorithm is adopted with the **bicubic interpolation** [83], which keeps the best image quality. These processed subimages are shown in Figure 5.12.

5.7.3 Feature Selection

There are two frontal image sets fa and fb in the FERET database, such that the fa set is used as the training dataset, and the fb set is used as the testing dataset.

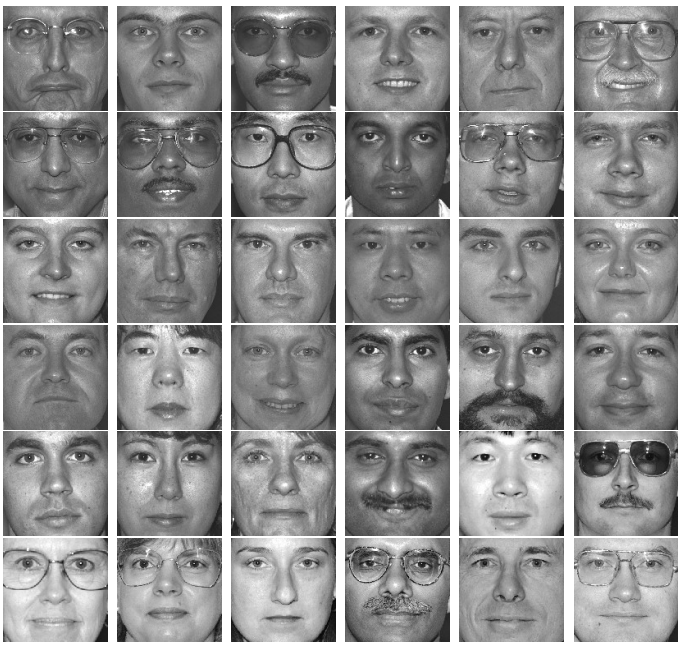


Figure 5.12: Samples of Segmented subimages from the FERET Database.

There are $1,364$ images with 994 different subjects in the training dataset, and $1,358$ images with 993 subjects in the testing dataset. The number of images - examples in both datasets is similar, and the condition of illumination and other environments are unchanged between the fa set and the fb set.

The training and testing on FERET is also client-based. To reduce the degree of curse of dimensionality and selection bias, the subject with ID00029 has been chosen as the client who has 11 images in the training set. There are a total of $1,364$ face images, so the ratio between the positive and negative is 11/1353. As in the XM2VTS database, the ratio is still very rare, but there are more positive examples in the training set. In the testing set, the subject with ID00029 also has 11 face images, and the total number of images in the testing set is $1,358$. The 22 face images in the training set and the testing set are shown in Figure 5.13.

After these 64×64 subimages are convolved with 40 Gabor wavelet kernels, the feature space is in the order of $163,840$ dimensions. If the Gabor-Boosting algorithm is directly applied on to this massive amount of features, the computational cost would be enormous. Therefore, a feature pre-selection scheme - **Scaling** is chosen. By applying the Scaling pre-selection scheme, the total number of features

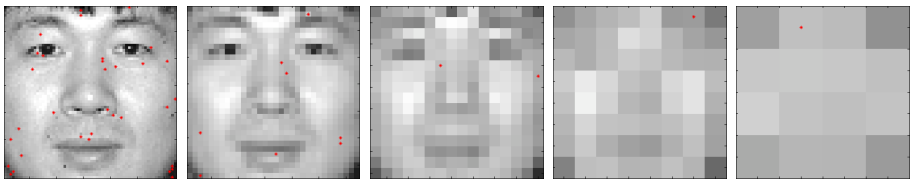
(a) The training set (11 images)

(b) The testing set (11 images)

Figure 5.13: The 22 face images for the ID00029 subject from the training set and the testing set respectively.

is reduced to 43, 648.

The Gabor-Boosting algorithm is applied on to the ID00029 subject with the *optimisation 2* in Section 5.2. The training is finished in 45 iterations, so that 45 Gabor wavelet features are selected as the significant features to verify the ID00029 subject. These 45 features are shown in Figure 5.14. To project the features with



(a) $\nu = -1$ (b) $\nu = 0$ (c) $\nu = 1$ (d) $\nu = 2$ (e) $\nu = 3$

Figure 5.14: The selected features from the ID00029 subject in the FERET database are projected on the face image with corresponding scales.

higher spatial frequency, the face image is scaled according to the frequencies. Most selected features are laid in the frequency $\nu = -1$. These features are around the eyes, nose, and mouth, and randomly spread on the face.

5.7.4 Classification

The performance of classification on the ID00029 subject in the FERET database is tested by the ensemble classifier consisting of Potsu weak learners and the SVM classifier.

There are 45 weak learners available for constructing an ensemble classifier with the corresponding importance α_t. The ensemble H made by 45 weak learners has achieved 100% accuracy on the training set, in which the false negative and the false positive are both zero. However, in the testing set, 6 out of 11 images for the subject are not recognised. The overall classification error rate is 0.4%, but the false negative rate is 54.5%. Comparing with the results from the XM2VTS, the false negative rate has been reduced and the performance of classification has been improved in the FERET database. The overfitting has been considerably suppressed in the FERET database testing. When the number of positive examples increases in the training set, the degree of curse of dimensionality is reduced, and the performance of the classification is improved.

However, the false negative rate 54.5% is still rather high. In Section 5.5, since the ensemble classifier is prone to overfitting, the best solution should be feature selection by AdaBoost algorithm and classification by SVM. So the SVM classifier is used to train the training data with these 45 features and test the performance on the testing dataset. Feeding the 45-dimensional training data into non-linear SVM with a polynomial kernel, the SVM classifier is well trained. In the testing, the false negative rate is zero, which means all positive examples are recognised correctly. The false positive only occurs 2 times out of total 1347 negative examples. These two images are 00516_940519_fb and 00588_940307_fb shown in Figure 5.15. Interestingly, both individuals are oriental. It may indicate

(a) 00516_940519_fb (b) 00588_940307_fb

Figure 5.15: The two false positive occur in the testing dataset.

that all oriental faces share similar characteristics on facial appearance. In general, the performance of classifier is greatly improved by using SVMs. Therefore, it proves that the AdaBoost algorithm is appropriate for feature selection, but the

SVMs are superior than AdaBoost in classification.

5.7.5 Summary of FERET Testing

In this section, experiments are performed on the FERET face database. To obtain the inner face subimages, pre-processing techniques are applied on the original face images. The significant features are selected by the Potsu weak learner based AdaBoost. With these selected features and corresponding weak learners, an ensemble classifier is constructed. The performance of ensemble classifier on the FERET face database is better than the performance on the XM2VTS face database, but the performance is not ideal. An SVM classifier is trained on the training set with respect to selected features. The recognition rate of the SVM classifier achieves 99.99% with only two false positives out of 1347 negative examples. It is concluded that the Gabor-Boosting algorithm combined with an SVM classifier can achieve high accuracy in face verification under conditions of frontal-view grayscale images, consistent illumination and a sufficient number of positive examples.

5.8 Summary

In this chapter, a novel weak learner - Potsu weak learner for AdaBoost training is presented. The Potsu weak learner satisfies the requirement of AdaBoost, which demands the minimal error over the training set. The Potsu weak learners are fast due to the simple perceptron prototype, and it is accurate when there is a large number of training examples. In the experiments on the XM2VTS and the FERET face databases, significant features are selected by the Potsu weak learner based AdaBoost training. However, the classification made by the ensemble classifiers in AdaBoost illustrates poor performance which is explained as the overfitting phenomenon. The overfitting is caused by two sources of factors: inferior training dataset and the classifier itself. In the inferior training dataset, the number of positive examples is much less than the number of features in classification, so that it leads to curse of dimensionality and selection bias. In the classifier, the AdaBoost is prone to overfitting due to its over-emphasis on noisy data and un-optimised weights. To suppress overfitting, cross-validation is used to improve the training dataset, and SVM classifiers are chosen as the classification solution instead of

ensemble classifiers. Both methods illustrate the improvement on the performance of classification. The experiments suggest that AdaBoost is appropriate for feature selection, while classification can be done by using an alternative classifier, such as SVM. Also, the Potsu weak learners based AdaBoost algorithm demonstrates superior performance over the AdaBoost algorithm with FLD weak learners.

Chapter 6

Multi-Class Gabor-Boosting

In the previous chapter, the introduction, the full explanation, the methods, and the performance evaluation of the two-class Gabor-Boosting classification are given. In this chapter, multi-class Gabor-Boosting classification is presented. Face recognition can be interpreted as either a way of binary classification, or a way of multi-class classification. In the way of binary classification, face verification is to compare an encounter face image with a given identity to decide the true person or impostor. In the way of multi-class classification, face identification is to find out the real identity of an encounter face image. To recognise a face image in a face database containing multiple subjects, it is expensive to build many binary classifiers for each subject in the way of binary classification. As the number of subjects goes up, the computational cost will be increased exhaustively. However, using multi-class techniques, it is unnecessary to build multiple subject-specific classifiers for multiple subjects, and only one classifier can tell which class the encounter face image is.

The following section introduces the multi-class AdaBoost and its two versions. Section 6.2 gives multi-class Gabor-Boosting algorithm for face recognition, in which a multi-class weak learner - mPotsu is proposed. Finally, there is a summary on the multi-class Gabor-Boosting algorithm in Section 6.3.

6.1 Multi-Class AdaBoost

AdaBoost has been a very prosperous approach for solving the two-classification problem. AdaBoost is extended to the multi-class case in which there are more

than two possible classes. There are two extensions of AdaBoost to the multi-class case.

6.1.1 AdaBoost.M1

The first extension is called **AdaBoost.M1** which is the most direct way to perform the classification. In AdaBoost.M1, after a weak learner is trained on the given training data, the weak learner is able to predict one of the k possible classes or labels on each example. The weak learners are required to have the prediction error less than $1/2$ with respect to the weights of examples. If the requirement is met, the error of the ensemble classifier will be decreased exponentially [47], as well as in the binary AdaBoost. However, the requirement should be much more restricted on the multi-class case than on the binary case. In the binary case, since the number of classes $k = 2$, the error is required to be smaller than $1/2$, but in the multi-class case, since the number of classes $k \geq 3$, the error should be smaller than $1/k$. Obviously, $1/2$ is greater than $1/k$ when $k \geq 3$. However, if the requirement is set to $1/k$, many weak learners which could improve the performance of the ensemble classifier will be rejected. Therefore, in the multi-class case, it is tolerant that the error rate can be slightly greater than $1/2$.

The algorithm for AdaBoost.M1 is shown in Table 6.1, and differs only slightly from the original AdaBoost in Table 3.1. The main differences is that, the binary weak learner h_t is replaced by the multi-class weak learner mh_t, also the error ε_t is evaluated by summing the examples' weights which have been classified falsely.

To improve the accuracy of AdaBoost.M1, it is possible to allow multi-class weak learners trained on each example x not only to generate predicted class label $mh(x) \in Y$, but also give a "confidence" $\kappa(x) \in [0, 1]$. In the ensemble classifier $MH(x)$, a better result can be obtained by combining these "confidence" $\kappa(x)$ with the corresponding weight ω_t.

6.1.2 AdaBoost.M2

The second version of multi-class AdaBoost is an extension from AdaBoost.M1. This version of multi-class AdaBoost is called AdaBoost.M2, which enhances the communication between the boosting algorithm and the weak learner. The AdaBoost.M2 is extended from AdaBoost.M1 from two perspectives.

Table 6.1: The AdaBoost.M1 algorithm

1: Given the training set $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the data of the ith example, and $y_i \in Y = \{1, \ldots, k\}$.
2: Initialise the weights $\omega_{1,i} = \frac{1}{n}$ for each example (x_i, y_i).
3: **for** $t = 1, \ldots, T$ **do**
4: Train a multi-class weak learner mh_t using the weights $\omega_{t,i}$.
5: Get the multi-class weak learner with error

$$\varepsilon_t = \sum_{i=0}^{n} \omega_{t,i} \text{ if } mh_t(x_i) \neq y_i \tag{6.1}$$

6: Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$
7: Update the weights

$$\omega_{t,i} = \omega_{t,i} \times \begin{cases} e^{-\alpha_t} & \text{if } mh_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } mh_t(x_i) \neq y_i \end{cases} \tag{6.2}$$

8: Normalise the weights

$$\omega_{t+1,i} = \frac{\omega_{t,i}}{\sum_{i=1}^{n} \omega_{t,i}} \tag{6.3}$$

where the normalisation can be expressed by a normalisation factor Z_t so that all $\omega_{t+1,i}$ will keep a probability distribution.
9: **end for**
10: The final "strong" (ensemble) classifier:

$$MH(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \alpha_t mh_t(x) \tag{6.4}$$

Firstly, the multi-class weak learners give output in a form of a k-element vector, rather than a single arbitrary class label in Y. Each element in a k length vector represents a "degree of belief" that indicates how the example would have the correct label as the label y. The elements with large values which are close to 1 indicate the examples more likely to have the correct label with the corresponding y. Likewise, the elements with small values which are close to 0 correspond to the labels to be implausible. The questionable labels may be assigned to around 1/2.

Secondly, AdaBoost.M2 not only focus on those examples which are "hard" to deal with, but also focus on some classes which are "hard" to discriminate. In AdaBoost.M2, not only each example is given a weight, but also each class is given a weight which indicates how difficult the subject to be discriminated from others. The weak learners is evaluated by a more sophisticated error measure rather than the usual prediction error. The more sophisticated error is called the pseudo-loss. The pseudo loss varies from example to example, and from one iteration to the next iteration. On each iteration, the pseudo-loss is calculated along with the distribution on the examples by the boosting algorithm.

The AdaBoost.M2 is developed based on above two considerations, and achieves boosting if each weak learner has pseudo-loss slightly better than random guessing, *i.e.*, the pseudo-loss must be less than 1/2.

6.2 mPotsu weak learner based multi-class Gabor-Boosting algorithm

In multi-class AdaBoost, a weak learner in the training is required to work for a multi-class classifier. The multi-class weak learner used in this section is called **mPotsu**, which is a multi-class extension of the binary Potsu weak learner.

6.2.1 From Binary to Multi-Class

Classifiers are often developed to distinguish between just two classes. In Bayes classifier, the probability density function is estimated on each class. The classification can directly be derived from the posterior probabilities by taking the difference between the posterior probabilities of two classes. In some other cases, the classification is not based on density estimation, but on fitting a discriminant

which separates two classes, like FLD, perceptron, and SVM. If the classification is to fit a discriminant, a multi-class classifier can be built on several binary classifiers.

