# Facial Expression Recognition System

by

Yuan Ren

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2008

Yuan Ren

# Abstract

A key requirement for developing any innovative system in a computing environment is to integrate a sufficiently friendly interface with the average end user. Accurate design of such a user-centered interface, however, means more than just the ergonomics of the panels and displays. It also requires that designers precisely define what information to use and how, where, and when to use it. Facial expression as a natural, non-intrusive and efficient way of communication has been considered as one of the potential inputs of such interfaces. The work of this thesis aims at designing a robust Facial Expression Recognition (FER) system by combining various techniques from computer vision and pattern recognition.

Expression recognition is closely related to face recognition where a lot of research has been done and a vast array of algorithms have been introduced. FER can also be considered as a special case of a pattern recognition problem and many techniques are available. In the designing of an FER system, we can take advantage of these resources and use existing algorithms as building blocks of our system. So a major part of this work is to determine the optimal combination of algorithms. To do this, we first divide the system into 3 modules, i.e. Preprocessing, Feature Extraction and Classification, then for each of them some candidate methods are implemented, and eventually the optimal configuration is found by comparing the performance of different combinations.

Another issue that is of great interest to facial expression recognition systems designers is the classifier which is the core of the system. Conventional classification algorithms assume the image is a single variable function of a underlying class label. However this is not true in face recognition area where the appearance of the face is influenced by multiple factors: identity, expression, illumination and so on. To solve this problem, in this thesis we propose two new algorithms, namely Higher Order Canonical Correlation Analysis and Simple Multifactor Analysis which model the image as a multivariable function.

The addressed issues are challenging problems and are substantial for developing a facial expression recognition system.

## Acknowledgements

I would like to express my best wishes to my advisor Prof. Dr. Fakhri Karray for his motivation, guidance, suggestions, and support anytime in the study.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Face recognition is a task that humans perform routinely and effortlessly in their daily lives. Robert Axelrod has also shown the ability to recognize those they have met before and distinguish them from strangers is one of the bases for humans to form cooperation [3]. The last decade has witnessed a trend towards an increasingly ubiquitous computing environment, where powerful and low-cost computing systems are being integrated into mobile phones, cars, medical instruments and almost every aspect of our lives. This has created an enormous interest in automatic processing of digital images and videos in a number of applications, including biometric authentication, surveillance, human-computer interaction, and multimedia management. Research and development in automatic face recognition follows naturally.

Face recognition is a visual pattern recognition problem where a three-dimensional object is to be identified based on its two-dimensional image. in recent years, significant progress has been made in this area; owing to better face models and more powerful computers, face recognition system can achieve good results under constrained situations. However because face images are influenced by several factors: illumination, head pose, expression and so on, in general conditions, face recognition is still challenging. From a computer vision point of view, among all these "noises" facial expression maybe the toughest one in the sense that expressions actually change the three-dimensional object while other factors, such as illumination and position, only affect imaging parameters. To get rid of expression “noise”, one first needs to estimate the expression of an image, this is called “Facial Expression Recognition”.

Another, maybe more important motivation of facial expression recognition is that expression itself is an efficient way of communication: it’s natural, non-intrusive, and [62] has shown that, surprisingly, expression conveys more information than spoken words and voice tone. To build a friendlier Human Computer Interface, expression recognition is essential.

## 1.2 Applications

Facial expression recognition can be useful in many areas, for research and application. Studying how humans recognize emotions and use them to communicate information are important topic in anthropology. And the emotion automatically estimated by a computer is considered to be more objective than those labeled by people and it can be used in clinical psychology, psychiatry and neurology.

As mentioned earlier, expression recognition can be embedded into a face recognition system to improve its robustness. In a real-time face recognition system where a series of images of an individual are captured, FER module picks the one which is most similar to a neutral expression for recognition, because normally a face recognition system is trained using neutral expression images. In the case where only one image is available, the estimated expression can be used to either decide which classifier to choose or to add some kind of compensation.

In a Human Computer Interface, expression is a great potential input. This is especially true in voice-activated control systems. Experiments in [62] show when people are speaking, 55% of communication happens via expression whereas only 7% happens via spoken words. This implies a FER module can markedly improve the performance of such systems. Customers’ facial expressions can also be collected by service providers as implicit user feedback to improve their service. Compared to a conventional questionnaire-based method, this should be more reliable and furthermore, has virtually no cost.

In the computer graphic area, facial expression estimated from real images can be used to animate synthetic characters [13]. This technique is useful in video telephony where bandwidth is limited. Instead of transmitting the video, we can just send the facial expression sequence, using which the original video can be reconstructed. Another application of this technique is in the movie industry where they use this to produce high quality computer animation.

There are some other possible applications, including emotion surveillance for employees in high work intensity industry (*e.g.*, long distance drivers), pain assessment, image and video database management and searching, lie detection and so on.

## 1.3 Objectives

The primary goal of this research is to design, implement and evaluate a novel facial expression recognition system using various statistical learning techniques. This goal will be realized through the following objectives:

1. System level design: In this stage, we'll be using existing techniques in related areas as building blocks to design our system.
    - A facial expression recognition system usually consists of multiple components, each of which is responsible for one task. We first need to review the literature and decide the overall architecture of our system, *i.e.*, how many modules it has, the responsibility of each of them and how they should cooperate with each other.
    - Implement and test various techniques for each module and find the best combination by comparing their accuracy, speed, and robustness.
2. Algorithm level design: Focus on the classifier which is the core of a recognition system, trying to design new algorithms which hopefully have better performance compared to existing ones.

## 1.4 Thesis Overview

The remainder of this thesis is structured as follows:

**Chapter 2** reviews the state of the art of facial expression recognition, first by dividing the system into modules, and then by describing and analyzing the various contributions to date of each module. As an important part of this area, face databases are also discussed in this chapter.

**Chapter 3** gives an overview of the system architecture and talks about the image preprocessing module. Face detection, alignment and normalization are performed in this module and for each of them 1 or 2 algorithms are implemented.

**Chapter 4** deals with feature extraction and selection. The Gabor filter and the Active Appearance Model are discussed: the former, which is a type of wavelet transformation, can be considered as a local feature, whereas the latter one is more holistic. Because sometimes the dimensionality of the features is too high, feature selection is employed to determine the most discriminating ones.

**Chapter 5** elaborates on classification algorithms and finalizes the system configuration. Two classification techniques: Support Vector Machine, and Higher Order Singular Value Decomposition, are implemented and we test their performance on different features. Based on the experimental results, the final system configuration is determined.

**Chapter 6** proposes two new higher order methods: Higher Order Canonical Correlation Analysis and Simple Multifactor Analysis. The former tries to make use of the training data more efficiently, while the latter attempts to simplify the computation under certain conditions.

**Chapter 7** summarizes the contributions of this thesis and introduces the focus of future research.

# Chapter 2

# Background and Literature Review

The importance of facial expression in social interaction and social intelligence is widely recognized. Facial expression analysis has been an active research topic since 19th century. The first automatic facial expression recognition system was introduced in 1978 by Suwa *et al.* [83]. This system attempts to analyze facial expressions by tracking the motion of 20 identified spots on an image sequence. Since then, a lot of work has been done in this domain. Various computer systems have been made to help us understand and use this natural form of human communication.

This chapter reviews the state of the art of what has been done in processing and understanding facial expression. When building an FER system, these main issues must be considered: face detection and alignment, image normalization, feature extraction, and classification. Most of the current work in FER is based on methods that implement these steps sequentially and independently. Before exploring what has been done in literature for implementing these steps, we will briefly describe the problem space for facial expression analysis.

## 2.1 Problem Space for Facial Expression Analysis

### 2.1.1 Level of Description

In general there are two types of method to describe facial expression.

## Facial Action Coding System

The facial action coding system [24] is a human-observer-based system widely used in psychology to describe subtle changes in facial features. FACS consists of 44 action units which are related to contraction of a specific set of facial muscles (Fig.2.1). Some of the action units are shown in Fig.2.2. Conventional, FACS code is manually labeled by trained observers while viewing videotaped facial behavior in slow motion. In recent years, some attempts have been made to do this automatically [69]. The advantage of FACS is its ability to capture the subtlety of facial expression, however FACS itself is purely descriptive and includes no inferential labels. That means in order to get the emotion estimation, the FACS code needs to be converted into the Emotional Facial Action System (EMFACS [28]) or similar systems.

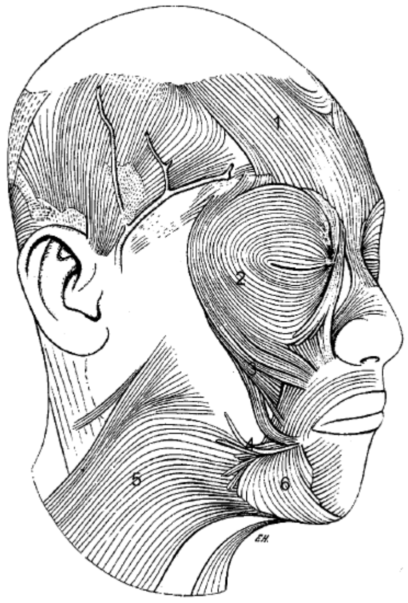
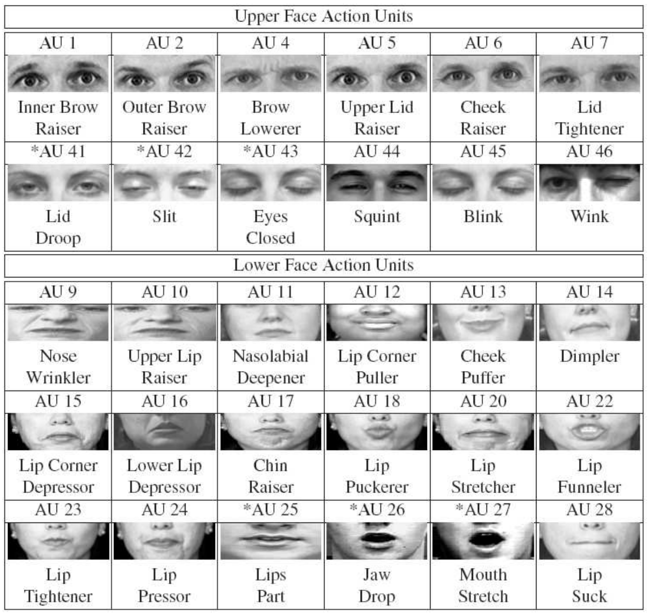


Figure 2.1: Muscles of facial expression. 1, frontalis; 2, orbicularis oculi; 3, zygomaticus major; 4, risorius; 5, platysma; 6, depressor anguli oris [33]



Upper Face Action Units

| AU 1 | AU 2 | AU 4 | AU 5 | AU 6 | AU 7 |
|---|---|---|---|---|---|
| Inner Brow Raiser | Outer Brow Raiser | Brow Lowerer | Upper Lid Raiser | Cheek Raiser | Lid Tightener |
| *AU 41 | *AU 42 | *AU 43 | AU 44 | AU 45 | AU 46 |
| Lid Droop | Slit | Eyes Closed | Squint | Blink | Wink |

Lower Face Action Units

| AU 9 | AU 10 | AU 11 | AU 12 | AU 13 | AU 14 |
|---|---|---|---|---|---|
| Nose Wrinkler | Upper Lip Raiser | Nasolabial Deepener | Lip Corner Puller | Cheek Puffer | Dimpler |
| AU 15 | AU 16 | AU 17 | AU 18 | AU 20 | AU 22 |
| Lip Corner Depressor | Lower Lip Depressor | Chin Raiser | Lip Puckerer | Lip Stretcher | Lip Funneler |
| AU 23 | AU 24 | *AU 25 | *AU 26 | *AU 27 | AU 28 |
| Lip Tightener | Lip Pressor | Lips Part | Jaw Drop | Mouth Stretch | Lip Suck |

Figure 2.2: FACS action units [35]

**Prototypic Emotional Expressions**

Instead of describing the detailed facial features, most FER systems attempt to recognize a small set of prototypic emotional expressions. The most widely-used set is perhaps human universal facial expressions of emotion which consists of six basic expression categories that have been shown to be recognizable across cultures (Fig.2.3).

These expressions, or facial configurations have been recognized in people from widely divergent cultural and social backgrounds [25], and they have been observed even in the faces of individuals born deaf and blind [23].

These 6 basic emotions, *i.e.*, disgust, fear, joy, surprise, sadness and anger plus "neutral" which means no facial expression are considered in this work. Given a facial image, our system either works as a conventional classifier to determine the most likely emotion or estimates the weights (or possibility) of each emotion as a fuzzy classifier does.

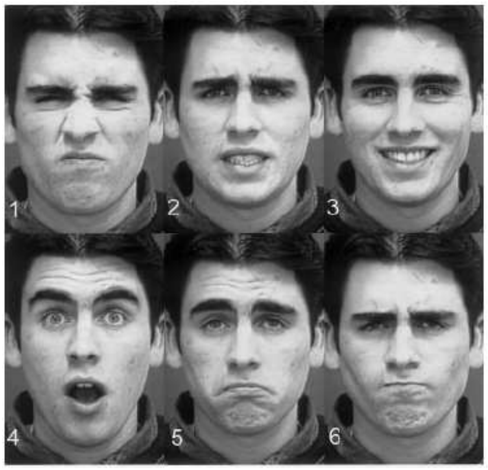


Figure 2.3: Basic facial expression phenotypes. 1, disgust; 2, fear; 3, joy; 4, surprise; 5, sadness; 6, anger [38]

## 2.2 System Structure

FER can be considered as a special face recognition system or a module of a face recognition system. So it should be instructive to look at the general architecture of a face recognition system. Normally, it consists of four components as depicted in Fig.2.4.

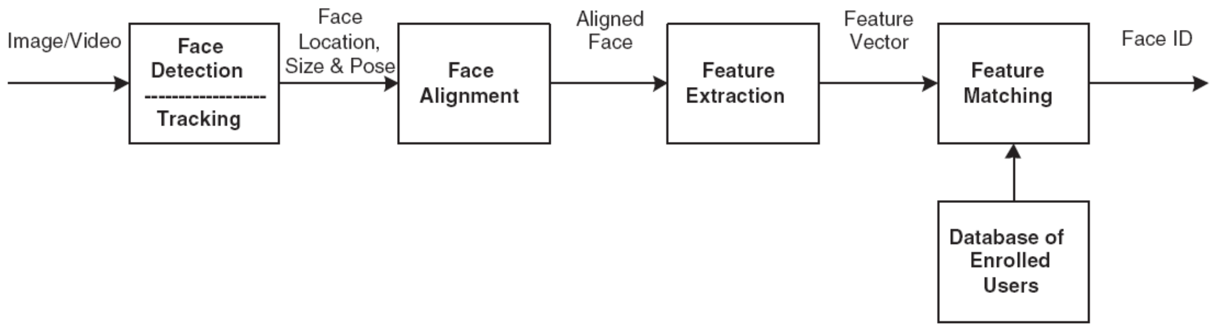


Figure 2.4: Face recognition processing flow [35]

Face detection finds the face areas in the input image. If the input is a video, to be more efficient and also to achieve better robustness, face detection is only performed on key frames and a tracking algorithm is applied on interval frames. Face alignment is very similar to detection, but it is aimed at achieving a more accurate localization. In this step, a set of facial landmarks (facial components), such as eyes, brows and nose, or the facial contour are located; based on that, the face image is rotated, chopped, resized and even warped, this is called geometrical normalization. Usually the face is further normalized with respect to photometrical properties such as illumination and gray scale.

Feature extraction is performed on a normalized face to provide effective information that should be useful for recognizing and classifying labels in which there is interest, such as identity, gender, or expression. The extracted feature vector is sent to a classifier and compared with the training data to produce a recognition output.

## 2.3 Face Detection

Face detection is the first step in face recognition. It has a major influence on the performance of the entire system [35]. Several cues can be used for face detection, for example, skin color, motion (for videos), facial/head shape, and facial appearance. Most successful face detection algorithms are based on only appearance [35]. This may be because appearance-based algorithms avoid difficulties in modeling 3D structures of faces. However, the variations of 3D structures due to facial expression and head pose actually heavily affect the facial appearance and make the face/non-face boundary highly complex [7]. To deal with this, a vast arrange of methods have been proposed since the 1990s.

Turk and Pentland [87] describe a detection system based on eigen decomposition which is also known as principal component analysis (PCA). In their method, an image is represented by an average face plus a set of weighted "eigenfaces". Whereas only the face images are considered in eigenface, Sung and Poggio [82] also consider the distribution of non-face images and apply Bayes' rule to obtain a likelihood estimation. Rowley *et al.* [72] use neural networks and Osuna *et al.* [68] trained a Kernel Support Vector Machine to classify face and non-face images. In these systems, a bootstrap algorithm is iteratively used to collect meaningful examples for retraining the detector.

Schneiderman and Kanade [75] use AdaBoost learning to construct a classifier based on wavelet representation of the image. This method is computationally expensive because of the wavelet transformation. To overcome this problem, Viola and Jones [93] replace wavelets with Haar features, which can be computed very efficiently [20] [80]. Their system is the first realtime frontal-view face detector [94].

Under Viola's framework, some improvements have been proposed. Lienhart *et al.* [52] use rotated Haar features to deal with in-plane rotation. Li *et al.* [51] [50] [52] propose a multiview face detection system which can also handle out-of-plane rotation using a detector-pyramid.

In the following sections, we will describe two face detection algorithms: Eigenface is one of the simplest methods and Viola's framework may be the most successful one. AdaBoost learning is an important component in Viola's framework and this algorithm will also be useful in the feature extraction module, so our presentation focuses on this part.

### 2.3.1 Eigenface and Template Matching

Eigenface assumes the face image $x = (x_1, x_2, \ldots, x_N)$ is amenable to a multivariate normal distribution from which the training images are identically independently drawn. This distribution can be described by the following probability density function:

$$f(x_1, x_2, \ldots, x_N) = \frac{1}{(2\pi)^{N/2} |\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)) \tag{2.1}$$

where $\Sigma$ is the covariance matrix and $\mu$ is the expectation of $x$.

Eigenface decomposes $\Sigma$ using eigen decomposition as

$$\Sigma = USU^T \tag{2.2}$$

where $U$ is a unitary matrix and $S = diag(s_1^2, s_2^2, \ldots, s_N^2)$ is a diagonal matrix with all elements non-negative. Each column of $U$, $U_i$, is called an Eigenface. A face image $x$ can be represented by $\mu$ and $U_i$ as

$$x = \mu + \sum_i a_i U_i \tag{2.3}$$

It can be shown that $\frac{a_i}{s_i}$ are i.i.d. standard normal variables. So the probability density function of $x$ is:

$$f(x) = \prod_i -\frac{1}{(2\pi)^{1/2}} \exp(\frac{1}{2}\frac{a_i^2}{s_i^2}) = \frac{1}{(2\pi)^{N/2}} \exp(-\frac{1}{2}\sum_i \frac{a_i^2}{s_i^2}) \tag{2.4}$$

Equation (2.4) can be used as a probability estimate and we can define a distance measure according to

$$D = \sum_i \frac{a_i^2}{s_i^2} \tag{2.5}$$

which is called normalized Euclidean distance. A large $D$ implies a small probability of being a face image and vice versa. Based on (2.5) Turk and Pentland [87] built a face detection system. Sung and Poggio's paper [82] used a similar idea, they assume images are produced by a mixture of Gaussian models: a face image Gaussian and a non-face image Gaussian. So they also estimate the probability of $x$ of being a non-face image $f^*(x)$ and the final decision is made using a Bayesian classifier.

If we further assume $\Sigma$ is an identity matrix, (2.5) degenerates into Euclidean distance which means the probability density function is controlled by, $|x-\mu|^2$, the variation of the image from the average face $\mu$. This gives the simplest detection algorithm: template matching, *i.e.*, finding a "face template" $\mu$, and then for each $x$ determine whether it is a face image by thresholding $|x-\mu|$ .

### 2.3.2 Viola's Framework

Almost all the state of the art face detection systems are developed upon Viola's Framework, based on AdaBoost and Haar features. The philosophy of AdaBoost is that if it is hard to find a good (strong) classifier directly, we can construct a lot of poor quality ones (weak classifier) $h_m(x)$ ,$m = 1, \ldots, M$ and use the combination of them to form a strong classifier [27]:

$$H_M(x) = \frac{\sum_m a_m h_m(x)}{\sum_m a_m} \tag{2.6}$$

where $a_m \geq 0$ are the combining coefficients. In the discrete version $h_m(x)$ gives either $-1$ or 1 whereas in the real version, the output can be a real number. Because we need a lot of weak classifiers, $h_m(x)$ is normally chose to have a simple form (so it is easy to construct). In Viola and Jones's work [93] [94], they use threshold classifiers, each of which works on one feature selected from an over-complete set of Haar wavelet-like features. We'll first introduce Haar Wavelet, then give the iterative algorithm of AdaBoost.

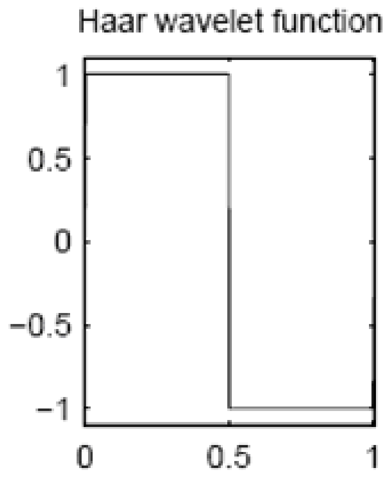
#### Haar Wavelet

Wavelet analysis is a tool for signal processing which can perform local analysis. It is useful in face detection because we want to focus on a localized area of the image and find whether it contains a face. Haar wavelet is the first known wavelet, and probably the simplest one. A one-dimensional Haar wavelet function is just a step function, as shown in Fig.2.5(a).

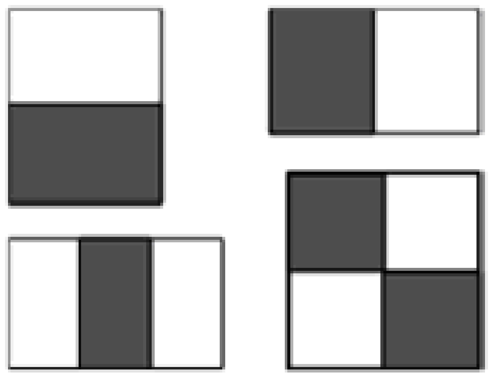
In image processing, two-dimensional wavelets are used which look like the ones in Fig.2.5(b). The functions in Fig.2.5(b) don't satisfy some conditions of a wavelet (because we don't need those in our application), so strictly speaking they are "Haar-like features".

An important property of Haar wavelet-like features is that they can be computed efficiently using an integral image [93]. The integral image $II(x, y)$ of image $I(x, y)$ is defined by:

$$II(x, y) = \sum_{x'<x, y'<y} I(x', x') \tag{2.7}$$



(a) 1-dimensional Haar wavelet function



(b) Four types of 2-d Haar wavelet-like features [35]

Figure 2.5: Haar wavelet and Haar features

This can be computed in one pass over the original image as follows [35]:

$$S(x,y) = S(x,y-1) + I(x,y) \tag{2.8}$$

where $S(x,y)$ is the cumulative sum of the $x$th row. Apparently $S(x,0) = 0$ and $II(0,y) = 0$. Any rectangular sum in an image can be expressed in terms of its integral image as illustrated in Fig.2.6.

The sum of the pixels within rectangle $D$ in Fig.2.6 can be computed using the integral image value on points $a$, $b$, $c$, $d$ as follows [93]:

$$II(x2,y2) = \sum_{x<x2,y<y2} I(x,y) = A + B + C + D \tag{2.9}$$

$$II(x2,y1) = \sum_{x<x2,y<y1} I(x,y) = A + C \tag{2.10}$$

$$II(x1,y2) = \sum_{x<x1,y<y2} I(x,y) = A + B \tag{2.11}$$

$$II(x1,y1) = \sum_{x<x2,y<y2} I(x,y) = A \tag{2.12}$$

$$D = II(x2,y2) + II(x1,y1) - II(x2,y1) - II(x1,y2) \tag{2.13}$$

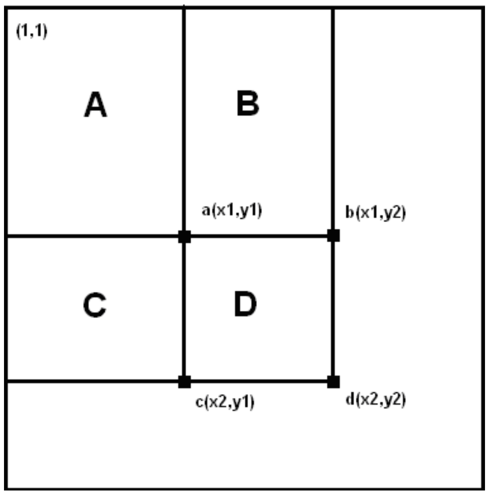


Figure 2.6: Compute rectangular sum using integral image

The use of integral images leads to enormous savings and makes it possible to use Haar wavelets in a real-time detection system.

### Constructing Weak Classifier

In Viola's Framework, "the AdaBoost learning procedure is used to solve the following three fundamental problems: 1) learning effective features from a large feature set; 2) constructing weak classifiers, each of which is based on one of the selected features; and 3) boosting the weak classifiers to construct a strong classifier" [35]. We'll talk about the first in this section.

Suppose a set of $N$ labeled training examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ is given, where $y_i = \{1, -1\}$ is the label of image $x_i$. We also assume a weight $w_i$ is assigned to each example, and how to compute this weight will be given in the next section. $K$ Haar features $F_k$, $k = 1, 2, \ldots K$ can be computed for each image; in Viola's detection system constructing a weak classifier means determining a feature $F_{k_m}$, a direction $b_m = \{1, -1\}$ and a threshold $f_m$:

$$h_m = \begin{cases} +1 & if\ (F_{k_m} - f_m) b_m > 0 \\ -1 & otherwise \end{cases} \tag{2.14}$$

This simple classifier is called a "stump". We choose the three parameters $k_m$, $b_k$ and

$f_k$ to minimize some objective function, for example weighted classification error. This optimization can be done in two steps: for each $F_k$ , find the best $b_k$, $f_k$ and corresponding weighted error $err_k$; then choose $(k_m, b_k, f_k)$ which corresponds to the smallest $err_k$.

### Boosting Strong Classifier

AdaBoost iteratively learns a sequence of weak classifiers $h_m$ and constructs a strong one $H_M$ using their linear combination. In each cycle AdaBoost does two things: find the best linear combination coefficients, and update the sample weights. Both of these tasks are based on the upper bound on classification error of $H_M$. [74] shows that the bound can be derived from the following exponential loss function:

$$J(H_M) = \sum_i e^{-y_i H_M(x_i)} = \sum_i e^{-y_i \sum\limits_m a_m h_m(x_i)} \tag{2.15}$$

Given the current strong classifier $H_{M-1}(x) = \sum\limits_{m=1}^{M-1} a_m h_m(x)$ and the newly learned weak classifier $h_M$, the best combining coefficient $a_M$ is the minimizer of the following optimization problem:

$$a_M = \underset{a}{\operatorname{argmin}}\, J(H_{M-1}(x) + a h_M(x)) \tag{2.16}$$

It can be shown that

$$a_M = \log \frac{1 - err_M}{err_M} \tag{2.17}$$

where $err_M$ is the weighted error of $h_M$ defined as

$$err_M = \sum_i w_i^{(M-1)} \frac{|\mathrm{sgn}(h_M(x_i)) - y_i|}{2} \tag{2.18}$$

Equation (2.17) suggests if the new weak classifier has a good performance (small $err_M$), we'll give it a large $a_M$; otherwise it will be assigned a small coefficient.

To update the weight of each training sample, let's rewrite (2.16) as follows:

$$J(H_M) = \sum_i e^{-y_i \sum\limits_m a_m h_m(x_i)} = \sum_i e^{-y_i H_{M-1}(x_i)} e^{-y_i a_M h_M(x_i)} \tag{2.19}$$

Equation (2.19) means the contribution in $J(H_M)$ of the $i$th example is the error of $h_M(x)$ on $x_i$, $e^{-y_i a_M h_M(x_i)}$, multiplied by $e^{-y_i H_{M-1}(x_i)}$. So we can define the weight of the $i$th example after the $M$th iteration as:

$$w^{(M)}(x_i) = e^{-y_i H_M(x_i)} \tag{2.20}$$

Intuitively this is saying if the current strong classifier already can classify an example $x_i$ correctly, then it doesn't really matter what performance the new weak classifier can achieve on $x_i$; on the other hand, if $H_{M-1}(x)$ doesn't work well on $x_i$, we can expect $h_M(x)$ to fix this problem.

So the updated formula of $w^{(M)}(x_i)$ is :

$$w^{(M)}(x_i) = w^{(M-1)}(x_i) e^{-y_i a_M h_M(x_i)} \tag{2.21}$$

Usually, the training set is unbalanced—the number of training images with $y_i = 1$ is larger(smaller) than the number of images with $y_i = -1$. In such cases, the $y_i = 1$ ($y_i = -1$) set will dominate the training process. To eliminate this asymmetry, one way is to initialize the sample weights such that:

$$\sum_i w_i^{(0)} y_i = 0 \tag{2.22}$$

The algorithm is summarized in Alg.1.

## 2.4 Face Alignment

The aim of face alignment is to achieve more accurate localization of the face, and usually the facial components (feature points). A typical result of face detection and face alignment is compared in Fig.2.7. As we can see, detection considers the images in terms of areas whereas alignment has a precision of pixels.

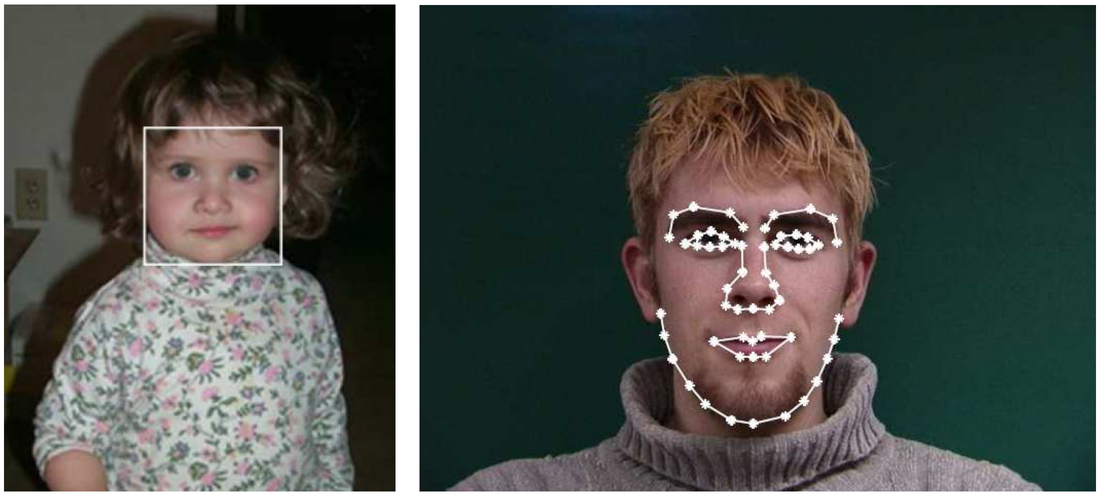
Various algorithms have been proposed since 1990's. Gu *et al.* [29] use histogram information to localize mouth corners and eye corners. In [26], Ian and Marian implement a preliminary alignment system using Gabor filter to detect pupils and philtrum. Curve fitting algorithms, especially Active Shape Model [18] and its offspring may be the most

```
**Input**:
Training examples $Z = \{(x_1, y_1), \dots, (x_N, y_N)\}$, where $N = a + b$, of which $a$
examples have $y_i = 1$ and $b$ examples have $y_i = -1$
The number $M$ of weak classifiers to be combined.
**Initialization**:
    $w_i^{(0)} = \begin{cases} \frac{1}{2a} & for\ those\ examples\ with\ y_i = 1 \\ \frac{1}{2b} & otherwise \end{cases}$
**end**
**Forward inclusion**:
    **for** $m = 1, \dots, M$ **do**
        Choose optimal $h_m$ to minimize the weighted error
        Choose $a_m$ according to (2.17)
        Update $w_i^{(m)} \leftarrow w_i^{(m-1)} \mathrm{e}^{-y_i a_m h_m(x_i)}$ and normalize to $\sum_i w_i^{(m)} = 1$
    **end**
**end**
**Output**:
Classification function: $H_M(x)$ as in (2.20)
Class label prediction: $\hat{y}(x) = \mathrm{sgn}(H_M(x))$
```

**Algorithm 1**: AdaBoost learning algorithm [35]

successful alignment methods nowadays. Cootes *et al.* [18] [45] [16] propose Active Shape Model and apply it to face image. After that, ASM is widely used in face image processing; a great amount of effort has been made to improve its speed, accuracy and robustness. In [36] the Active Shape Model is combined with the Gabor filter; Li *et al.* [78] propose Direct Appearance Model; [56] and [101] enhance ASM by using 2-D local textures feature for local search.

This section will mainly focus on curve-fitting types of method, in particular Active Shape Model and its ancestor Active Contour Model.



(a) Face detection [35] (b) Face alignment [66]

Figure 2.7: Detection Vs. Alignment

### 2.4.1 Curve Fitting

The basic problem of curve fitting is to locate the contour of an object in an image. Most curve fitting methods (explicitly or implicitly) consider two "forces": internal force (elastic force) caused by deformation and external force (image force) caused by density gradient. As in physics, both internal force and external force cause potential energy, and the curve fitting is achieved by minimizing the overall energy functional (Fig.2.8).

In general, a contour can be represented by $c = c(s)$, parameterized by its arc length, $s$. The energy function can be expressed by:

$$\varepsilon = E_{elastic}(c) + E_{image}(c) = \int_c (a(s)e_{elastic}(c,s) + b(s)e_{image}(c,s))ds \tag{2.23}$$

So the optimization problem can be expressed as

$$\operatorname*{argmin}_c E_{elastic}(c) + E_{image}(c) \tag{2.24}$$

Usually, (2.24) is solved using iterative searching algorithms. A large number of curve fitting methods have been proposed using a wide range of energy functions and searching schemes.

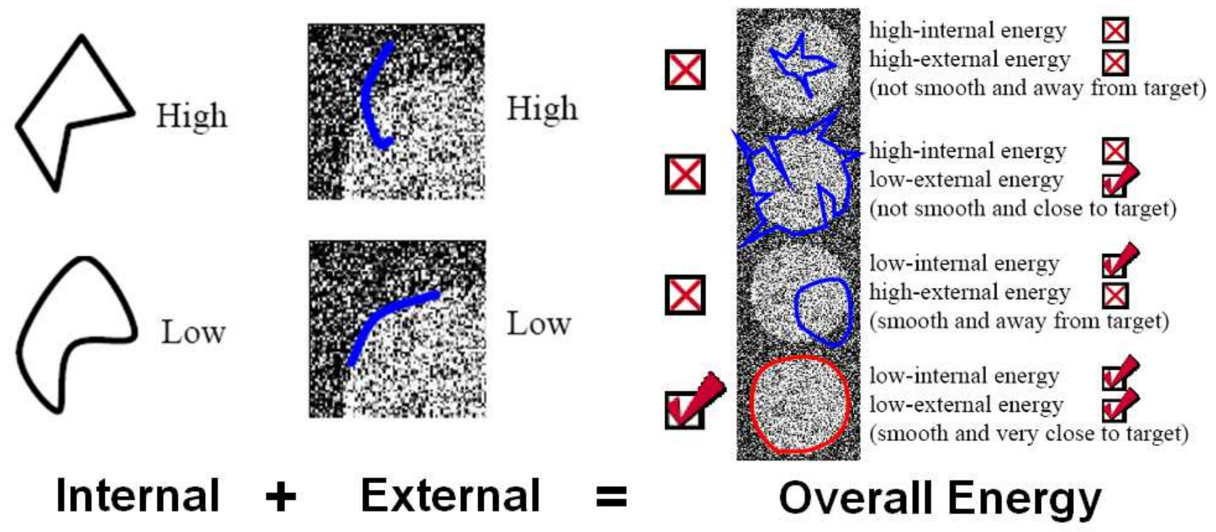


Figure 2.8: Energy function and curve fitting

It's worth pointing out that sometimes the energy function is not explicitly given, instead it's implicitly defined in the searching scheme, for example, in Active Shape Model.

## 2.5 Feature Extraction

"Feature extraction converts pixel data into a higher-level representation of shape, motion, color, texture, and spatial configuration of the face or its components. The extracted representation is used for subsequent classification. Feature extraction generally reduces the dimensionality of the input space. The reduction procedure should (ideally) retain essential information possessing high discrimination power and high stability" [14]. In the face recognition area, various features have been used.

The coefficients of Eigenface can be used as features and recently, an extension of Eigenface [90] defines Tensorface which has shown a promising choice of feature. Active Appearance Model [17] decomposes the facial image into "shape" and "texture". The shape vector which is coded using ASM describes the contour of the facial components, whereas the texture vector gives the "shape-free" facial texture. Matsuno *et al.* [41] extract features using a two-dimensional mesh, called Potential Net. All the above-mentioned methods are considered as holistic features, because they are related to the overall structure of the image. There is another kind of features called local features, each of which focuses only on a small region. The most straightforward idea may be to directly use image sub-windows as local features: for example, in [15] Colmenarez *et al.* use nine sub-windows located around the facial components. Wavelet filters have been used too, the most popular of which is the Gabor filter which has been shown [22] [37] to be a reasonable model of visual processing in primary visual cortex. Yin and Wei use topographic primitive features to represent faces [103]. In [104], instead of defining features ahead of time, Yu and Bhanu use a evolutionary algorithm to generate features automatically. For video-based FER, the dynamic of expression can also serve as features. [44] proposes Geometric Deformation Feature which represents the geometrical displacement of certain selected landmark nodes. In [1], Aleksic and Katsaggelos use Facial Animation Parameters which are based on Active Shape Model.

In this section, we'll briefly introduce some of these feature extraction methods

## 2.5.1 Tensorface

Tensorface [90] is a multilinear extension of Eigenface. Instead of representing the face image using a linear equation

$$x = \bar{x} + \sum_i a_i x_i \tag{2.25}$$

It models the face by a multilinear system which is equivalent to

$$x = \bar{x} + \sum_{i_1} \dots \sum_{i_N} a_{i_1} \dots a_{i_N} x_{i_1 \dots i_N} \tag{2.26}$$

Compared to Eigenface which ignores the label of images, Tensorface analyzes a face ensemble with respect to its underlying factors(labels): for example, identities, views, and illuminations. The “principal components” in this multilinear system are referred to as Tensorfaces which are shown in Fig.2.9. The Tensorface coefficients $a_{i_k}$ can be used as features in a recognition task, and because the original image can be reproduced using 2.26, Tensorface coefficients can also be used for image synthesis where we first generate a set of coefficients, then use them to synthesize images.

Since Tensorface is shown to be a promising method in face recognition, some improvements have been proposed. [92] proposes Multilinear Independent Component Analysis where they try to find the independent directions of variation. In [76] Shashua *et al.* introduce Non-Negative Tensor Factorization which is a generalization of Non-negative Matrix Factorization.

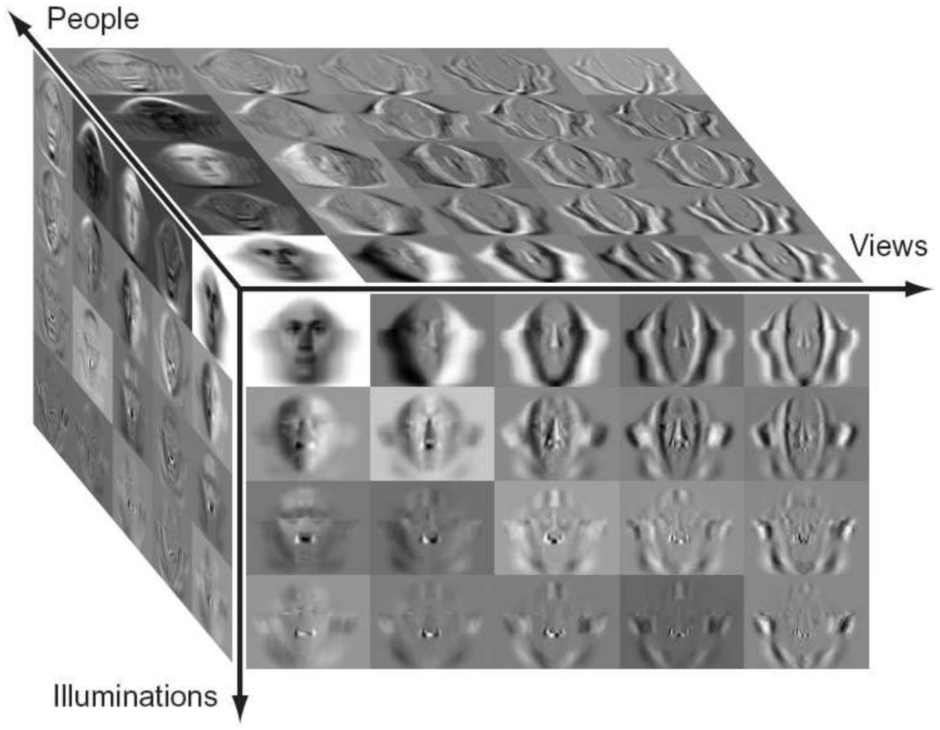


Figure 2.9: A partial visualization of TensorFaces bases for an ensemble of 2,700 facial images spanning 75 people, each imaged under 6 viewing and 6 illumination conditions [92]

### 2.5.2 Potential Net

Matsuno *et al.* [61] [41] propose Potential Net to extract facial features. As shown in Fig.2.10, Potential Net is a two dimensional mesh of which nodes are connected to their four neighbors with springs, while the most exterior nodes are fixed to the frame of the Net. Similar to curve fitting, Potential Net considers two forces: each node in the mesh is driven by the external forces which come from the image gradient and the elastic forces of springs propagate local deformation throughout the Net. Eventually, equilibrium is reached, and the nodal displacements represent the overall pattern of facial image.

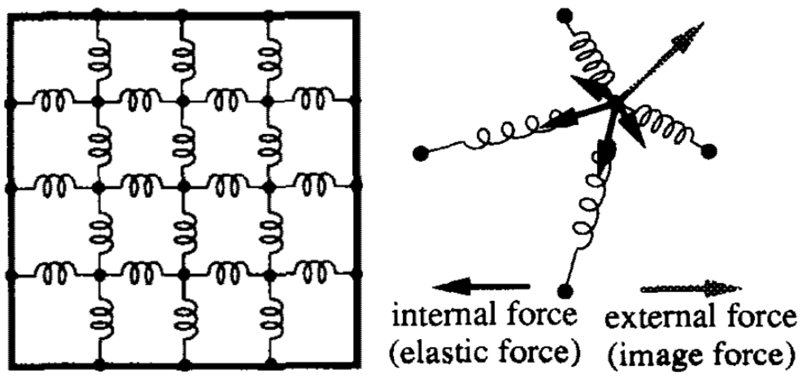


Figure 2.10: Structure of Potential Net and nodal deformation [61]

### 2.5.3 Active Appearance Model

Cootes *et al.* propose Active Appearance Model [17] to represent objects which have deformable shape as well as diverse textures. In Active Appearance Model, the shape variation, texture variation and their interdependence are modeled using three linear systems respectively as depicted in Fig.2.11. The shape of the facial contour is coded into a shape vector $V_{shape}$ by applying Active Shape Model, then the texture is warped to have identical shape. This "shape-free texture" is converted into a texture vector $V_{texture}$ using PCA, finally $V_{texture}$ is concatenated with $V_{shape}$ and translated into AAM coefficient.

AAM coefficient has fairly small dimensionality, normally 30-40 depending on the training parameters; also it is intuitive and easy to visualize. It has some limitations too, the major one is that AAM is purely linear, which means it is incapable of modeling nonlinear variation of the facial image.

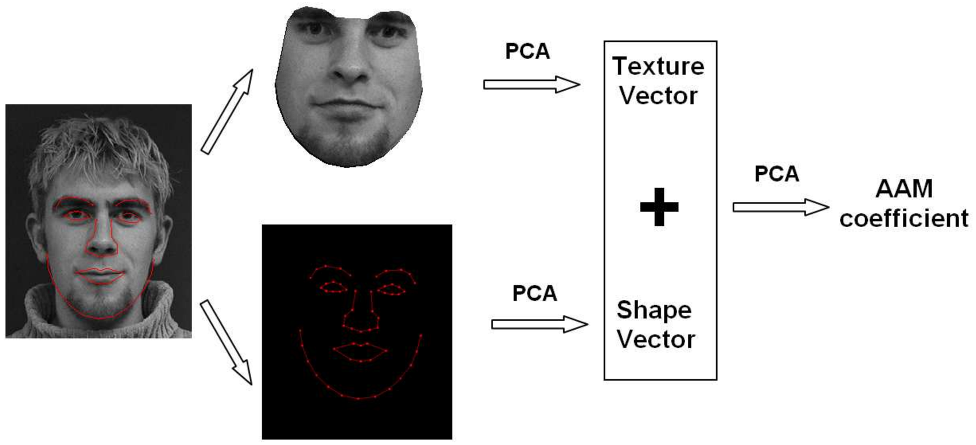


Figure 2.11: Active Appearance Model processing flow

### 2.5.4 Sub-window Features

In [15] the face is divided into nine facial features grouped in 4 regions, *i.e.*, Right eyebrow, Left eyebrow, Eyes&Nose and Mouth. The appearance of each facial feature is provided by the image sub-window located around its position. Fig.2.12 illustrates the four facial regions and the nine facial features.

## 2.6 Classification

A wide range of classifiers have been applied to the automatic expression recognition problem: In [61], Matsuno *et al.* classify expression by thresholding the Normalized Euclidean distance in the feature space. [15] implements a Bayesian recognition system where they find the facial expression that maximizes the likelihood of a test image. Other methods which have been used in facial expression recognition include Fisher discrimination analysis [77], Locally Linear Embedding [100], Higher Order Singular Value Decomposition [95] and so on [103] [1] [44]. Among them the most successful ones are Neural Network [60] [106] [42] [34] and Support Vector Machine [102] [63] [4]. All the above-mentioned methods are widely-used in statical learning, so we only briefly introduce Support Vector Machine because it will be used in other systems.

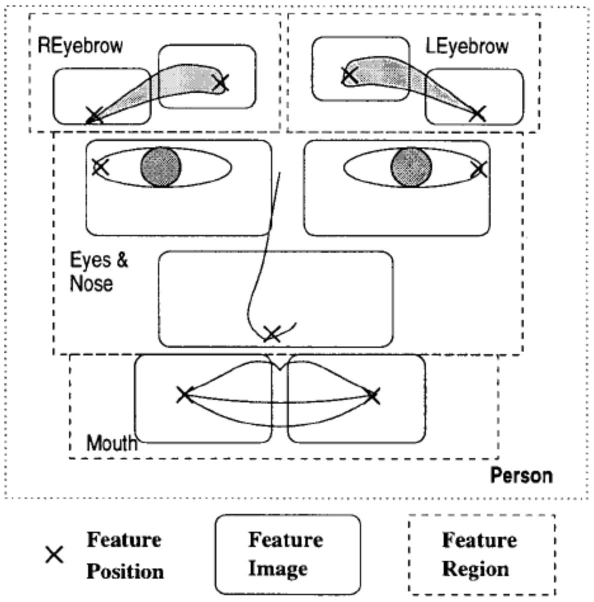


Figure 2.12: Scheme of the Facial Features and Regions [15]

### 2.6.1 Support Vector Machine

Support Vector Machine attempts to construct a linear classifier which maximizes the margin between two classes, so it's also known as Optimal Margin Classifier [9]. Fig.2.13 gives an SVM classifier where $\frac{1}{|w|}$ gives the margin and samples along the hyper-planes are called the support vectors.

It has been proven that SVM minimizes the Structural Risk Function which is considered as a better error estimation than the normally-used Empirical Risk Function in terms of generalization capacity.

We consider data points of the form: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $y_i$ is either 1 or $-1$, a label denoting the class to which the point $x_i$ belongs. The basic version of SVM can be written as

$$\begin{aligned} \underset{w,b_0}{\operatorname{argmax}} \quad & \frac{1}{\|w\|} \\ s.t. \quad & y_i(x^T w - b_0) \geq 1, \quad \forall i \end{aligned} \tag{2.27}$$

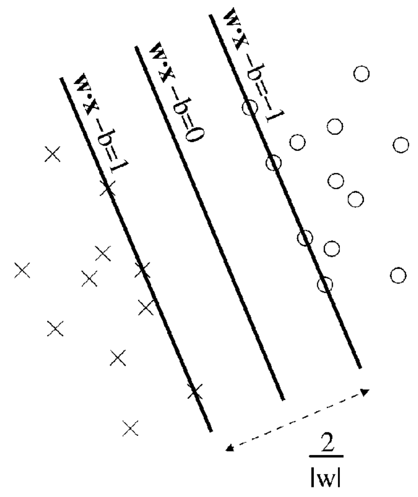


Figure 2.13: Maximum-margin hyper-planes for a SVM trained with samples from two classes [99]

$\|\ldots\|$ in (2.27) can be replaced by any distance measure. If norm-2 is used, the problem is equivalent to

$$\begin{aligned} \underset{w,b_0}{\operatorname{argmin}} \quad & \frac{1}{2}\|w\|^2 \\ s.t. \quad & y_i(x^T w - b_0) \geq 1, \quad \forall i \end{aligned} \tag{2.28}$$

Equation (2.28) is a quadratic programming and according to the strong duality theorem it can be converted to:

$$\begin{aligned} \underset{\alpha_i}{\operatorname{argmax}} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j \\ s.t. \quad & \alpha_i \geq 0 \quad \forall i \end{aligned} \tag{2.29}$$

which is called the dual problem of (2.28).In practice, we always use (2.29) as it is easier to handle numerically. Moreover, in (2.29) all the computation of $x_i$ is written in terms of inner product, and that means it can be generalized to a nonlinear case by employing Kernel technique.

## 2.7 Face Database

"Because of its nonrigidity and complex three-dimensional structure, the appearance of a face is affected by a large number of factors including identity, face pose, illumination, facial expression, age, occlusion, and facial hair. The development of algorithms robust to these variations requires databases of sufficient size that include carefully controlled variations of these factors. Furthermore, common databases are necessary to comparatively evaluate algorithms. Collecting a high quality database is a resource-intensive task: but the availability of public face databases is important for the advancement of the field" [35]. In this section we briefly review some *publicly* available databases for face recognition, face detection, and facial expression analysis, and we'll mainly focus on the three databases which we will use in this thesis.

To facilitate this statement, we divide face databases into two categories according to their designing goals. In the first part, we'll introduce databases which are normally used for face recognition; those which are dedicated to expression recognition will be discussed in the second part. As only a few databases are of the second type, and FER system shares some common modules with identity recognition system, in this work we also use some databases of the first type.

### 2.7.1 Databases For Identity Recognition

Most face databases are of this category (Table.2.1). To test for robustness, some of them are captured under different poses, illuminations and expressions. However, because they're mainly designed for identity recognition, the expressions are added as noise and usually not well controlled. So in general these databases are considered not suitable for FER research. In our work, we only use them to train peripheral modules (Preprocessing and Feature Extraction).

#### The IMM Face Database [66]

The IMM Face Database comprises 240 still images of 40 individuals (7 females and 33 males), all without glasses. For each person, 6 images are provided:

Table 2.1: Some of the most popular Face Recognition Databases [35]

| Database | No. of subjects | Pose | Illumination | Facial Expressions |
|---|---|---|---|---|
| AR | 116 | 1 | 4 | 4 |
| BANCA | 208 | 1 | ++ | 1 |
| CAS-PEAL | 66-1040 | 21 | 9-15 | 6 |
| CMU Hyper | 54 | 1 | 4 | 1 |
| CMU PIE | 68 | 13 | 43 | 3 |
| Equinox IR | 91 | 1 | 3 | 3 |
| FERET | 1199 | 9-20 | 2 | 2 |
| Harvard RL | 10 | 1 | 77-84 | 1 |
| IMM Face | 40 | 3 | 2 | 3+ |
| KFDB | 1000 | 7 | 16 | 5 |
| MIT | 15 | 3 | 3 | 1 |
| MPI | 200 | 3 | 3 | 1 |
| ND HID | 300+ | 1 | 3 | 2 |
| NIST MID | 1573 | 2 | 1 | ++ |
| ORL | 10 | 1 | ++ | ++ |
| UMIST | 20 | ++ | 1 | ++ |
| U. Texas | 284 | ++ | 1 | ++ |
| U. Oulu | 125 | 1 | 16 | 1 |
| XM2VTS | 295 | ++ | 1 | ++ |
| Yale | 15 | 1 | 3 | 6 |
| Yale B | 10 | 9 | 64 | 1 |

- Frontal face, neutral expression, diffuse light.
- Frontal face, happy expression, diffuse light.
- Face rotated approx. 30 degrees to the person's right, neutral expression, diffuse light.
- Face rotated approx. 30 degrees to the person's left, neutral expression, diffuse light.

- Frontal face, neutral expression, spot light added at the person's left side.
- Frontal face, "joker image" (arbitrary expression), diffuse light.

The images are stored in 640 × 480 JPEG files. Owing to technique problems, most images are RGB, but some are grey-scale [66]. One good thing about this database is that manually labeled face contour is available. The following facial structures were annotated using 58 landmarks: eyebrows, eyes, nose, mouth and jaw. These landmarks are divided into seven point paths; three closed and four open as shown in Fig.2.14. In our work, this database will be used to train the ASM and AAM model.

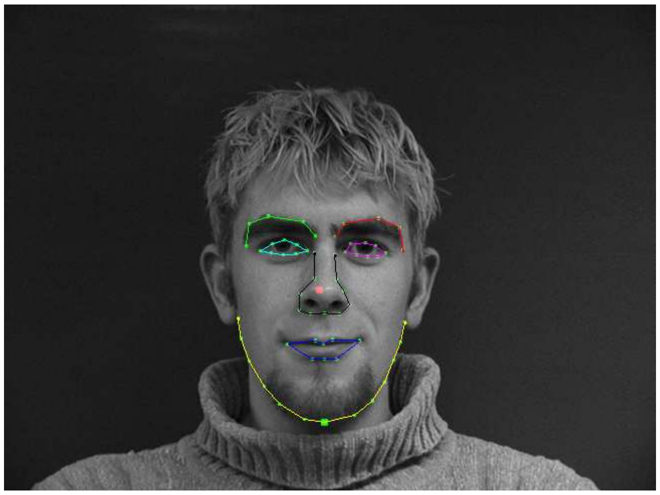


Figure 2.14: Example image from IMM face database

**CMU Pose, Illumination, and Expression Database [79]**

The CMU-PIE database is among the most comprehensive databases in this area. It systematically samples a large number of pose and illumination conditions along with a variety of facial expressions. The PIE database was captured under 21 illuminations (lit by 21 flashes) from 13 directions (using 13 synchronized cameras). In total, there are 41,368 images obtained from 68 individuals. In our experiment, we only use a sub-set of this database which consists of images of 62 people. 25 images were selected for each individual with 5 different viewpoints and 5 different illuminations. Part of the data set is shown in Fig.2.15.

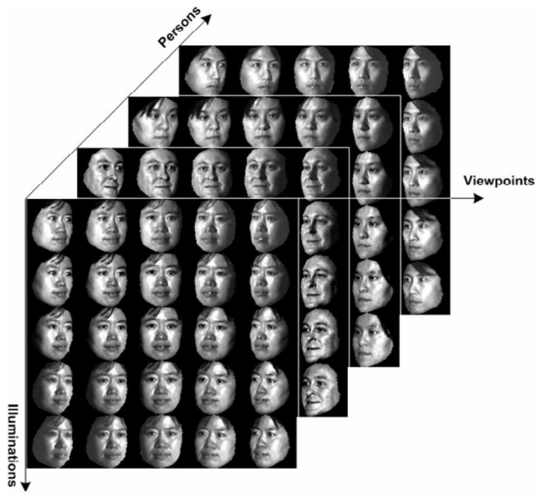


Figure 2.15: A subset of CMU PIE database [53]

### 2.7.2 Databases for Expression Recognition

"The human face is able to display an astonishing variety of expressions. Collecting a database that samples this space in a meaningful way is a difficult task" [35]. As a result, there are many fewer databases available for expression recognition (Table 6.1). As mentioned in 2.1.1, there are two ways to describe facial expressions. Available databases can be categorized into two classes according to the description they used. In one group [38] expressions are coded in FACS, while in the other group [57] images are labeled by their prototypic emotional expressions.

#### Japanese Female Facial Expression Database [57]

The JAFFE database contains 213 images of 10 Japanese female models. Their images are labeled by emotions: six basic emotions (anger, disgust, fear, joy, happy, sad and surprise) are considered and "Neutral" is added as the 7th emotion which is defined through the absence of expression. Fig.2.16 shows example images for one subject along with emotion

Table 2.2: Commonly used expression recognition databases [35]

| Database | No. of subjects | No. of Expressions | Image Resolution | Video/Image |
|---|---|---|---|---|
| JAFFE | 10 | 7 | $256 \times 256$ | Image |
| U. Maryland | 40 | 6 | $560 \times 240$ | Video |
| Cohn-Kanade | 100 | 23 | $640 \times 480$ | Video |

labels. The images were originally printed in monochrome and then digitized using a flatbed scanner.

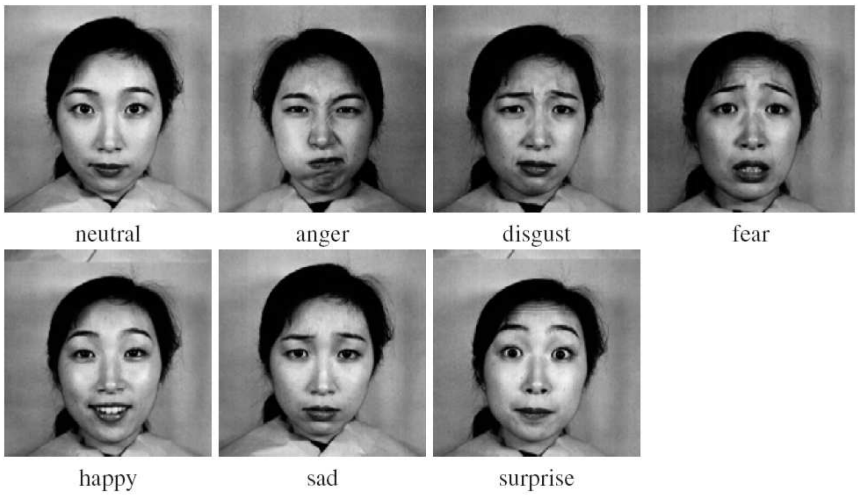


Figure 2.16: Example images from JAFFE database [35]

## 2.8 Chapter Summary

In this chapter, we first talked about the background of facial analysis, then gave an overview of the development in this area, and we also briefly introduced some state of the art techniques which might be useful for our system. At the end, we had a glance at some face databases for identity and expression recognition. Starting in the next chapter, we'll discuss the design of our FER system.