When it is assumed that the output of classifier is binary decision, there are two basic strategies to construct k-class classifiers. In the first strategy, given an example x, a classifier just needs to tell x belonging to a class j or not. The classifier between one class and rest $k-1$ classes is trained. This is called *one-against-rest* strategy. In the second strategy, the classifier is trained for each possible pair of classes. Every possible pair between classes is collected. Based on each pair (i, j), the classifier is trained just to tell i or j. The approach is called *one-against-one* strategy.

Examples are classified by applying a combining rule on a set of decisions. One common rule is to use voting, where the example is labelled to the class with the highest number of votes. In the *one-against-rest* strategy, a class can have 1 or 0 votes, so that an example could be labelled to multiple classes. In the *one-against-one* strategy, a class has accumulate votes.

To construct a multi-class classifier, either the *one-against-rest* or the *one-against-one* can be chosen. In the *one-against-rest*, k binary classifiers are built. Due to the pair-wise structure of *one-against-one* strategy, $k(k-1)/2$ binary classifiers are needed. It should be considered that the multi-class AdaBoost is also a time-consuming process. More binary classifiers involved in the construction will cost more computational time. Hence, the *one-against-rest* strategy is adopted for constructing multi-class weak learners.

6.2.2 mPotsu

A multi-class weak learner - **mPotsu** (multi-class Potsu) is built on the *one-against-rest* strategy. In the training data of the XM2VTS database, there are 200 subjects, *i.e.*, 200 classes along with 4 images per subject. In one mPotsu weak learner, there are 200 Potsu binary weak learners. Each Potsu weak learner is built on a particular subject against rest subjects. For example, the first Potsu weak learner is built on the first client against all other clients, which is called $C1$[1], and the second Potsu weak learner on $C2$, and so on. Figure 6.1 shows the diagram of the mPotsu weak learner. Each Potsu weak learner is also built on

[1]The C represents the "client", and the digit 1 represents the first client against others.

Figure 6.1: mPotsu weak learner containing 200 binary Potsu weak learners.

a single Gabor wavelet feature, such that the training data in each Potsu weak learner is one-dimensional. The training data is the same in the training for all binary Potsu weak learners, but the only difference is the labelling. For the subject $C1$ (the first client), the examples belonging to $C1$ are labelled as 1, *i.e.*, the positive examples, and examples other than the first client are labelled as 0, *i.e.*, the negative examples.

Each binary Potsu weak learner in mPotsu gives one output indicating either belonging to the subject or not, so that mPotsu will give 200 outputs of weak learners. The decision making of mPotsu is to combine these 200 decisions. Ideally, if an mPotsu weak learner is built on a very excellent feature which distinguishes most subjects, there will be only one positive output and the output of rest 199 Potsu weak learners are negative. In the ideal case, the mPotsu weak learner gives out only one plausible class or label. However, in most real situations, the mPotsu weak learner has multiple positive outputs. For example, given an input example, an mPotsu weak learner may output multiple labels, such as $C1$, $C2$, $C3$ and $C4$. It indicates that the example is recognised as the subject $C1$, $C2$, $C3$ and $C4$. The true identity of the example is possibly among these four subjects. In the case of multiple output labels, it is rare to output only one plausible class for an mPotsu weak learner.

In this thesis, the mPotsu weak learner outputs a label vector η instead of one single label.

$$\eta = (y_1, y_2, \ldots, y_{200}) \tag{6.5}$$

The label vector η contains 200 components. Each component y_k is corresponding to each subject k, and has a boolean value 1 or 0. The boolean value 1 indicates

recognition acceptance on the corresponding subject k, and the value 0 indicates recognition rejection on the corresponding subject k. By introducing the label vector η, the mPotsu weak learner is able to coexist multiple decisions.

After an mPotsu weak learner is trained with a given training data[2], an evaluation method assesses the performance of mPotsu weak learner. The performance is evaluated by the sum of training errors with respect to the corresponding weights on training examples. In this thesis, a loose method on classification is applied. Given an example with its label l, the mPotsu weak learner outputs a label vector η. If the label l appears in the multiple output labels from the label vector η, the classification will be considered as being classified correctly, otherwise as false classification. For example, an example x with its true label $C1$ is feed into an mPotsu weak learner, and the mPotsu gives the plausible label vector η $(1, 1, 1, 1, 0, \ldots, 0)$. The first element in the vector η is 1, *i.e.*, the mPotsu has recognised x as $C1$, the classification is considered as true, regardless that the mPotsu also recognises x as $C2$, $C3$ and $C4$. Since the mechanism of classification is quite loose, the accuracy of the mPotsu is low through assigning multiple labels to an example. However, since AdaBoost can boost the performance from a set of weak classifiers, the low accuracy on mPotsu is accumulated into higher accuracy. After many iterations, the performance of mPotsu can be enhanced.

6.2.3 Multi Gabor-Boosting Algorithm

The multi-class Gabor-Boosting algorithm is a variant from AdaBoost.M1 with some improvements on multiple labels from AdaBoost.M2. The algorithm of multi-class Gabor-Boosting algorithm is given in Table 6.2. In each iteration, a significant feature is selected with the lowest error ε_t. After the multi-class Gabor-Boosting training, there are T significant features. Each feature can be built into an mPotsu weak learner $mh(x)$ which outputs a label vector η indicating multiple plausible labels. Given an example (x_i, y_i), $y_i \in mh_t(x_i)$ means that the label vector η_t from the mPotsu weak learner contains the corresponding label y_i, *i.e.*, $y_i \in mh(x_i)$ specifying the training example (x_i, y_i) is classified correctly. The error ε is calculated by summing the weights of misclassified examples. The importance α_t for each significant feature is evaluated by the lowest error ε_t in the iteration t. The

[2]Actually, the training of mPotsu weak learner involves the training of 200 Potsu weak learners. The training is based on each individual Potsu weak learner, but not on mPotsu. The mPotsu provides a mechanism to combine the multiple outputs.

Table 6.2: Multi-class Gabor-Boosting algorithm for feature selection and classification

1: Given example $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the data of the ith example, which are contributed by k features $\{j_1, \ldots, j_k\}$, and $y_i \in Y = \{1, \ldots, c\}$ for c subjects (classes)
2: Initialise the weights $\omega_{1,i} = \frac{1}{n}$ for each example (x_i, y_i).
3: **for** $t = 1, \ldots, T$ **do**
4: Normalise the weights, $\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{i=1}^{n} \omega_{t,i}}$ so that ω_t form a probability distribution.
5: **for all** $\{j_1, \ldots, j_k\}$ **do**
6: Train an mPotsu weak learner mh_j built with one single feature j with the weights $\omega_{t,i}$.
7: The error is calculated as $\varepsilon_j = \sum_{i=1}^{n} \omega_{t,i}$ if $y_i \in mh_t(x_i)$.
8: **end for**
9: Choose the optimal mPotsu weak learner mh_t with the lowest error ε_t from all mh_j.
10: Select the corresponding feature j_t of the mPotsu weak learner mh_t as a significant feature.
11: Remove the feature j_t from the feature set $\{j_1, \ldots, j_k\}$.
12: Update the weights $\omega_{t+1,i} = \omega_{t,i}\beta_t^{1-e_i}$, where $e_i = 0$ if $y_i \in mh_t(x_i)$ and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
13: Choose the importance $\alpha_t = \log \frac{1}{\beta_t}$, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
14: **end for**
15: The final "strong" (ensemble) classifier

$$MH(x) = \sum_{1}^{T} \alpha_t mh_t(x)$$

where $MH(x)$ outputs a sum of label vectors η_t with the importance. The label for the examples x is the one with the highest vote in $\sum_1^T \alpha_t \eta_t$.

ensemble classifier is a linear combination of all collected mPotsu weak learners with corresponding importance α_t. Each mPotsu weak learner mh_t outputs a label vector η_t which is either 1 or 0 in each component. The final combined label vector $\boldsymbol{\eta}$ is the sum of each label vectors η_t with the corresponding importance α_t, as

$$\boldsymbol{\eta} = \sum_{1}^{T} \alpha_t \eta_t \tag{6.6}$$

The final label vector $\boldsymbol{\eta}$ is a vector with 200 components. Each component is a real value rather than the boolean value, after each boolean label vector η_t is multiplied by a real importance α_t. The values of components are votes for the corresponding subject to be the plausible label, and the final vector is the accumulated votes with weight, *i.e.*, α_t. The index of component with the highest vote is the most plausible label.

6.2.4 Feature Selection

Feature selection on mPotsu weak learner based Gabor-Boosting algorithm is very time consuming, because each mPotsu weak learner contains 200 Potsu binary weak learners for the XM2VTS database. The computational time on training an mPotsu weak learner is equivalent to time cost on training 200 individual binary Potsu weak learners. The AdaBoost feature selection algorithm searches over the feature set exhaustively in each iteration, such that the computational time will be 200 times more than the Binary Gabor-Boosting feature selection. In this thesis, Grid Computing technology has been applied to feature selection on mPotsu weak learner based Gabor-Boosting algorithm, such that the computational cost has been reduced dramatically. The Grid computing is introduced in Appendix D.

Among various Grid computing applications, the Condor grid [158] is used in this thesis. The Condor grid is a research project at the University of Wisconsin-Madison (UW-Madison), and it was first installed as a production system in the UW-Madison Department of Computer Sciences nearly 15 years ago. The Condor is a technical workload management system for compute-intensive jobs. The Reading University Grid is a Condor Pool consisting of between 200 and 500 nodes. The aim of the Campus Grid is to provide researchers with a High Throughput Computing (HTC) resource.

In the Multi-class Gabor-Boosting training, all features are evaluated by build-

ing an mPotsu weak learner and calculating the error within each iteration. The evaluation on each feature is independent, because the result from the evaluation of the current feature does not affect the result of the next feature. Hence, the evaluation of each feature can be performed in a parallel way. There are $30,240$ features evaluated in each iteration. The whole set of features are split into 120 subsets, and each subset contains 252 features. The input data for each job is the face images from the XM2VTS face database. The output data for each job is a text file which records the feature and the error of the corresponding mPotsu weak learner. After all 120 jobs are finished, an `update` program is to combine all log files into one large file, find the minimum error and the corresponding feature, and update weights for the next iteration. The diagram of the Multi-class Gabor-Boosting training is displayed in Figure 6.2.

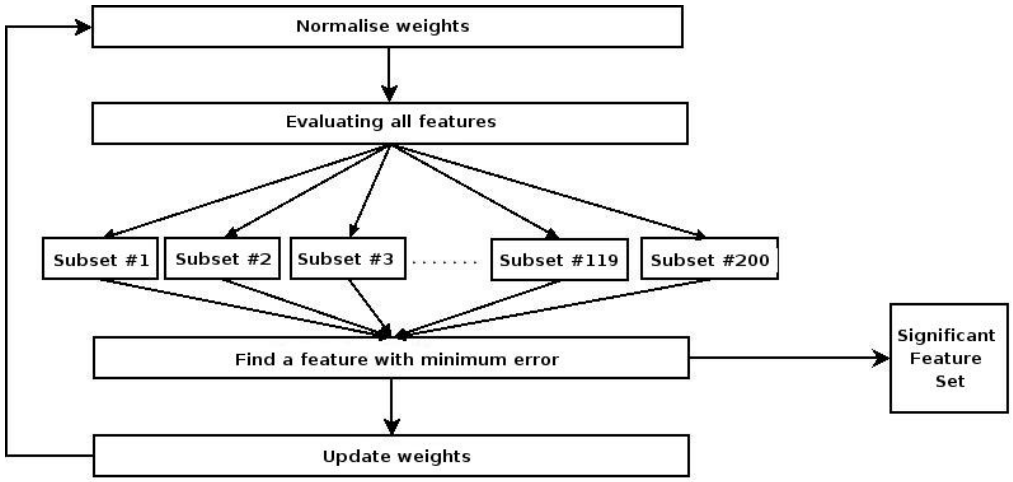
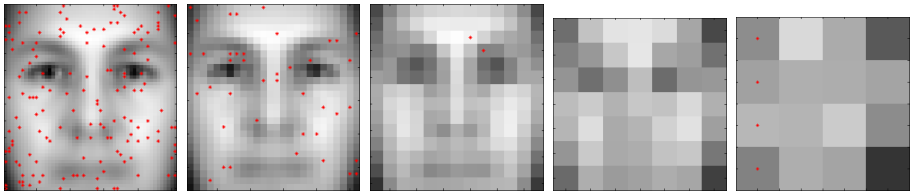


Figure 6.2: The Multi-class Gabor-Boosting training on the Condor Grid

If only one *Intel Pentium 4* 2.8 GHz machine is used for the training, each iteration consisting of $30,240$ features will take $332,640$ seconds (30240×11 seconds), *i.e.*, 92.4 hours. To select 200 features from the Multi-class Gabor-Boosting training, the 200 iterations will take 770 days, *i.e.*, roughly two year and one month. It is assumed that on the Condor Gird, every node has the same computational capability equal to the *Intel Pentium 4* 2.8 GHz machine. Theoretically, each iteration will take 46.2 minutes. To select 200 features, 200 iterations will be finished in $9,240$ minutes (46.2×200), *i.e.*, less than one week. Actually, the Condor Grid

computation takes more time, but comparing to over two years computation, the Condor Grid still displays its advantages over the conventional computation.

After 200 iterations of the mPotsu weak learner based Gabor-Boosting, 200 features are selected as shown in Figure 6.3. These 200 features are general for all



(a) $\nu = -1$ (b) $\nu = 0$ (c) $\nu = 1$ (d) $\nu = 2$ (e) $\nu = 3$

Figure 6.3: The 200 features are selected from mPotsu weak learner based Gabor-Boosting algorithm and are projected on an mean face image.

200 subjects in the XM2VTS face database. The feature projection plane in Figure 6.3 is a mean face image generated from the training set. Because the **Scaling** feature pre-selection scheme is applied before the feature selection, most features are associated with lower spatial frequencies ν.

The first nine features are displayed in Figure 6.4 from the mPotsu weak learner based Gabor-Boosting algorithm. Eight from nine features are with the spatial frequency $\nu = -1$, and only one feature (the 8th feature) is with the spatial frequency $\nu = 0$. Each feature is referenced according to the mean face image.

Figure 6.4: The first nine selected features from mPotsu weak learner based Gabor-Boosting algorithm

The first selected feature is in a place between two eyebrows on the face image, which responses to the difference between eyebrows. Eyebrows are proved to be the most important for recognition among the different facial features. In this section, the first Gabor wavelet feature appears in the middle of eyebrows, which makes accordance with this human vision finding in [148].