# Chapter 3

# System Architecture and Image Preprocessing

## 3.1 System Architecture

Although a vast array of algorithms are available for each module of face recognition system, how to combine them is still a challenging problem, especially for an expression recognition task, where ambiguity is an essential issue we need to face. Fig.3.1 depicts the overall system architecture developed in this work to solve these problems. Our system is divided into three major components, *i.e.*, Image Reprocessing, Feature Extraction and Classification. Almost all modules are based on statistical methods and they will be trained using a labeled training set. Under this framework, we'll compare the performance of various existing algorithms and find out the optimal configuration of a FER system. This is a major contribution of this work.

### 3.1.1 Image Preprocess

Face images from different databases have diverse resolutions, backgrounds and are captured under varying illumination; to make the images more comparable preprocessing is necessary. In our system, this module consists of three components: face detection, face alignment and normalization. Because our system is designed to work on well defined

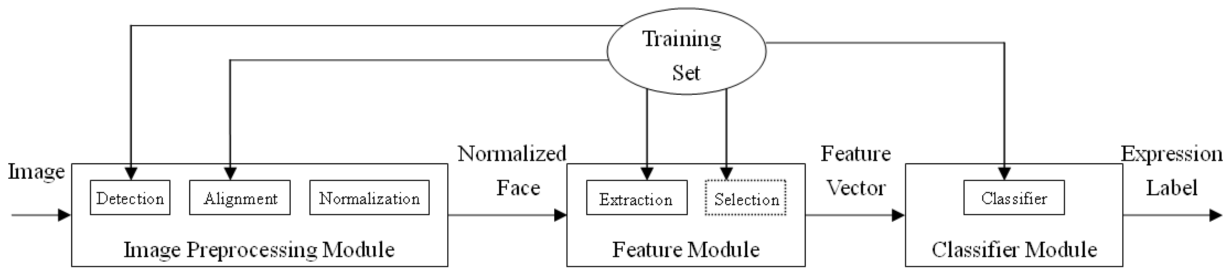


Figure 3.1: System architecture

databases where face images have already been centered, a very sophisticated detection system is not necessary. So template matching method is used in our system. In the alignment part, two algorithms are implemented: feature points localization and Active Shape Model. The aim of the feature point localization algorithm is to determine a set of feature points from the image using image processing techniques. Active Shape Model also attempts to find a set of points, however these points are considered as a contour so the ASM search is more holistic.

### 3.1.2 Feature Extraction and Selection

To achieve better robustness and accuracy, before feeding the normalized facial images into classifier, they are converted into some other, hopefully more separable representations which are referred to as features. In this way, the image processing and classification can be tackled separately. Two types of features are tested in our work: Gabor features and Active Appearance Model coefficient.

After features have been computed, AdaBoost methods can be employed to find out which features are useful for the specific recognition task. Sometimes, this feature selection is considered as a part of classifier design, but here we put it with feature extraction because of the following considerations: 1) Feature selection is only performed in the training stage, then the selection is stored in feature extraction modules so that only the useful features are computed in the final system; 2) In our framework, the same feature selection algorithm is used for different classifiers, so it's only related to the property of features.

### 3.1.3 Classification

The core of a FER system is its classifier. In this module we will discuss two kinds of methods, in particular Support Vector Machine and higher order methods. Support Vector Machine is one of the most popular and well-developed methods in machine learning, so in this work it's used mainly as a benchmark. Higher order methods on the other hand is a series of emerging algorithms which are based on multilinear algebra. In this group, first we will discuss Higher Order Singular Value Decomposition methods and then propose two new algorithms which are called Higher Order Canonical Correlation Analysis and Simple Multifactor Analysis. This is another major contribution of this thesis.

### 3.1.4 Summary

In this section, we've decided the outline of our FER system:

- Input: Face image
- Output: Expressions which are represented by prototypic emotions
- Main components and their responsibilities:
  - Image Preprocessing: Preparing the image for the following modules. So actually its detailed tasks are determined by what kind of input is needed in Feature Extraction.
  - Feature Extraction: Converting the original image into a more compact, and hopefully more separable representation.
  - Classification: Determining the facial expression for a given feature vector.

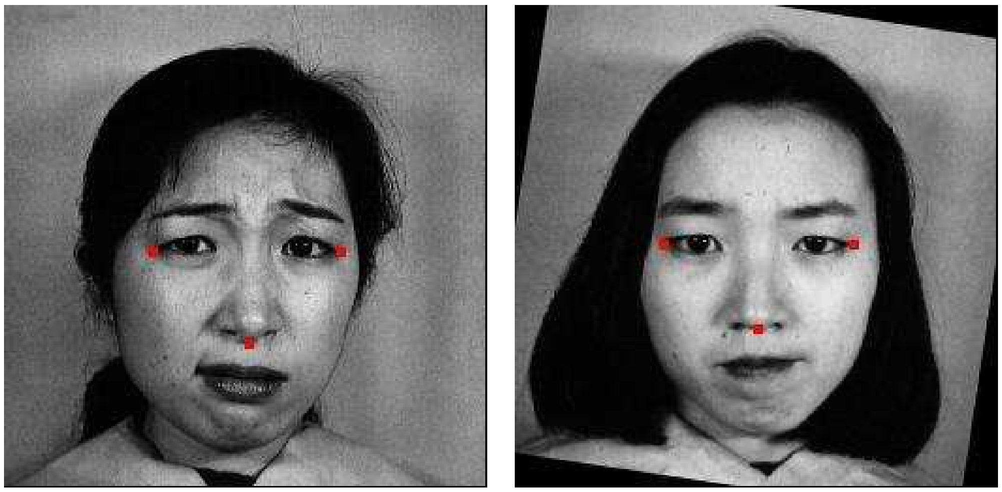
In the remainder of this chapter we'll talk about the image preprocessing module which is responsible for the following three tasks: face detection, face alignment and image normalization. we'll discuss the implementations of these three components one by one.

## 3.2 Face Detection

In this thesis, we choose a comparatively simple detection algorithm because our system is designed to work on well-defined databases in which face images have already been centered. Template matching algorithm [29] is used in our system and experiment is conducted on Japanese Female Facial Expression database [57].

### 3.2.1 Template Training

3 feature points: outside eye-corners and philtrum are manually labeled on each training image as shown in Fig.3.3. Then each image is rotated such that its 2 eye-corners are on the same horizon.



(a) Manually labeled training image (b) Rotated training image

Figure 3.2: Labeled training image

Then the facial area is extracted according to scheme shown in Fig.3.4(a) and resized to $100 \times 56$. After that histogram equalization is performed and typical output is shown in Fig.3.4(b). Then the average image of these normalized training faces are computed and because the cheek area usually has very small variation, only the eye and nose area is kept which gives the final template shown in Fig.3.5.

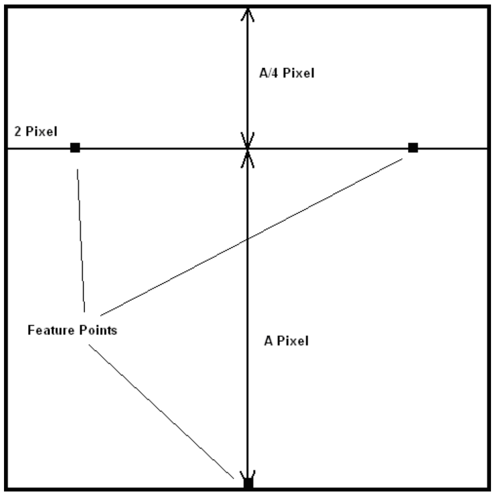
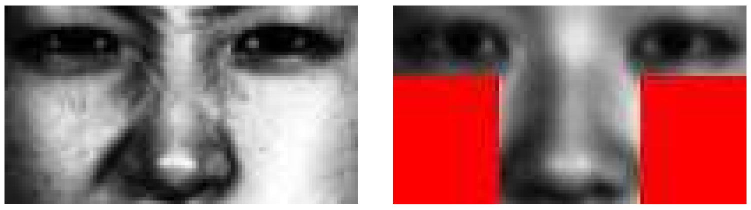


Figure 3.3: Chopping scheme



(a) Normalized (b) Eye-nose template

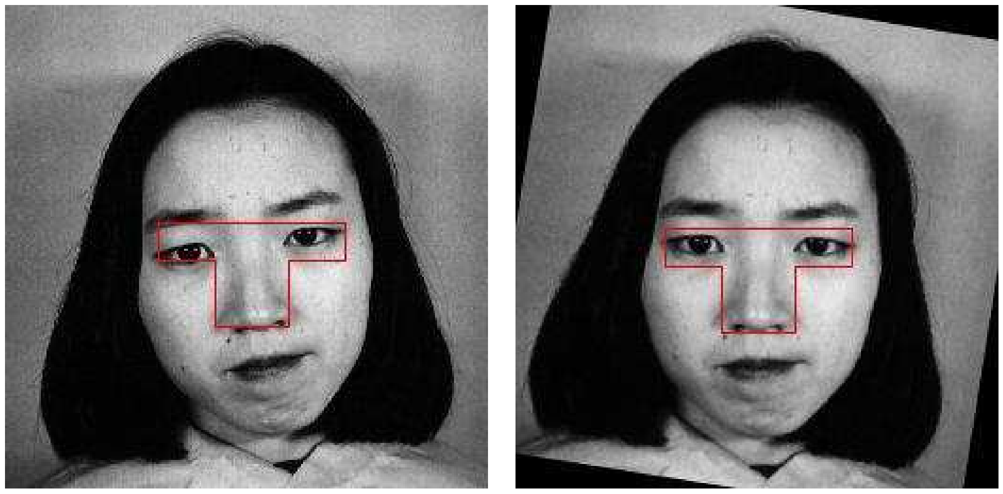
Figure 3.4: Face template training

### 3.2.2 Template Matching

In matching stage, we try to find the area of image which minimizes the sum squared error, *i.e.*,

$$\arg\min_{area} \sum_{i \in area} [I(i) - template(i)]^2 \tag{3.1}$$

This optimization is solved in 2 steps: first minimize (3.1) in all the upright areas, the optimal area is denoted by $area_1$. Then rotate $area_1$ by a small angles ($-12°$–$12°$) to $area'_1$ and search in a $9 \times 9$ neighborhood of $area'_1$ to find the final matching $area_2$. Typical result is shown in Fig.3.5.



(a) $area_1$ (b) $area_2$

Figure 3.5: Template matching

Besides sum squared error (norm-2 distance), we've also tried norm-1 distance

$$|I(i) - template(i)| \tag{3.2}$$

and normalized Euclidean distance

$$D = \sum_i \frac{a_i^2}{s_i^2} \tag{3.3}$$

where $a_i$ are Eigenface coefficients and $s_i$ are corresponding eigenvalues. Experiment shows that these 3 distance measures give similar results, so in the following tests only sum squared error is used.

## 3.3 Face Alignment

In this section, feature points localization method and Active Shape Model are implemented and compared.

### 3.3.1 Feature Points Localization

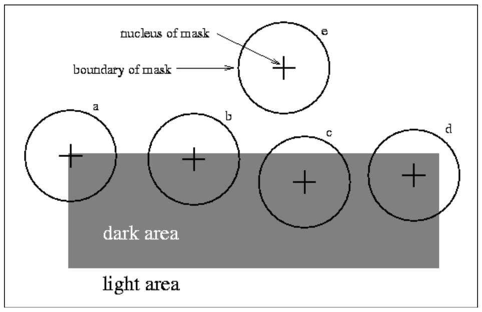
By locating the feature points from a face image we can tell the characteristics of the organs easily: we can normalize the image and find the different organs on the face. So the selection of face feature points is crucial to the face recognition. We should pick the feature points which represent the most important characteristics on the face and can be extracted accurately. Eye corners, especially outside eye corners have been shown robust feature points because they have fairly stable positions with respect to other organs and as "corners" of eyes they can be localized by corner detection which is extensively studied in computer vision area and well developed algorithms are available. In our system Smallest Univalue Segment Assimilating Nucleus (SUSAN) operator is used because of its robustness and simplicity.

#### SUSAN Detector

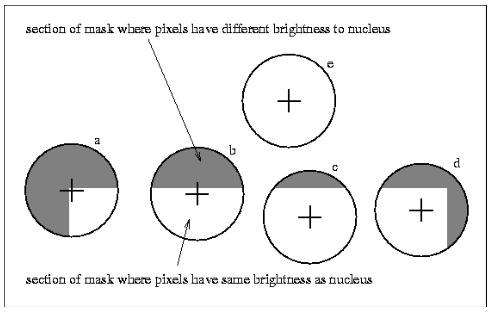
The basic assumption of SUSAN detector is that each image point has associated with it a local area of similar brightness. This area is known as Univalue Segment Assimilating Nucleus (USAN) and contains much information about the structure of the image. From the size, centroid and second moment of the USAN, corners and edges can be detected. In practice, USAN is computed using a circular masks which has the same brightness as the nucleus: all pixels in the circle is considered as the neighbors of the "nucleus" (the center pixel) and the area where the image has the same (or similar) brightness as the mask is the "USAN". The process is shown in Fig.3.6(a) and Fig.3.6(b): Fig.3.6(a) shows a dark rectangle on a white background and a circular mask is put at five image positions. In Fig.3.6(b) each mask is illustrated with its USAN shown in white.

As shown above, the USAN area has a maximum value when the nucleus lies in a flat region and achieves its minimum near corners. This property gives rise to Smallest Univalue Segment Assimilating Nucleus (SUSAN) operator which is used as the main determinant of the presence of corners.

For each pixel $(x_0, y_0)$, SUSAN operator is computed in 3 steps:



(a) Four circular masks at different places on a simple image [81]


section of mask where pixels have different brightness to nucleus
a
b
c
d
e
section of mask where pixels have same brightness as nucleus

(b) Four circular masks with similarity coloring; USANs are shown as the white parts of the masks [81]

Figure 3.6: SUSAN detector

1. Find USAN

$$c(x,y) = \begin{cases} 1 & if\ |I(x,y) - I(x_0,y_0)| \le t \\ 0 & if\ |I(x,y) - I(x_0,y_0)| > t \end{cases} \qquad for\ \ |(x,y) - (x_0,y_0)| \le r$$

2. Compute area of USAN

$$n(x_0,y_0) = \sum_{|(x,y)-(x_0,y_0)|<r} c(x,y) \tag{3.4}$$

3. Find corner strength by

$$R(x_0,y_0) = (g - n(x_0,y_0))^+ \triangleq \max(g - n(x_0,y_0), 0) \tag{3.5}$$

As one can see, there are only 3 parameters in SUSAN operator, and each of them has a clear meaning:

- $r$ is the radius of the circular mask, only points in this region are considered as "near" $(x_0, y_0)$.
- $t$ defines similarity of brightness.
- $g$ is the threshold of USAN area. Only the points which have USAN areas smaller than $g$ are labeled as corners and given a corner strength.

In our experiment, $r$, $t$, $g$ are set to 3, 20, 7, respectively. According to the detection result, SUSAN operator is applied on left eye area and right eye area respectively. The inside and outside eye-corners are localized on the output, typical result is shown in Fig.3.7.

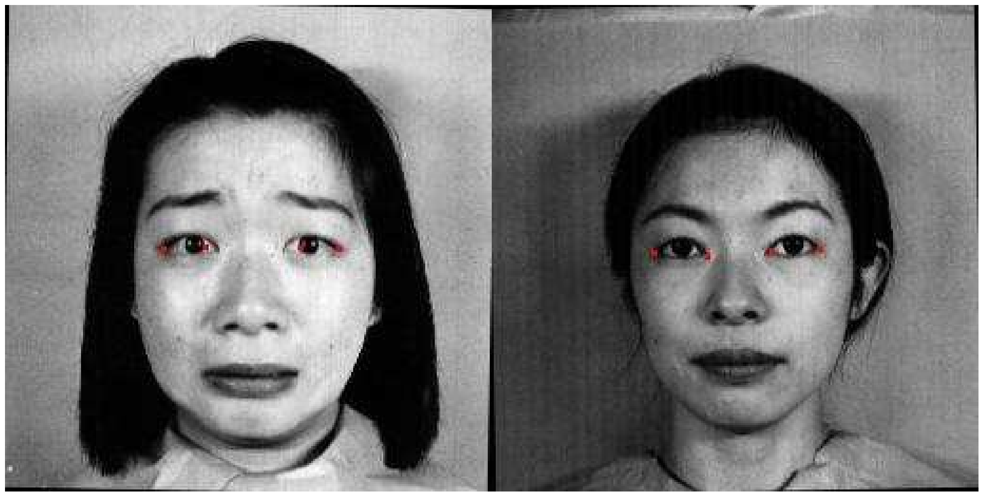


Figure 3.7: Eye corners localization

It can be seen that, the insider eye corners are sometimes confused with the edges of eyeball. The detection of the outside eye corner is more robust, and fortunately this is what we need.

## 3.3.2 Active Shape Model

**Intuition**

As mentioned in Chapter 2, as a curve-fitting algorithm Active Shape Model attempts to fit the contour in an image by minimizing an energy function. However, instead of defining the energy function ahead of time, Active Shape Model learns it from training data. The elastic (internal) force term is computed based on the shape distribution of training contours while the image (external) force term is estimated by comparing the normal direction profiles of current contour to those of training contours.

**Training**

In the training stage, ASM learns two models: a shape model, and an image model, which are used to define internal energy and external energy respectively.

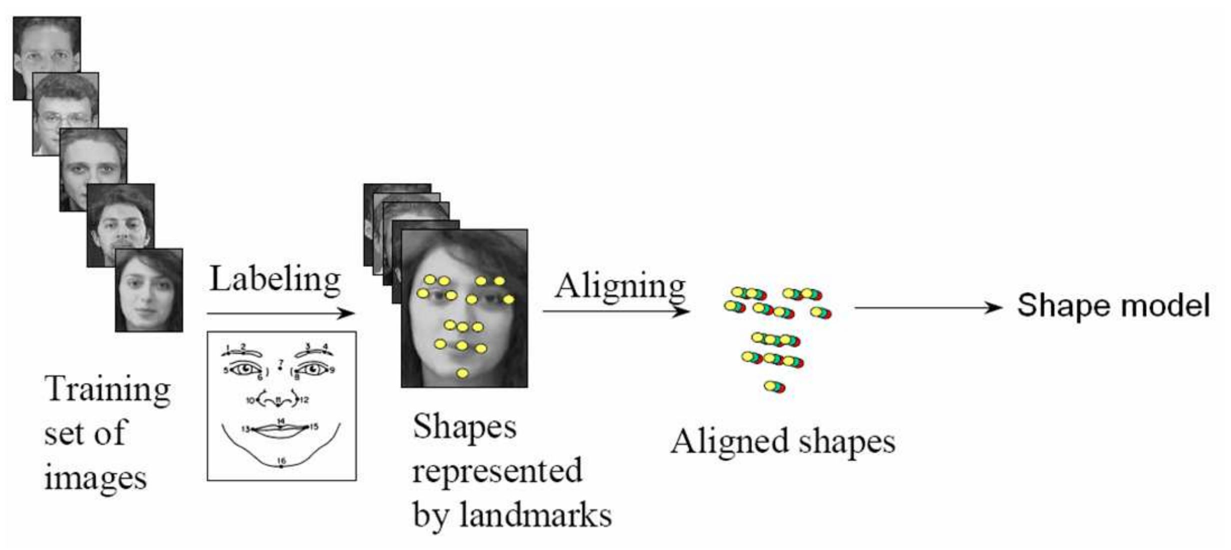


Figure 3.8: Shape model training [67]

**Shape Model Learning** The procedure of shape model training is depicted in Fig.3.8. The contour is represented by $N$ ordered landmarks (as a $2N$ dimensional vector). To eliminate the influence of rotation, scaling and translation, all training contours are aligned [18] with some preset landmarks. Then the distribution of the aligned shapes are analyzed using PCA to give:

- Mean Shape: $\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
- Covariance Matrix: $S = \frac{1}{N} \sum_{i=1}^{N} dx_i dx_i^T$
- Mode of Shape Variation: $Sp_k = \lambda_k p_k, \quad k = 1, \ldots, 2n$

Any shape can be represented by the mean shape and a weighted sum of these $Sp_k$ as:

$$x = \overline{x} + Pb \tag{3.6}$$

where $P = (p_1, p_2 ... p_{2n})$, $b = (b_1, b_2 ... b_{2n})^T$ is the vector of weights.

And the probability of $x$ can be expressed by:

$$\Pr(x) = C \exp(-\sum_{k=1}^{2n} \frac{b_k^2}{\lambda_k}) \tag{3.7}$$

Obviously, when $Pr(x)$ is large the internal energy function should be small. So ASM defines its energy function $E_{elastic}$ as

$$E_{elastic}(c) = D_m^2 = \sum_{k=1}^{2n} \frac{b_k^2}{\lambda_k} \tag{3.8}$$

For the sake of simplicity, instead of minimizing $D_m^2$, ASM sets a threshold $D_{\max}^2$ and consider a shape as a legal shape iff $D_m^2 \leq D_{\max}^2$. In other words, ASM solves the following optimization problem, which is slightly different from (2.24):

$$\underset{c}{\operatorname{argmin}} \quad E_{image}(c) \tag{3.9}$$

$$s.t. \quad E_{elastic}(c) = D_m^2 < D_{\max}^2 \tag{3.10}$$

**Image Model Learning** Image model studies the grey levels around the landmarks, in plain ASM this local grey-level information is represented by the normal profile located at each landmark as depicted in Fig.3.9.

The image model is trained in 5 steps.

1. Extract profile along the normal direction of the contour. Grey level profile for landmark j in image i is a vector.

$$g_{ij} = [g_{ij1}, g_{ij2}, ... g_{ijnp}]^T \tag{3.11}$$

2. To make it less sensitive to the lighting condition, take the derivative

$$dg_{ij} = [g_{ij2} - g_{ij1}, ... g_{ijnp} - g_{ijnp-1}]^T \tag{3.12}$$

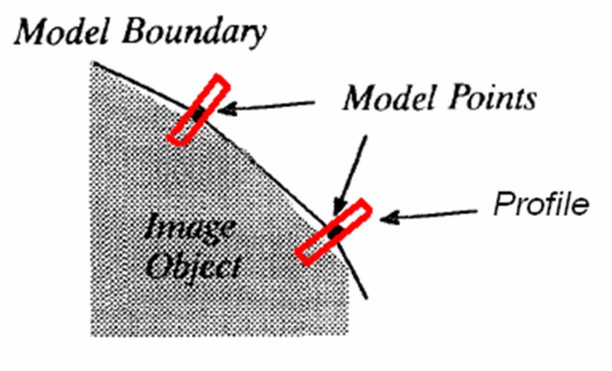


Figure 3.9: Extracting profiles from training image

3. Normalize

$$y_{ij} = \frac{dg_{ij}}{\sum_{k=1}^{np-1} |dg_{ij1}|} \tag{3.13}$$

4. Compute the average profile for landmark j:

$$\overline{y_j} = \frac{1}{N} \sum_{i=1}^{N} y_{ij} \tag{3.14}$$

5. Compute the covariance matrix for landmark j:

$$S_j = \frac{1}{N} \sum_{i=1}^{N} (y_{ij} - \overline{y_j})(y_{ij} - \overline{y_j})^T \tag{3.15}$$

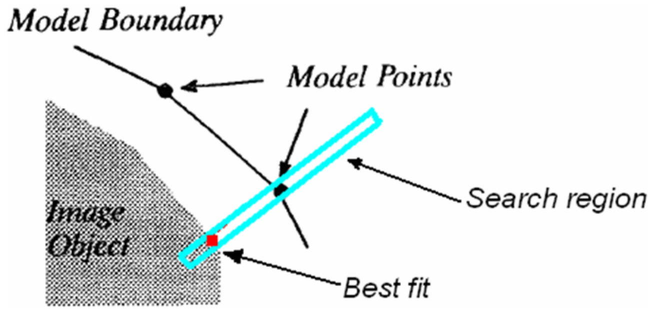
When a new sample $y$ is given, its probability can be calculated by:

$$\Pr(y) = C \exp(-M(y)) \tag{3.16}$$

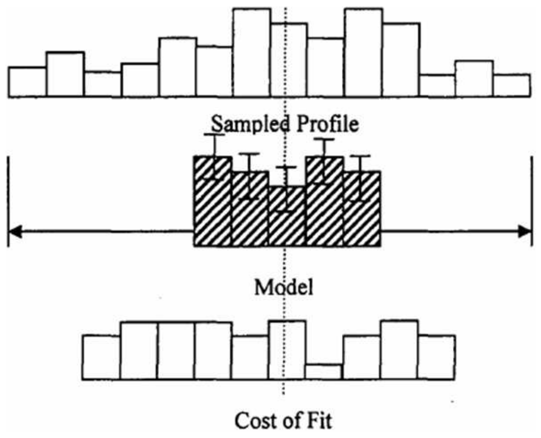
where $C$ is a constant and M(y) is the Mahalanobis Distance defined as:

$$M(y) = (y - \overline{y_j})^T S_j^{-1} (y - \overline{y_j}) \tag{3.17}$$

ASM uses $M(y)$ as the external energy function $E_{image}$.



(a) Search for better fit



(b) Cost of fit [17]

Figure 3.10: ASM local search

**Fitting The Shape**

In the fitting stage ASM alternately considers (3.9) and (3.10): first minimize $E_{image}$ by searching in a small neighborhood; then adjust if needed to make sure $E_{elastic}$ satisfies the constraint. The algorithm is summarized in Alg.2.

```
Input:
A face image
Initialization:
    Set shape tolerance $D^2_{\max}$; search region $n_s$; converge threshold $P_{close}$
    Set shape parameter $b$; scaling factor$T_s$; rotation $T_r$; translation $T_x$, $T_y$

        $$b = (0, ...0), \quad (Ts, Tr, Tx, Ty) = (1, 0, 0, 0) \tag{3.18}$$

    Flag=0
end
Shape searching:
    while Flag=0 do
        For each landmark in current shape, search in a region of $[-n_s, n_s]$ along the
        normal direction. This gives the suggested new locations:

            $$X_S = X + dX \tag{3.19}$$

        if less than $P_{close}$ landmarks are suggested to be moved then
            Flag=1
        end
        Update mapping parameters to better fit $X_s$ with current shape:

            $$X_S \approx M(Ts, Tr, Tx, Ty)[\overline{x} + Pb] \tag{3.20}$$

        Compute the suggested new shape parameters $b_s$:

            $$b_S = P^T (M(Ts, Tr, Tx, Ty)^{-1} X_S - \overline{x}) \tag{3.21}$$

        Apply the constraint on the shape parameters:

            $$b = \begin{cases} b_S & D^2_m \leq D^2_{\max} \\ b_S \cdot \frac{D^2_{\max}}{D^2_m} & o/w \end{cases} \tag{3.22}$$

    end
end
Output:
Facial contour represented by a set of landmarks
```

**Algorithm 2**: ASM shape searching [18]

**Multi-resolution Framework (Pyramid)**

To improve the robustness and efficiency, a Pyramid technique is utilized in ASM. The idea is to train the models on different resolution images produced using Gaussian smoothing and sub-sampling (Fig.3.11). In the fitting stage, ASM first searches on a low resolution version of the image to get a coarse result, then refines the result on higher resolution versions. This has two advantages: 1) With the same parameter, because the size of the low resolution image is much smaller the equivalent searching step is larger and convergence rate is higher; 2) Using this coarse-to-fine scheme, it is less likely to be trapped in local minimum.

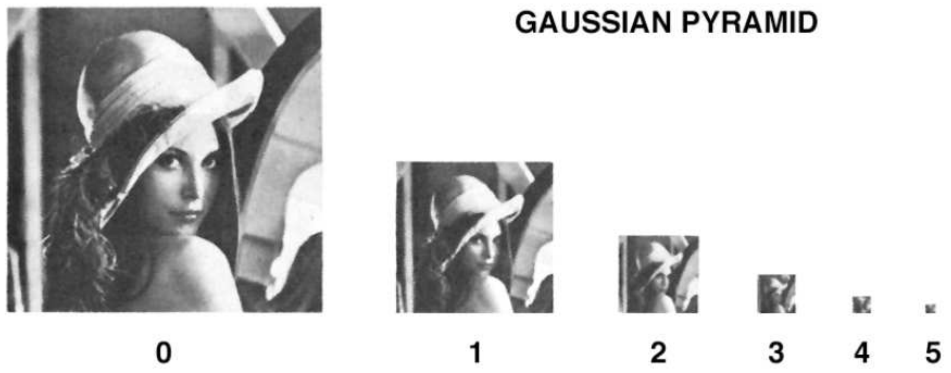


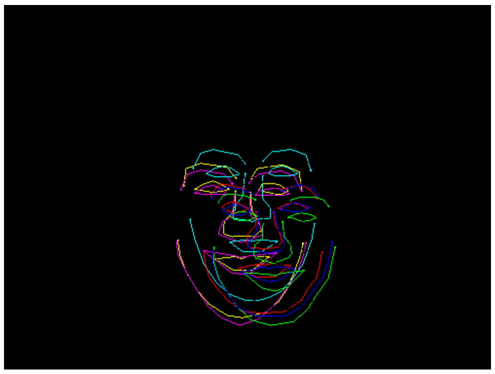
Figure 3.11: Gaussian Pyramid [10]

**Implement and Experiment**

Because ASM requires labeled landmarks for training images, in this part we conduct our experiments on the IMM face database [66].

**Shape Model Learning** Part of the shapes before and after alignment are shown in Fig.3.12. As one can see in the right figure, aligned shapes have similar location, orientation and size.

Then PCA is applied to extract the distribution of training shapes, the results are shown in Fig.3.13. Fig.3.13(a) is the average shape $\overline{x}$ and Fig.3.13(b) illustrates the variance $\lambda_k$


(a) Before alignment

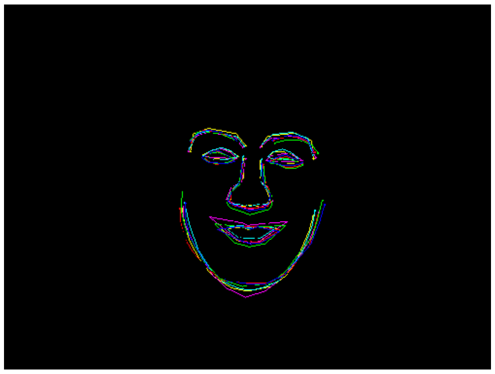
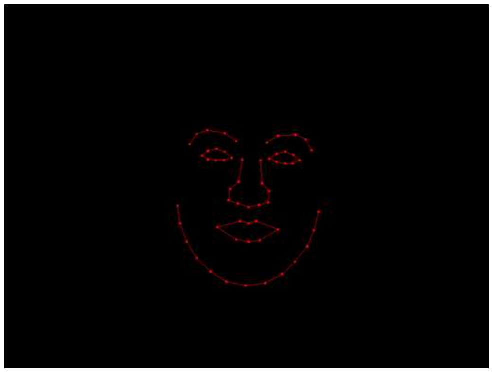
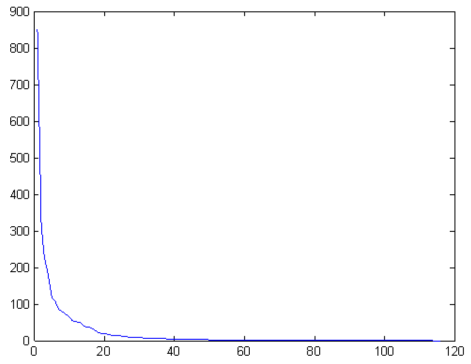

(b) Aligned shapes

Figure 3.12: Shape before and after alignment

of each mode.


(a) Mean Shape



(b) Aligned shapes

Figure 3.13: ASM shape model

**Grey-level Profile Learning** The red lines in Fig.3.14 indicate the normal direction on each landmark. Profile model can be computed using the formulas in .

**Shape Fitting** The parameters used in our work are:

- Pyramid level: 5, resolution decreases by a factor of 2 in each level

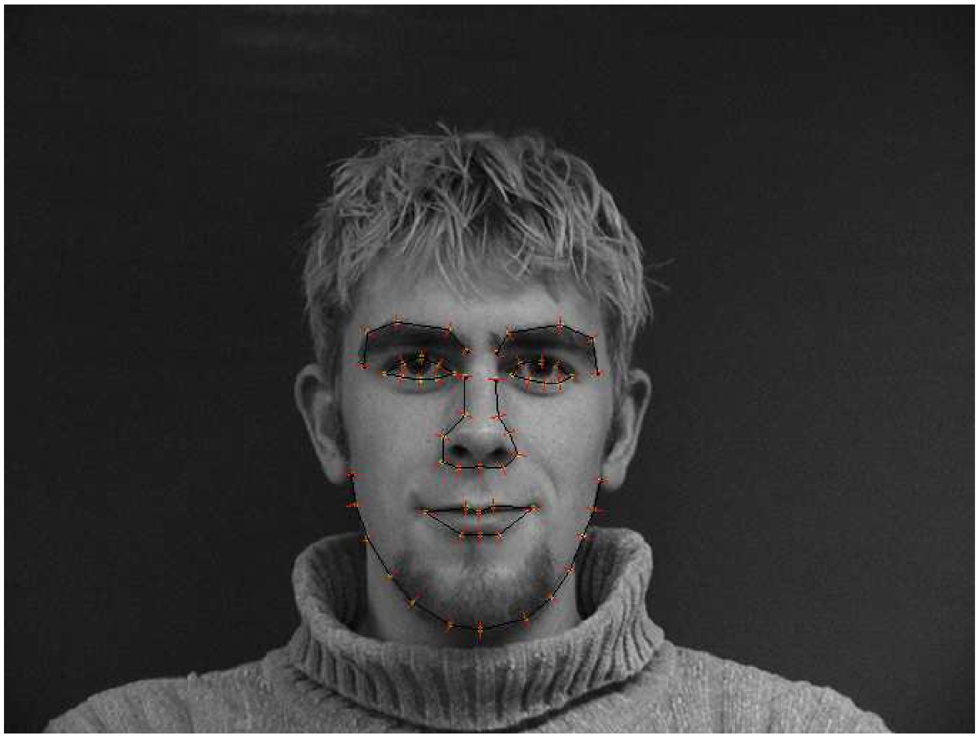
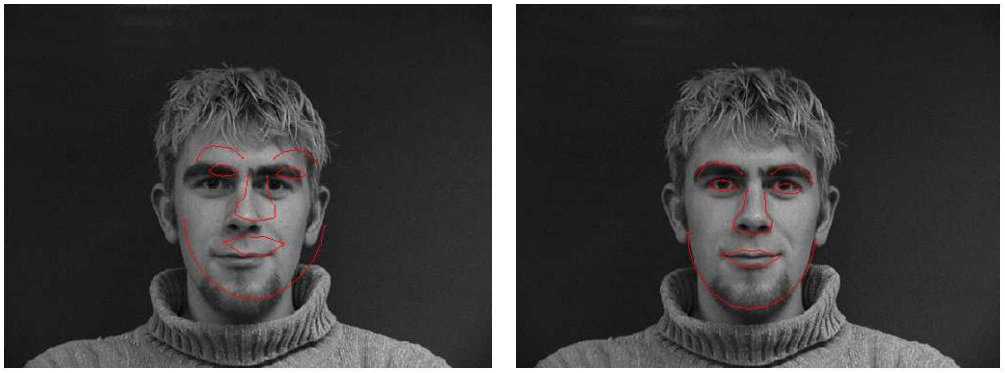


Figure 3.14: Contour and normal direction

- Max search step in each level: 10, to make it quicker, don't always wait until it converges.
- Profile length: [-3,3], which means each profile has 7 pixels.
- Search range: [-2,2]. Search for better profile fitting in [-2,2] along the normal direction
- Number of modes: 30. Keep first 30 modes of shape variation.
- $D^2_{\max}$: 5

Typical result is shown in Fig.3.15.



(a) Initial guess (mean shape) (b) Final result

Figure 3.15: ASM search result

## 3.4 Normalization

To make the images more comparable and also to reduce their size, normalization is performed based on the alignment result. It consists of two steps: geometric normalization and photometrical adjustment.

### 3.4.1 Geometric Normalization

It has been shown that using outer eye corners to normalize the facial image is more accurate than using other features [15] [29]. In our system, the face images are rotated and chopped based on their outside eye corners according to the scheme shown in Fig.3.16.

### 3.4.2 Photometrical Adjustment

In this stage, the chopped facial area is resized to $180 \times 140$. Then histogram equalization is applied to enhance contrast. After that a simple noise reduction is performed using media filter. The final output images are shown in Fig.3.17.

If the face image is aligned using ASM, usually Active Appearance Model will be used in feature extraction. Because AAM already contains some normalization steps, in that case the normalization given in this section is optional.

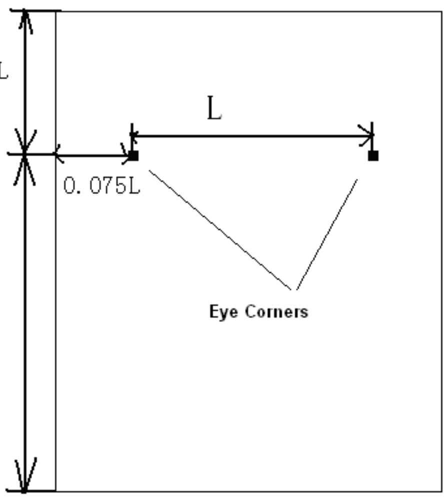


Figure 3.16: Normalizing scheme

## 3.5 Chapter Summary

In this chapter, we first gave the overall architecture of our system, then talked about image preprocessing, *i.e.*, face detection, face alignment and image normalization. Because currently our system is designed to be used on well-defined databases, we choose a simple detection algorithm which is based on template matching. Experiment shows that its output is good enough for our subsequent process. In the alignment part, two algorithms are implemented: eye corner localization which is based on the SUSAN operator is computationally efficient and easy to control. However, it gives very little information, so it is only suitable to be followed by Gabor filter based feature extraction. On the other hand, Active Shape Model is a more sophisticated method which finds a set of feature points as well as the facial contour. In general, ASM alignment is more robust and accurate. However all these benefits come at a price of more complex implementation and higher computational cost. So we'll only use this when Active Appearance Model is used for feature extraction. After alignment, the images pass through a stand normalization procedure which consists of a noise reduction (median filter), a geometric normalization (rotating and resizing) and

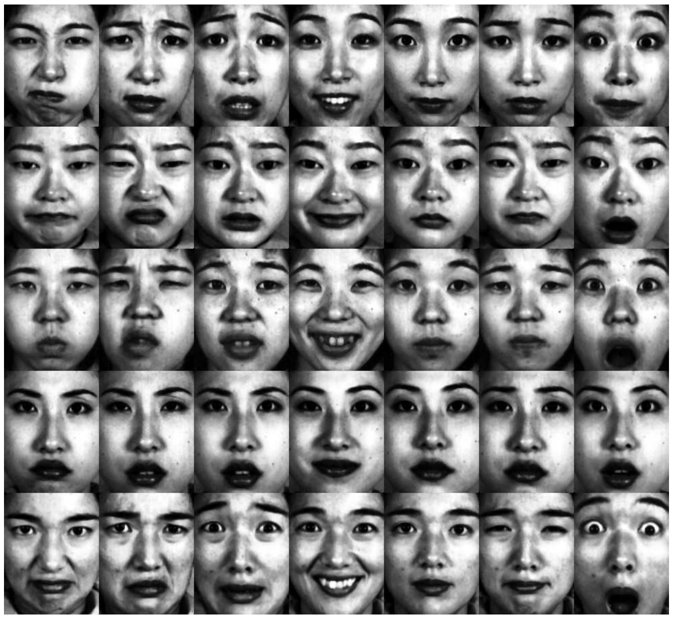


Figure 3.17: Part of the normalized database

a photometrical adjustment (histogram equalization).

# Chapter 4

# Feature Extraction and Selection

To achieve better robustness and accuracy, before the normalized facial images are fed into classifier, they are projected into some feature spaces in which, hopefully, the faces are more separable. In this way, the image processing and classifier designing can be tacked separately. In this module, we only focus on the image processing task, so when designing the classifier we can consider it as a general recognition task and won't need to think about the image properties. Two types of features are tested in our work: Gabor coefficients and Active Appearance Models. Gabor coefficients are an over-complete set of features. So, in the training stage, feature selection is performed to determine which Gabor filters are useful, so only these features are computed in the final system.

## 4.1 Gabor Filter Based Method

A wavelet filter has the ability to decompose an image into relevant texture features. Multi-channel filtering has been widely used for classification because it can mimic the human visual system which is sensitive to orientation and spatial-frequency. Gabor wavelet [21] [49] [57] has gained increasing attention recently.

### 4.1.1 Gabor Function

In space domain Gabor function [21] [49] [57] is a Gaussian-shaped "envelop" multiplied by a complex sinusoidal "carrier":

$$g(x,y) = \frac{1}{2\pi\delta_x\delta_y}\exp\{-\frac{1}{2}[(\frac{x}{\delta_x})^2 + (\frac{y}{\delta_y})^2] + i(ux+vy)\} \tag{4.1}$$

In frequency domain, this is a shifted and rotated 2-D Gaussian function—a band-pass filter.

$$\hat{g}(\omega_x,\omega_y) = \exp\{-2\pi^2[\delta_x^2(\omega_x-u)^2 + \delta_y^2(\omega_y-v)^2]\} \tag{4.2}$$

From a signal process perspective, it can be proven that the Gabor function is the only function that achieves the lower limits of the uncertainty relations. "For a given area in the space domain it provides the maximum possible resolution in the frequency domain, and vice-versa" [65].

Biology research also showed that Gabor functions fit very well the receptive field weight functions found in simple cells in cats' striate cortices.

For the above reasons, Daugman [21] introduced 2-D Gabor wavelet in vision area. Gabor wavelet transformation uses a set of Gabor functions with different sizes and orientations as its basis functions. It is able to represent the frequency information of the image while maintaining its spacial structure. Since then Gabor wavelet as a feature extraction and image representation method has been widely used in recognition and image analysis.

A two dimensional Gabor function can be decomposed into a real part and an imaginary part [57] [59]:

$$G_{\vec{k}}(\vec{r}) = G_{\vec{k},+}(\vec{r}) + iG_{\vec{k},-}(\vec{r}) \tag{4.3}$$

where

$$G_{\vec{k},+}(\vec{r}) = \frac{k^2}{\delta^2}\exp(\frac{k^2\left\|\vec{r}-\vec{r}_o\right\|^2}{-2\delta^2})\cos[\vec{k}(\vec{r}-\vec{r}_o)] \tag{4.4}$$

$$G_{\vec{k},-}(\vec{r}) = \frac{k^2}{\delta^2}\exp(\frac{k^2\left\|\vec{r}-\vec{r}_o\right\|^2}{-2\delta^2})\sin[\vec{k}(\vec{r}-\vec{r}_o)] \tag{4.5}$$

In which $\vec{k} = k\exp(j\theta_v)$. $k$ controls the scale whereas $\theta$ determines the rotation.

$G_{\vec{k},+}$ and $G_{\vec{k},-}$ satisfy the quadrature phase condition, and are also called even-symmetric and odd-symmetric filter, respectively (Fig.4.1).

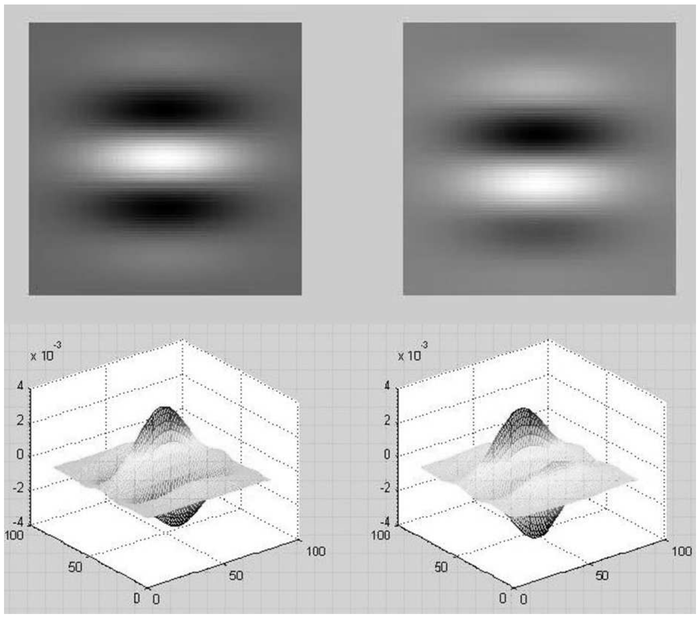


Figure 4.1: Real and imaginary part of 2-D Gabor Function

Because $G_{\vec{k},+}$ has a non zero mean, a compensation term is added to make the even-symmetric filter insensitive to overall brightness change of the image. And this gives:

$$G_{\vec{k},+}(\vec{r}) = \frac{k^2}{\delta^2}\exp(\frac{k^2\left\|\vec{r}-\vec{r}_o\right\|^2}{-2\delta^2})\{\cos[\vec{k}(\vec{r}-\vec{r}_o)] - \exp(-\frac{\delta^2}{2})\} \tag{4.6}$$

### 4.1.2 Feature Extraction Using Gabor Filter

According to research results reported in [21] [5] [54] [55], Gabor wavelet decomposition is useful in facial image analysis, and it's also shown that Gabor filters with different parameters carry complementary information. Consequently, it is reasonable to use a group of Gabor functions to represent images. In our system, 40 Gabor filters are used using 5 scales ($k = \frac{\pi}{2\times\sqrt{2}^r}$,$r = 1, 2, \ldots, 5$), 8 directions ($\theta_v = \frac{u\pi}{8}$, $u = 0, 1, \ldots, 7$) and $\delta = \pi$. The filter bank is shown in Fig.4.2 where rows correspond to different scales and columns correspond to different directions.

The convolution of the input image and each filter in the filter bank is computed. For

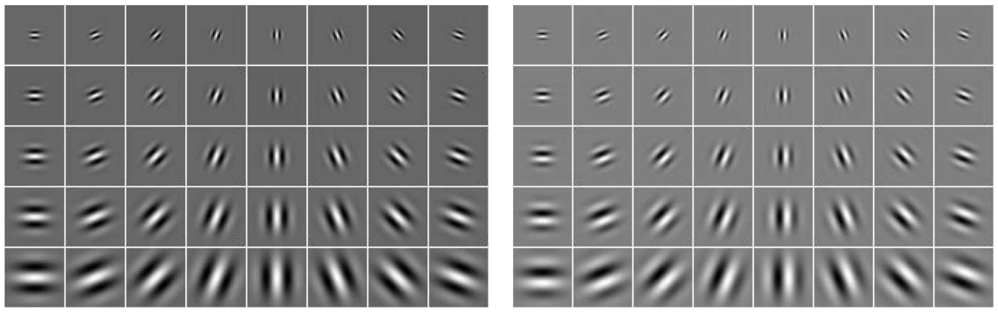

(a) Even-symmetric filters (b) Odd-symmetric filters

Figure 4.2: Gabor filter bank

each pair of filtered image, the amplitude is used as final output:

$$R_{\vec{k},\pm}(\vec{r}_0) = \int G_{\vec{k},\pm}(\vec{r}_0, \vec{r}) I(\vec{r}) d\vec{r}, \quad R_{\vec{k}} = \sqrt{R^2_{\vec{k},+} + R^2_{\vec{k},-}} \tag{4.7}$$

So for each input image we get 40 output images which are shown in Fig.4.3.

Because of the spatial correlation between adjacent points, we don't need to keep all of them. For each output image 48 samples are drawn from an $8 \times 6$ uniform grid (Fig.4.4). In total, each input image is represented by a $40 \times 48 = 1920$ dimensional feature vectors.

### 4.1.3 Feature Selection Using AdaBoost

Since the dimensionality of $x_i$ is very high, usually feature selection is performed using AdaBoost. The algorithm is similar to Alg.1 we introduced in Chapter 2. However, the output is slightly different: here the strong classifier is not needed; instead, we want the feature index $k_m$ for each weak classifier. According to each expression, the training data are labeled as either 1 (belongs to this class) or $-1$ (otherwise), and 100 features are found using AdaBoost.

The algorithm is summarized in Algorithm 3.

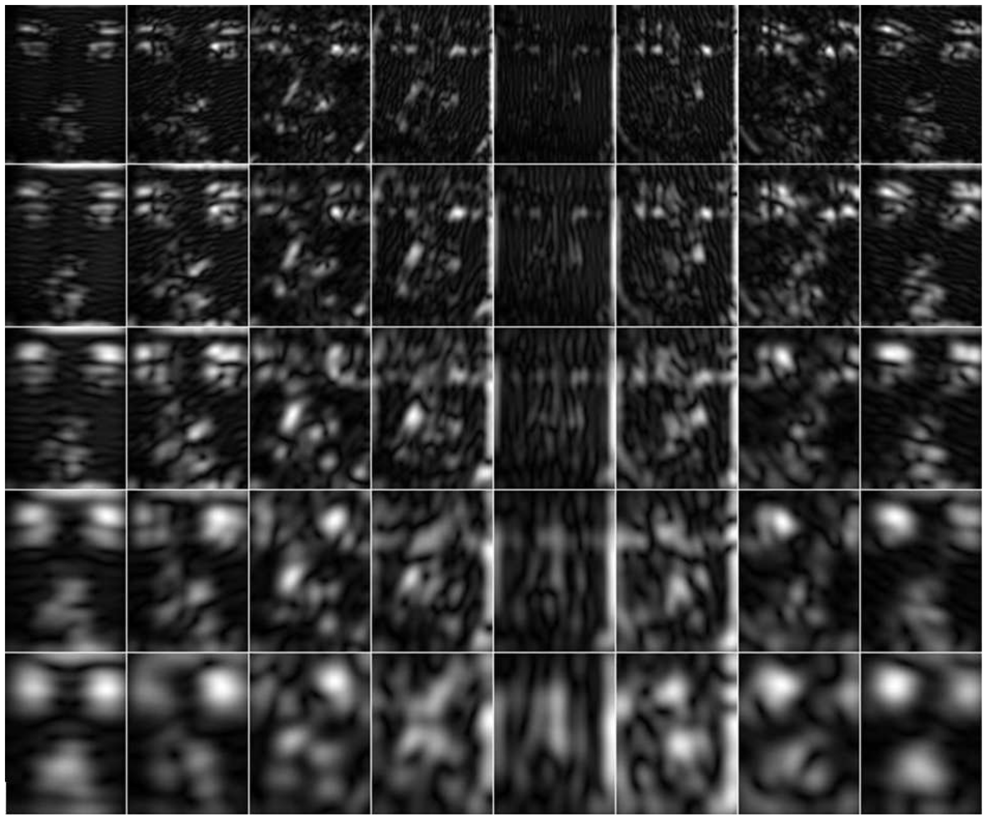


Figure 4.3: Gabor filter output images

## 4.2 Active Appearance Model Based Method

In this section, we use Active Appearance Model to represent the facial image. The basic idea of AAM has already been introduced in Chapter 2. In some ways, AAM can be considered as an extension of ASM; the major difference is that AAM uses a global texture model instead of a set of local image models. Another great improvement in AAM is that the shape model and texture model are combined. This is because shape and grey-level variations are correlated. For example, the shape mode of variation responsible for opening and closing the mouth is correlated with the texture mode responsible for the appearance of teeth [17].

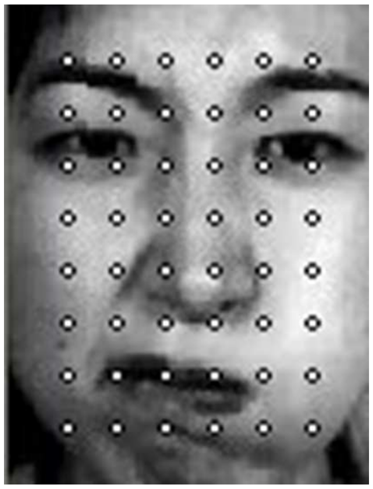


Figure 4.4: Sampling scheme of Gabor features

### 4.2.1 Model Training

In AAM, training consists of three steps: a) train the shape model; b) train the texture model; c) combine shape model and texture model.

#### Training Shape model

This is done exactly as in ASM.

#### Training Texture Model

As the facial textures have different shapes, first we need to convert them to have an identical shape; this is called image warping. There are two main kinds of image warping algorithms: Piece-wise Affine and Thin Plate Splines [8].

**Piece-wise Affine** Piece-wise Affine algorithm cuts the image into small pieces and warps each one of them using an affine projection.

Normally, triangular pieces are used. We still need to figure out how to cut this image. The short answer is using something called Delaunay triangulation. Basically this method

```
Input:
potential features $x_j$ and their corresponding class label $y_j^{(i)}$,$i = 1, \ldots, 7$
number of features we want for each expression $M$
for $i = 1to7$, consider a classification problem $(x, y^(i))$ do
    Construct M weak classifier $h_m(x)$,$m = 1, \ldots, M$ using AdaBoost. The index of
    feature for $h_m(x)$ is $k_m$
    Define the feature sets of $i$th expression as $F_i = \{k_m | m = 1, \ldots, M\}$
end
Output:
feature selection $F = \bigcup_i F_i$
```

**Algorithm 3**: AdaBoost based feature selection

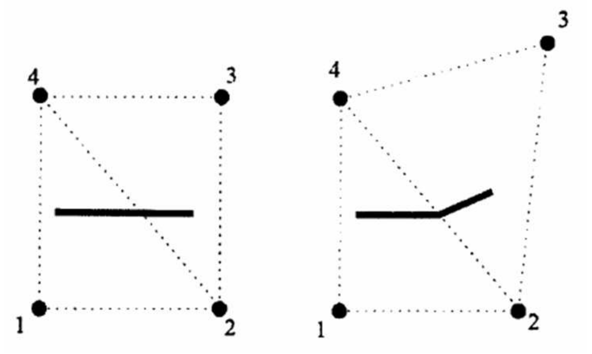


Figure 4.5: Piece-wise Affine

tries to avoid narrow triangles. For a more comprehensive yet understandable reading, please refer to [98].

Piece-wise Affine is a conceptually simple method but its problem is obvious. As one can see in Fig.4.5, after warping the line is not smooth near the edge of pieces. To solve this problem, spline method is proposed which guarantees at least the first-order derivative is continuous.

**Thin Plate Splines [8]** Boostein introduced this method in 1989 and it has been widely used since then. The idea is to construct an elastic energy function on the image, and when some landmarks are moved, other points will automatically shift following them to minimize the energy function. Here we only give the formulas and make a very short comment; for the detailed derivation please refer to [8].

The deformation under elastic forces should satisfy the biharmonic equation:

$$\nabla^4 \phi = 0 \tag{4.8}$$

So we want to construct a solution to 4.8 which also satisfies the movements of landmarks. First we choose a fundamental solution to the biharmonic equation:

$$U(r) = \begin{cases} 0 & r = 0 \\ r^2 \log r^2 & o/w \end{cases} \tag{4.9}$$

As should be evident, any linear combination of $U$ is also a solution to 4.8. In the following steps we are going to construct such a solution which satisfies the movements of landmarks. In general it looks like

$$\begin{cases} x_{new}(x,y) = a_{0x} + a_{xx}x + a_{yx}y + \sum_{i=1}^{n} w_{ix} U(|P_{oldi} - (x,y)|) \\ y_{new}(x,y) = a_{0y} + a_{xy}x + a_{yy}y + \sum_{i=1}^{n} w_{iy} U(|P_{oldi} - (x,y)|) \end{cases} \tag{4.10}$$

where $P_{old1} = (x_{old1}, y_{old1}), \ldots, P_{oldn} = (x_{oldn}, y_{oldn})$ are the old locations of landmarks and $(x_{new}, y_{new})$ is the new location of $(x, y)$.

The parameters $a$, $w_{ix}$ and $w_{iy}$ can be computed by solving the following equation:

$$LS = Y \tag{4.11}$$

where $S$ is the parameter matrix

$S = \begin{bmatrix} W \\ A \end{bmatrix}$, in which $W = \begin{bmatrix} w_{1x} & w_{1y} \\ \vdots & \vdots \\ w_{nx} & w_{ny} \end{bmatrix}$, $A = \begin{bmatrix} a_{0x} & a_{0y} \\ a_{xx} & a_{xy} \\ a_{yx} & a_{yy} \end{bmatrix}$ And $L$ is the fundamental solution matrix:

$$L = [\begin{matrix} K & P \\ P^T & O \end{matrix}] \tag{4.12}$$

In which $P = \begin{bmatrix} 1 & P_{old1} \\ \vdots & \vdots \\ 1 & P_{oldn} \end{bmatrix}$ and $K = \begin{bmatrix} 0 & U(r_{12}) & \dots & U(r_{1n}) \\ U(r_{21}) & 0 & \dots & U(r_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ U(r_{n1}) & U(r_{n2}) & \dots & 0 \end{bmatrix}$, $(r_{ij} = |P_{oldi} - P_{oldj}|)$

And $Y = \begin{bmatrix} V \\ O_{3\times2} \end{bmatrix}$, in which $V = \begin{bmatrix} P_{new1} \\ \vdots \\ P_{newn} \end{bmatrix}$

As we can see here, this algorithm is more complex than the Piece-wise Affine.

Anyhow, the image is warped into a so-called "shape-free patch" which is raster-scanned into a texture vector $g'$. To make the model insensitive to exposure conditions, we normalize it using:

$$g = \frac{g' - \mu_g 1}{\sigma_g} \tag{4.13}$$

where 1 is a vector with ones, $\mu_g$ and $\sigma_g^2$ are the mean and variance of elements of $g'$.

PCA is applied on $g$ to build a texture model. Any texture $g$ can be approximated by the mean texture $\overline{g}$ and $t$ modes of variation $P_g$

$$g \approx \overline{g} + P_g b_g \tag{4.14}$$

### Combining Shape Model and Texture Model [17]

Recall that the shape of an image can be represented by shape parameter $b_s$:

$$s \approx \overline{s} + P_s b_s \tag{4.15}$$

Similarly, the texture can be represented by texture parameter $b_g$:

$$g \approx \overline{g} + P_g b_g \tag{4.16}$$

Then vector $c = \begin{bmatrix} b_g \\ b_s \end{bmatrix}$ can represent both shape and texture:

$$\begin{bmatrix} g \\ s \end{bmatrix} = \begin{bmatrix} \overline{g} \\ \overline{s} \end{bmatrix} + \begin{bmatrix} P_g & 0 \\ 0 & P_s \end{bmatrix} c \tag{4.17}$$

To find the correlation between shape and texture, PCA is applied on $c$ :

$$c = \overline{c} + P_c b_c \tag{4.18}$$

$P_c$ gives the modes of appearance variation, and the facial image (both its shape and texture) is encoded in $b_c$ which is referred to as the "AAM coefficient".