6.2.5 Classification

A multi-class classifier is able to give a label of multiple classes rather than only positive or negative. Hence, the multi-class classification mechanism is better used in face identification, but not in face verification. In this section, the XM2VTS face database is used to train and test a classifier for the multi-class classification purpose. In the training data, 200 different subjects contain four images for each one. The other four images for each subject remain for the testing purpose. There are 800 images in the training set as well as the testing set.

Ensemble classifier

With 200 selected features and corresponding weights for each iteration, 200 mPotsu weak learners are constructed. These 200 mPotsu weak learners are combined into an ensemble classifier with the corresponding importance α_t. Given a testing example, each mPotsu weak learner gives a label vector η_t. The ensemble classifier combines the label vector η_t with the corresponding α_t and outputs the final label vector $\boldsymbol{\eta}$. The plausible class label is the highest one in the final label vector $\boldsymbol{\eta}$. The ensemble classifier is tested with the training data and the testing data. However, the combined mPotsu ensemble classifier demonstrates low recognition rate. It indicates that the ensemble classifier is not appropriate for multi-class classification. To perform multi-class classification correctly, other classifiers are retrained with the training data, and better performance is achieved.

SVM classifier

The classification is then performed by the SVM classifier as applied in Chapter 4 and Chapter 5. After 200 features are selected from the multi-class Gabor-Boosting algorithm, the training data is learnt by an SVM with a polynomial kernel of degree of 3. The well built SVM classifier with 200 features has evaluated the training set and the testing set, respectively. In the training set, 762 examples out of 800 examples are classified correctly by the SVM classifier. However, in the testing set, only 467 out of 800 examples are classified correctly. The acceptance rates in the training set and the testing set are 95.25% and 58.38%. As mentioned in Chapter 5, the SVM classifier performs relatively well in the training set.

To find out how many features are sufficient for face recognition to obtain a better performance, the number of features for SVM is adjusted. Figure 6.5 shows

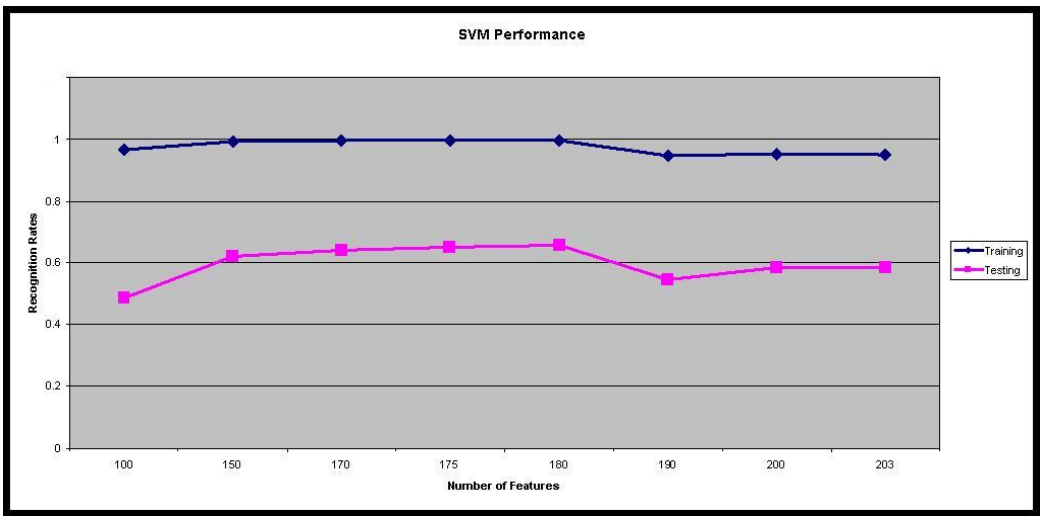


Figure 6.5: The SVM performance with respect to the number of features

the recognition rate against to the number of features. The number of features are sampled at 100, 150, 170, 175, 180, 190 and 200. In the result, it is expected that the performance on the training set is better than the performance on the testing set. The recognition rates on the training set are above 0.8 and close to 1.0. In the training set, when the number of features is increased from 100 to 180, the recognition rates are increasing. When the number of features reaches 180, the maximum recognition rate (99.75%) has been achieved in the training set. When the number of features is more than 180, the recognition rate decreases. The recognition rates in the testing set are around 0.6. In the testing set, when the number of features are increased from 100 to 180, the performance is boosted. When the number of feature reaches 180, the recognition rate has achieved its highest one, *i.e.*, 65.75%. With more than 180 features, the performance of SVM classification is decreased. It indicates that for face recognition using SVM classifiers, the appropriate number of features should be around 180. Although 200 features are obtained in feature selection, only the first 180 features are sufficient to represent each individual.

k-NN classifier

As non-linear classifiers, k nearest neighbour (k-NN) classifiers perform very well in the domain of multi-class classification. A k-NN classifier is trained with the training set with respect to the selected features. The well built k-NN classifier is

able to recognise different subjects. The assumption for the k-NN is that example points near an unclassified point should imply the class of that point. In the k-NN classifier, an example is classified by using the class of the nearest examples with the known class. Table 6.3 shows the algorithm of a k-NN classifier to classify a testing example. To determine how to finding the k nearest examples, the choice

Table 6.3: The algorithm for k-NN classifying testing examples.

1: Given a testing example x
2: determine the k training examples that are nearest, $x_1, \ldots, x_k$;
3: determine the class c that has the largest number of examples in the set $x_1, \ldots, x_k$;
4: classify x as c.

of distance is needed to consider. If obviously the features in examples are in the same type, for instance, different lengths, any usual metric is good enough to measure the distance, such as *Euclidean distance*. But if the features in examples are not in the same type, one possible solution to use a covariance estimate to compute *Mahalanobis distance*. In the 2-D weak learner, each example contains a Gabor feature pair which consists of two features. The features are apparently in the same type, so the Euclidean metric is used to measure the distance between examples.

In the k-NN classifier, the key problem is to determine the number of the nearest neighours k. The choice of k depends upon the data so that a good k can be selected by heuristic techniques. To obtain the best performance, different k and different number of features are used. Table 6.4 displays the recognition rates in the training set and the testing set with various k and the number of features. There are two findings from Table 6.4.

The first finding is, with k increasing, the recognition rates for both the training set and the testing set drop. It is because the k-NN classifier determines the class by voting. Among the k nearest neighbours, the class with the highest vote is taken as the plausible one. The number of examples for each class is only four, such that the cluster of each class is not very representative. The nearest neighbours of an example are not only examples from the same class, but also examples from other classes. If more nearest neighbours are counted in, more examples from

Table 6.4: The performance of k-NN classifier with respect to the number of features and the number of the nearest neighbour k.

Number of Features	$k = 1$		$k = 5$		$k = 10$		$k = 32$	
	training	testing	training	testing	training	testing	training	testing
200	99.88%	58.88%	76.50%	52.25%	68.13%	50.00%	43.75%	32.25%
190	99.63%	55.75%	77.00%	49.88%	65.38%	48.13%	42.63%	30.50%
180	100%	64.38%	79.75%	55.00%	67.00%	51.13%	42.00%	33.00%
175	100%	64.50%	79.38%	55.25%	66.88%	50.50%	41.88%	33.13%
170	100%	64.50%	79.00%	54.50%	66.50%	51.38%	41.63%	33.50%
150	100%	63.38%	76.00%	52.75%	64.38%	47.13%	40.88%	32.13%
100	100%	47.63%	68.38%	41.38%	56.63%	38.38%	36.00%	26.50%
50	99.25%	33.13%	54.63%	27.88%	48.75%	27.00%	32.63%	20.00%
20	94.13%	21.00%	44.88%	17.88%	38.75%	18.88%	24.50%	14.75%

other classes will be brought as the neighbours. When the number of examples from other classes exceeds the number of examples from the same class, the misclassification will occur. Hence, the best performance only occurs when $k = 1$, *i.e.*, the example is simply assigned to the class of its one nearest neighbour.

The second finding is, with the number of features increasing, the recognition rates on the training set and the testing set are increasing firstly, then decreasing. The recognition rates are achieved on its optimum when there are between 170 and 180 features used in the k-NN classifier. When the number of features is more than 180, the recognition rates drop slowly. It indicates that the best number of features used in the k-NN classifier should be between 170 and 180.

Among various combinations with the number of features and nearest neighbours, the best performance is achieved at 100% recognition rate in the training set and 64.50% recognition rate in the testing set with 170 to 175 features and $k = 1$. Compared with the performance of SVM classification, the k-NN classification is slightly better in the training set, but slightly worse in the testing set.

Loose Face Recognition

Both SVM and k-NN classifiers have been implemented for the multi-class face recognition. The recognition rates on both classifiers are not ideal, since multi-class classification is a very difficult pattern recognition task. With no prior knowledge

on these 200 individuals, even a human vision system may not able to recognise these faces correctly. However, if the multi-class face recognition has a loose requirement, the recognition performance will be increased. The loose requirement: given a face image, a multi-class classifier will give a number of most possible classes rather than only one exclusive class. If the true identity of the face is among these possible classes, the classification is considered as correct recognition. Giving a number of most possible classes, the k-NN classifier is adopted as the multi-class classifier. The nearest examples are considered as the examples from the same class. The k nearest examples may contain multiple classes.

This kind of loose control face recognition scenario is tested by the XM2VTS face database. To test how many nearest neighbours can improve the recognition performance, k is assigned to 5, 10 and 32, which are small, moderate and maximum numbers of neighbours. Also, to find how many features are sufficient for the purpose of multi-class face recognition, the number of features is assigned to 150, 170, 175, 180, 190 and 200 respectively. Both the training set and the testing set with 800 images are tested. With the training set, the recognition rates have been achieved 100% no matter how the number of nearest neighbours and the number of features are varied. With the testing set, the recognition rates under different combinations of nearest neighbours and features are given in Table 6.5. Similar to

Table 6.5: The recognition rates of loosely controlled multi-class face recognition.

Number of Features	$k = 5$	$k = 10$	$k = 32$
200	78.25%	84.50%	91.88%
190	76.50%	83.88%	91.88%
180	81.00%	86.00%	92.50%
175	81.00%	85.88%	92.38%
170	80.75%	85.88%	92.38%
150	79.88%	84.75%	92.38%

the classification in SVM and k-NN, the best recognition performance is obtained with 180 significant features. When more nearest neighbours are counted for possible classes, the recognition performance is increased. When only five nearest neighbours are available in the possible classes, the optimal recognition rate has achieved at 81%. When ten nearest neighbours are counted for the possible classes, the optimal recognition rate is 86%. When 32 nearest neighbours are available,

the optimal recognition has achieved at 92.50% which is a rate for 200-class classification[3]. From Table 6.5, it can be found that with $k = 32$ and 180 features, such a loosely controlled multi-class face recognition has achieved the accuracy of 92.5%.

The possible application for such loosely controlled multi-class face recognition algorithm is a face identification system with humans' cooperation. For instance, given an unknown face image, the face identification system can output a set of individuals from a database who "look like" the unknown face, then system operators can select the most plausible one and ensure the identity of the unknown face. The face identification system provides a set of possible candidates for a testing face, but the final decision of identification is made by human beings rather than machines. Such a kind of systems combine machine vision perception and human vision perception. In the current state of the arts in computer vision, machine vision perception is much weaker than human vision perception. However, the superiority of machine vision perception is its speed and consistency. Hence, the system utilises the "weak" machine vision perception to provide a small number of possible candidates out of a large number of persons in the database, and utilises more advanced human vision perception to make a complicated decision among these small set of individuals.

6.3 Summary

In this chapter, the multi-class Gabor-Boosting algorithm for face recognition is presented. The multi-class AdaBoost algorithm is firstly introduced to the multi-class task. In the multi-class AdaBoost, a multi-class weak learner is required, such that the mPotsu weak learner is proposed. The multi-class Gabor-Boosting algorithm has been used to select some important features for the multi-class classification. The classification is committed by the ensemble, the SVM and the k-NN classifiers. The ensemble classifier has lower performance than SVMs and k-NN in the multi-class classification. It indicates that AdaBoost is not appropriate for classification, but only for feature selection. The performance of the SVM is

[3]In practice, the number of nearest neighbours does not equal to the number of possible classes. When $k = 5$, the number of possible classes is varied from 2 to 5. When $k = 10$, the number of possible classes is varied from 4 to 10. When $k = 32$, the number of possible classes is varied from 15 to 29.

slightly better than the k-NN in the testing set. The multi-class classification is a challenging pattern recognition task, so that a loosely controlled classification mechanism is introduced. The loosely controlled face recognition combines the machine vision and the human vision, and gives very high recognition rates.

Chapter 7

Conclusions and Future Work

This chapter concludes the work of the Gabor-Boosting face recognition, and presents the future work. The first section gives significant findings and achievements throughout the thesis. The second section discusses the observations in the experiments. Following by the conclusion in the third section, the final section proposes some directions for future research aimed at solving the remaining problems.

7.1 Achievements

In this thesis, the Gabor-Boosting face recognition algorithm is proposed, developed and tested. From experiments on the Gabor-Boosting face recognition algorithm, some important findings were:

- Face recognition can be introduced as a two-class classification problem. A set of images in training set are divided into two classes, *i.e.*, clients or impostors. A client is the given individual with claimed identity, and impostors are all other persons except the client. This effectively avoids the difficulties of building multi-class classifiers in face recognition.

- The Gabor wavelet transform is beneficial for feature extraction, and reflects salient changes between pixel values. This makes the algorithm robust against global luminance varying between images.

- The number of Gabor wavelet features can be reduced by neglecting some

high frequency features due to small local variations between high-frequency feature's responses.

- AdaBoost is excellent at selecting significant features from a large set of features.
- The computation in AdaBoost training can be optimised by setting criteria to reject some features in each iteration. By optimising the AdaBoost training, the computational time has been significantly reduced. With *optimisation 1*, the computational cost has been reduced to around 50% compared to the cost without any optimisation. The *optimisation 2* has produced a saving on the computational cost of 3.44% to 5.55% of the computational time without any optimisation.
- The Potsu weak learner satisfies the requirement of AdaBoost, which demands the minimal error over the training data. The Potsu weak learner is fast and accurate due to the simple perceptron prototype and large number of examples.
- The AdaBoost training based on the Potsu weak learners displays its superior performance over the AdaBoost algorithm on other weak learners, such as FLD weak learners.
- The ensemble classifier made from AdaBoost training is not appropriate for classification, which suffers from the overfitting in the testing result. It is because of inferior training data and classification mechanism of ensemble. Overfitting can be by-passed by either cross-validation or SVM evaluation.
- SVM is very good for classification with respect to selected features from AdaBoost training. The overfitting effect is suppressed by using SVMs through fewer support vectors and optimised weighting schemes.
- The multi-class Gabor-Boosting face recognition algorithm is based on multi-class weak learner - mPotsu. The mPotsu weak learner consists of multiple Potsu binary weak learners, and outputs a label vector instead of only one exclusive label.
- The multi-class Gabor-Boosting training selects significant features for the multi-person recognition purpose. The training algorithm is a variant from AdaBoost.M1 with the improvement on multiple labels from AdaBoost.M2.