AAM is a more comprehensive model, it deals with both the shape and the texture of the object. And it's invertible, *i.e.*, the original image can be constructed from the AAM coefficient.

### 4.2.2 Experiment

Because Active Appearance Model needs labeled images for training, this experiment is conducted on the IMM face database.

Shape model training is exactly the same as ASM experiment in Chapter 3. Both Piece-wise Affine and Plate Spline are implemented for image warping; the results are compared in Fig.4.6. Most times they give similar results. Not surprisingly, the affine algorithm is much faster than the spline algorithm—intuitively an affine function is a linear function while a spline function is high-order. As there're not many straight lines in a face image, the smoothness problem is not very obvious in our experiment . Considering the complexity, Piece-wise Affine is used in subsequent experiment.

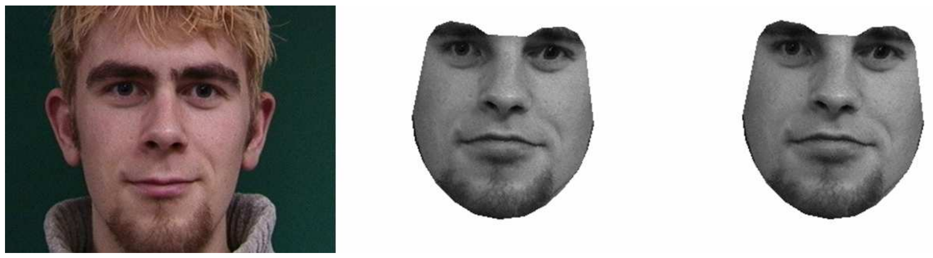


Figure 4.6: Original/ Warped using Piece-wise affine/ Warped using Plate splines

In our experiment, the "shape-free texture" is raster scanned into a $30000 \times 1$ vector on which PCA is applied. The mean texture is shown in Fig.4.7.

Eventually, a shape vector and a texture vector are combined to generate an AAM coefficient which can be used as a compact representation of a face image. The reconstructed

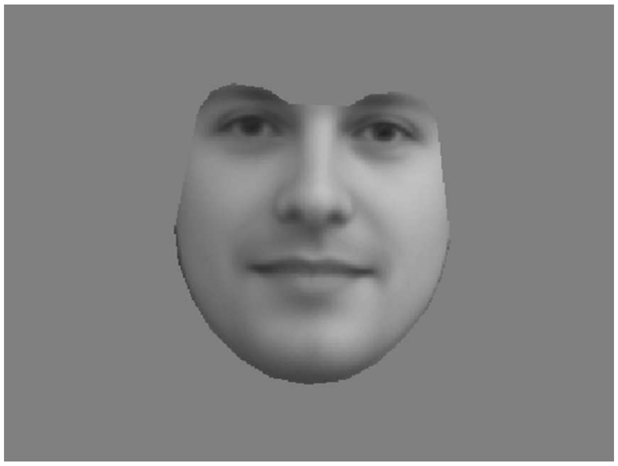
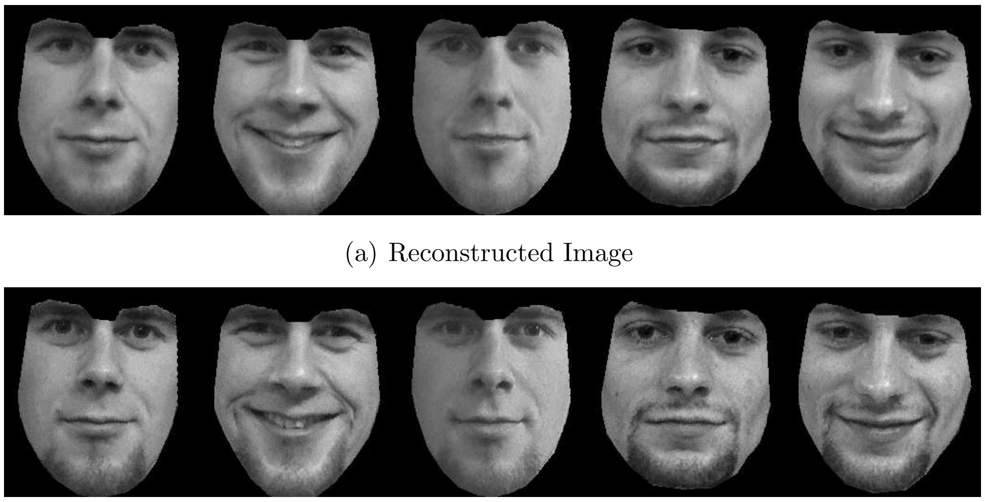


Figure 4.7: Mean texture

image is compared with the original one in Fig.4.8. Note that the original image has a size of $640 \times 480$ while its AAM coefficient is only a 50 dimensional vector!



(a) Reconstructed Image

(b) Original Facial Image

Figure 4.8: AAM reconstructed image

### 4.2.3 Feature Selection

In general, feature selection is not necessary for AAM coefficient because of the following reasons: first, AAM coefficient is already very compact and its dimensionality is not very high; second, a good property of AAM is that using it a face image can be reconstructed, which is important in some applications such as image synthesis, so we don't want to lose this characteristic in feature selection.

## 4.3 Chapter Summary

In this chapter, we implemented two, maybe the two most popular feature extraction algorithms: Gabor filters and Active Appearance Model. Gabor features are considered to be local features, each of which represent the grey level information within a small area. The dimensionality of Gabor features is very high, so feature selection was performed using AdaBoost. Another feature we investigated in this chapter is Active Appearance Model coefficient which is a very compact representation of the original facial image. At this stage, we still cannot decide which one is more suitable for FER problem and we'll find this out in the next chapter by comparing the performance of these two features using different classifiers.

# Chapter 5

# Classification

In this chapter, we will talk about the implementation of the classification module. Support Vector Machine is the first discussed, which is a very popular method in machine learning and has been widely used in facial recognition. In the second part, we introduce higher order methods, which are a new type of classification algorithms and have shown very good performance in the computer vision area.

## 5.1 Support Vector Machine Classifier

The basic idea and formulae of Support Vector Machine have been given in Chapter 2. SVM was originally designed for binary classification and here we are dealing with a multi-class classification. In this section, we'll first briefly look at how to solve this problem, then show the FER result by applying SVM on Gabor features and AAM coefficient.

### 5.1.1 Multi-class Support Vector Machine

SVM is a powerful tool and widely used in many areas, however it can only handle binary classification. Great effort has been put into finding how to effectively extend SVM for multi-class classification, and several techniques have been proposed [32] [19] [70] [88] [97]. The simplest idea may be to divide an $n$-class problem into a series of binary classifications. There are two ways: we can consider it either as $n$ "one-against-all" classifications or as

$C_n^2$ "one-against-one" problems. There are some other methods that consider all classes at once [88] [97]. Comparison in [32] has shown one-against-one is more suitable, so this is the method we used in this work.

### 5.1.2 Kernel Support Vector Machine

As mentioned earlier, SVM can be extended to a nonlinear case by utilizing a Kernel technique. It can be shown that, replacing the inner product with a Kernel function is equivalent to projecting the data into some higher dimensional space where hopefully the data are more separable. Widely-used Kernel functions include:

- Linear Kernel: $k(x, y) = x^T y$;
- Monomial Kernel: $k(x, y) = (\langle x, y \rangle)^d$
- Polynomial Kernel: $k(x, y) = (1 + \langle x, y \rangle)^p$
- Gaussian Kernel: $k(x, y) = \exp(-\left\| x - y \right\|^2 / \sigma)$
- Sigmoid Kernel: $k(x, y) = \tanh(\gamma x^T y + r)$

Among them, the Gaussian Kernel is by far the most popular. By applying the Gaussian Kernel, the data are projected to an infinite dimensional space and it delivers good results in most applications. In our experiment, only the Gaussian Kernel and the Linear Kernel (which is equivalent to no kernel) are considered.

### 5.1.3 Experiment

In this part ,we test Support Vector Machine on both Gabor features and AAM features. Our experiment is conducted on the Japanese Female Facial Expression (JAFFE) Database and the famous toolbox LIBSVM [11] is used as our SVM implementation. Two thirds of the database (143 images) are used for training the model and remaining one third (70 images) serve as a testing set. The recognition result is shown in Table 5.1.

From these results, we can see that:

Table 5.1: Recognition accuracy of Support Vector Machine

| Features | Selection | Kernel | Accuracy |
|---|---|---|---|
| Gabor features | Yes | Gaussian | 66/70=94% |
| | | No | 66/70=94% |
| | No | Gaussian | 62/70=88% |
| | | No | 65/70=92% |
| AAM coefficient | No | Gaussian | 63/70=90% |
| | | No | 58/70=82% |

- Feature selection is important for Gabor feature: it not only simplifies computation but also increases classification accuracy.
- Kernel method works well on AAM coefficient, but for Gabor features it's not necessary.
- Gabor features outperform AAM coefficient in recognition tasks.

## 5.2 Higher Order Method

In reality, facial images are influenced by the interaction of multiple underlying factors such as identity, illumination and viewpoint. People can extract this multifactor structure despite significant variation, for example we can recognize a person under different illumination conditions and even imagine the appearance of his face from a new viewpoint. However, it is still challenging to perform these tasks by computer. Conventional systems usually study these factors independently, and one problem is this: what might be noise in one context, might be signal in another context. From an information theory perspective, multiple factors embedded in one image can be viewed as multiple signals transmitted on the same channel. When one factor is studied, the image variances caused by other factors act like noise, the same as what happens in a multiple access channel. This “interference” is severe in computer vision, because different factors are often tightly entangled. That means it's hard or even impossible to extract a feature which is related to only one factor.

Higher Order methods [95] [53] [96] [86] [89] [90] [91] [90] [84] which study these underlying factors simultaneously have been shown to be a promising solution to this problem.

Multifactor analysis—modeling the interaction of multiple factors—was introduced by Tenebaum and Freeman in [86]. They proposed a bilinear model to analyze a data set influenced by two interacting variables which are generically called "content" and "style". However, in reality there are usually more than two factors and also identifying which variables correspond to "style" or "content" is somewhat arbitrary. Later, multilinear algebra was used in vision research. It is capable of handling any number of factors, and of treating them in a symmetrical way. Higher Order Singular Value Decomposition (HOSVD) [46] and its refinement Multilinear Independent Components Analysis (MICA) [92]have been used to model images of faces [95] [90] [92], motion capture sequences [89], and textures [91]. These models are multilinear in the factors, but linear with respect to any single factor. Li *et al.* [53] extended it to handle nonlinear data by adopting a kernel approach. They applied it to model the mapping between facial image and its 3 underlying factors: persons, illuminations and viewpoints, and then used it to perform recognition and synthesis tasks. Kernel-based HOSVD delivered better results than its multilinear counterpart, however this good quality came at a price of high complexity which hindered its popularization.

In this part, we discuss the most popular Higher Order Methods: HOSVD and Kernel HOSVD. The rest of this section is organized as follows: 5.2.1 reviews the mathematical background; the formulas of HOSVD and Kernel HOSVD are introduced in 5.2.2 followed by the experimental result in 5.2.3.

### 5.2.1 Principle of Multilinear Algebra

Multilinear algebra is the algebra of higher order tensors. A tensor is a higher order generalization of a matrix of which the elements are addressed by more than two indices. Let $A \in R^{I_1 \times I_2 \times \ldots \times I_N}$ denote a tensor. The order of $A$ is $N$, the $j$th dimension of $A$ is $I_j$. By convention, the element of tensor is symbolized by corresponding lower-case letters, i. e $(A)_{i_1 i_2 \ldots i_N} = a_{i_1 i_2 \ldots i_N}$.

**Definition 1.** *The mode-n unfolding [46]:assume an* $N$*th-order tensor* $A \in R^{I_1 \times I_2 \times \ldots \times I_N}$. *The mode-n unfolding* $A_{(n)} \in R^{I_n \times (I_{n+1} I_{n+2} \ldots I_N I_1 I_2 \ldots I_{n-1})}$ *contains the element* $a_{i_1 i_2 \ldots i_N}$ *at the* *position with row number $i_n$ and column number equals to*

$$(i_{n+1}-1)I_{n+2}\dots I_N I_1\dots I_{n-1}+(i_{n+2}-1)I_{n+3}\dots I_N I_1\dots I_{n-1}+\dots$$
$$+(i_N-1)I_1\dots I_{n-1}+(i_1-1)I_2\dots I_{n-1}++i_{n-1}$$

**Definition 2.** *The mode-n product [46] of a tensor $A \in R^{I_1\times I_2\times\dots\times I_N}$ by a matrix $U \in R^{J_n\times I_n}$, denoted by $A\times_n U$ is an $(I_1\times I_2\times\dots\times I_{n-1}\times J_n\times I_{n+1}\times\dots\times I_N)$-tensor whose elements are:*

$$(A\times_n U)_{i_1\dots i_{n-1}j_n i_{n+1}\dots i_N}=\sum_{i_n=1}^{I_n} a_{i_1\dots i_{n-1}i_n i_{n+1}\dots i_N}u_{j_n i_n} \tag{5.1}$$

The mode-n product has following properties:

$$(A\times_n U)_{(n)}=UA_{(n)} \tag{5.2}$$

$$(A\times_n U)\times_n V=A\times_n(VU) \tag{5.3}$$

$$(A\times_n U)\times_m V=(A\times_m V)\times_n U \tag{5.4}$$

**Definition 3.** *The outer product [46] of $N$ vectors $U^{(1)},U^{(2)},\dots,U^{(N)}$ denoted by $U^{(1)}\circ U^{(2)}\circ\dots\circ U^{(N)}$ is an $N$th-order tensor and*

$$a_{i_1 i_2\dots i_N}=u^{(1)}_{i_1}u^{(2)}_{i_2}\dots u^{(N)}_{i_N} \tag{5.5}$$

**Definition 4.** *The scalar product [46] $\langle A,B\rangle$ of two tensors $A,B\in R^{I_1\times I_2\times\dots\times I_N}$ is defined as:*

$$\langle A,B\rangle=\sum_{i_1,i_2,\dots i_N} a_{i_1 i_2\dots i_N}b_{i_1 i_2\dots i_N} \tag{5.6}$$

*$A$ and $B$ are orthogonal if their scalar product equals* 0

**Definition 5.** *If we imagine an $N$th-order tensor $A\in R^{I_1\times I_2\times\dots\times I_N}$ as an $(N+1)$th-order tensor $C\in R^{I_1\times I_2\times\dots\times I_N\times 1}$, we call the transpose of C's mode-(N+1) unfolding the vectorization of A, i. e.*

$$vec(A)=(C_{(N+1)})^T$$

As is apparent, the following holds:

$$\langle A,B\rangle=\langle vec(A),vec(B)\rangle=vec(A)^T vec(B) \tag{5.7}$$

### 5.2.2 HOSVD Based Method

#### Multilinear Version [90]

A shortcoming of Principal Components Analysis (PCA) is that it only makes use of the training images, while totally ignoring the labels which contain important discriminative information. HOSVD solves this by using labels as indices to arrange the images into a tensor. However, this implies the labels must satisfy the following condition.

**Definition 6.** *Regular Sample Condition: A set of training images controlled by $k$ factors $y^{(i)}$ , $i = 1, \dots, k$ satisfies Regular Sample Condition if:*

1. *Each factor has $I_i$ distinct states $e_j^{(i)}$, $j = 1, \dots, I_i$, which means every $y^{(i)}$ in the training data is one of $e_j^{(i)}$—it can not be the superposition of multiple states.*

2. *For each combination of states $(e_{j_1}^{(1)}, e_{j_2}^{(2)}, \dots, e_{j_k}^{(k)})$, one and only one sample is present in the training set.*

A direct inference of RSC is the sample size must equal to $I_1 I_2 \dots I_k$.

RSC is a very strong condition in the sense that it not only regulates the number of samples, but also defines which exact $I_1 I_2 \dots I_k$ label combinations are required.

When RSC is satisfied, the training sample can be stacked together to form a $(k+1)$th-order tensor $Z \in R^{I_1 \times I_2 \times \dots \times I_k \times I_{image}}$, where $I_{image}$ is the dimensionality of the image.

HOSVD as defined in [46] is employed to decompose $Z$ as:

$$Z = D \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_k U^{(k)} \tag{5.8}$$

where $U^{(i)}$ represents the subspace of factor $y^{(i)}$ and $D$ is their interaction tensor. Each row vector of $U^{(i)}$ represents a state of corresponding factor, *i.e.*, $u_j^{(i)T}\Big|_{1 \leq j \leq I_i}$ of $U^{(i)} = [u_1^{(i)}, u_2^{(i)}, \dots, u_{I_i}^{(i)}]^T$ represents the $j$th state of the $i$th factor, $e_j^{(i)}$. Letting $Z^{j_1 j_2 \dots j_k} \in R^{1 \times 1 \times \dots \times 1 \times I_{image}}$ denote the tensor containing the image in state $(e_{j_1}^{(1)}, e_{j_2}^{(2)}, \dots, e_{j_k}^{(k)})$, one has:

$$Z^{j_1 j_2 \dots j_k} = D \times_1 (u_{j_1}^{(1)})^T \times_2 (u_{j_2}^{(2)})^T \dots \times_k (u_{j_k}^{(k)})^T \tag{5.9}$$

Given a new image $Z^{new}$, its "implied factor" vectors are $u^{(i)}$ that satisfy:

$$Z^{new} = D \times_1 (u^{(1)})^T \times_2 (u^{(2)})^T \dots \times_k (u^{(k)})^T \tag{5.10}$$

```
Input:
Subspace matrices $U^{(i)}$; interaction tensor $D$
New image $Z^{new}$ (denote its vector form as $z^{new} = vec(Z^{new})$)
Initialization:
    $u^{(i)} = u_1^{(i)}$
    Convergence $flag \leftarrow 0$
end
while flag=0 do
    for i=1 to k do
        $T^{(i)} = D \times_1 (u^{(1)})^T \times_{i-1} (u^{(i-1)})^T \ldots \times_{i+1} (u^{(i+1)})^T \times_k (u^{(k)})^T$
        $u_{new}^{(i)} = [(T^{(i)})_{(i)}^T]^+ z^{new}$
        if $\left| u_{new}^{(i)} - u^{(i)} \right| < \Delta$ then
            $flag \leftarrow 1$
        end
        $u^{(i)} \leftarrow u_{new}^{(i)}$
    end
end
Output:
Implied factor vectors $u^{(i)}$
```

**Algorithm 4**: Multilinear translation for HOSVD [90]

Extracting $u^{(i)}$ is called multilinear "translation" [53] [86] and can be done by iteratively updating $u^{(i)}$ [53]. For example, $u^{(1)}$ can be updated from (5.10) by assuming $u^{(2)}, \ldots u^{(k)}$ are known as:

$$T^{(1)} = D \times_2 (u^{(2)})^T \ldots \times_k (u^{(k)})^T \in R^{I_1 \times 1 \times \ldots \times 1 \times I_{image}} \tag{5.11}$$

$$u^{(1)} = [(T^{(1)})_{(1)}^T]^+ z^{new} \tag{5.12}$$

where $[\ldots]^+$ denotes the inverse function and $z^{new}$ is the vector form of $Z^{new}$. Other $u^{(i)}$ can be updated analogously. The algorithm is summarized in Alg.4.

After $u^{(i)}$ have been estimated, by changing some of them with known factor vectors $u_j^{(i)}$ , one can synthesize new images using(5.10).

**Nonlinear Version [53]**

In Kernel HOSVD, image data are mapped into a higher $I_{feature}$-dimensional feature space $F$ via a nonlinear mapping $\Phi$ implicitly defined by a Kernel function $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$. As in HOSVD, training data form a $(k+1)$th-order tensor $W \in R^{I_1 \times I_2 \times \ldots \times I_k \times I_{feature}}$. Kernel HOSVD decomposes $W$ as:

$$W = D \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_k U^{(k)} \tag{5.13}$$

where $U^{(i)}$ can be found by the SVD of $B^{(i)} \in R^{I_i \times I_i}$ defined according to:

$$(B^{(i)})_{m,n} = \sum_{j_t, t \neq i} K(z^{j_1 \cdots j_{i-1} m j_{i+1} \cdots j_k}, z^{j_1 \cdots j_{i-1} n j_{i+1} \cdots j_k}) \tag{5.14}$$

$D$ is not computable; however, it can be expressed as a linear combination of $\Phi(z^{j_1 j_2 \cdots j_k})$.

Similar to HOSVD, the following holds

$$\Phi(z^{j_1 j_2 \cdots j_k}) = D \times_1 (u_{j_1}^{(1)})^T \times_2 (u_{j_2}^{(2)})^T \ldots \times_k (u_{j_k}^{(k)})^T \tag{5.15}$$

Given a new image $z^{new}$ , Kernel-based "translation" is to find $u^{(i)}$, $i = 1, \ldots, k$ to satisfy

$$\Phi(z^{new}) = D \times_1 (u^{(1)})^T \times_2 (u^{(2)})^T \ldots \times_k (u^{(k)})^T \tag{5.16}$$

This can also be performed using an iterative estimation algorithm, but because some parts are not directly computable the updating procedure is much more complex as shown in Alg.5.

Experiments in [53] showed that kernel HOSVD delivered significantly better results compared to its multilinear counterpart, however this benefit came at the cost of increased complexity: $E^{(i)}$ and $F^{(i)}$ are hard to compute and they need to be re-evaluated in each iteration. Further, HOSVD also needs the data to satisfy RSC.

### 5.2.3 Experiments

Now we test the performance of HOSVD in a system which recognizes identity and facial expression simultaneously. In this experiment, we assume the facial image is controlled by two factors: identity $y^{per}$ and expression $y^{exp}$ and this relationship $I = f(y^{per}, y^{exp})$

```
Input:
Subspace matrices $U^{(i)}$; training images $z^{j_1 j_2 \cdots j_k}$
New image $Z^{new}$
Initialization:
    $u^{(i)} = u_1^{(i)}$
    Convergence $flag \leftarrow 0$
end
while flag=0 do
    for i=1 to k do
        $S^{(t)} = U^{(t)} u^{(t)}, t \neq i$
        Compute $E^{(i)} \in R^{I_i \times I_i}$ and $F^{(i)} \in R^{I_1 \times 1}$
        $(E^{(i)})_{m,n} = \sum_{j_t, t \neq i} \sum_{l_t, t \neq i} K(z^{j_1 \cdots j_{i-1} m j_{i+1} \cdots j_k}, z^{l_1 \cdots l_{i-1} n l_{i+1} \cdots l_k}) \prod_{s \neq i} S_{j_s}^{(s)} S_{l_s}^{(s)}$
        $(F^{(i)})_m = \sum_{j_t, t \neq i} K(z^{j_1 \cdots j_{i-1} m j_{i+1} \cdots j_k}, z^{new}) \prod_{s \neq i} S_{j_s}^{(s)}$
        In which $S_{j_t}^{(t)}$ denotes the $j_t$th element of $S^{(t)}$
        $u_{new}^{(i)} = (U^{(i)})^T [E^{(i)}]^+ F^{(i)}$
        if $\left| u_{new}^{(i)} - u^{(i)} \right| < \Delta$ then
            $flag \leftarrow 1$
        end
        $u^{(i)} \leftarrow u_{new}^{(i)}$
    end
end
Output:
Implied factor vectors $u^{(i)}$
```

**Algorithm 5**: Kernel based translation for KHOSVD [53]

is modeled using HOSVD. Given a testing image, identity and expression recognition is performed by finding the implied factors $y^{per}$ and $y^{exp}$. To evaluate the accuracy, three recognition measures are defined:

$$R = \Pr\{c_p^* = c_p, c_e^* = c_e\} \tag{5.17}$$

$$R^p = \Pr\{c_p^* = c_p\} \tag{5.18}$$

$$R^e = \Pr\{c_e^* = c_e\} \tag{5.19}$$

Apparently, $R \leq \min\{R^e, R^p\}$.

The experiment is conducted on the JAFFE database. Because HOSVD requires training data to satisfy RSC which implies the size of training set must be exactly $10 \times 7 = 70$, in our experiment 70 images are used for training and 143 for testing. The results are shown in Table 5.2 (where "N/A" means program doesn't converge).

Table 5.2: Recognition accuracy of HOSVD

| Feature | Selection | Kernel | Accuracy(%) | | |
|---|---|---|---|---|---|
| | | | $R$ | $R^p$ | $R^e$ |
| Gabor Features | Yes | Gaussian | 89.5 | 95.8 | 91.6 |
| | | No | 87.4 | 93.7 | 88.1 |
| | No | Gaussian | 86.0 | 94.4 | 87.4 |
| | | No | 83.9 | 88.1 | 85.3 |
| AAM coefficient | No | Gaussian | 80.4 | 99.3 | 80.4 |
| | | No | N/A | N/A | N/A |

From Table 5.2 we can see the following things:

- Identity recognition accuracy is always higher than the expression recognition rate, which implies FER is a more challenging problem than identity recognition. This is reasonable if we consider the subtlety of expression, experiments in [6] [57] have shown even trained observers can not distinguish facial expression perfectly.

- Feature selection can improve recognition accuracy. We can understand this from the information theory point of view: feature selection tries to keep the features which contain most useful information, by doing this the Signal-to-Noise ratio is increased and error rate decreases.

- The effect of the Kernel function highly depends on the feature we're dealing with. If the features are already linearly separable, Kernel function doesn't affect the result too much; otherwise the Kernel function can have crucial importance. For example, when AAM coefficient is used, without Kernel function HOSVD doesn't even converge.

Comparing Table 5.1 to Table 5.2 we can further see that for FER task the recognition rates of SVM are slightly higher than that of HOSVD. However, higher order methods have two advantages: first, they take into consideration the interaction between different factors and model the influence of multiple factors in their entirety, so Higher Order methods can perform multiple recognition tasks at once; second, higher order models are invertible, *i.e.*, given a set of factors one can generate the corresponding features and this can be used for image synthesis. As in most applications, when FER is integrated with each identity recognition system or with an image synthesizer program, we choose a higher order method as the classifier. Actually, in the next chapter we'll propose some new higher order methods and show that they can also achieve high accuracy.

## 5.3 System Configuration

Our experiment results boil down to the following two configurations which depend on the application in which the FER system is to be used.

This first configuration (Fig.5.1) is suitable if the FER is implemented as a stand alone system which only performs recognition tasks (it can handle both face recognition and expression recognition if needed). It is a detailed version of Fig.3.1: face detection and alignment are implemented using template matching and eye corner localization, respectively; image normalization consists of 3 steps, *i.e.*, Median filter, Rotating and Resizing and Histogram Equalization; Gabor filters are utilized for feature extraction, controlled by a feature selection module using AdaBoost; and a higher order method is chosen to be the core classifier and gives both expression and identity of the image. In this configuration, there are three statistical methods which need to be trained. It is worthwhile pointing out that the Gabor filter itself doesn't need training (parameters $k$,$\theta$ and $\sigma$ are fixed). We only train the feature selection module and use it to control the Gabor filters.

As shown in the experiments, this configuration can achieve a recognition rate of 95% for expression recognition which is among the best results [100] [55] reported. And compared with existing methods, a great advantage of our system is that it can perform a high accuracy identity recognition at the same time.

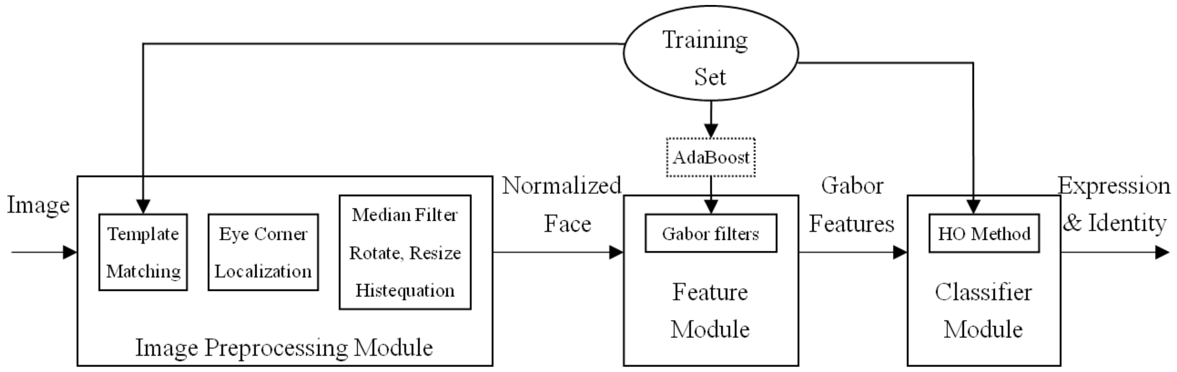


Figure 5.1: FER system configuration I

The second configuration is designed for the cases where the FER is integrated with other systems. Because most applications of face processing involve image coding/compression or image synthesis/animation, it is beneficial to work on some features which can be used to reconstruct the image. So in this case, AAM coefficient is used, and other modules have been adjusted accordingly. As shown in Fig.5.2, this configuration has fewer components: the preprocessing model only consists of an ASM fitting engine, no detection or normalization is necessary. The facial contour found by ASM, together with the original image, is sent into an AAM engine where the AAM Coefficient is computed. And in the classification stage, we also use higher order methods.