- When using face recognition based on multi-class classifiers, it is better to have loose requirements with k-NN classifiers, which give a number of most plausible individuals rather than only one exclusive individual. When 10 nearest individuals are given, the recognition rate has been achieved as 84.5%.
- The computational time in computer vision, especially in training, can be greatly reduced by Grid computing technology.

7.2 Observations

One phenomenon observed from the experiments is that the ensemble classifier built from AdaBoost training suffers from overfitting. Because AdaBoost over-emphasises noisy examples and has un-optimised weighted voting, it is apt to overfit the training data. By giving a small size of data, the ensemble quickly recognises all positive and a few negative examples in the first few iterations, then excludes these negative examples in following iterations. In addition, curse of dimensionality and selection bias in the dataset lead to overfitting. The overfitting could be by-passed by using cross-validation or using alternative classifiers. To suppress overfitting, the training data is trained by SVM classifiers with the selected features. The Gabor-Boosting face recognition algorithm has achieved an extremely strong recognition performance. Therefore, ensemble classifier from AdaBoost training is not a good choice for classification due to the overfitting effect in this case, but AdaBoost is considered as an excellent algorithm for feature selection.

7.3 Conclusions

In the thesis, a highly accurate appearance-based grey scale front-view face recognition algorithm - Gabor-Boosting face recognition, is presented. The strong recognition performance in the face recognition is highlighted by three key leading edge techniques - *Gabor wavelet transform*, *AdaBoost*, *Support Vector Machine (SVM)* in computer vision, machine learning, and pattern recognition. The Gabor wavelets are of similar shape as the receptive fields of simple cells in the primary visual cortex, and demonstrate significant potential in biometrics such as iris recog-

nition and face recognition. AdaBoost is one of the most successful and popular learning algorithms. The classification is designed to construct a "strong" classifier from "weak" learning algorithms. AdaBoost has also been successfully applied in face detection to select features representing human faces. The SVM is a powerful learning algorithm for solving a variety of learning and function estimation problems, such as pattern recognition and regression estimation. The SVM provides non-linear classification by mapping the input into high-dimensional feature space where a special type of hyperplane is constructed. The overall recognition performance is benefited by using these three key components. The Gabor wavelet transform extracts features which describe texture variations in human faces. The AdaBoost algorithm selects the most significant features which represent different individuals. The SVM constructs a hyperplane for classification with respect to selected features.

The Gabor-Boosting face recognition algorithm is robust under small number of example images per subject and selection-bias in training data. In the situation of extremely small number of positive examples, such as in the XM2VTS database, the system has achieved nearly 0 false positive rate and moderate false negative rate. In the general situation of few positive examples, such as in the FERET database, the system performance has been improved with nearly 0 false negative rate and 0 false positive rate, *i.e.*, approximately 100% recognition accuracy under conditions of a sufficient number of examples per class. Potential applications of the algorithm are possibly in highly secure face authentication systems dedicated to a small group of clients. The systems thus have "no false detections on impostors" and "an appropriate acceptance detection rate on clients".

Therefore, it is clear that the successful AdaBoost algorithm in face detection can be transplanted into face recognition, however some modifications are needed to apply due to different specifications.

- Face recognition can be performed in the face verification scenario. In face verification, the system only needs to distinguish clients and impostors, so that a multi-class classification issue is changed into two-class classification issue.

- The feature extraction is improved by Gabor wavelet transform rather than using Haar features. The Gabor wavelet features provide more precise and accurate description on human face than Haar features, so that they are

very suitable for the face recognition purpose.

- In the training dataset, the number of client examples, *i.e.*, positive examples, should be sufficiently large. Otherwise, overfitting will be happened.
- The ensemble classifier built on weak learners suffers from overfitting which reduce the performance, but the alternative SVM classifier gives nearly perfect performance with the selected features. Hence, the AdaBoost training is excellent in feature selection, whilst the classification is better done by SVMs.

7.4 Future work

Future work could be extended in many aspects, such as face pre-processing, investigation on the training dataset, exploration of more types of features, improvement on feature selection and more advanced classification techniques, experiments on other databases.

- In face pre-processing, the coordinates of two pupils on faces are located manually or provided by database ground truth. Although this is conducted in the offline training process, it would be more desirable to make it a fully automatic process for obtaining the location of eyes on face images. Hence, automatic eye detection approaches could be implemented in the face recognition system.
- In Chapter 5, the face recognition suffers from overfitting, which is partially due to small number of positive examples in the training dataset, while the performance of face recognition has greatly increased with increasing on the number of positive examples. An investigation can be done for finding how many examples are sufficient for face recognition to gain a satisfactory recognition rate.
- Rather than Gabor wavelet features, other types of features may be explored and used in the proposed algorithm. Local Binary Patterns (LBP) [117, 4, 5] is widely used in face recognition recently, *i.e.*, LBP features representing face images. The different types of feature give comparable results which help to choose the best type of feature for face recognition.

- The feature selection is done by a discrete version of AdaBoost, in which the weak learners only give two outputs - either positive or negative. The feature selection could be improved by using a real version of AdaBoost [139], in which the weak learners are confidence-rated from a real-value space. It is believed that using a real value rather than a boolean value would improve the performance of feature selection.

- From Section 5.5.2, it is observed that there is still redundancy among the selected features. Hence, during feature selection, the redundancy between candidate features and selected features is an important factor to examine whether the candidate features would be selected. Mutual Information [146, 145] or Float Search [129, 92] could be implemented for calculating the redundancy.

- The SVM classifier could be improved by Geometric Programming [110]. A geometric framework for the SVM classification problem allows existing geometric algorithms to be directly and practically applied to solve not only separable, but also non-separable classification problems both accurately and efficiently.

- The performance of face recognition could be enhanced by classifier fusion [100], in which more classifiers (LDA, Naive Bayes, Neural Network, etc) can be built up separately, and combined into a complex classifier.

- Although the Gabor-Boosting is an appearance based approach for face recognition, some model-based approach could be adopted, such as EBGM [167] into the algorithm to improve its performance at the feature selection stage.

- In the experiments, only two databases - XM2VTS and FERET are tested. To demonstrate the performance of the proposed algorithm, other larger and more realistic databases may be used in testing, such as the Yale database [55] and the PIE database [147].

With development of more advanced computer vision technology, more powerful computational competence, and more precise image acquisition, Gabor-Boosting face recognition holds promise to make facial recognition systems more robust in

practice. Face recognition is an exciting and challenging research topic, and it is believed that there is a large market waiting for the research outcomes.

Appendices

Appendix A

PCA and Canonical Variate

In this section, two algorithms are developed to recognise human face from the ORL face database. Two algorithms are used to recognise these 40 subjects. One is PCA, another is the canonical variates. The results showed that the canonical variates algorithm has a better performance than PCA on face recognition.

A.1 PCA

The PCA (principal component analysis), also known as the Karhunen-Loeve transform [99], is a classical method in multivariate statistics. In machine vision, PCA is a good solution for feature selection. The goal of feature selection is to obtain a smaller set of features that accurately represents the original set. The smaller and new set of features will capture as much of original set's variance as possible. Also if a value of one feature can be predicted precisely from the value of the others features, this feature is clearly redundant and needs to be removed. In PCA, the data are taken as points in a very high dimensional space (which is depend on how many features are in the original set), and construct a lower dimensional linear subspace. This subspace describes the main variation of these data points from their mean data point. In other form of explaining PCA, the principle components which are orthogonal to each other from original data, are generated. Each principle component represents one dimension in the original feature space. The original data points can be projected as new points on an axis along this dimension. In this new coordinate, these new points may gather tightly or separate widely, which depends on the principle component chosen. If a few

principle components are chosen, that is a few dimensions are chosen, they span a lower dimensional subspace described before as lower dimensional linear subspace. The locations of these new points in the new linear subspace represent the new values of each features selected.

A.1.1 Generate Eigenvectors

The face images in the ORL face database are two-dimensional 92×112 arrays in grey level intensity values. Each image can be considered as a one dimensional array, so that the images become vectors which contain 10304 digits. Each digit is considered as an feature in the image array. Equivalently, an image becomes a point in 10304-dimensional space. Obviously, it is a very high dimensional feature space. The very huge space is far more enough describing human face. Therefore, it is possible to reduce the dimensionality of the high dimensional feature space.

The training set of PCA algorithm is the ORL face database. In the database, there are 40 subjects and each subject has 9 images. In the training set, there are $M = 360$ face images, and for each image, there are $N = 10304$ pixels. Each image is considered as a vector. In the training set, the face images are Γ_1, Γ_2, ..., Γ_M. The mean face is defined by

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n \tag{A.1}$$

The mean face image is shown in Figure.A.1. The difference vector describes the

Figure A.1: The mean face image over 360 face images in the ORL face database.

degree of each face differs from the mean face by

$$\Phi_i = \Gamma_i - \Psi \tag{A.2}$$

Here, the covariance matrix of the training set is defined by

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T \tag{A.3}$$

The covariance matrix is a matrix of covariances between features of a vector. It is the natural generalisation to higher dimensions of the concept of the variance. The covariance matrix measures how much two features vary together. If two features tend to vary together (that is, when one of them is above its expected value, then the other features tends to be above its expected value too), then the covariance between the two features will be positive. On the other hand, if one of them is above its expected value and the other feature tends to be below its expected value, then the covariance between the two features will be negative. The covariance matrix C is an $N \times N$ symmetric matrix which characterise the scatter of the training set. A non-zero vector u is an eigenvector of the covariance matrix C, if it satisfies the condition:

$$Cu_k = \nu_k u_k \tag{A.4}$$

where the corresponding eigenvalues are ν_k.

The size of covariance matrix C is $N \times N$ ($N = 10304$). It is extremely huge matrix, and the computation cost is tremendous. To reduce the cost, Turk and Pentlands approach [159] is adapted to calculate an alternative smaller covariance matrix L rather than original one. In Equation A.3, the matrix C can be decomposed as

$$C = AA^T \tag{A.5}$$

where the matrix $A = [\Phi_1, \ldots, \Phi_M]$. The eigenvector v_i is considered as such that

$$A^T A v_i = \mu_i v_i \tag{A.6}$$

The both side of the above Equation is pre-multiplied by A, then

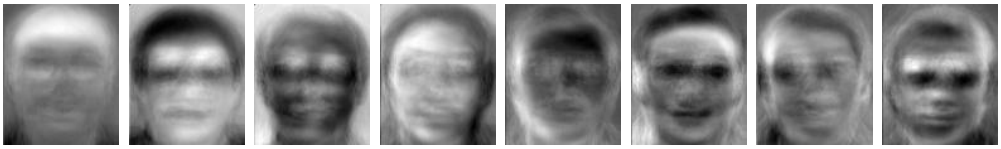
$$AA^T A v_i = \mu_i A v_i \tag{A.7}$$

where Av_i is the eigenvector of $C = AA^T$. Following the analysis, an $M \times M$ matrix $L = A^T A$ is constructed. The M eigenvectors v_i of L are found. Therefore, the technique uses a smaller covariance matrix L rather than the original huge matrix C. The computation of finding eigenvectors on the original matrix C becomes the computation of finding eigenvectors on the smaller alternative covariance matrix L. The size of the matrix is reduced from $N \times N$ to $M \times M$, so that a lot of computational time is saved.

These vectors determine linear combinations of the M training examples to construct the eigenvectors u_l

$$u_l = \sum_{k=1}^{M} v_{lk} \Phi_k \tag{A.8}$$

Because the size of L is $M \times M$, M eigenvectors can be generated from the technique. With the corresponding eigenvalues, the eigenvectors can be sorted. Each eigenvector u_l is an N length vector. If the eigenvectors are transformed into two-dimensional array with the same size of the images, these arrays display a face-like appearance as "ghost face". Hence, these eigenvectors are also called eigenfaces. Some examples of these eigenfaces' are shown in Figure A.2. These eigenfaces



(a) 1st (b) 2nd (c) 3rd (d) 4th (e) 5th (f) 6th (g) 7th (h) 8th

Figure A.2: The top 8 eigenfaces generated from the ORL face database.

are meaningful regarding visual information described in the images. The first eigenface with the biggest corresponding eigenvalue has brighter area across the top of head, which is the area where hair cover. It indicates that in the training set, the variance on the hair area is the most significant. The third eigenface with the third biggest corresponding eigenvalue has two brighter areas locating on the two eyes. It indicates the variance on eyes is significant in the training set.

A.1.2 Image Representation

A face image Γ is transformed into its eigenface components (projected into "face space") by a simple operation

$$\omega_k = u_k^T(\Gamma - \Psi) \tag{A.9}$$

where $k = \{1, \ldots, M'\}$ and $M' \leq M$. The vector $\Omega = [\omega_1, \ldots, \omega_{M'}]$ that describes the contribution of each eigenface in representing the face image Γ. The M' eigenfaces are treated as the basis for face images. Hence, an N length image vector is reduced into an M' length vector. The dimensionality has been reduced by choosing M' the number of eigenfaces. The M' eigenfaces span a subspace in which the vector Ω resides and represents the original image Γ. In term of face recognition, the subspace is called "face space", because all the training examples are face images. The eigenfaces span a lower dimensional subspace which could contains the representations of all possible face images. The vector Ω is considered as the projection of the N-dimensional image space into the M'-dimensional face space, so that the features are selected, or the dimensionality is reduced[1].

Each individual face can be represented exactly in terms of a linear combination of the eigenfaces. If the vector Ω and the eigenfaces are available, the difference between individual one and the mean face is approximately described by

$$\Phi' = \sum_{i=1}^{M'} \omega_i u_i \tag{A.10}$$

The image Γ' can be approximately reconstructed by

$$\Gamma' = \Phi' + \Psi \tag{A.11}$$

The quality of reconstructed image Γ' is controlled by the number of eigenfaces M'. Obviously, the more eigenfaces are adopted for the reconstruction, the higher quality of reconstructed image is acquired. The quality of reconstructed image can be modelled by the loss of image energy

$$\text{Loss} = \frac{\sum_{x,y} \|\Gamma' - \Gamma\|^2}{\sum_{x,y} \|\Gamma\|^2} \tag{A.12}$$

[1]In terms of information theory, the image is encoded into M' random variables

For better reconstructed quality, the loss of image energy should be zero as close as possible. However, there is a trade-off between the higher quality and the more computational cost.