Although this configuration has a lower recognition rate (90%), it is considered to be more versatile and can better cooperate with other systems.

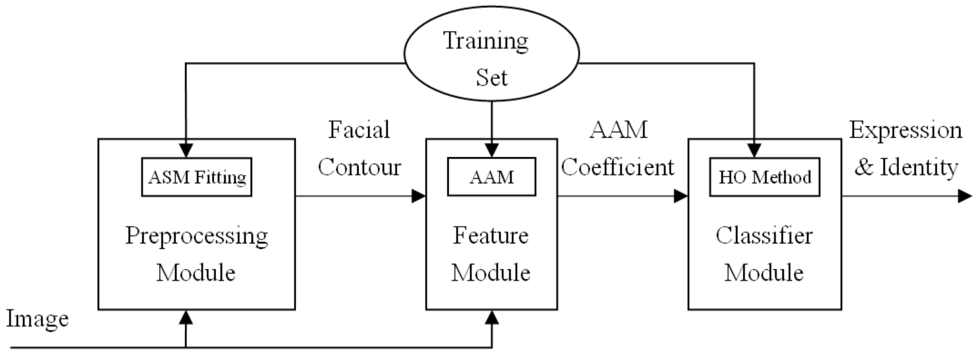


Figure 5.2: FER system configuration II

## 5.4 Chapter Summary

Two classifiers, in particular Support Vector Machine and HOSVD, are talked about in this chapter. They are used to evaluate the performance of different features which give two system configurations.

We have also shown that, although HOSVD has a lower recognition rate than SVM, this new type of method has some very favorable characteristics, so the next chapter will focus on higher order methods. We will propose two algorithms of this type and show that they can also achieve high accuracy.

# Chapter 6

# More On Higher Order Methods

As shown in the last chapter, higher order methods are promising techniques for face recognition. In this chapter, we'll propose two new higher order algorithms, in particular Higher Order Canonical Correlation Analysis and Simple Multi-factor Analysis. Owing to the scarcity of facial expression databases, some experiments in this Chapter are conducted using other data sets—both simple synthetic and real face images are used, however these methods can also be used for FER. And as we'll explain later, actually it's necessary to use these new methods in a real system where the size of the training set is large.

## 6.1 Higher Order Canonical Correlation Analysis

All existing HOSVD-based approaches have two inherent limits: first, they only work on a training set that satisfies RSC, which is a very strong constraint. Another difficulty of HOSVD comes from scaling. Just as with matrix SVD, HOSVD is sensitive to the magnitude of each dimension; however, the choice of scaling factors is somewhat arbitrary, which in turn makes the output of HOSVD arbitrary.

In this section, we propose a new multifactor analysis algorithm called Higher Order CCA, which is a natural generalization of Canonical Correlation Analysis (CCA). Our work is partially motivated by [107], where CCA was used to perform the Facial Expression Recognition task. CCA is a classical multivariate method concerned with describing linear dependencies between two variables. Here we extend it by adopting multilinear algebra to study the dependencies between observed image and its latent parameters. We also Kernelized our method to deal with nonlinear problems. Our preliminary experiments have demonstrated that the CCA-based framework is superior to previous HOSVD-based ones.

Kim *et al.* [40] have recently proposed another extension of CCA using multilinear algebra: so called "Tensor CCA". Although it's not used for multifactor analysis, mathematically it is complementary to the method proposed here: in their paper a set of vectors are found for a given tensor, while conversely we fixed vectors and found a tensor. Another significant difference is that theirs handles only linear cases, while ours is nonlinear.

The remainder of this section is organized as follows: 6.1.1 gives the definition of Canonical Correlation Analysis which is extended to higher order in 6.1.2. The details of the Kernel-based extension are addressed in 6.1.3. In 6.1.4, our method is used for recognition and synthesis tasks, followed by a short summary in 6.1.5.

### 6.1.1 Canonical Correlation Analysis

Canonical Correlation Analysis (as introduced by Hotelling) is concerned with describing linear dependency between two zero-mean random variables $x \in R^{n_x}$ and $y \in R^{n_y}$. From a machine-learning perspective, it may be more familiar to think of $x$ as observation while $y$ describes the underlying parameter. The aim of CCA is to determine a pair of directions $\omega_x$ and $\omega_y$ such that the correlation $\rho(x, y; \omega_x, \omega_y)$ between the two canonical variables $a = \omega_x^T x$ and $b = \omega_y^T y$ is maximized [107], where:

$$\rho(x, y; \omega_x, \omega_y) = \frac{E\{\omega_x^T x y^T \omega_y\}}{\sqrt{E\{\omega_x^T x x^T \omega_x\}}\sqrt{E\{\omega_y^T y y^T \omega_y\}}} \tag{6.1}$$

Suppose that $\{x_i, y_i\}_{i=1,\dots,N}$ are $N$ observations of $x$ and $y$. Then the aim of CCA is equivalent to finding $\omega_x$ and $\omega_y$ that maximize:

$$\rho(x, y; \omega_x, \omega_y) = \frac{\omega_x^T X Y^T \omega_y}{\sqrt{\omega_x^T X X^T \omega_x}\sqrt{\omega_y^T Y Y^T \omega_y}} \tag{6.2}$$

where: $X = [x_1, x_2, \dots, x_N]$,$Y = [y_1, y_2, \dots, y_N]$

Because only the direction of $\omega_x$ and $\omega_y$ are of concern, they are rescaled to satisfy:

$$\omega_x^T X X^T \omega_x = \omega_y^T Y Y^T \omega_y = 1 \tag{6.3}$$

CCA is formulated as a constraint optimization problem:

$$\begin{aligned} \underset{\omega_x, \omega_y}{\operatorname{argmax}} \quad & \omega_x^T X Y^T \omega_y \\ s.t. \quad & \omega_x^T X X^T \omega_x = \omega_y^T Y Y^T \omega_y = 1 \end{aligned} \tag{6.4}$$

The solution of (6.4) gives the first pair of canonical vectors $(\omega_{x1}, \omega_{y1})$ and $a_1 = \omega_{x1}^T x$, $b_1 = \omega_{y1}^T y$ are the corresponding canonical variates. Up to $r = \min\{\dim(X), \dim(Y)\}$ pairs of canonical vectors $(\omega_{xj}, \omega_{yj})$ can be recursively defined subject to $\omega_{xi} \perp \omega_{xj}, \omega_{yi} \perp \omega_{yj}, 1 \leq i < j$. $span\{\omega_{x1}, \omega_{x2}, \ldots \omega_{xr}\}$ defines a subspace within which $x$ is closely related to $y$. Compared to PCA which tends to find a subspace that maximizes variation, the objective function of CCA, correlation, is scaling invariant. CCA abandons those dimensions in which $x$ is independent from $y$ even if the variation of $x$ is large. In general, this is favorable because there is little useful information as $x$ and $y$ are independent.

### 6.1.2 Higher Order CCA for Multifactor Analysis

In this part, we first extend CCA to multilinear cases, then explain how to use it to perform image analysis and synthesis.

#### HOCCA

CCA only analyzes the dependency between observation $x$ and one factor $y$. We extend it to handle multifactor cases where $x$ is influenced by $k$ underlying factors $y^{(i)} \in R^{n_y^{(i)}}, i = 1, \ldots, k$. The basic idea is to find a direction $\omega$ in which the variation of $x$ can be best approximated by a multilinear function $f(y^{(1)}, y^{(2)} \ldots y^{(k)})$. Similar to CCA, we maximize the correlation between $\omega^T x$ and $f(y^{(1)}, y^{(2)} \ldots y^{(k)})$ :

$$\rho(f(y^{(1)}, y^{(2)} \ldots y^{(k)}), \omega^T x) = \frac{E\{\omega^T x f(y^{(1)}, y^{(2)} \ldots y^{(k)})\}}{\sqrt{E\{\omega^T x x^T \omega\}} \sqrt{E\{f(y^{(1)}, y^{(2)} \ldots y^{(k)})^2\}}} \tag{6.5}$$

Here, we suppose both $\omega^T x$ and $f(y^{(1)}, y^{(2)} \dots y^{(k)})$ are centered. Centering of $\omega^T x$ is trivial, it will be shown later how to center $f(y^{(1)}, y^{(2)} \dots y^{(k)})$.

Generally, $f(y^{(1)}, y^{(2)} \dots y^{(k)})$ can be expressed by a $k$th-order tensor $T \in R^{n_y^{(1)} \times n_y^{(2)} \times \dots \times n_y^{(k)}}$ as

$$f(y^{(1)}, y^{(2)} \dots y^{(k)}) = T \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T} \tag{6.6}$$

Thus, the optimization problem of HOCCA is:

$$\operatorname*{argmax}_{\omega_x, T} \frac{E\{\omega^T x T \times_1 y^{(1)T} \dots \times_k y^{(k)T}\}}{\sqrt{E\{(\omega^T x)^2\}}\sqrt{E\{(T \times_1 y^{(1)T} \dots \times_k y^{(k)T})^2\}}} \tag{6.7}$$

It can be proved that when all $y^{(1)}, y^{(2)} \dots y^{(k)}$ are column vectors, the following holds:

$$T \times_1 y^{(1)T} \dots \times_k y^{(k)T} = \left\langle T, y^{(1)} \circ \dots \circ y^{(k)} \right\rangle \tag{6.8}$$

where $\langle A, B \rangle$ is the scalar product of two tensors and $y^{(1)} \circ y^{(2)} \circ \dots \circ y^{(k)}$ is the outer product of $y^{(1)}, y^{(2)} \dots y^{(k)}$. The proof of (6.8) is outlined at the end of the subsection.

According to (5.7) and (6.8):

$$T \times_1 y^{(1)T} \dots \times_k y^{(k)T} = T_{vec}^T vec(y^{(1)} \circ \dots \circ y^{(k)}) \tag{6.9}$$

where $T_{vec}$ is the vectorization of $T$, *i.e.*, $T_{vec} = vec(T)$.

Thus solving a Tensor CCA problem is equivalent to:

$$\begin{aligned} \operatorname*{argmax}_{T_{vec}, \omega} \quad & E\{\omega^T x T_{vec}^T vec(y^{(1)} \circ y^{(2)} \circ \dots \circ y^{(k)})\} \\ s.t. \quad & E\{(T_{vec}^T vec(y^{(1)} \circ y^{(2)} \circ \dots \circ y^{(k)}))^2\} = 1 \\ & E\{(\omega^T x)^2\} = 1 \end{aligned} \tag{6.10}$$

Despite the difference in notation, (6.10) is the same optimization problem as (6.4) and can be solved by translating into an eigenvalue problem. The detail is omitted here, as it can also be considered as a special case of Kernel HOCCA and solved using the method that will be given in the next section.

**Multifactor Analysis Using HOCCA**

Somehow, $r$ pairs of $(\omega_j, T_j)$ have been found and $\rho_1, \rho_2, \dots \rho_r$ the $r$ corresponding correlation values. Let $a_j = \omega_j^T x$ and $b_j = T_j \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T}$. Suppose that $\rho_j \approx 1$, which means $a_j$ and $b_j$ are linearly correlated. So we can find $\theta_j$ and $d_j$ such that:

$$a_j = \theta_j b_j + d_j, \quad (j = 1, 2, \dots, r) \tag{6.11}$$

Note that both $a_j$ and $b_j$ are centered, according to regression analysis [48]:

$$d_j = E\{a_j\} - E\{b_j\} = 0 \tag{6.12}$$

$$\theta_j = \frac{E\{a_j b_j\}}{E\{b_j^2\}} \tag{6.13}$$

Thus $x$ can be expressed by $y$ as:

$$x = \sum_{j=1}^{r} \omega_j \theta_j (T_j \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T}) + \varepsilon_\perp \tag{6.14}$$

where $\varepsilon_\perp$ is orthogonal to $span\{\omega_1, \omega_2, \dots, \omega_r\}$ thus weakly related to $y^{(1)}, y^{(2)} \dots y^{(k)}$. Then the best multilinear approximation of $x$ is:

$$x = \sum_{j=1}^{r} \omega_{xj} \theta_j (T_j \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T}) \tag{6.15}$$

Equation (6.15) can be further simplified to:

$$x = vec(Z \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T}) \tag{6.16}$$

where $Z \in R^{n_y^{(1)} \times n_y^{(2)} \times \dots \times n_y^{(k)} \times n_x}$ is a (k+1)th-order tensor defined as:

$$z_{i_1 i_2 \dots i_k t} = \sum_{j=1}^{r} \theta_j (T_j)_{i_1 i_2 \dots i_k} (\omega_j)_t \tag{6.17}$$

in which $(T_j)_{i_1 i_2 \dots i_k}$ and $(\omega_j)_t$ are elements of $T_j$ and $\omega_j$, respectively. It's worth pointing out that the vectorization operation in (6.16) is merely for the sake of strictness: $Z \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T}$ is a tensor with first $k$ dimensions singletons. In some articles [53] [95], this conversion is implicit.

Equation (6.11) and (6.16) can be used for either image synthesis or pattern classification. In a synthesis task, we infer the observation on a set of new parameters. One simple case is that in which $y^{(1)}, y^{(2)} \dots y^{(k)}$ are explicitly given, thus $x$ can be calculated using (6.16) directly. In a more complex case, some parameters are implicit in an image $x^{old}$; for example, given a facial image, we want to imagine this person's facial appearance with a different expression. Then the first step is to find the implied factors $y^{(1)}, y^{(2)} \dots y^{(k)}$ which solve:

$$x^{old} = vec(Z \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T}) \tag{6.18}$$

This can be done using multilinear "translation" (Alg.4). After $y^{(1)}, y^{(2)} \dots y^{(k)}$ have been estimated, we can synthesize a new image by changing some instances of $y^{(i)}$.

In a recognition task, "translation" can also be used. However, here we give an alternate solution based on vectorization, in which no iteration is needed.

Given $(\omega_j, T_j)$, $\theta_j$ and the image of interest $x$, $a_j$ are computable, in turn $b_j$ can be calculated from (6.11). Thus we have $r$ equations:

$$b_j = T_j \times_1 y^{(1)T} \times_2 y^{(2)T} \dots \times_k y^{(k)T}, \quad (j = 1, 2, \dots, r) \tag{6.19}$$

From (6.9) and (6.19), we obtain:

$$vec(y^{(1)} \circ y^{(2)} \circ \dots \circ y^{(k)}) = \begin{bmatrix} vec(T_1)^T \\ vec(T_2)^T \\ \vdots \\ vec(T_r)^T \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_r \end{bmatrix} \tag{6.20}$$

Because in classification only the class indices are of interest, we can directly choose the element of $y^{(1)} \circ y^{(2)} \circ \dots \circ y^{(k)}$ which has the largest absolute value, *i.e.*, :

$$(c_1, \dots, c_k) = \underset{i_1,\dots,i_k}{\operatorname{argmax}} \left| (y^{(1)} \circ \dots \circ y^{(k)})_{i_1 \dots i_k} \right| \tag{6.21}$$

where $c_1, \dots, c_k$ give the class indices of each factor.

In fact $y^{(1)}, y^{(2)}, \dots, y^{(k)}$ can also be estimated from (6.20) by finding the rank-1 approximation [47] of $y^{(1)} \circ y^{(2)} \circ \dots \circ y^{(k)}$ . Although the algorithm used for rank-1 decomposition involves iteration, however, it's useful in Kernel HOCCA which will be shown in next section. After $y^{(1)}, y^{(2)}, \ldots, y^{(k)}$ are obtained, class labels can also be found by

$$c_j^* = \underset{m}{\operatorname{argmax}} \left|(y^{(j)})_m\right| \tag{6.22}$$

Usually, $c_j^* = c_j$, but that's not always the case. This will be shown later in experiment.

*Proof of (6.8).* Consider the left hand side:

$$(T \times_1 y^{(1)T})_{1i_2i_3\ldots i_k} = \sum_{j_1} y_{j_1}^{(1)} u_{j_1i_2i_3\ldots i_k} \tag{6.23}$$

$$(T \times_1 y^{(1)T} \times_2 y^{(2)T})_{11i_3i_4\ldots i_k} = \sum_{j_2} y_{j_2}^{(2)} \sum_{j_1} y_{j_1}^{(1)} u_{j_1j_2i_3\ldots i_k} \tag{6.24}$$

By induction, one can obtain:

$$(T \times_1 y^{(1)T} \times_2 y^{(2)T} \ldots \times_k y^{(k)T})_{11\ldots 1} = \sum_{j_1,j_2\ldots j_k} y_{j_1}^{(1)} y_{j_2}^{(2)} \ldots y_{j_k}^{(k)} u_{j_1j_2\ldots j_k} \tag{6.25}$$

From the definition of outer product and scalar product, right hand side of (6.8) equals:

$$\sum_{i_1,i_2\ldots i_k} (y^{(1)} \circ y^{(2)} \circ \ldots \circ y^{(k)})_{i_1i_2\ldots i_k} u_{i_1i_2\ldots j_k} = \sum_{i_1,i_2\ldots i_k} y_{i_1}^{(1)} y_{i_2}^{(2)} \ldots y_{i_k}^{(k)} u_{i_1i_2\ldots i_k} \tag{6.26}$$

Despite minor notational difference, (6.25) and (6.26) are equivalent. □

### 6.1.3 Kernel HOCCA

The goal of this this section is to enable HOCCA to handle nonlinear problem. Similar to last section, here we first give the formulas of Kernel HOCCA then explain how to use their output. HOCCA and the multilinear "translation" can be extended by employing Kernel approach. Imagine $x$ is mapped into a Hilbert space $F$ through a nonlinear mapping $\Phi : R^{n_x} \to F, x \to \Phi(x)$. Also, we suppose that $\Phi(x)$ is centered in $F$. For a method to center $\Phi(x)$, see [73].

Similar to (6.7), the correlation function in $F$ can be formulated as:

$$\rho(f(y^{(1)}, \ldots, y^{(k)}), \omega_{\Phi(x)}^T \Phi(x)) = \frac{E\{\omega_{\Phi(x)}^T \Phi(x) T \times_1 y^{(1)T} \ldots \times_k y^{(k)T}\}}{\sqrt{E\{(\omega_{\Phi(x)}^T \Phi(x))^2\}} \sqrt{E\{(T \times_1 y^{(1)T} \ldots \times_k y^{(k)T})^2\}}} \tag{6.27}$$

Suppose that $\{x_i; y_i^{(1)}, \ldots, y_i^{(k)}\}_{i=1,\ldots,N}$ are $N$ observations of $x$ and $y^{(i)}$. Define $X$, $\Phi(X)$ and $Y_{vec}$ as:

$$X = [x_1, \ldots, x_N], \quad \Phi(X) = [\Phi(x_1), \ldots, \Phi(x_N)]$$

$$Y_{vec} = \left[vec(y_1^{(1)} \circ \ldots \circ y_1^{(k)}), \ldots, vec(y_N^{(1)} \circ \ldots \circ y_N^{(k)})\right]$$

Recall that we assumed $E\{T \times_1 y^{(1)T} \times_2 y^{(2)T} \ldots \times_k y^{(k)T}\} = 0$, by (6.9) this can be guaranteed by centering $Y_{vec}$ .

The optimization problem of Kernel HOCCA can then be expressed as:

$$\underset{\omega_{\Phi(x)}, T_{vec}}{\operatorname{argmax}} \frac{\omega_{\Phi(x)}^T \Phi(X) Y_{vec}^T T_{vec}}{\sqrt{\left\|\omega_{\Phi(x)}^T \Phi(X)\right\|^2} \sqrt{\|T_{vec}^T Y_{vec}\|^2}} \tag{6.28}$$

Therefore, the goal of Kernel Higher Order CCA is equivalent to finding $\omega_{\Phi(x)}$ and $Y_{vec}$ that maximize $\omega_{\Phi(x)}^T \Phi(X) Y_{vec}^T T_{vec}$ under the constraints:

$$\omega_{\Phi(x)}^T \Phi(X) \Phi(X)^T \omega_{\Phi(x)} = T_{vec}^T Y_{vec} Y_{vec}^T T_{vec} = 1 \tag{6.29}$$

Since $\Phi(\ldots)$ is not explicitly known (instead, only the Kernel function is given which is defined by the dot production in $F$ as, $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle = \Phi(x_i)^T \Phi(x_j)$) all computation in $F$ should be formulated as dot products. According to [30], there exists $\alpha_{\Phi(x)}$, such that:

$$\omega_{\Phi(x)} = \Phi(\mathrm{X}) \alpha_{\Phi(x)} \tag{6.30}$$

From (6.29) and (6.30), one obtains:

$$\omega_{\Phi(x)} \Phi(\mathrm{X}) Y_{vec}^T T_{vec} = \alpha_{\Phi(x)}^T K Y_{vec}^T T_{vec} \tag{6.31}$$

$$\omega_x^T \Phi(X) (\Phi(X))^T \omega_x = \alpha_{\Phi(x)}^T K K \alpha_{\Phi(x)} = 1 \tag{6.32}$$

where:

$$K = \Phi(X)^T \Phi(X) \tag{6.33}$$

$K$ is known as the Gram matrix and is computable from Kernel function by:

$$(k)_{ij} = \Phi(x_i)^T \Phi(x_j) = k(x_i, x_j) \tag{6.34}$$

Now Φ(X) has been eliminated and the Kernel HOCCA problem is converted into:

$$\begin{aligned} \underset{\alpha_{\Phi(x)}, T_{vec}}{\text{argmax}} \quad & \alpha_{\Phi(x)}^T K Y_{vec}^T T_{vec} \\ s.t. \quad & T_{vec}^T Y_{vec} Y_{vec}^T T_{vec} = 1 \\ & \alpha_{\Phi(x)}^T K K \alpha_{\Phi(x)} = 1 \end{aligned} \tag{6.35}$$

The corresponding Lagrangian is:

$$L(\alpha_{\Phi(x)}, T_{vec}, \lambda, \mu) = \alpha_{\Phi(x)}^T K Y_{vec}^T T_{vec} - \frac{\lambda}{2}(T_{vec}^T Y_{vec} Y_{vec}^T T_{vec} - 1) - \frac{\mu}{2}(\alpha_{\Phi(x)}^T K K \alpha_{\Phi(x)} - 1)$$

Taking derivatives with respect to $\alpha_{\Phi(x)}$ and $T_{vec}$ and setting to zeros, one obtains:

$$\frac{\partial L}{\partial \alpha_{\Phi(x)}} = K Y_{vec}^T T_{vec} - \mu K K \alpha_{\Phi(x)} = 0 \tag{6.36}$$

$$\frac{\partial L}{\partial T_{vec}} = Y_{vec} K \alpha_{\Phi(x)} - \lambda Y_{vec} Y_{vec}^T T_{vec} = 0 \tag{6.37}$$

From (6.36) and (6.37), one has:

$$\lambda = \mu = \alpha_{\Phi(x)}^T K Y_{vec}^T T_{vec} \tag{6.38}$$

$$T_{vec} = \frac{[Y_{vec} Y_{vec}^T]^{-1} Y_{vec} K \alpha_{\Phi(x)}}{\lambda} \tag{6.39}$$

$$K Y_{vec}^T [Y_{vec} Y_{vec}^T]^{-1} Y_{vec} K \alpha_{\Phi(x)} = \lambda^2 K K \alpha_{\Phi(x)} \tag{6.40}$$

Apparently, (6.40) is a generalized eigen problem which can be solved by QZ factorization [2]. Equation (6.38) implies the correlation coefficient is equal to the square root of the corresponding eigenvalue, *i.e.*, $\rho_j = \sqrt{\lambda_j^2}$, so we want to choose the $r$ largest eigenvalues $1 \geq \lambda_1^2 \geq \lambda_2^2 \geq \ldots \geq \lambda_r^2 \geq 0$. After $\alpha_{\Phi(x)}$ has been obtained, $T_{vec}$ and, in turn, $T$ can be found by (6.39).

After $r$ pairs of $(\alpha_{\Phi(x)-j}, T_j)$ have been found, define $a_j$ and $b_j$ according to:

$$\begin{aligned} a_j &= \omega_{\Phi(x)}^T \Phi(x) = \alpha_{\Phi(x)-j}^T \Phi(X)^T \Phi(x) \\ b_j &= T_j \times_1 y^{(1)T} \times_2 y^{(2)T} \ldots \times_k y^{(k)T} \end{aligned}$$

As both $a_j$ and $b_j$ are computable, as for HOCCA, formally $\Phi(x)$ can be approximated by:

$$\Phi(x) = \sum_{j=1}^{r} \Phi(\mathrm{X})\alpha_{\Phi(x)}\theta_j(T_j \times_1 y^{(1)T} \times_2 y^{(2)T} \ldots \times_k y^{(k)T}) \tag{6.41}$$

In which:

$$\theta_j = \frac{E\{a_j b_j\}}{E\{b_j^2\}}$$

Note that, $\Phi(\mathrm{X})$ in (6.41) should be substituted. To do this, multiply (6.41) by $\Phi(X)^T$, *i.e.*, $\Phi(X)^T \cdot$(6.41) which gives:

$$K_x = K \sum_{j=1}^{r} \alpha_{\Phi(x)-j}\theta_j(T_j \times_1 y^{(1)T} \times_2 y^{(2)T} \ldots \times_k y^{(k)T}) \tag{6.42}$$

In which:

$$K_x = \Phi(X)^T\Phi(x) = \begin{bmatrix} k(x_1, x) \\ k(x_2, x) \\ \vdots \\ k(x_N, x) \end{bmatrix} \tag{6.43}$$

Equation (6.42) can be rewritten into a tensor form:

$$K_x = vec(Z_{\text{Kernel}} \times_1 y^{(1)T} \times_2 y^{(2)T} \ldots \times_k y^{(k)T}) \tag{6.44}$$

where $Z_{\text{Kernel}} \in R^{n_y^{(1)} \times n_y^{(2)} \times \ldots \times n_y^{(k)} \times N}$ is a (k+1)th-order tensor defined according to:

$$z_{i_1 i_2 \ldots i_k t} = \sum_{n=1}^{N}\sum_{j=1}^{r} \theta_j (T_j)_{i_1 i_2 \ldots i_k} k_{tn} (\alpha_{\Phi(x)-j})_n \tag{6.45}$$

In synthesis-related tasks, given $y^{(1)}, y^{(2)} \ldots y^{(k)}$, $K_x$ can be obtained from (6.44), and the pre-image $x$ can be recovered from $K_x$ using gradient descent methods [64].

For implied factor estimation, *i.e.*, "translation" and classification, a vectorization-based approach is favored here: after $a_j$ and $b_j$ have been calculated, no Kernel function is involved. Therefore, (6.19)-(6.21) can be used exactly the same way as in the multilinear case.

As shown above, despite calculation of Gram matrix $K$, solving Kernel HOCCA is not significantly harder than solving HOCCA, compared to Kernel HOSVD [53] which is very complicated.

### 6.1.4 Experiments

In this experiment, we use HOCCA to implement a face recognition system, which analyzes person identity and facial expression simultaneously.

#### Data Preparing

Our experiments were conducted on the Japanese female facial expression (JAFFE) database [58]. As illustrated earlier, Gabor features have better accuracy in recognition task, so most of our experiment in this section using these features.

Besides an identity label and an expression label assigned to each image, the JAFFE database also provided semantic expression ratings evaluated by psychologists in which a quantitative evaluation is given for each of the six basic expressions (neutral excluded). Person identity and expression are considered as two underlying factors $y^p$ and $y^e$, respectively. $y^p$ is represented by a 10-dimensional vector: assume that $y_i^p$ corresponds to the $i$th training image $x_i$, and let $(y_i^p)_{(k)}$ denote the $k$th element of $y_i^p$. If $x_i$ belongs to person $j$, set $(y_i^p)_{(j)} = 1$, otherwise, set $(y_i^p)_{(j)} = 0$. Expression label is translated into a 7-dimensional vector $y^{e-1}$ analogously. The semantic expression rating can be considered as a 6-dimensional vector $y^{e-2}$ as it is.

In our experiment, an identity and expression recognition system is realized using HOCCA. We test its performance with different parameters and compare it with an HOSVD-based method. In the first part of this experiment, the semantic expression rating is used, whereas the second part uses the expression label, and in all experiments Leave-One-Out validation is used unless otherwise noted.

#### Experiment Using Expression Rating

As mentioned above, HOSVD-based methods only works where $y^{(j)}$ is pure state, which is not the case for semantic ratings. Therefore, HOSVD is not capable here and this experiment is devoted to testing the performance of Kernel HOCCA using different parameters.

Because only six basic emotions are used in the semantic rating, we exclude neutral expression images in this experiment (as we cannot define their accuracy) and use the remaining 183 images.

In the first testing, we estimate the implied identity vector $y^p$ and expression vector $y^e$ using (6.22), which we will refer to as the original scheme in contrast to the simplified scheme (6.21). A typical result is shown in Fig.6.1, where the left column illustrates the semantic ratings given by people (which are considered as ground truth), whereas the right column shows the implied $y^p$ and $y^e$ (all figures are normalized into the scope of [0,1]).

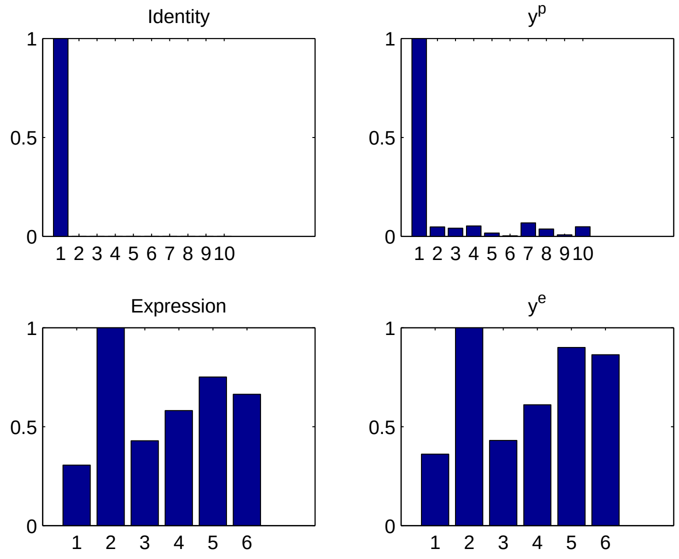


Figure 6.1: Semantic ratings. Figures in the left column are ground truth, while those in the right column are implied factors. 1-10 in upper figures represent 10 individuals; 1-6 in lower figures represent: happiness, sadness, surprise, anger, disgust and fear, respectively.

From Fig.6.1 one can see: first, that identity recognition is much easier than expression recognition, expression is never purely an expression of just one emotion, but is always an admixture of different emotions, which even people can not distinguish perfectly. Second, the estimated results closely approximate the ground truth. Based on Fig.6.1, this image should be classified as: $c_p^* = 1$, $c_e^* = 2$ which means 1st person, 2nd expression (sadness). These outcomes are compared with identity label $c_p$ and expression label $c_e$, 3 different accuracies $R$, $R^p$ and $R^e$ are calculated as defined in 5.2.3

Recognition results of both HOCCA and Kernel HOCCA are shown in Table 6.1.

Table 6.1: Recognition accuracy using semantic rating (original scheme)

| | Accuracy(%) | | |
|---|---|---|---|
| | R | $R^p$ | $R^e$ |
| HOCCA | 77.1 | 100.0 | 77.1 |
| KHOCCA(Monomial Kernel d=2) | 80.0 | 100.0 | 80.0 |
| KHOCCA(Monomial Kernel d=3) | 78.7 | 100.0 | 78.7 |
| KHOCCA(Polynomial Kernel p=2) | 80.0 | 100.0 | 80.0 |
| KHOCCA(Polynomial Kernel p=3) | 78.7 | 100.0 | 78.7 |
| KHOCCA(Gaussian Kernel $\sigma = 2e5$ ) | 80.0 | 99.5 | 80.0 |
| KHOCCA(Gaussian Kernel $\sigma = 2e6$ ) | 79.2 | 100.0 | 79.2 |

Table 6.2: Recognition accuracy using semantic rating (simplified scheme)

| | Accuracy(%) | | |
|---|---|---|---|
| | R | $R^p$ | $R^e$ |
| HOCCA | 78.7 | 100.0 | 78.7 |
| KHOCCA(Monomial Kernel d=2) | 80.0 | 100.0 | 80.0 |
| KHOCCA(Monomial Kernel d=3) | 79.2 | 100.0 | 79.2 |
| KHOCCA(Polynomial Kernel p=2) | 80.0 | 100.0 | 80.0 |
| KHOCCA(Polynomial Kernel p=3) | 79.7 | 100.0 | 79.7 |
| KHOCCA(Gaussian Kernel $\sigma = 2e5$) | 78.7 | 100.0 | 78.7 |
| KHOCCA(Gaussian Kernel $\sigma = 2e6$) | 79.2 | 100.0 | 79.2 |

From Table 6.1, one can see that the Kernel HOCCA always delivers better results than the multilinear version, though in general the accuracy of different Kernel functions are close; HOCCA can be considered as KHOCCA with a linear Kernel. And again, it's shown that the recognition accuracy of identity is significantly higher than that of expression. This is partly because expression is harder to recognize, and partly because of the fuzziness in the semantic ratings.

In the second testing, the class label is estimated using the simplified scheme: $c_p$ and $c_e$ are determined directly from $c_p \circ c_e$ using (6.21). Results are shown in Table 6.2.

Comparing Table 6.1 and Table 6.2, one can see that the simplified scheme can achieve similar accuracy. The advantage of the original method is obvious: $y^p$ and $y^e$ can be used as a fuzzy classification result which contains much more information than $c_p$, $c_e$; also a reliability assessment can be calculated which is of special importance for expression recognition. However, this benefit comes at the cost of increased computation to estimate $y^p$ and $y^e$: performed using “translation” or rank-1 decomposition, both involve iteration.

### Experiment Using Class label

In this scenario, class label information is used for both identity and expression, and the proposed method is compared with an HOSVD-based method [53]. Kernel versions are used for both methods and only the results using Gaussian Kernel ($\sigma = 2e5$) are listed, since the others are very similar.

It’s worthwhile emphasizing that because of the RSC requirement, for HOSVD-based method 70 images serve as training data whereas the other 143 images are used as testing data.

Table 6.3 shows that the proposed method achieved higher accuracy than the HOSVD-based method, which is reasonable because: a) HOCCA picks directions most related to underlying factors regardless of the magnitude of vector, while HOSVD attempts to find the direction with the largest variation which is sensitive to re-scaling; b) HOCCA makes full use of training data, whereas a HOSVD-based method has various constraints. Comparison between Table 6.1, Table 6.2 and Table 6.3 also suggests recognition accuracy is higher when the class label is used. Similar results have been reported in [107] and these can be explained as: recognition using semantic rating suffers from the fuzziness of training data and this problem is “effectively” solved in Class Label information by forcing each image to have clear classification; nevertheless, this “defuzzification” process is itself arbitrary.

The result based on AAM coefficient is tabulated in Table 6.5. Only simplified scheme result is shown, the original scheme gives very similar result. To facilitate the comparison, the data in Table 5.2 are re-listed.

By Comparing Table 5.1 and 6.5 we can further see that for an FER task the recognition rates of SVM and HOCCA are very close.

Table 6.3: Recognition accuracy using class label

| | Accuracy(%) | | |
|---|---|---|---|
| | R | $R^p$ | $R^e$ |
| KHOCCA(Original Scheme) | 94.3 | 100.0 | 94.8 |
| KHOCCA(Simplified Scheme) | 95.3 | 100.0 | 95.3 |
| KHOSVD | 89.5 | 95.8 | 91.6 |

Table 6.4: HOCCA Vs. HOSVD

| Feature | Selection | Method | Kernel | Accuracy(%) | | |
|---|---|---|---|---|---|---|
| | | | | R | $R^p$ | $R^e$ |
| Gabor Feature | Yes | HOCCA | Gaussian | 95.3 | 100.0 | 95.3 |
| | | | No | 93.4 | 99.1 | 93.4 |
| | | HOSVD | Gaussian | 89.5 | 95.8 | 91.6 |
| | | | No | 87.4 | 93.7 | 88.1 |
| | No | HOCCA | Gaussian | 93.9 | 100.0 | 95.3 |
| | | | No | 85.4 | 90.6 | 86.4 |
| | | HOSVD | Gaussian | 86.0 | 94.4 | 87.4 |
| | | | No | 83.9 | 88.1 | 85.3 |
| AAM coefficient | No | HOCCA | Gaussian | 89.7 | 99.1 | 89.7 |
| | | | No | 10.8 | 28.6 | 22.1 |
| | | HOSVD | Gaussian | 80.4 | 99.3 | 80.4 |
| | | | No | N/A | N/A | N/A |

### 6.1.5 Comment

HOCCA is superior to HOSVD in two senses: first it releases the RSC constraint on training data, which makes it capable of handling general applications; Also, HOCCA always delivers better results than HOSVD, which is expected because HOCCA doesn't need its training set to satisfy the RSC condition, and so it can make use of available data more efficiently.

HOCCA is more favorable than SVM, especially in a system where both facial expression and person identity are needed.

However, there's a problem too: HOCCA is also based on eigen-decomposition which is in general very computational. In the next section we'll explore how to simplify the computation by utilizing the special structure of training data.

## 6.2 Simplify Multifactor Analysis

Most existing multifactor analysis methods are computational, and the nonlinear versions in particular are too complex in either a temporal [53] or spatial [71] sense. In this section, we discuss how can we simplify those methods by utilizing the special structure of training image.

A problem in many vision researches is the so called Small Sample Size(SSS), *i.e.*, the size of the training set is smaller than the dimensionality of image data. Many statistical methods suffer from Small Sample Size when dealing with high-dimensional data. Intuitively, this is because there are insufficient training samples to provide enough information about the structure of the data. Normally, SSS is referred to as a problem needing to be solved [12] [105] [39], however if we look at it from a more positive perspective, we can make use of SSS to simplify computation. The most well-known example may be the correlation method for computing PCA.

In this section, we first study the geometry of Higher Order Canonical Correlation Analysis(HOCCA) and define a Modified HOCCA which can be considered as an approximation of HOCCA under SSS conditions. In Modified HOCCA, a large tensor optimization is divided into a series of independent simple linear equations so that space complexity is no longer an issue. And then we also study how to take advantage of the Regular Sample Condition(RSC), which is a more special case, and also the most commonly-used one in the literature [95] [90] [91] [53] [92]. Under RSC, Modified HOCCA is further reduced to a Simple Multifactor Analysis (SMA). Despite notational difference, SMA actually models the same multilinear structure as HOSVD-based methods [95] [53] and this equivalence is shown both mathematically and experimentally.

The rest of the section is organized as follows: In 6.2.1 we'll briefly restate the formulas of HOCCA and point out its limitation. 6.2.2 gives the formulas of Modified HOCCA under

SSS, which are further simplified to SMA in 6.2.3. In 6.2.4, the proposed methods are tested on both synthetic and real images and compared with a state-of-the-art algorithm [53].

## 6.2.1 HOCCA

HOCCA deals with general training data in which underlying factors can be either pure states or superpositions of multiple states. Given $N$ images and their underlying factors $\{(x_n; y_n^{(1)}, y_n^{(2)}, \ldots, y_n^{(k)})\}_{n=1,\ldots,N}$ , the aim of HOCCA is to determine $r$ pairs of $(\omega_i, T_i)$ that maximize the correlation:

$$\rho(\omega^T x, T \times_1 (y^{(1)})^T \ldots \times_k (y^{(k)})^T) = \frac{\sum_{n=1}^{N} \omega^T x_n T \times_1 (y_n^{(1)})^T \ldots \times_k (y_n^{(k)})^T}{\sqrt{\sum_{n=1}^{N} (\omega^T x_n)^2} \sqrt{\sum_{n=1}^{N} (T \times_1 (y_n^{(1)})^T \ldots \times_k (y_n^{(k)})^T)^2}} \tag{6.46}$$

It has been proved in 6.1.2 that (6.46)is equivalent to:

$$\begin{aligned} \underset{\omega, T_{vec}}{\text{argmax}} \quad & \omega^T X Y_{vec}^T T_{vec} \\ s.t. \quad & \omega^T X X^T \omega = 1 \\ & T_{vec}^T Y_{vec} Y_{vec}^T T_{vec} = 1 \end{aligned} \tag{6.47}$$

where:

$$X = [x_1, x_2, \ldots, x_N], T_{vec} = vec(T)$$
$$Y_{vec} = [vec(y_1^{(1)} \circ \ldots \circ y_1^{(k)}), \ldots, vec(y_N^{(1)} \circ \ldots \circ y_N^{(k)})]$$

Here we suppose $X$ and $Y_{vec}$ are centered.

This optimization leads to the eigen problems:

$$[XX^T]^+ X Y_{vec}^T [Y_{vec} Y_{vec}^T]^+ Y_{vec} X^T \omega = \lambda^2 \omega \tag{6.48}$$

$$[Y_{vec} Y_{vec}^T]^+ Y_{vec} X^T [XX^T]^+ X Y_{vec}^T T_{vec} = \lambda^2 T_{vec} \tag{6.49}$$

After $r$ pairs of $(\omega_i, T_{vec-i})$ have been found, the relationship between an image and its underlying factors can be approximated by:

$$x = \sum_{i=1}^{r} \omega_i \theta_i T_{vec-i}^T vec(y^{(1)} \circ y^{(2)} \circ \ldots \circ y^{(k)}) \tag{6.50}$$

where $\theta_i = \frac{\omega_i^T X Y_{vec}^T T_{vec-i}}{T_{vec-i}^T Y_{vec} Y_{vec}^T T_{vec-i}}$. Similar to HOSVD-based methods, (6.50) can be used for finding the implied underlying factors or to synthesize new image.

Compared to HOSVD based methods, HOCCA releases the constraint on training data; however, solving the eigenproblems in (6.48) and (6.49) is non-trivial. When the dimensionality of $X$ is high, it would need a huge amount of memory to compute $[XX^T]^+$ and HOCCA becomes impractical.

### 6.2.2 Modified HOCCA

#### The Geometry of HOCCA

To gain insight into the geometry of the method it is advantageous to contemplate (6.48) and (6.49) with respect to the spaces spanned by the rows of matrices $X$ and $Y_{vec}$. To illustrate the notation used, let $A \in R^{m\times n}$ be an arbitrary matrix, then $L\{A^T\} = \{\, A^T\alpha \big| \, \alpha \in R^m\}$ will be referred to as the row-space of $A$ [31]. Define $a = X^T\omega$ and $b = Y_{vec}^T T_{vec}$. Apparently, $a \in L\{X^T\}$, $b \in L\{Y_{vec}^T\}$. Equation (6.46) equates to the cosine of the angle between the variates $a$ and $b$. Maximizing this cosine can be interpreted as minimizing the angle between $a$ and $b$, which in turn is equivalent to minimizing the distance between variates of unit length [43]:

$$\begin{aligned} \underset{a\in L\{X^T\}, b\in L\{Y_{vec}^T\}}{\operatorname{argmin}} \quad & \|a-b\| \\ s.t. \quad & \|a\| = \|b\| = 1 \end{aligned} \tag{6.51}$$

In SSS cases, the dimensionality of data $I_{image}$ is larger than the number of samples $N$, which implies $L\{X^T\}$ is almost the whole space $R^N$. If we also assume $I_1 I_2 \dots I_k < N$ which is usually true, $L\{Y_{vec}^T\}$ will be almost the whole space too. So the intersection of $L\{X^T\}$ and $L\{Y_{vec}^T\}$ is $(I_1 I_2 \dots I_k - 1)$-dimensional[1] and $r = I_1 I_2 \dots I_k - 1$ pairs of $(a_i = b_i)$ can be found:

$$\omega_i^T X = a_i^T = b_i^T = T_{vec-i}^T Y_{vec}, \quad i = 1, \dots, r \tag{6.52}$$

Which gives:

$$X \approx D_{mat} Y_{vec} \tag{6.53}$$

[1]Because $X$ and $Y_{vec}$ are centered, they are orthogonal to $[1, 1, \dots, 1]$

where $D_{mat} \in R^{I_{image} \times I_1 \dots I_k}$ is defined as

$$D_{mat} = [\omega_1, \dots, \omega_r] \begin{bmatrix} T_{vec-1}^T \\ \vdots \\ T_{vec-r}^T \end{bmatrix} \tag{6.54}$$

Note that, $(\omega_i, T_{vec-i})$ here have different scaling factors to the ones in (6.50). However their directions are the same.

**Compute** $D_{mat}$

Equation (6.53) suggests that the correspondence between $x$ and $y^{(i)}$ can be approximated by a $I_{image} \times I_1 \dots I_k$ matrix $D_{mat}$. Under SSS condition, the rank of $D_{mat}$ is usually $I_1 I_2 \dots I_k - 1$ and if $X$ and $Y_{vec}$ are not centered, in general $D_{mat}$ is full rank, *i.e.*, rank $I_1 I_2 \dots I_k$.

Computing $D_{mat}$ from eigenproblem (6.48), (6.49) is impractical, since normally $XX^T$ is very large. Therefore, we approximate $D_{mat}$ from (6.53) as:

$$\underset{D_{mat} \in R^{I_{image} \times I_1 \dots I_k}}{\operatorname{argmin}} \|X - D_{mat} Y_{vec}\| \tag{6.55}$$

It's notable the optimization in (6.55) is not exactly the same as the one of HOCCA, in fact, it's more like the intercross of PCA and HOCCA: PCA considers only the variation of $X$ regardless of label information $Y_{vec}$, whereas HOCCA maximizes correlation between $X$ and $Y_{vec}$ but ignores the magnitude; (6.55) takes both label information and variation into consideration and trade-off in a least squared error sense.

Another good property of (6.55) is that it can be decomposed into $I_{image}$ independent sub-problems:

$$\underset{\overline{d_j} \in R^{1 \times I_1 \dots I_k}}{\operatorname{argmin}} \left\| \overline{x_j}^T - Y_{vec}^T \overline{d_j}^T \right\|, \quad j = 1, \dots, I_{image} \tag{6.56}$$

where $\overline{x_j}$ and $\overline{d_j}$ are rows of $X$ and $D_{mat}$ , respectively.

Analytically, (6.56) leads to:

$$\overline{d_j}^T = [Y_{vec} Y_{vec}^T]^+ Y_{vec} \overline{x_j}^T, \quad j = 1, \dots, I_{image} \tag{6.57}$$

Or equivalently:

$$D_{mat}^T = [Y_{vec}Y_{vec}^T]^+ Y_{vec}X^T \tag{6.58}$$

Equations (6.57) and (6.58) preserve an attractive property of HOCCA: they're capable of handling arbitrary label information as training data, and by making use of the SSS condition, $D_{mat}$ can be approximated without solving eigen problems. Furthermore using (6.57), $D_{mat}$ can be calculated row by row, which implies memory usage is no longer a problem.

### 6.2.3 A Simple Multifactor Analysis Method

#### Multilinear case

RSC is a more special case compared to SSS, in which $D_{mat}$ can be directly constructed from training data instead of solving (6.58). We call this method Simple Multifactor Analysis(SMA). In SMA each state $e_j^{(i)}$ is represented by natural basis $u_j^{(i)*}$ which is a vector that has a 1 at the $j$th position and 0 elsewhere. Recall that in HOSVD-based method $e_j^{(i)}$ is represented by the row vector of $U^{(i)}$.

By the first condition of RSC, all training images are in pure states which means their underlying factors $y^{(i)}$are natural bases thus $vec(y_n^{(1)} \circ \ldots \circ y_n^{(k)}), n = 1, 2, \ldots N$ are natural bases too. And by the second condition, there should be $I_1 I_2 \ldots I_k$ images each of which has a unique combination of states, hence unique $vec(y_n^{(1)} \circ \ldots \circ y_n^{(k)})$. This implies one can re-order the training set to satisfy:

$$Y_{vec} = I^{I_1 I_2 \ldots I_k} \tag{6.59}$$

where $I^{I_1 I_2 \ldots I_k}$ is a $I_1 I_2 \ldots I_k \times I_1 I_2 \ldots I_k$ identity matrix. From (6.58) and (6.59), one obtains:

$$D_{mat} = X \tag{6.60}$$

Define a $(k+1)$th order tensor $D^* \in R^{I_1 \times I_2 \times \ldots \times I_k \times I_{image}}$ such that:

$$D^*_{(k+1)} = D_{mat} \tag{6.61}$$

It can be proved that a multilinear equation similar to (5.9) holds:

$$Z^{j_1 j_2 \ldots j_k} = D^* \times_1 (u_{j_1}^{(1)*})^T \times_2 (u_{j_2}^{(2)*})^T \ldots \times_k (u_{j_k}^{(k)*})^T \tag{6.62}$$

where $Z^{j_1 j_2 \dots j_k}$ as defined in 5.2.2.

Now let's show the relationship between (5.9) and (6.62) more clearly. It can be verified that $Z_{(k+1)} = X$, where $Z$ as defined in 5.2.2. From (6.60) and (6.61) one can obtain:

$$D^* = Z \tag{6.63}$$

Replacing $Z$ in (5.10) gives:

$$D^* = D \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_k U^{(k)} \tag{6.64}$$

By the definition of $u_{j_i}^{(i)}$ and $u_{j_i}^{(i)*}$ one can check the following is true:

$$u_{j_i}^{(i)} = (U^{(i)})^T u_{j_i}^{(i)*} \tag{6.65}$$

Putting (6.64) and (6.65) into (6.62), one obtains:

$$\begin{aligned} Z^{j_1 j_2 \dots j_k} \quad & = D \times_1 U^{(1)} \times_1 (u_{j_1}^{(1)*})^T \dots \times_k U^{(k)} \times_k (u_{j_k}^{(k)*})^T \\ & = D \times_1 [(u_{j_1}^{(1)*})^T U^{(1)}] \dots \times_k [(u_{j_k}^{(k)*})^T U^{(k)}] \\ & = D \times_1 (u_{j_1}^{(1)})^T \dots \times_k (u_{j_k}^{(k)})^T \end{aligned} \tag{6.66}$$

As illustrated in (6.64) and (6.66), (6.62) is just a reformulation of (5.9). They interpret the same intrinsic multilinear structure but apparently (6.62) is much simpler because $D^*$ can be constructed directly whereas in (5.9) a series of SVD is needed. In most applications, for example the ones in [95] [53], both methods will work, so apparently SMA is favorable.

To perform the "translation" task, one can use an iteratively updating algorithm similar to Alg.4. Alternatively, a vectorization-based algorithm can be used [71]. According to [71], for any $k$th order Tensor and $k$ column vectors $y^{(1)}, y^{(2)} \dots y^{(k)}$, the following holds:

$$T \times_1 y^{(1)T} \dots \times_k y^{(k)T} = \left\langle T, y^{(1)} \circ \dots \circ y^{(k)} \right\rangle = vec(T) \cdot vec(y^{(1)} \circ \dots \circ y^{(k)}) \tag{6.67}$$

From (6.67), the aim of multilinear "translation" is equivalent to: given an image $z^{new} \in R^{I_{image} \times 1}$, finding $u^{(i)}$ that satisfy

$$z^{new} = D_{mat} \cdot vec(u^{(1)} \circ \dots \circ u^{(k)}) \tag{6.68}$$

Equation (6.68) can be approximated by a two step minimization: first solving $u$ from:

$$z^{new} = D_{mat}u \tag{6.69}$$

Then finding $u^{(i)}$ that minimize:

$$\left\| u - vec(u^{(1)} \circ \ldots \circ u^{(k)}) \right\| \tag{6.70}$$

by Rank-1 approximation problem [47].

For multilinear cases the computational complexity of the HOSVD based method and SMA don't significantly differ because (5.8) only needs to be solved once. However, we'll show that for nonlinear cases, Kernel SMA is markedly simpler than Kernel HOSVD.

**Nonlinear Case**

SMA can be used for nonlinear cases by employing Kernel approach. Nonlinear mapping $\Phi : R^{I_{image}} \rightarrow F$ and its corresponding Kernel function $K(x, y)$ are defined as in 5.2.2. Denotes $\Phi(X) = [\Phi(z^{1\ldots1}), \ldots, \Phi(z^{I_1 \ldots I_k})]$ , similar to (6.60)-(6.61), define $W$ and $W_{mat}$ according to:

$$W_{(k+1)} = W_{mat} = \Phi(X) \tag{6.71}$$

It can be proved that:

$$\Phi(Z^{j_1 j_2 \cdots j_k}) = W \times_1 (u_{j_1}^{(1)*})^T \ldots \times_k (u_{j_k}^{(k)*})^T \tag{6.72}$$

In which $\Phi(Z^{j_1 j_2 \cdots j_k})$ is a tensor containing $\Phi(z^{j_1 j_2 \cdots j_k})$.

For a new image $\Phi(Z^{new})$, Kernel-based "translation" aims to determine its implied $u^{(i)}$ that satisfy:

$$\Phi(Z^{new}) = W \times_1 u^{(1)} \times_2 u^{(2)} \ldots \times_k u^{(k)} \tag{6.73}$$

To make it computable, mode-$(k + 1)$ multiply (6.73) by $\Phi(X)^T$ , *i.e.*, (6.73)$\times_{k+1}\Phi(X)^T$:

$$\Phi(Z^{new}) \times_{k+1} \Phi(X)^T = W \times_1 u^{(1)} \ldots \times_k u^{(k)} \times_{k+1} \Phi(X)^T \tag{6.74}$$

The left hand side can be expressed in terms of $K(x, y)$ as:

$$\begin{aligned}
& (\Phi(Z^{new}) \times_{k+1} \Phi(X)^T)_{(k+1)} \\
= \; & \Phi(X)^T \Phi(Z^{new})_{(k+1)} = \Phi(X)^T \Phi(z^{new}) \\
= \; & \begin{bmatrix} \Phi(z^{1\dots1})^T \\ \vdots \\ \Phi(z^{I_1\dots I_k})^T \end{bmatrix} \Phi(z^{new}) \\
= \; & \begin{bmatrix} K(z^{1\dots1}, z^{new}) \\ \vdots \\ K(z^{I_1\dots I_k}, z^{new}) \end{bmatrix}
\end{aligned} \tag{6.75}$$

And the right hand side can be rewritten as:

$$(W \times_{k+1} \Phi(X)^T) \times_1 u^{(1)} \dots \times_k u^{(k)} = W_K \times_1 u^{(1)} \dots \times_k u^{(k)} \tag{6.76}$$

where

$$\begin{aligned}
(W_K)_{(k+1)} \quad & = (W \times_{k+1} \Phi(X)^T)_{(k+1)} = \Phi(X)^T W_{(k+1)} \\
& = \Phi(X)^T \Phi(X) = K
\end{aligned} \tag{6.77}$$

Equation (6.77) suggests $W_K$ is a reorganized version of Gram matrix $K$, which is defined by:

$$K = \begin{bmatrix} K(z^{1\dots1}, z^{1\dots1}) & \cdots & K(z^{I_1\dots I_k}, z^{1\dots1}) \\ \vdots & \ddots & \vdots \\ K(z^{1\dots1}, z^{I_1\dots I_k}) & \cdots & K(z^{I_1\dots I_k}, z^{I_1\dots I_k}) \end{bmatrix} \tag{6.78}$$

If we define $K^{new}$ according to:

$$K^{new}_{(k+1)} = \begin{bmatrix} K(z^{1\dots1}, z^{new}) \\ \vdots \\ K(z^{I_1\dots I_k}, z^{new}) \end{bmatrix} \tag{6.79}$$

Equation (6.74) can be written as:

$$K^{new} = W_K \times_1 u^{(1)} \dots \times_k u^{(k)} \tag{6.80}$$

All elements in (6.80) are computable, and $u^{(i)}$ can be estimated using an iteratively updating algorithm similar to Alg.4. As both $K^{new}$ and $W_K$ are constant, the computational complexity of Kernel SMA is the same as its multilinear counterpart compared to Kernel HOSVD, where $E^{(i)}$ and $F^{(i)}$ need to be recalculated in each iteration.

$u^{(i)}$ can also be calculated by a vectorization based method; the formulas are exactly the same as (6.67)–(6.70).

After $u^{(i)}$ are known, one can change some of them and synthesize a new image. For example, replace $u^{(1)}$ by $u^{(1)*}$:

$$\Phi(Z^{syn}) = W \times_1 u^{(1)*} \times_2 u^{(2)} \ldots \times_k u^{(k)} \tag{6.81}$$

From (6.67) and (6.71), this can be rewritten into a vector form as

$$\Phi(z^{syn}) = \Phi(X) \cdot vec(u^{(1)*} \circ u^{(2)} \circ \ldots \circ y^{(k)}) \tag{6.82}$$

The synthetic image $z^{syn}$ is now represented as a linear combination of the training samples in $F$. So $z^{syn}$ can be obtained by approximating the pre-image which is specified in [64].

### 6.2.4 Experiment

In order to compare the computational efficiency, we need to use large databases. Unfortunately, there are no available large facial expression databases. So the experiments in this section are conducted on other image databases, but the conclusion is available for FER too. All algorithms are implemented in not especially well-optimized Matlab code and run on a notebook with 1.7G centrino CPU, 1GB of RAM.

#### Image Synthesis Using Modified HOCCA

In the first experiment, Modified HOCCA(6.55) is used to learn a multilinear model for an ensemble of images and then synthesize new images.

We generated a corpus of images of different items, with different colors, and under different illumination conditions. There are four items (duck, dice, rocket, flower), hence the first underlying factor is represented by a four dimensional vector $y^I$; each item is painted using two RGB colors, so color factor $y^C$ is $2 \times 3 = 6$ dimensional; and illumination factor $y^L$ is five dimensional since we have five lights (upper right, upper left, down left, down right and frontal).

Unlike the test setting of HOSVD methods [95] [53] [89] [90] [91], we don't restrict underlying factors to be in pure states; they can be superpositions of multiple states. For example, $y^C = [2, 0, 0, 1, 1, 1]$ means the item is painted with dark red ([0.5, 0, 0]) and white ([1, 1, 1]). Similarly for illumination: in general the item is lit by multiple light sources each with different illuminance. We restrict $y^I$ to be pure state, merely because it is counterintuitive to define a new item as the combination of multiple objects; however, mathematically it's feasible to do so. 200 images are generated using *random* underlying factors. Compared to the setting in [95] [90] [91] [53] [92], ours is more realistic. Part of the data set is shown in Fig.6.2.