Therefore, any collection of face images can be approximately reconstructed by storing a small collection Ω for each face and a small set of eigenfaces. The Ω describing each face is found by projecting the face image onto each eigenface. In terms of image compression, an image Γ can be compressed as Γ' with some energy loss.

A.1.3 Classification

The vector Ω may then be used in a standard pattern recognition algorithm to find predefined face classes, if any, best describes the face. The simplest method for determining which face class provides the best description of an input face image is to find the face class k that minimises the Euclidian distance

$$\varepsilon_k = \|\Omega - \Omega_k\|^2 \tag{A.13}$$

where a vector Ω_k represents the kth face class. The vector is calculated by averaging each individual vector of the eigenface representation within the kth face class. A face is classified as belonging to the class k when ε_k is below some chosen threshold θ_k. Because creating the vector of weights is equivalent to projecting the original face image onto a low-dimensional face space, many images which might look nothing like a face will project onto a given pattern vector. Also the image belonging to the face space is also concerned. The distance between the image and the face space is simply the squared distance between the mean adjusted input image Φ and Φ', the reconstructed one. The distance is

$$\varepsilon^2 = \|\Phi - \Phi'\|^2 \tag{A.14}$$

Therefore, for recognising a known individual, calculate the class vector Ω_k by averaging the eigenface pattern vectors Ω calculated from the original images of individual. Choose a threshold θ_ε that defines the maximum allowable distance from any face class, and a threshold that defines the maximum allowable distance from face space. For each new face image to be identified, calculate its pattern vector Ω, the distance ε_i to each known class, and the distance ε to face space.

If the minimum distance $\varepsilon_k < \theta_\varepsilon$ but distance $\varepsilon < \theta_\varepsilon$, then the image may be classified as unknown, and optionally used to begin a new face class.

The training set is 360 images from the ORL face database with 40 subjects. For each subject, nine images were taken into the training set, and rest one image taken into the testing set. In the testing set, there are 40 face images and each subject just contains one image. The classifier is the Nearest Neighbour classifier in the PCA implement. The difficulty of PCA is the number of Eigenfaces M'. Different number of Eigenfaces gives the different performance on the classification. An exhaustive search on the optimal number of Eigenfaces is given by letting $M' \in \{2, \ldots, 220\}$. The performance of classification on different number of Eigenfaces is given in Figure A.3. The number of eigenfaces is started from 2 to 220. From

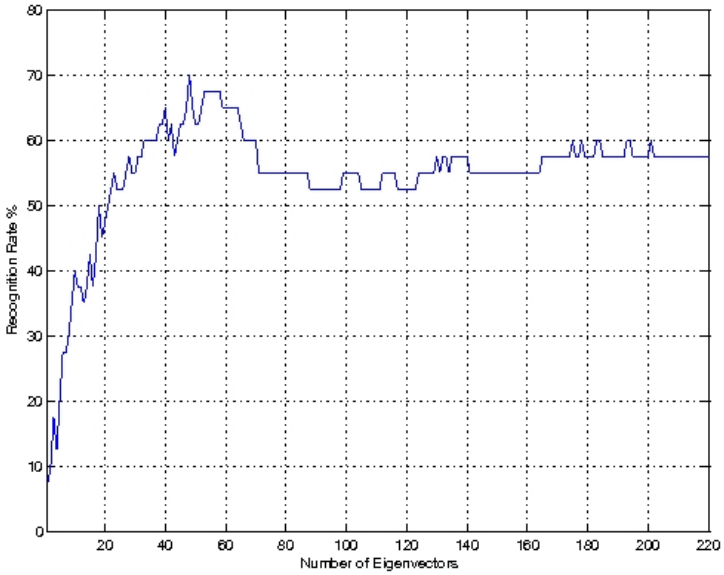


Figure A.3: The recognition rates of the PCA approaches with different number of eigenfaces

$M' = 2$ to $M' = 48$, the recognition rate goes up rapidly. When the M' reaches 49, the recognition rate has its maximum - 70%. After the $M' = 49$, the recognition rate drops slowly, and converge around 55%. It indicates that, in PCA approach on the ORL face database, the optimal number of eigenfaces is 49, where it achieves the best performance.

A.2 Canonical Variate

PCA yields a set of linear features of a particular dimension that best represents the variance in a high-dimensional dataset. There is no guarantee that this set of features is good for classification. Linear features that emphasise the distinction between classes are known as canonical variates [45].

To construct canonical variates, assume there are a set of data with g different classes. There are n_k examples in each class, and an example from the kth class is $x_{k,i}$, for $i = 1, \ldots, n_k$. The jth class has mean μ_j. It is assumed that there are p features (i.e. that the examples are p-dimensional vectors). Write for the mean of the class means, that is

$$\bar{\mu} = \frac{1}{g} \sum_{j=1}^{g} \mu_j \tag{A.15}$$

The variance of the class means gives

$$B = \frac{1}{g-1} \sum_{j=1}^{g} (\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T \tag{A.16}$$

In the simple case, it is assumed that each class has the same covariance, and that this has full rank. It is likely to obtain a set of features where the clusters of examples belonging to a particular class group together tightly, while the discontinuous classes are widely separated. This involves finding a set of features that maximises the ratio of the separation (variance) between the class means to the variance within the class means to the variance within each class. The separation between the class means is typically referred to as *between-class variance*, and the variance within a class is typically referred as *within-class variance*.

The canonical variate approach is concerned on the linear functions of the features as

$$\upsilon(x) = \nu^T x \tag{A.17}$$

It is likely to maximise the ratio of the between-class variance to the within-class variances for ν. It is assumed that each class has the same covariance Σ, which is either known or estimated by

$$\Sigma = \frac{1}{N-1} \sum_{c=1}^{g} \sum_{i=1}^{n_k} (x_{c,i} - \mu_j)(x_{c,i} - \mu_j)^T \tag{A.18}$$

The total number of examples across all class is N. The unit eigenvectors of $\Sigma^{-1}B$ are represented by $\{\nu_1, \nu_2, \ldots, \nu_d\}$. The order of eigenvectors is given by the order of the corresponding eigenvalues, where ν_1 has the largest eigenvalue. Given a set of features, the projection onto the k eigenvectors spans a k-dimensional linear subspace that best separates the class means. The canonical variates are these eigenvectors which can group examples belonging to a particular class together tightly, and widely separate the different classes.

In the experiment of the canonical variates, the ORL face database is adopted. All 400 images are used to generate the canonical variates. The difficulty of canonical variates is to compute the inverse matrix of the covariance matrix Σ, because when the data is in a very high dimensional feature space, the covariance matrix is tend to be singular or close to singular. To solve the problem of singular matrix, the size of images is reduce from 92×112 to 18×22.

From analysis on canonical variates in the ORL face database, the first eigenvector ν_1 (variate) is useless for distinguishing the different classes. The second eigenvector ν_2 displays an excellent performance on classifying the examples. The third eigenvector ν_3 is complex because the covariance matrix is not symmetric. In the third eigenvector, the real part is adopted, but the imaginary part is ignored. With the second eigenvector ν_2 and the real part of the third eigenvector $\Re\{\nu_3\}$, a linear subspace is spanned. In the linear subspace, the 39 subjects are well separated as Figure A.2 shown, which indicates the excellent performance for the canonical variates on classification.

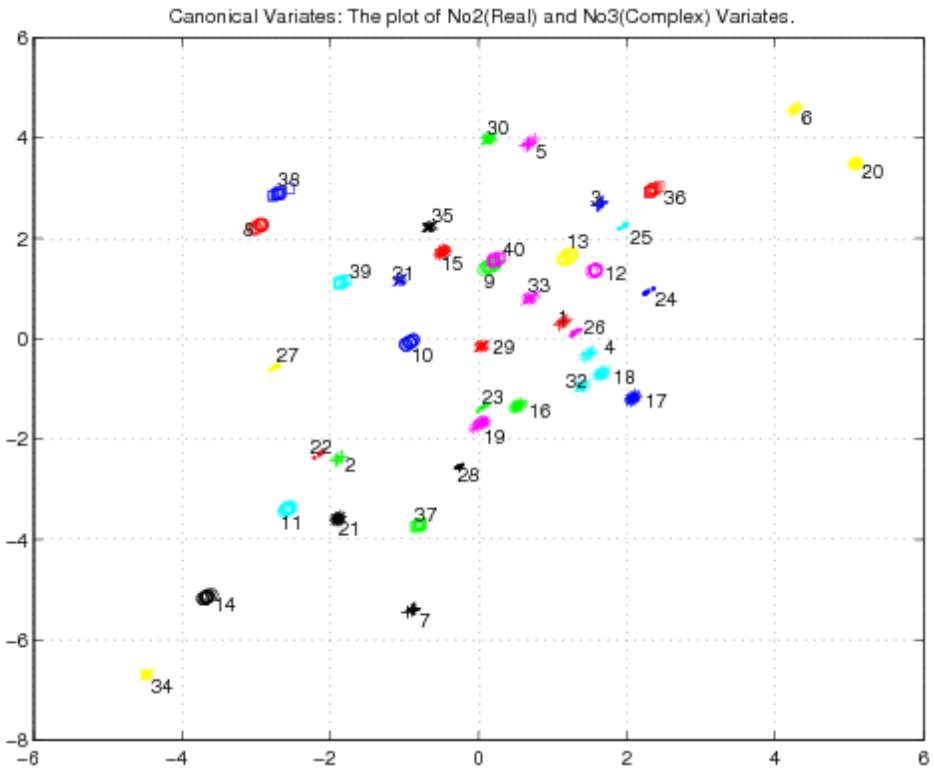


Figure A.4: The 40 classes of the ORL face database on 2D space constructed by Canonical Variates

Appendix B

Weak learner using LLR

Some weak learners are designed to minimise the error ε. In [141, 50], the weak learner is derived as

$$h(x) = \frac{1}{2} \log \frac{P(y = +1|x, \omega)}{P(y = -1|x, \omega)} \tag{B.1}$$

where positive examples are labelled as $y = +1$, and negative examples are labelled as $y = -1$. In this thesis, the positive class is $y = 1$ and the negative class is $y = 0$. The Equation B.1 is changed to

$$h(x) = \begin{cases} 1 & \text{when} \quad \frac{1}{2} \log \frac{P(y=1|x,\omega)}{P(y=0|x,\omega)} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

To train a weak learner, the posterior probabilities $P(y = +1|x, \omega)$ and $P(y = -1|x, \omega)$ are obtained according to Bayes' theorem, which is

$$P(y|x, \omega) = \frac{p(x|y, \omega)P(y)}{p(x|\omega)} \tag{B.3}$$

The likelihood $p(x|y, \omega)$ is learnt from the training examples. The prior $P(y)$ and the evidence $p(x|\omega)$ are also learnt. In [92], a boosted approach using backtrack mechanism, called *FloatBoost*, is proposed. The weak learner is expressed as

$$h(x) = L(x) + T \tag{B.4}$$

where

$$L(x) = \frac{1}{2} \log \frac{p(x|y = +1, \omega)}{p(x|y = -1.\omega)} \tag{B.5}$$

$$T = \frac{1}{2} \log \frac{P(y=+1)}{P(y=-1)} \tag{B.6}$$

Here the positive and negative examples are labelled as $y = +1$ and $y = -1$, respectively. The log likelihood ratio (LLR) $L(x)$ is trained from the training examples of two classes. The threshold T is the log ratio of prior probabilities.

In [68], the weak learner is implemented with a Look-Up Table (LUT), which also requires the likelihood from the training examples.

From the above description, it can be seen that, to build an optimal weak learner, the likelihood $p(x|y, \omega)$ is required to construct the posterior probability in a probability model. Histogram search is a simple way to build a probability model for a classifier [45]. If a histogram is divided by a number of bins, a representation of the class-conditional probability density is obtained. When the size of training set gets larger and the bins gets smaller, the histogram will almost certainly converge to the probability density function. Since the size of the training set is normally fixed, the only way to increase similarity between the histogram and the probability density function is to increase the number of bins.

There are two variables x and ω to mapping, *i.e.*, a 2D space. If the number of bins is b in each variable, the number of two-dimensional bins will be b^2 in the 2D feature space. The computational cost will be quadratically increased. Therefore, the histogram search needs to balance between the accuracy of computed model and the computational cost. A faster and more precise method is needed for the learning of weak learners rather than the LLR concept is used in training weak learners.

Appendix C

Investigation of Overfitting

AdaBoost has its potential to overfit the training set because its objective is to minimise error on the training set [24]. At each iteration, AdaBoost focuses on classifying the mis-classified examples, which might result in fitting noises during the training. Since the AdaBoost classifier is a combination of many weak learners, the classification results demonstrate the overall performance among these weak learners. Ideally, without considering overfitting, more weak learners would lead a better classification performance. With more iterations, classification performace would be improved. However, more weak learners may bring the loss on the classification accuracy, because these learners trend to fit noise better than to fit the data itself. The reason of overfitting is investigated by exploring false negative $\mathcal{FN}$ and false positive $\mathcal{FP}$ on each iteration. Table C.1 shows the performance of the training and testing dataset on each iteration with corresponding false negative $\mathcal{FN}$ and false positive $\mathcal{FP}$. For all eight clients, the number of false negative $\mathcal{FN}$ in the training set are zero after the first and second iterations. It means that after the training of the first two iterations, all positive examples in the training set have been classified correctly, and there are some negative examples which are mis-classified as the positive. In the first two iterations, the classifier has more focused on the positive examples. The classifier does not concern on how many negative examples are misclassified, such that the false positive rate is the highest among all iterations. At the beginning of AdaBoost training, the weights are initialised according to number of examples involved. Hence, the weights on positive examples are much higher than negative examples ($\frac{1}{8}$ v.s. $\frac{1}{1592}$). At the first two iterations, the AdaBoost training is more focused on the positive

Table C.1: The performance of each iteration in the training and testing set by the AdaBoost classifier.

(a) The 1st client

	Training Set (%)		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	1.13	75	0.77
2	0.0	9	3	0.77
3	0.0	1.13	75	0.25
4	0.0	0.0	25	0.19
5	0.0	0.0	50	0.13
6	0.0	0.0	75	0.0
7	0.0	0.0	100	0.0

(b) The 2nd client

	Training Set (%)		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	1.0	100	0.19
2	0.0	1.0	100	1.6
3	0.0	0.0	100	0.0
4	0.0	0.0	100	0.0
5	0.0	0.0	100	0.0
6	0.0	0.0	100	0.0

(c) The 3rd client

	Training Set (%)		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	0.5	100	0.58
2	0.0	0.5	100	0.58
3	0.0	0.0	75	0.25
4	0.0	0.0	100	0.0
5	0.0	0.0	100	0.0

(d) The 4th client

	Training Set (%		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	0.13	25	0.96
2	0.0	0.13	25	0.96
3	0.0	0.0	25	0.13
4	0.0	0.0	25	0.19
5	0.0	0.0	75	0.0
6	0.0	0.0	75	0.0
7	0.0	0.0	100	0.0

(e) The 5th client

	Training Set (%)		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	1.5	75	2.3
2	0.0	1.6	25	3.5
3	0.0	0.0	25	0.39
4	0.0	0.0	25	0.39
5	0.0	0.0	25	0.0
6	0.0	0.0	25	0.0
7	0.0	0.0	50	0.0
8	0.0	0.0	50	0.0
9	0.0	0.0	75	0.0

(f) The 6th client

	Training Set (%)		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	0.25	25	0.51
2	0.0	0.25	25	0.51
3	0.0	0.0	100	0.0

(g) The 7th client

	Training Set (%)		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	0.50	100	0.51
2	0.0	0.50	100	0.51
3	0.0	0.0	100	0.13
4	0.0	0.0	100	0.96
5	0.0	0.0	100	0.13
6	0.0	0.0	100	0.0
7	0.0	0.0	100	0.13
8	0.0	0.0	100	0.0

(h) The 8th client

	Training Set (%)		Testing Set (%)	
Iter	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$
1	0.0	0.25	75	1.3
2	0.0	0.25	100	0.45
3	0.0	0.0	75	0.0
4	0.0	0.0	100	0.0

examples, which means the training is trying to classify the positive examples correctly, but regardless to the negative examples. After these two iterations, the weights on those mis-classified negative examples are increased, but the weights on the positive examples are decreased. After the third iterations, the false positive rate has dropped to zero, because the AdaBoost training has changed to focus on those negative examples. Generally, AdaBoost focuses on the examples with higher weights. When these examples are classified, the weights will be updated across the train set, and the AdaBoost training shifts its focus onto other examples. In this case, AdaBoost firstly focuses on positive examples, then to add more constraints on negative examples and to achieve zero error in the training set. Consequently, the mechanism leads the overfitting on the training set.

From Table C.1, it is seen that the performance of AdaBoost classifier gets stable only after three iterations. When the training is going on further more than three iterations, the overall error (false negative and false positive) is still kept as zero. More iterations means more selected features, *i.e.*, more weak learners are combined into the AdaBoost classifier H. Finally, the performance on the testing set all ends at a point of 100% false negative rate and zero false positive except the 5th client. The performance on different clients converges to the point with different iterations. It indicates that some clients are difficult to be recognised due to the variance between face images. Among these clients, AdaBoost has achieved high performance in early iterations. For the 1st and 4th client, after the 4th iteration, the $\mathcal{FP}$ is 25% (only one positive example not being recognised) and the $\mathcal{FN}$ is 0.19% (only 3 out of 1400 negative examples not being recognised). For the 5th client, after five iterations, the $\mathcal{FP}$ is 25% and the $\mathcal{FN}$ is 0%. However, the performance on some clients (like the 2nd and 7th clients) is persistently low from the start to the end. These clients are "hard" clients in the XM2VTS face database due to large variance between face images within the client.

For both training set and testing set, the performance converges at a certain point at which the performance of classifiers gets stable. The number of iterations for the classifier being stable is much less than the one in the training.

Appendix D

Grid Computing on Computer Vision

In contemperate computer vision, images are always with high resolution, or video sequences contain long duration and high quality frames. There are massive amount of data in computer vision which are processed in high-dimension, multi-scales, etc. With these massive data, the computational time is enormous. Especially in feature selection, a small number of features are selected from hundreds of thousands features. For instance, Young and Ferryman [175] present that AdaBoost takes a significant amount of time to select Haar features and to learn a good classification. With a 2.8 GHz CPU, the learning for AdaBoost takes about two weeks, and with their optimisation on AdaBoost, the computational time has been improved to 6 hours 47 minutes.

It is clear that the AdaBoost training requires huge amount of computational time, with which there are some difficulties with AdaBoost. First of all, the long duration computational training is very vulnerable to the exceptions of system environment. A standalone long-running program may be corrupted due to memory leak, data loss, processor errors, I/O errors in computer systems. Although these problems are less common in a well designed program, the longer the program is running, the more likely the collapse will occur. For very long duration training, this type of problems is unavoidable. The second difficulty is that there are more computational time required in real-time than expected. In practice, the AdaBoost training is not an "once a time" training. It includes many times training with tuning parameters and bias. The actual tuning time is several times

larger than the original expected "once a time" training time. Grid computing technology is applied to use more computing resources to process the training data simultaneously.

This chapter is organised as follows: Section D.1, Grid computing is briefly introduced with its characteristics. In Section D.2, the general Condor Grid, the Condor Grid at University of Reading and a way to design a computer program for the Condor Grid are introduced. In Section D.3, two computer vision tasks are solved by utilising the Condor Grid. Finally, the advantage and disadvantage of Grid computing are given.

D.1 Brief Introduction to Grid Computing

The term Grid computing is originated in the early 1990s as a metaphor for making computer power as easy to access as an electric power grid. As a term in distributed computing, Grid computing is explained as several meanings. Some meanings refer to multiple independent computing clusters acting like a "grid" because resource nodes are not located within a single domain. Some meanings refer to the creation of a "virtual supercomputer" by using spare computing resources within an organisation. And some meanings explain that Grid computing is the creation of a "virtual supercomputer" by using a network of geographically dispersed computers. Among the various meanings, the definition of Grid Computing from IBM is considered as the best one appropriate to the application in this thesis, which says

> *A grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of resources distributed across "multiple" administrative domains based on their (resources) availability, capacity, performance, cost and users' quality-of-service requirements.*

Grid computing has several characteristics.

- Network Centred Parallel Computing
- Massive Multiple Computation Power and Data Storage
- Lower Cost
- CPU Scavenging

Network centred parallel computing Grid computing is a type of network centred parallel computing. There is a group of computers which are connected through the network, and processing the parallel tasks. This eases transactions on limited resources as in utility computing, or makes it easier to assemble volunteer computing networks. For example, volunteer computing was popularised in 1999 by the project SETI@home [7] which is to solve CPU-intensive research problems, to harness the power of networked personal computers worldwide. One drawback of this feature is that the computers which are involved in performing the computations might not be reliable. The Grid system must be designed to prevent malfunctions or malicious participants from producing false, misleading, or erroneous results. Also, due to the lack of central control, there is no way to guarantee that nodes (computers) will not drop out of the network at random times. Some nodes, like laptops or dialup Internet customers, may also be unavailable for computation when there is no network connection available for unpredictable periods. These drawbacks lead computations fail. In many cases, the participating nodes must be trusty, not to abuse the access that is being granted by the centre system. In this thesis, the grid computation is deployed on a dedicated computer cluster to avoid mistrust and availability difficulties.

Massive multiple computation power and data storage Grid computing systems contain massive multiple computation power and data storage. Grids offer a way to solve challenge problems like protein folding, financial modelling, earthquake simulation, and climate/weather modelling. Grids offer a way of using the information technology resources optimally inside an organisation. Some current grid computing systems at least contain hundreds of nodes. Each node is a computer with modern CPU and storage. One of the most famous networks - SETI@home uses more than 3 million computers to achieve 23.37 sustained teraflops in September 2001. With the massive computation horsepower and huge storage capability, Grid computing is able to tackle some questions which was untouchable technically.

Lower cost (Grid computing v.s. Supercomputers) Grid computing costs much less than conventional supercomputers, which includes many processors connected by a high-speed local computer bus. Each node can be purchased as an individual personal computer. When these nodes are connected, they can pro-

duce similar computing resources to a multiprocessor supercomputer, but with a lower cost. This is due to the economies of scale of producing personal computers, compared to less economic design and manufacture of customised supercomputers. Conventional supercomputers also cause some physical problems on supplying sufficient electricity energy and cooling capacity in a single location. In Grid computing, the nodes may allocate across several places, and energy consumption for each node is equal to home appliance. The infrastructure and programming considerations on each type of platform are different. It is expensive and difficult to write programs so that they are run in the environment of a supercomputer, because a supercomputer may have a customised operating system, or have some special supported hardware. In Grids, each node has the same infrastructure both on software and hardware due to the standardisation in the IT industry. It is not difficult and not expensive to write programs for common operation systems and hardware.

CPU scavenging CPU scavenging is also called *cycle scavenging*, *cycle stealing*, or *shared computing*. CPU scavenging creates a "grid" from the unused resources in a network of participants administrative domain. Typically this technique uses computers effectively and prevents computing resources from wasting at night, during lunch, or even in the scattered seconds throughout the day when the computer is waiting for user input or slow devices. The NASA Advanced Supercomputing facility has run genetic algorithms using the Condor cycle scavenger running on about 350 Sun and SGI workstations. Each workstation runs a daemon that watches user I/O and CPU load. When a workstation has been idle for two hours, a job from the batch queue is assigned to the workstation and will run until the daemon detects a keystroke, mouse motion, or high non-Condor CPU usage. At that point, the job will be removed from the workstation and placed back on the batch queue.

D.2 Condor Grid

Among various Grid computing projects, the Condor grid [158] is used in this thesis. In this section, the Condor grid is firstly introduced. Then the Condor grid at University of Reading is briefly introduced. The key part is how to design a program for the Condor Grid, which is also given in this section.

D.2.1 Condor

Condor is a research project at the University of Wisconsin-Madison (UW-Madison), and it was first installed as a production system in the UW-Madison Department of Computer Sciences nearly 15 years ago. According to usage statistics on a typical day, Condor delivers more than 650 CPU days to UW researchers. Today, hundreds of organisations in industry, government, and academia have used Condor to establish compute installations ranging in size from a handful to well over one thousand workstations. Most flavours of Unix are supported, as well as Windows NT/2K and XP.

Condor is a technical workload management system for compute-intensive jobs. Like other batch systems, Condor provides job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy.

D.2.2 Condor at UoR

The Reading University Grid is a Condor Pool consisting of between 200 and 500 nodes. The aim of the Campus Grid is to provide researchers with a High Throughput Computing (HTC) resource. HTC differs from High Performance Computing (HPC) for that it is designed to process tasks that require fairly short processing times (usually minutes or hours), and repeated processes. For example, a program of genetic algorithm takes a sample of data from a data-set, analyses the sample and then stores a result. This process takes 30 minutes but needs to be done 1000 times. This requires 500 hours of processing time.

Although nodes in the Condor pool naively run Windows XP, Cooperative Linux (CoLinux) [6] is installed as a Microsoft Windows service on these machines to provide the Linux environment. The CoLinux is an open source software for optimally running Linux on Windows. Specifically, the CoLinux is a port of the Linux kernel that allows it to run together with another operating system on a single machine. For instance, it allows one to freely run Linux on Windows 2000/XP, without using other commercial PC visualisation software such as VMware or Virtual Box. In the University Grid, each node runs on Windows XP, and CoLinux as a diskless Linux image is stored on each node. Once a Windows XP node is idle for a fixed period (such as no I/O access, no keybroad or mouse input), the

CoLinux image will be deployed into the memory of nodes. The CoLinux will take over the control of the node. The node becomes running at linux so that it accepts jobs from the Condor pool.

D.2.3 Program Design for Condor Grid

The Condor Grid is designed to perform the HTC. It is possible that some HPC is also performed in the Condor Grid. However some modifications are required, and the programs are required to follow some specifications of the Condor Grid. The road-map of running with the Condor Grid is

1. Preparing code
    - separate the whole task into small jobs
    - define input and output
    - make the program resume-able
    - link the program with static-library and compile it
2. Choosing Runtime Environments (Universe)
    - standard universe
    - vanilla universe
3. Generating Submission Description File
    - executable file
    - program arguments
    - input and output data
    - number of jobs to run
4. Submitting the jobs
5. Maintaining, Debugging, Adjusting, etc

Preparing code

An executable program on a stand alone machine can not directly be deployed on the Condor Grid, because the Condor Grid contains a cluster of parallel and distributed individual machines. The original program on a stand alone machine is modified to fit for the specification of the Condor Grid. If the program is designed under an extendable and flexible framework, the modification is minor. However, if the program is designed under a restricted framework, the deployment of whole program is needed to be done. Each machine processes only small part of computation. Therefore, the initial design for the program running on the Condor Grid is to separate the whole task into a large number of small jobs. The number of these jobs should be equal to the number of nodes planned to utilise. Each small job runs on a node.

For any scientific computer program, there must be an input and an output, so when designing a program on the Condor Grid, the input and output of the program must be defined.

Even if a dedicated grid is allocated, the computation on the grid is not trustworthy. The computation on nodes may fail due to exception of program running, power cuts, system down, and other unpredictable reasons. If computation is broken on a node in a middle way, it has to restart manually from the beginning. It obviously wastes the computational resource. A wise method is to make the program resume-able. A resume-able program uses a log file to record the status of the program at an interval time. If the program is failed in a middle way, it is no need to restart it from the beginning. The program checks the log file, and starts from where it fails.

In compiling the code, the program is linked with static-library. A static library or statically-linked library is a set of routines, external functions and variables which are resolved in a caller at compile-time. The library is copied into a target application by a compiler. Using the static library linking, the program is more independent from different platforms. The compiler produces an object file and a stand-alone executable file. Programs compiled with linking static-libraries normally takes more compiling time and more storage space.

Choosing Runtime Environments

The Condor has several runtime environments called universes. There are two universes to choose when submitting a job to the Condor Grid. One is called *standard universe*, and the other is called *vanilla universe*. The standard universe allows a job running under the Condor Grid to handle system calls by returning them to machines where a job is submitted. The standard universe also provides the mechanisms necessary to take a checkpoint and migrate a partially completed job. To use the standard universe, it is necessary to re-link the program with the Condor library and to use the compiler `condor_compile` provided with Condor. The vanilla universe provides a way to run jobs which do not requires to relink. There is no way to take a checkpoint or migrate a job executed under the vanilla universe. For access to input and output files, the jobs must either use a shared file system, or use Condor's File Transfer mechanism. The standard universe provides more flexibilities such as checkpoint and job migration, but it brings some complexity on building the program. The vanilla universe has many constraints on managing the jobs, but the program can be build with any compiler, such as **GNU GCC** [58]. In this thesis, the vanilla universe is chosen due to only **GNU GCC** compiler available.

Generating Submission Description File

When the runtime environment is decided, jobs are submitted to the Condor pool. The arguments for the submission is described in the submission description file. This file contains commands and keywords to direct the queueing of jobs. In the submission description file, the Condor Grid finds everything it needs to know about the job, such as the name of the executable file, the initial working directory, command-line arguments to the executable file, the directory of input and output data, and the number of jobs to run. It is easy to submit multiple small jobs of a whole task to the Condor pool. The example of the submit description file is below

```
Universe = vanilla
Executable = train
Arguments = -job $(Process)
log = log.txt
Error = Error/err.$(Process)
```

```
Queue 120
```

This submission description file is to queue 120 jobs of an executable file `train`. These jobs require the Condor Grid to run the program on machines with different arguments. Each of the 120 jobs of the program is given its own process arguments, starting with process number 0. The `$(Process)` represents the current number of process. A log file is assigned to be the file `log.txt`, containing entries about when and where Condor runs, checkpoints, and migrates processes for the 120 queued jobs. The error message for each job is stored in the file `err.$(Process)` under the directory `Error`. The number of jobs is expressed as `Queue 120`.

Submitting the jobs Jobs are submitted for execution to the Condor Grid using the `condor_submit` command.

Maintaining, Debugging, Adjusting, etc. Once jobs are submitted, the Condor Grid does the rest toward running these jobs, *e.g.*, monitors the jobs' progress with the `condor_q` and `condor_status` commands. The order of these jobs is modified with the `condor_prio` command. The jobs need to be monitored, so that it can find that some jobs stop exceptionally, or some jobs produce wrong output. These jobs need to be stopped and re-submitted manually. This step takes the longest duration in the Condor Grid computation.

D.3 Computer Vision Application using Condor Grid

In this section, the Condor Grid has been applied on two computer vision cases which demand large amount of computation. One is to compute mutual information between Gabor wavelet features, the other is to compute the multi-class Gabor-Boosting for feature selection.

D.3.1 Computing Mutual Information

Refer to Mutual information computation on a whole set of Gabor Feature Pairs (GFPs), the number of GFPs is massive. The purpose of computing mutual information over all GFPs is to find a small group of GFPs which have the lowest

mutual information, so that the group of GFPs are formed into a classifier to recognise human faces. There are $30,240$ Gabor features available to construct GFPs. To calculate the mutual information on these $457,213,680$ GFPs, it is necessary to use the Condor Grid utilising the power of hundreds machines.

Among these pairs, they are independent to each other, which means the mutual information on one pair does not affect the mutual information on another pair. Therefore, the whole set of pairs are computed in a parallel way. The whole set of pairs are separated into 121 subsets. Each subset is equivalent to a condor job - roughly $3,700,000$ pairs. If there are enough unclaimed nodes available, 121 jobs can run simultaneously. The input data for these jobs is the face images from the XM2VTS face database. The output data is stored as a text file recording the quantity of mutual information of each pair for each job. The 121 jobs are assigned to 121 nodes in the Condor Grid. Figure D.1 shows the diagram of the Mutual Information Computing with Condor Grid. Through the network, each node takes the input data extracted from the XM2VTS face Database on a machine called `kalypso.rdg.ac.uk`, and stores the output data on a machine called `jupiter.rdg.ac.uk`.

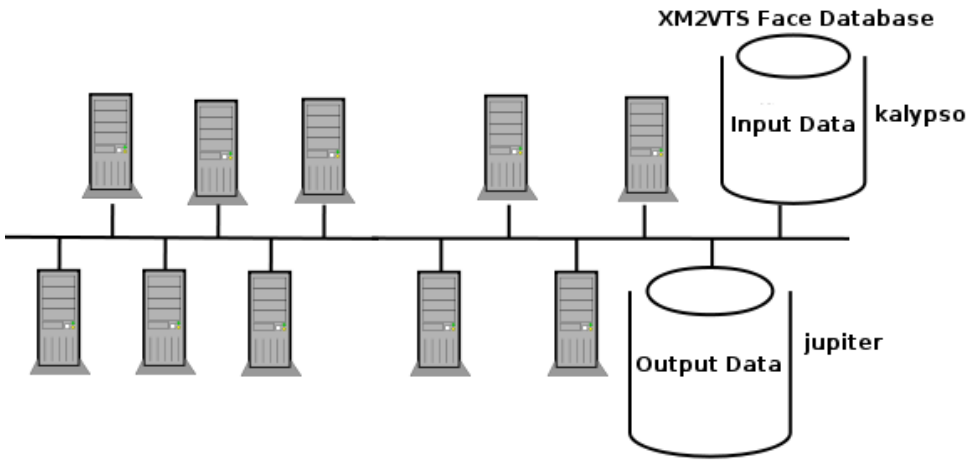


Figure D.1: Computing Mutual Information in the Condor Grid

If only one stand alone machine is used to calculate the whole set of pairs, it will take a very long duration. For instance, using an *Intel Pentium 4* 2.8 GHz computer, the calculation on each pair takes 20 milliseconds. The whole set of

pairs contains $457,213,680$ GFPs, so that

$$\begin{aligned} &20 \text{ millisecond} \times 457213680 \approx 9144273 \text{ seconds} \\ &\approx 152404 \text{ minutes} \approx 2540 \text{ hours} \approx 105 \text{ days} \end{aligned}$$

The whole computation will take 105 days to complete, however it is the optimistic estimation, because the program may failed, be stopped or broken due to some exception happened in the 105 days.

It is assumed that every node has the same computational capability equal to the *Intel Pentium 4* 2.8 GHz machine on the Condor Gird. Theoretically, the Condor Grid computation on mutual information will take 20 hours to finish. However, in practice, it takes more time, because the nodes in the Grid are not always dedicated to the Grid computation. Some nodes may be occupied from the Grid computing due to other usages. Also, due to the different hardware configuration on different nodes, the computation capability is different, so that some jobs are finished earlier than others. The whole duration for Condor Grid has been extended for waiting some slow jobs to finish. It indicates that the computational time for the Condor Grid is difficult to predict precisely.

D.3.2 Multi-class Gabor-Boosting Feature Selection

The second application is to perform the training on Multi-class Gabor-Boosting Feature Selection. The Multi-Class Gabor-Boosting algorithm already has been introduced in Section 6.2. The training of mPotsu weak learner is very time consuming, because it includes 200 individual training on each binary Potsu weak learner respectively. On an *Intel Pentium 4* 2.8 GHz machine, the training on an mPotsu weak learner takes 11 seconds. It is infeasible to run the training of Multi-class Gabor-Boosting on a stand-alone machine. Therefore, the Condor Grid computation is necessary.

The detailed description of Condor grid implementation is introduced in Section 6.2.4 of Chapter 6.

D.4 Summary

Grid computing speeds up computer vision application significantly, especially in feature selection. With the Condor Grid, the total computational time is reduced into approximately $1/n$ of the original time (n is the number of nodes) theoretically. In the first computer vision application - mutual information over GFPs, the total computation time has been reduced to 20 hours instead of 105 days. In the second computer vision application - Multi-class Gabor-Boosting training, the total computation time becomes less than one week rather than two years. The Grid computing is less expensive comparing to the conventional supercomputers. The Condor Grid consists of a cluster of desktop computers, network connecting nodes, and a sever managing the jobs. The implementation of the Condor Grid is not as difficult as the conventional supercomputers.

However, Grid computing has some shortcomings. For Grid computing, the algorithm must have independent elements so that these elements can be processed in a parallel way. In feature selection, each feature is an element which is independent to each other, so that features are processed in a parallel way. The program running in the Condor Grid needs to be processed a small set of data with the cooresponding arguments. Grid computing is not entirely trustworthy. There might be some miss links which lead failure to access input data. In the Condor Grid, the cluster of nodes are not dedicated only for the Grid computing, so that some machines probably are not available after jobs are assigned to them. In this situation, the jobs have to be removed from the Condor Grid, and resubmit to the Condor pool. The performance of the Grid computing is unpredictable, due to different software and hardware configurations on different nodes. It is very hard to predict the computational time of Grid computing. In the Condor Grid computation, the computational time on each node should be kept as small as possible, because when the computational time becomes longer, the more opportunities for the jobs encounter exceptions in running.

In general, Grid computing largely improves computational performance for scientific research. Although it has some drawbacks, these drawbacks can be overcome. Grid computing brings a lot of advantages on computer vision applications.

References

[1] Biometric Consortium, http://www.biometrics.org.

[2] International biometrics group, http://www.biometricgroup.com.

[3] Y. Adini, Y. Moses, and S. Ullman. Face recognition: The problem of compensating for changes in illumination direction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):721–732, 1997.

[4] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face recognition with local binary patterns. In *Proceedings of European Conference on Computer Vision*, 2004.

[5] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2037–2041, 2006.

[6] Dan Aloni. Cooperative linux. In *Proceedings of the Linux Symposium*, 2004.

[7] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, November 2002.

[8] Edgar Anderson. The irises of the gaspe peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.

[9] Taichi Asami, Koji Iwano, and Sadaoki Furui. Stream-weight optimization by LDA and Adaboost for multi-stream speaker verification. In *Proceedings of International Conference on Speech Communication and Technology (Interspeech 2005)*, pages 2185–2188, Lisbon, Portugal, 2005.

[10] F.R. Bach and M.I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[11] Kyungim Baek, Bruce A. Draper, J. Ross Beveridge, and Kai She. PCA vs. ICA: A comparison on the FERET data set. In *In Proceedings of Joint Conference on Information Sciences*, 2002.

[12] Randall C. Ballard. Television system. Patent, 3 1939. Patent number: 2152234.

[13] M.S. Bartlett, H.M. Lades, and T.J. Sejnowski. Independent component representations for face recognition. *Proceedings of SPIE*, 3299:528–539, 1998.

[14] M.S. Bartlett, J.R. Movellan, and T.J. Sejnowski. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13:1450–1464, 2002.

[15] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.

[16] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[17] B. Bell and L. Mass. Firms forge alliance to enhance. Businessworld, 29 Dec 2001.

[18] J.R. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center, Princeton, 1990.

[19] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the SIGGRAPH'99*, pages 187–194, August 1999.

[20] Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1063–1074, September 2003.

[21] W.W. Bledsoe. The model method in facial recognition. Technical Report PRI:15, Panoramic Research Inc., Palo Alto, CA, 1964.

[22] A. C. Bovik, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:55–73, 1990.

[23] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[24] Tom Bylander and Lisa Tate. Using validation sets to avoid overfitting in AdaBoost. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, pages 544–549, 2006.

[25] Jean-Francois Cardoso and Antoine Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM*, 17(1):161–164, 1996.

[26] H. Cevikalp and M. Wilkes. Face recogntion by using discriminative common vectors. In *Proceedings of International Conference on Pattern Recognition*, 2004.

[27] L.F. Chen, H.Y.M. Liao, J.C. Lin, M.T. Ko, and G.J. Yu. A new LDA-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33:1713–1726, 2000.

[28] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[29] T.F. Cootes and C.J. Taylor. Locating faces using statistical features. In *Proceedings of Second International Conference on Automatic Face and Gesture Recognition*, pages 204–209, 1996.

[30] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Technical report, University of Manchester, 2000.

[31] J. Daugman. Uncertainty relation for resolution in space, spatial frequency and orientation optimized by two-dimensional visual cortical filters. *Journey of the Optical Society of American A*, 2:1160–1169, July 1985.

[32] J. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1148–1160, November 1993.

[33] John G Daugman. Complete discrete 2-D gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36:1169–1179, 1988.

[34] E. R. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Academic Press, 1990.

[35] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.

[36] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–137, 1997.

[37] H. Drucker, R.E. Schapire, and P. Y. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705–719, 1993.

[38] D. Dunn and W. Higgins. Optimal gabor filters for texture segmentation. *IEEE Transactions on Image Processing*, 4(7):947–964, July 1995.

[39] Dennis Dunn, William E. Higgins, and Joseph Wakeley. Texture segmentation using 2-D Gabor elementary functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:130–149, 1994.

[40] G.J. Edwards, T.F. Cootes, and C.J. Taylor. Face recognition using active appearance models. In *Proceedings of European Conference on Computer Vision*, 1998.

[41] B. Efron and K. McKusick. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[42] W. Fan, Y.H. Wang, W. Liu, and T.N. Tan. Combining null space-based gabor features for face recognition. In *Proceedings of International Conference on Pattern Recognition*, 2004.

[43] N. Fell and P. Dempsey. Whitehall dithers on airport security. Times, 14 Jan 2002.

[44] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[45] D.A. Forsyth and J. Ponce. *Computer Vision A Modern Approach*. Prentice Hall, 2003.

[46] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[47] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt ?5*, pages 23–37. Springer-Verlag, 1995.

[48] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14:771–780, 1999.

[49] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Stanford University, 1998.

[50] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

[51] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. New York: Academic Press, 1990.

[52] S. I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

[53] Francis Galton. Personal identification and description. *Nature*, 21:173–177, June 1888.

[54] I. Gauthier and C.A. Nelson. The development of face expertise. *Cognitive Neuroscience*, 11:219–224, 2001.

[55] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:643–660, 2001.

[56] C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society B*, 53(2):285–339, 1991.

[57] John Gorsuch. Improved security through technology. Overhaul & Maintenance, 20 Nov 2001.

[58] Brian J. Gough. *An Introduction to GCC*. Network Theory Limited, 2004.

[59] Daniel B. Graham and Nigel M. Allinson. Characterizing virtual eigen signatures for general purpose face recognition. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors, *Face Recognition: From Theory to Applications*, volume 163, pages 446–456. NATO ASI Series F, Computer and Systems Sciences, 1998.

[60] Ralph Gross. Face databases. In S. Li and A.K. Jain, editors, *Handbook of Face Recognition*. Springer, New York, February 2005.

[61] A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[62] P. Hallinan. *A Deformable Model for Face Recognition Under Arbitrary Lighting Conditions*. PhD thesis, Harvard University, 1995.

[63] P.J. Hancock, V. Bruce, and A.M. Burton. Recognition of unfamiliar faces. *Trends in Cognitive Sciences*, 4:330–337, 2000.

[64] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision*, volume 2. Addison-Wesley, 1992.

[65] H. Hong, H. Neven, and C. von der Malsburg. Online facial expression recognition based on personalized galleries. In *Proceedings of International Conference on Automatic Face and Gesture Recognition*, 1998.

[66] B. Horn. *Robot Vision*. MIT Press, 1986.

[67] B.K.P. Horn and B.G. Schunck. Determing optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[68] Chang Huang, Haizhou Ai, Bo Wu, and Shihong Lao. Boosting nested cascade detector for multi-view face detection. In *In Proceedings of International Conference on Pattern Recognition 2004*, 2004.

[69] J. Huang, B. Heisele, and V. Blanz. Component-based face recognition with 3D morphable models. In *Proceedings of the 4th International Conference on Audio- and Video-Based Biometric Person Authentication, AVBPA 2003*, pages 27–34, Guildford, UK, June 2003.

[70] Rui Huang, Qingshan Liu, Hanqing Lu, and Songde Ma. Solving the small sample size problem of LDA. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 3, 2002.

[71] A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999.

[72] A. Hyvarinen, J Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[73] K. Ichikawa, T. Mita, and O. Hori. Component-based robust face detection using Adaboost and decision tree. In *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition, FGR 2006*, 2006.

[74] A. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.

[75] A.K. Jain, P.W. Robert, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[76] A.K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14:4–20, 2004.

[77] Anil K. Jain and Farshid Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, 4:1167 – 1186, December 1991.

[78] J. Jones and L. Palmer. An evaluation of two-dimensional gabor filter model of simple receptive fields in cat strait cortex. *Journal of Neurophysiology*, 58:1233–1258, 1987.

[79] A. Kadyrov and M. Petrou. The trace transform and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):811–828, August 2001.

[80] Takeo Kanade. Computer recognition of human faces. *Interdisciplinary Systems Research*, 47:1–106, 1977.

[81] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formula and finite automata. *Journal of the Association for Computing Machinery*, 41:67–95, 1994.

[82] M.D. Kelly. Visual identification of people by computer. Stanford AI Project AI-130, Stanford University, 1970.

[83] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Signal Processing, Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, December 1981.

[84] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, Jan 1990.

[85] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995.

[86] H. Kong, L. Wang, E. Teoh, J. Wang, and R. Venkateswarlu. A framework of 2D fisher discriminant analysis: Application to face recognition with small number of training samples. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[87] Hui Kong, Xuchun Li, Jian-Gang Wang, and Chandra Kambhamettu. Ensemble LDA for face recognition. In *Proceedings of the International Conference on Biometrics Authentication (ICBA)*, 2006.

[88] Volker Krueger. *Gabor Wavelet Networks for Object Representation.* PhD thesis, Christian-Albrechts University, Kiel, Germany, 2001.

[89] M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, C.V.D. Malsburg, and R. Wurtz. Distortion invariant object recognition on the dynamic link architecture. *IEEE Transactions on Computers*, 42:300–311, 1993.

[90] Jerome Landre. Programming with Intel IPP and Intel OpenCV under GNU Linux: A beginner's tutorial. Technical report, Universite de Bourgogne, 2003.

[91] A. Lanitis, C.J. Taylor, and T.F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):743–756, 1997.

[92] Stan Z. Li and ZhenQiu Zhang. FloatBoost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Learning*, 26(9):1112–1123, 2004.

[93] Ping Sung Liao, Tse Sheng Chen, and Pau Choo Chung. A fast algorithm for multilevel thresholding. *Journal of Information Science and Engineering*, 17:713–727, 2001.

[94] C. Liu. Enhanced independent component analysis and its application to content based face image retrieval. *IEEE Transactions on Systems, Man, and Cybernetics- Part B: Cybernetics*, 34(2):1117–1127, April 2004.

[95] C. Liu. Gabor-based kernel pca with fractional power polynomial models for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:572–581, 2004.

[96] C. Liu and H. Wechsler. Gabor feature based classification using enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, 11:467–476, 2002.

[97] C. Liu and H. Wechsler. Independent component analysis of gabor features for face recognition. *IEEE Transactions on Neural Networks*, 14:919–928, 2003.

[98] Chengjun Liu and Harry Wechsler. A gabor feature classifier for face recognition. In *Proceedings of the Eighth International Conference on Computer Vision (ICCV'01)*, volume 2, 2001.

[99] M. M. Loeve. *Probability Theory*. Princeton N.J, 1955.

[100] J. Lu, K.N. Plataniotis, and A.N. Venetsanopoulos. Face recognition using LDA-based algorithms. *IEEE Transactions on Neural Networks*, 14(1):195–200, January 2003.

[101] X.G. Lu, Y.H. Wang, and A.K. Jain. Combining classifiers for face recognition. In *Proceedings of IEEE International Conference on Multimedia and Expo*, volume 3, pages 13–16, 2003.

[102] Xiaoguang Lu. *3D Face Recognition across Pose and Expression*. PhD thesis, Michigan State University, 2006.

[103] M.J. Lyons, J. Budynek, and S. Akamatsu. Automatic classification of single facial image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:1357–1362, 1999.

[104] J. Ma, Y. Zhao, S. Ahalt, and D. Eads. OSU SVM: A support vector machine toolbox for Matlab. Online, 2001.

[105] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.

[106] Aleix Martinez and Avinash Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:228–233, 2001.

[107] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithm as gradient descent in function space. Technical report, Department of Systems Engineering, Australian National Univeristy, 1999.

[108] J. Matas, M. Hamous, and K. Jonsson. Comparison of face verification results on the xm2vts database. In *Proceedings of the International Conference on Pattern Recognition*, 2000.

[109] Jiri Matas and Jan Sochman. Adaboost. Technical report, Centre for Machine Perception, Czech Technical University, Prague, 2004.

[110] M.E. Mavroforakis and S. Theodoridis. A geometric approach to support vector machine (SVM) classification. *IEEE Transactions on Neural Networks*, 17:671–682, 2006.

[111] K. Messer, J. Matas, J. Kittler, and K. Jonsson. XM2VTSDB: The extended M2VTS database. In *Audio- and Video-based Biometric Person Authentication, AVBPA'99*, pages 72–77, Washington, D.C., March 1999. 16 IDIAP–RR 99-02.

[112] B. Moghaddam. Principal manifolds and probabilistic subspaces for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:780–788, 2002.

[113] B. Moghaddam, T. Jebara, and A. Pentland. Bayesian face recognition. *Pattern Recognition*, 33:1771–1782, November 2000.

[114] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, July 1997.

[115] A. Nefian and M. Hayes. Hidden markov models for face recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'98*, volume 5, pages 2721–2724, May 1998.

[116] A. Nefian and M. Hayes. Maximum likelihood training of the embedded HMM for face detection and recognition. In *Proceedings of the IEEE International Conference on Image Processing, ICIP 2000*, volume 1, pages 33–36, Vancouver, BC, Canada, September 2000.

[117] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transcations on Pattern Analysis and Machine Intelligence*, 24:971–987, 2002.

[118] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[119] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, June 1997.

[120] N. Otsu. A threshold selection method from gray level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9:62–66, March 1979. minimize inter class variance.

[121] P. S. Penev and J. J. Atick. Local feature analysis: A general statistical theory for object representation. *Neural Systems*, 7:477–500, 1996.

[122] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[123] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J.Rauss. The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104, October 2000.

[124] P. Jonathon Phillips, Patrick J. Rauss, and Sandor Z. Der. FERET (face recognition technology) recognition algorithm development and test results. Technical report, Army Research Lab, 1996.

[125] P.J. Phillips, H. Moon, P. Rauss, and S.A. Rizvi. The FERET methodology for face recognition algorithm. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 137–143, 1997.

[126] P.J. Phillips, H. Wechsler, J. Huang, and P.J. Rauss. The FERET database and evaluation procedure for face recognition algorithm. *Image and Vision Computing*, 16(5):295–306, 1998.

[127] Chris Pope. Ways to keep the terrorist grounded. *Professional Engineering*, 14:20–21, 2001.

[128] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[129] P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 11:1119–1125, 1994.

[130] Gunnar Ratsch, Sebastian Mika, Bernhard Scholkopf, and Klaus-Robert Muller. Constructing boosting algorithms from SVMs: An application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1184 – 1199, 2002.

[131] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceeding of International Conference on Neural Network*, 1993.

[132] S.A. Rizvi, P.J. Phillips, and H. Moon. The FERET verification testing protocol for face recognition algorithm. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, number 48-53, 1998.

[133] S.A. Rizvi, P.J. Phillips, and H. Moon. A verification protocol and statistical performance analysis for face recognition algorithm. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 833–838, 1998.

[134] A. R.Martinez and R. Benavente. The AR face database. Technical Report 24, Computer Vision Center, 1998.

[135] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:3860408, 1958.

[136] P.K. Sahoo, S. Soltani, A.K. Wong, and Y.C. Chan. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41:233–260, 1988.

[137] Ferdinando S. Samaria and Andy C. Harter. Parameterisation of a stochastic model for human face identification. In *IEEE 2nd Workshop on Applications of Computer Vision*, page 138ï142, Sarasota (Florida), December 1994.

[138] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[139] R. E. Schapire and Y. Singer. Improved boosting algorithm using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

[140] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[141] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of The 11th Annal Conference on Computational Learning Theory*, number 80-91, 1998.

[142] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[143] B. Scholkopf, A. Smola, and K.R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[144] A. Shashua. *Geometry and Photometry in 3D Visual Recognition*. PhD thesis, Massachusetts Institute of Technology, 1994.

[145] Linlin Shen and Li Bai. MutualBoost learning for selecting gabor features for face recognition. *Pattern Recognition Letters*, 27:1758–1767, 2006.

[146] Linlin Shen, Li Bai, Bardsley Daniel, and Yangsheng Wang. Gabor feature selection for face recognition using improved Adabooost learning. In *Proceedings of International workshop on biometric recognition systems*, 2005.

[147] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615–1618, December 2003.

[148] Pawan Sinha, Benjamin Balas, Yuri Ostrovsky, and Richard Russell. Face recognition by humans: Nineteen results all computer vision researchers should know about. *Proceedings of The IEEE*, 94(11):1948–1962, November 2006.

[149] P. Sollich and A. Krogh. Learning with ensembles: How over-fitting can be useful. In D. Touretsky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 190–196. MIT Press, 1995.

[150] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, September 1998.

[151] S. Srisuk and W. Kurutach. Face recognition using a new texture representation of face images. In *Proceedings of Electrical Engineering Conference*, pages 1097–1102, Cha-am, Thailand, November 2003.

[152] S. Srisuk, M. Petrou, W. Kurutach, and A. Kadyrov. Face authentication using the trace transform. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*, pages 305–312, Madison, Wisconsin, USA, June 2003.

[153] D.L. Swet and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:831–836, 1996.

[154] John Shawe Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[155] A. Tefas, C. Kotropoulos, and I. Pitas. Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):735–746, July 2001.

[156] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *SCIENCE*, 290:2319–2323, 2000.

[157] Andreas Teuner, Olaf Pichler, and Bedrich J. Hosticka. Unsupervised texture segmentation of image using tuned matched gabor filters. *IEEE Transactions on Image Processing*, 4(6):863–870, 1995.

[158] Douglas Thain, Todd Tannenbaum, and Miron Livny. Condor and the Grid. In Fran Berman, Geoffrey Fox, and Tony Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., December 2002.

[159] M. Turk and A. Pentland. Eigenfaces for recognition. *Journey of Cognitive Neuroscience*, 3(1):71–86, 1991.

[160] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[161] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[162] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[163] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[164] H. Wang, J. Yin, J. Pei, P. S. Yu, and J. X. Yu. Suppressing model overfitting in mining concept-drifting data streams. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*, 2006.

[165] B. Weyrauch, J. Huang, B. Heisele, and V. Blanz. Component-based face recognition with 3D morphable models. In *Proceedings of CVPR Workshop on Face Processing in Video (FPIV'04)*, 2004.

[166] Wikipedia. Netpbm format — wikipedia, the free encyclopedia, 2008. [Online; accessed 21-April-2008].

[167] L. Wiskott, J.M. Fellous, N. Kruger, and C. Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:775–779, 1997.

[168] L. Wiskott, J.M. Fellous, N. Kruger, and C. Malsburg. Face recognition by elastic bunch graph matching. In L.C. Jain et al, editor, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, pages 355–396. CRC Press, 1999.

[169] Bo Wu, Haizhou Ai, and Ran Liu. Glasses detection by boosting simple wavelet features. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, pages 292–295. Washington, DC, USA, IEEE Computer Society, 2004.

[170] X.J. Wu, J. Kittler, J.Y. Yang, K. Messer, and S.T. Wang. A new direct LDA (D-LDA) algorithm for feature extraction in face recognition. In *Proceedings of International Conference on Pattern Recognition*, 2004.

[171] J. Yang, D. Zhang, A.F. Frangi, and J. Yang. Two-dimensional PCA: a new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:131–137, 2004.

[172] M.H. Yang. Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.

[173] Ming-Hsuan Yang. Face recognition using extended isomap. In *Proceedings of Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002.

[174] Peng Yang, Shiguang Shan, Wen Gao, Stan Li, and Dong Zhang. Face recognition using Ada-Boosted gabor features. In *Proceedings of the 6th*

IEEE International Conference on Automatic Face and Gesture Recognition, 2004.

[175] David Young and James Ferryman. Faster learning via optimised Adaboost. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005. AVSS 2005.*, 2005.

[176] Z. Zhang, M. Lyons, M. Schuster, and S. Akamastsu. Comparison between geometry-based and gabor wavelets-based facial expression recognition using multi-layer perceptron. In *Proceeding of International Conference on Automatic Face and Gesture Recognition*, volume 1, pages 454–457, 1998.

[177] W. Zhao and R. Chellappa. SFS based view synthesis for robust face recognition. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, pages 285–292, 2000.

[178] W. Zhao, R. Chellappa, and A. Krishnaswamy. Discriminant analysis of principal components for face recognition. In *Proceedings of the 3rd IEEE International Conference on Face and Gesture Recognition, FG'98*, 1998.

[179] W. Zhao, R. Chellappa, A. Rosenfeld, and P.J. Phillips. Face recognition: A literature survey. *ACM Computing Surveys*, 1:399–458, 2003.

[180] Mian Zhou and Hong Wei. Face verification using gabor wavelets and Adaboost. In *Proceedings of 18th International Conference on Pattern Recognition*, 2006.

[181] S. Zhou, R. Chellappa, and B. Moghaddam. Intra-personal kernel space for face recognition. In *Proceedings of the 6th International Conference on Automatic Face and Gesture Recognition, FGR2004,*, 2004.

[182] Ji Zhu, Saharon Rosset, Hui Zhou, and Trevor Hastie. Multi-class Adaboost. A multiclass generalization of the Adaboost algorithm, based on a generalization of the exponential loss, 2006.