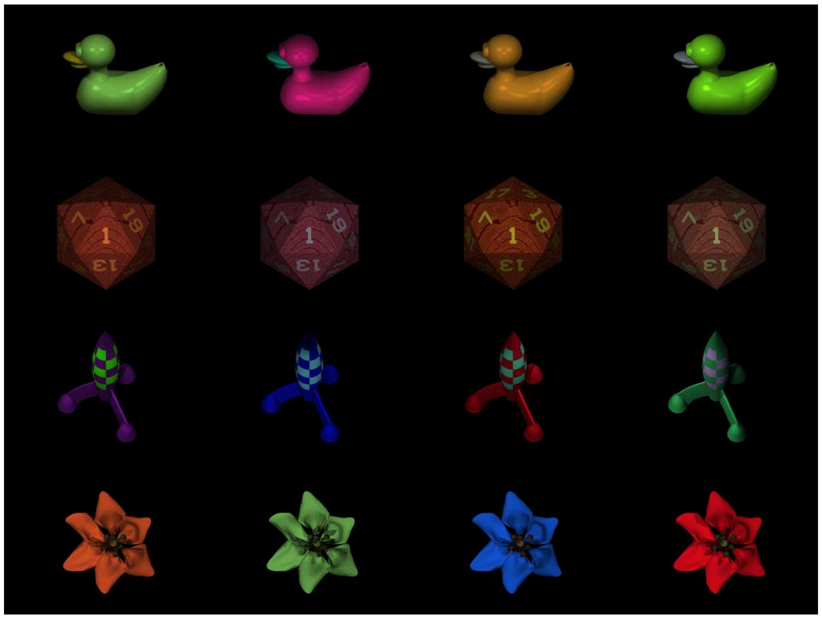


Figure 6.2: Part of the training data

The training data here doesn't satisfy RSC, so HOSVD-based methods are not applicable here. Because the size of the image is $320 \times 240$, $I_{image} = 320 \times 240 \times 3 = 230400$, a HOCCA-based method will need too much memory ( $I^2_{image}$ ), so it is virtually impossible too. Because $I_{image} > N$, we can use the modified HOCCA (6.57) to learn a multilinear model that satisfies:

$$X = T \times_1 y^I \times_2 y^C \times_3 y^L \tag{6.83}$$

Equation (6.83) can be used to synthesize an image from arbitrary underlying factors. Fig.6.3 illustrates a red duck with green bill and a blue-red rocket, both are illuminated from the front.

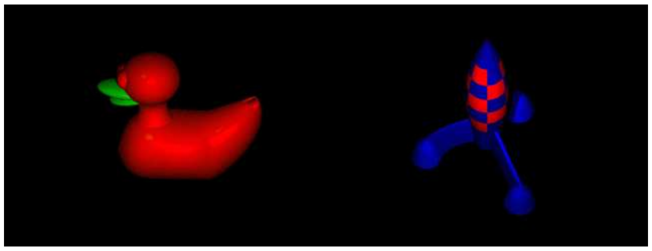


Figure 6.3: Synthetic toy images. Left: red duck with green bill; Right: Blue rocket painted with red pattern

Fig.6.4 and Fig.6.5 are dedicated to showing the influence of $y^L$: the same dice is lit from different directions. Four figures in Fig.6.4 are lit using the first 4 lights, respectively. Fig.6.5 illustrates the results under multiple light sources: the left one is lit using 2 strong light sources, one from upper right, one from down left; the right figure is lit by a strong light from upper left and a weak frontal one.

It takes about 30 seconds to train the model and only 1.5 seconds to synthesize each image.

### Image Analysis Using Kernel SMA

In this part we re-implement the experiment in [53] using Kernel SMA and show that SMA can achieve a similar result with much less complexity. Our experiments were conducted with sixty-two individuals from the CMU PIE database [79]. 25 images were selected for each individual, with 5 different viewpoints and 5 different illuminations. In the pre-processing stage, images are translated into AAM coefficients and the remainder of this experiment works on these AAM coefficients.

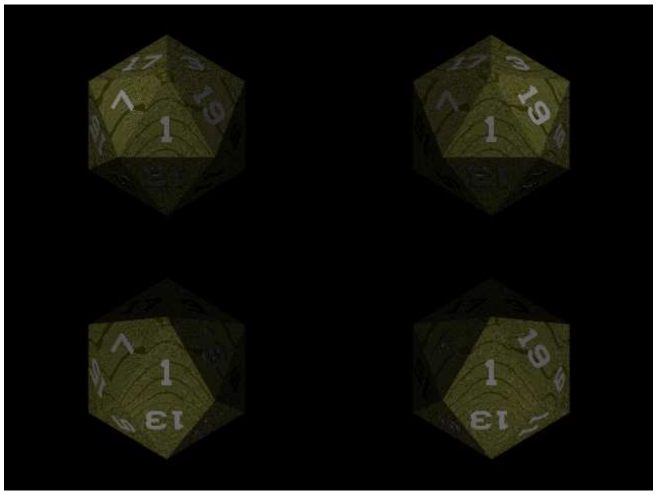


Figure 6.4: Dice under single light. The upper-left is lit from its nearest corner, as are the others respectively

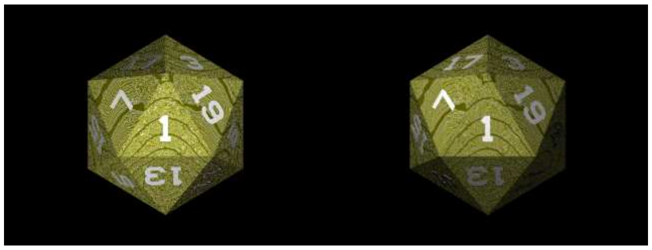


Figure 6.5: Dice under multiple lights

To make the comparison more meaningful, the Kernel HOSVD based method [53] and Kernel SMA use the same Gaussian Kernel, and an iteratively updating algorithm is used in both methods, also the parameters are set to be the same.

**Image Synthesis** In this test, a Leave-One-Out method was adopted. Specifically, while one facial image was selected as the testing image, the other sixty-one people's facial images under the remaining four viewpoints and four illuminations were served as a training set, and twenty-four novel images were synthesized.

Figure 6.6 gives a comparison between the synthetic results generated with Kernel

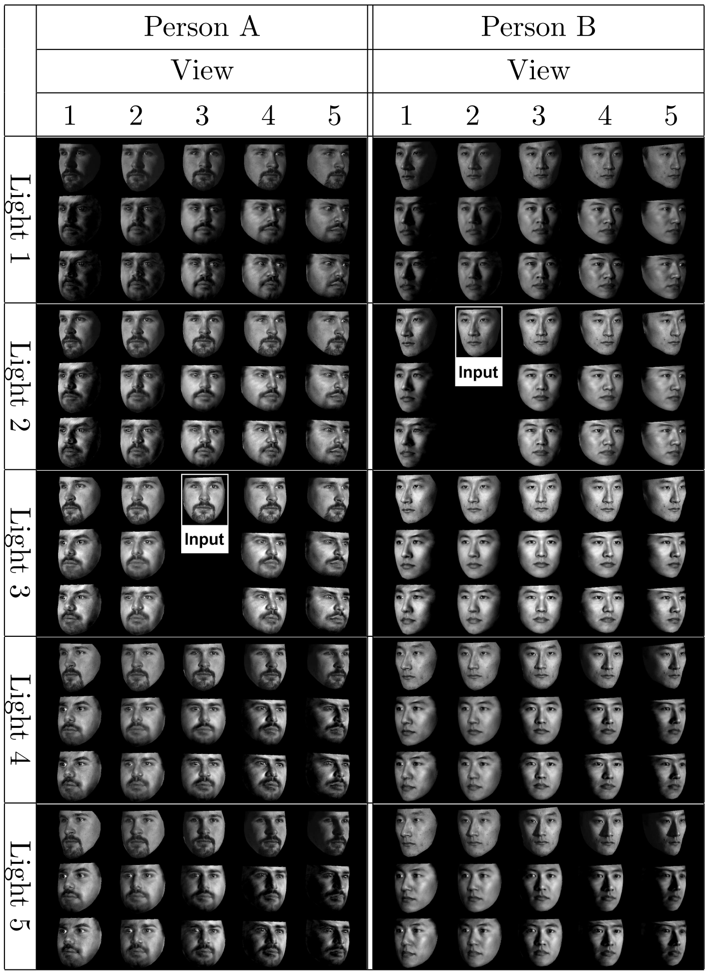


Figure 6.6: Comparison between the synthetic results of the Kernel SMA and Kernel HOSVD. For each view, the first line is the ground truth, the second line is given by Kernel HOSVD and the third line is given by Kernel SMA.

HOSVD and Kernel SMA. It can be seen that they are very similar. Both Kernel HOSVD and Kernel SMA converge well, and the local optimum problem reported in [53] doesn't occur in Kernel SMA, which is expected as it's also a Kernel method.

The first 2 rows of Table 6.5 illustrate the time costs. Because calculating (5.16) and (6.81) takes less than 0.1 second, only the time costs of implied factor estimation, namely the iteratively updating algorithms, are listed. As we can see, SMA is at least 20 times faster than HOSVD because it updates $u^{(i)}$ more efficiently.

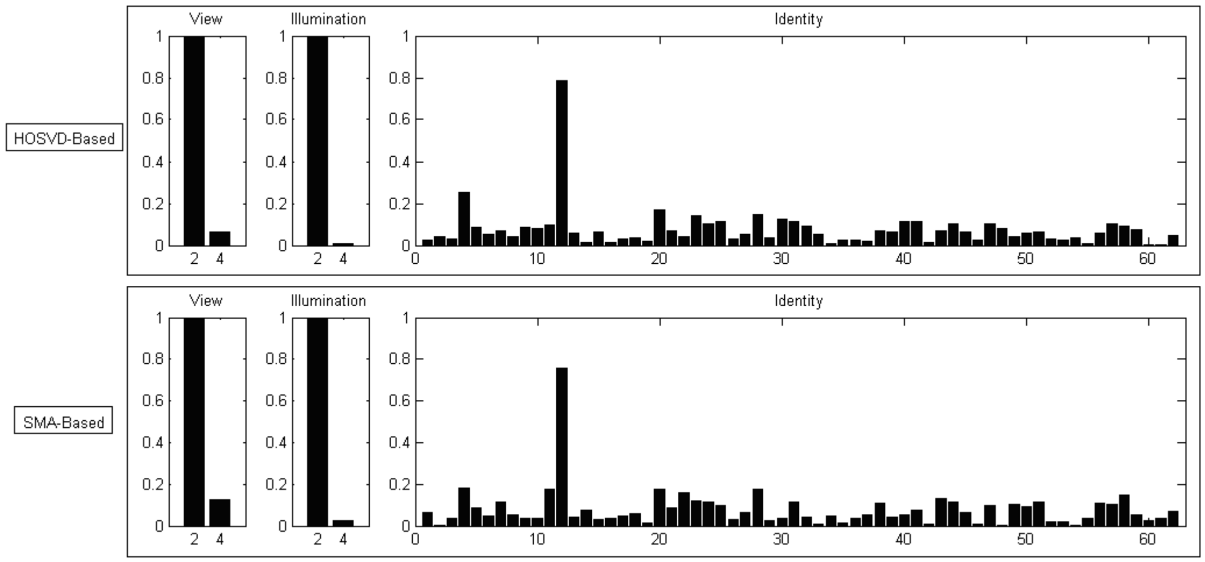
Table 6.5: Time cost of HOSVD based and SMA based method

| Task | Size of Training Set | Method | Time Cost(s) |
|---|---|---|---|
| Synthesis | $4 \times 4 \times 61 = 976$ | HOSVD | 380 |
| | | SMA | 15 |
| Recognition | $2 \times 2 \times 62 = 248$ | HOSVD | 30 |
| | | SMA | $< 1$ |

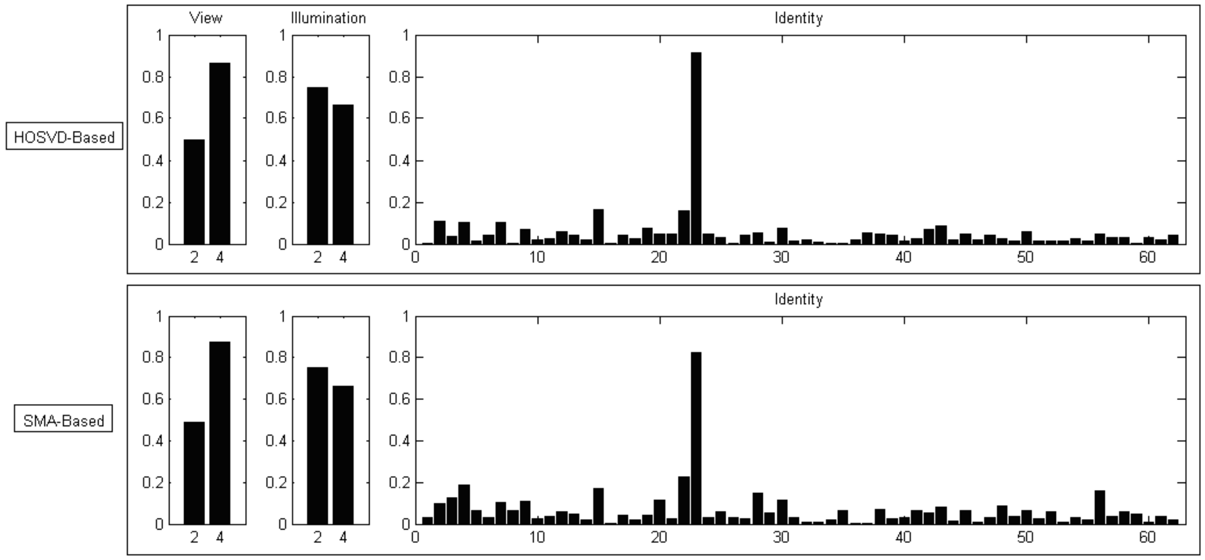
**Face Recognition** SMA and HOSVD are also compared in a face recognition task. For SMA, after identity factor $u^{per*}$ is estimated by "translation", recognition can be performed by choose the index $c^{per}$ which corresponds to the largest element of $u^{per*}$. For HOSVD, a "cosine similarity" method [95] is used which is equivalent to finding the largest element of $U^{per}u^{per}$ where $U^{per}$ is the subspace matrix of identity and $u^{per}$ is the person vector. In this test, the facial images produced by the second and fourth viewpoint with the second and fourth illumination served as the training set while the others were used for testing.

Typical results are shown in Figure 6.7. In part (a), the input is the 12th individual's image under 1st viewpoint and 1st illumination. Because this new viewpoint has not been seen by the computer, the algorithms try to "approximate" it by the combination of known viewpoints. As we can see, they did a decent job as they both realized the new viewpoint is similar to the 2nd viewpoint but different from the 4th one. Similarly for illumination factor. For the identity vector, the 11th element is large (about 0.8), while the others are fairly small. In part (b), the 23th person's image, produced by 3rd viewpoint and 3rd illumination is used.

As shown in Fig 6.7, the outputs of SMA and HOSVD are always very close, but not exactly the same. We've mentioned in 6.2.3 these 2 methods actually model the same intrinsic multilinear relationship; so why are the outputs different? Here's a short explanation: both methods try to find an equation in the form of (5.9), however because the real data is not perfectly linear, this becomes an optimization problem which is implicitly solved by an iteratively updating algorithm. "Implicitly" refers to the fact that the objective function is hidden in the updating formula rather than explicitly given. Because the



(a) 12th Person, 1st View, 1st Illumination



(b) 23th Person, 3rd View, 3rd Illumination

Figure 6.7: Implied underlying factors. All normalized to have unit norm

updating schemes of SMA and HOSVD based methods are not the same, they are virtually using different objective functions, thus giving slightly different results. It can be predicted that if the training data are exactly multilinear, SMA and HOSVD-based method would give the same results.

It's worth pointing out that a hidden optimization is hard to control, so in some situations a vectorization algorithm might be preferable, as $vec(u^{(1)} \circ \ldots \circ u^{(k)})$ can be found by explicitly minimizing (6.70).

Average time costs are shown in the last 2 rows of Table 6.5. As the size of the training data is smaller, both algorithms run faster, but the speed advantage of SMA is still significant.

### 6.2.5 Comment

In this section, we've discussed how to perform multifactor analysis more efficiently when the number of training data is small or the dimensionality of the image is large. We proposed a modified version of HOCCA which can be used under Small Sample Size condition. Then, in Regular Sample Condition, we further simplified it to what we called Simple Multifactor Analysis. The proposed algorithms were tested in image recognition and synthesis applications using both synthetic images and real data and delivered similar results as existing methods, but with much faster speed.

## 6.3 Chapter Summary

In this chapter we introduced three Higher Order Methods as summarized in Table 6.6.

Table 6.6: Different higher order methods

| Method | Sample Condition | Complexity |
|---|---|---|
| HOSVD | RSC | Highest |
| HOCCA | ALL | Higher |
| Modified HOCCA | SSS | Lower |
| SMA | RSC | Lowest |

For research use, since a lot databases satisfy the RSC condition, probably SMA is enough. But in real application; RSC is almost never satisfied; fortunately, in the vision area, because the dimensionality of the image is very high, SSS is usually true, so Modified HOCCA should be the best choice.

# Chapter 7

# Conclusion and Future Work

This work provided a framework for facial expression recognition that can effectively maximize information gathered about the emotion change and minimize the impact of person identity. Two main issues that are relevant to the design of FER systems were addressed. The first issue is the right combination of algorithms and features; the second issue is the design of classifier. Both are challenging problems, and significant research effort has been directed toward finding appropriate solutions for them.

## 7.1 Summary and Contributions

### 7.1.1 System Configuration

This work first decided the overall architecture of the system which consists of three modules: preprocessing, feature extraction and classifier. To find the appropriate algorithm for each module, various combinations of algorithms are implemented and compared. Our experimental results boil down to two configurations, which depend on the application where the FER system is used: The first configuration uses Gabor features and can achieve higher accuracy (95%), so it is suitable when the FER is implemented as a stand alone system which only performs recognition tasks; The second configuration is designed for those cases where the FER is integrated with other systems. AAM coefficient is used, which has a lower accuracy (90%) but is considered to be more versatile.

### 7.1.2 Classifier Design

The core of an FER system is its classifier; the conventional algorithms study the underlying factors independently. However, because in reality the influence of these factors is usually entangled, this "divide and conquer" strategy has some natural limitations. In this work, we proposed two new methods, namely, Higher Order Canonical Correlation Analysis and Simple Multifactor Analysis, to model this multi-variable relationship. By using this new proposed classifier in our system, we achieved recognition rates of 100% for identity and 95% for expression.

## 7.2 Future Work

Our current system is designed only for constrained condition and to be used in practice a lot of issues still need to be solved:

- A more sophisticated face detection algorithm is necessary. Because our system is only worked on well-defined databases, we choose a comparatively simple detection algorithm. To be used in real image or video, this component must be strengthened.
- Active Shape Model needs to be enhanced and retrained. Research in [56] [101] [17] has shown that the plain ASM had some limitations, and various improved versions have been proposed. To achieve a better robustness and accuracy, we should consider adopting some of these improvements. Another concern is that now our model is trained using a very small training set captured in some constrained conditions, to make it capable of handling general cases, more training data are necessary.
- Efficiency needs to be considered. Most face image processing applications are video-based: to perform real-time analysis both algorithm and code need to be optimized. And sometimes it is inevitable to trade off between accuracy and speed.
- Find other potential applications of HOCCA. Although our experiment is on an FER task, HOCCA itself is designed to be a general multifactor analysis algorithm. It can be used in modeling, recognition, synthesis and interpolation. So we can try this method in other application.

# Bibliography

[1] P. Aleksic and A. Katsaggelos. Automatic facial expression recognition using facial animation parameters and multistream hmms. *Information Forensics and Security, IEEE Transactions on*, 1:3–11, March 2006.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK's user's guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[3] R. Axelrod. *The Evolution of Cooperation*. Basic Books, September 1985.

[4] M. Bartlett, G. Littlewort, I. Fasel, and J. Movellan. Real time face detection and facial expression recognition: Development and application to human-computer interaction, 2003.

[5] M. S. Bartlett, P. A. Viola, T. J. Sejnowski, B. A. Golomb, J. Larsen, J. C. Hager, and P. Ekman. Classifying facial action. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 823–829. The MIT Press, 1996.

[6] J. Bassili. Emotion recognition: the role of facial movement and the relative importance of upper and lower areas of the face. *J Pers Soc Psychol*, 37(11):2049–58, 1979.

[7] M. Bichsel and A. P. Pentland. Human face recognition and the face image sets topology, 1994.

[8] F. L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(6):567–585, 1989.

[9] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM.

[10] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31,4:532–540, 1983.

[11] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[12] L.-F. Chen, H.-Y. M. Liao, M.-T. Ko, J.-C. Lin, and G.-J. Yu. A new lda-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33(10):1713–1726, 2000.

[13] S.-M. Choi and Y.-G. Kim. An affective user interface based on facial expression recognition and eye-gaze tracking. In Tao et al. [85], pages 907–914.

[14] F. B. Claude C. Chibelushi. Facial expression recognition: A brief tutorial overview, 2003. [Online; accessed 20-November-2007].

[15] A. Colmenarez, B. Frey, and T. S. Huang. A probabilistic framework for embedded face and facial expression recognition. pages 592–597.

[16] T. Cootes, C. Taylor, and A. Lanitis. Multi-resolution search with active shape models. pages A:610–612, 1994.

[17] 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.

[18] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Comput. Vis. Image Underst.*, 61(1):38–59, 1995.

[19] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learing Theory*, pages 35–46, 2000.

[20] F. Crow. Summed-area tables for texture mapping, 1984.

[21] J. Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision Research*, 20:847–856, 1980.

[22] R. De-Valois and K. De-Valois. *Spatial Vision.* Oxford Univ. Press, New York, 1990.

[23] I. Eibl-Eibesfeldt. Human ethology. pages 1–103, 1989. chaps 1 and 2.

[24] P. Ekman and W. Friesen. The facial action coding system: A technique for the measurement of facial movement, 1978.

[25] P. Ekman, E. T. Rolls, D. I. Perrett, and H. D. Ellis. Facial expressions of emotion: an old controversy and new findings. *Facial Expressions of Emotion: An Old Controversy and New Findings*, 335(1273):63–69, Jan. 1992.

[26] I. Fasel, M. Bartlett, and J. Movellan. A comparison of gabor filter methods for automatic detection of facial landmarks, 2002.

[27] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1997.

[28] W. Friesen and P. Ekman. Emfacs-7: emotional facial action coding system, 1983.

[29] H. Gu, G. Su, and C. Du. Feature points extraction from faces. *Proceedings of Image and Vision Computing New Zealand Conference*, pages 154–158, November 2003.

[30] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis an overview with application to learning methods. Technical Report CSD-TR-03-02, Royal Holloway University of London, 2003.

[31] D. A. Harville. *Matrix Algebra from a Statistician's Perspective.* Springer-Verlag, New York, 1997.

[32] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines, 2001.

[33] E. Huber. *Evolution of facial musculature and facial expression.* John Hopkins Press, Baltimore, 1931.

[34] S. Y. T. Ichimura, T.; Oeda. Construction of emotional space from facial expression by parallel sand glass type neural networks. *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, 3:2422–2427, 2002.

[35] A. K. Jain and S. Z. Li. *Handbook of Face Recognition.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[36] F. Jiao, S. Li, H.-Y. Shum, and D. Schuurmans. Face alignment using statistical models and wavelet features. *cvpr*, 01:321, 2003.

[37] J. Jones and L. Palmer. An evaluation of two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1233–1258, 1987.

[38] T. Kanade, J. Cohn, and Y. Tian. Comprehensive database for facial expression analysis, 2000.

[39] S.-W. Kim and R. P. W. Duin. On combining dissimilarity-based classifiers to solve the small sample size problem for appearance-based face recognition. In Z. Kobti and D. Wu, editors, *Canadian Conference on AI*, volume 4509 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 2007.

[40] T. Kim, S. Wong, and R. Cipolla. Tensor canonical correlation analysis for action classification. pages 1–8, 2007.

[41] S. Kimura and M. Yachida. Facial expression recognition and its degree estimation. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 295, Washington, DC, USA, 1997. IEEE Computer Society.

[42] H. Kobayashi, A. Tange, and F. Hara. Real time recognition of six basic facial expressions. *Proceedings of the IEICE General Conference*, 1995:256, 19950327.

[43] U. Kockelkorn. *Lineare Statistische Methoden*. Oldenbourg, München, 2000.

[44] I. Kotsia and I. Pitas. Facial expression recognition in image sequences using geometric deformation features and support vector machines. *IEEE Transactions on Image Processing*, 16(1):172–187, 2007.

[45] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):743–756, 1997.

[46] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[47] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

[48] J. M. Lattin, P. E. Green, and J. D. Carroll. *Analyzing Multivariate Data*. Duxbury Pr, Pacific Grove, CA, 2003.

[49] T. S. Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.

[50] S. Li, Z. Q. Zhang, H.-Y. Shum, and H. Zhang. Floatboost learning for classification, 2002.

[51] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection, 2004.

[52] S. Z. Li, L. Zhu, Z. Q. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection, 2002.

[53] Y. Li, Y. Du, and X. Lin. Kernel-based multifactor analysis for image synthesis and recognition. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 114–119, Washington, DC, USA, 2005. IEEE Computer Society.

[54] C. Liu and H. Wechsler. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition, 2002.

[55] W. Liu and Z. Wang. Facial expression recognition based on fusion of multiple gabor features. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 536–539, Washington, DC, USA, 2006. IEEE Computer Society.

[56] Y. Liu, Y. Li, L. Tao, and G. Xu. Multi-view face alignment guided by several facial feature points. In *ICIG '04: Proceedings of the Third International Conference on Image and Graphics (ICIG'04)*, pages 238–241, Washington, DC, USA, 2004. IEEE Computer Society.

[57] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *FG '98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, page 200, Washington, DC, USA, 1998. IEEE Computer Society.

[58] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets, 1998.

[59] M. J. Lyons, J. Budynek, and S. Akamatsu. Automatic classification of single facial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1357–1362, 1999.

[60] L. Ma and K. Khorasani. Facial expression recognition using constructive feedforward neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(3):1588–1595, 2004.

[61] K. Matsuno, C.-W. Lee, S. Kimura, and S. Tsuji. Automatic recognition of human facial expressions. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 352, Washington, DC, USA, 1995. IEEE Computer Society.

[62] A. Mehrabian. Communication without words. *Psychology Today*, 2(4):53–56, 1968.

[63] P. Michel and R. E. Kaliouby. Real time facial expression recognition in video using support vector machines. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 258–264, New York, NY, USA, 2003. ACM.

[64] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de–noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[65] J. R. Movellan. Tutorial on gabor filters, 2002.

[66] M. M. Nordstrøm, M. Larsen, J. Sierakowski, and M. B. Stegmann. The IMM face database - an annotated dataset of 240 face images. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, may 2004.

[67] I. A. G. C. U. of Technology. Active contour models active shape models (snakes smart snakes). [Online; accessed 20-November-2007].

[68] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection, 1997.

[69] M. Pantic and I. Patras. Dynamics of facial expression: Recognition of facial actions and their temporal segments from face profile image sequences. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 36(2):433–449, 2006.

[70] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification, 2000.

[71] Y. Ren. Private communication ii, 2007.

[72] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection, 1998.

[73] B. Sch, o Smola, and K. uller. Nonlinear component analysis as a kernel eigenvalue problem, 1998.

[74] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods, 1998.

[75] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars, 2000.

[76] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 792–799, New York, NY, USA, 2005. ACM.

[77] Y. Shinohara and N. Otsu. Facial expression recognition using fisher weight maps. In *FGR*, pages 499–504. IEEE Computer Society, 2004.

[78] Y. ShuiCheng and Q. Cheng. Multi-view face alignment using direct appearance models. In *FGR '02: Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, page 324, Washington, DC, USA, 2002. IEEE Computer Society.

[79] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression database, 2003.

[80] P. Y. Simard, L. Bottou, P. Haffner, , and Y. L. Cun. Boxlets: a fast convolution algorithm for signal processing and neural networks, 1998.

[81] S. M. Smith and J. M. Brady. SUSAN – A new approach to low level image processing. Technical Report TR95SMS1c, Chertsey, Surrey, UK, 1995.

[82] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection, 1998.

[83] M. Suwa, N. Sugie, and K. Fujimora. A preliminary note on pattern recognition of human emotional expression, 1978.

[84] D. Tao, X. Li, X. Wu, and S. Maybank. General tensor discriminant analysis and gabor features for gait recognition. *IEEE Trans. PAMI*, 29(10):1700–1715, October 2007.

[85] J. Tao, T. Tan, and R. W. Picard, editors. *Affective Computing and Intelligent Interaction, First International Conference, ACII 2005, Beijing, China, October 22-24, 2005, Proceedings*, volume 3784 of *Lecture Notes in Computer Science*. Springer, 2005.

[86] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

[87] M. A. Turk and A. P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[88] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[89] M. A. O. Vasilescu. Human motion signatures: Analysis, synthesis, recognition. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3*, page 30456, Washington, DC, USA, 2002. IEEE Computer Society.

[90] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *ECCV (1)*, pages 447–460, 2002.

[91] M. A. O. Vasilescu and D. Terzopoulos. Tensortextures: Multilinear image-based rendering, 2004.

[92] M. A. O. Vasilescu and D. Terzopoulos. Multilinear independent components analysis. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 547–553, Washington, DC, USA, 2005. IEEE Computer Society.

[93] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features, 2001.

[94] P. Viola and M. Jones. Robust real time object detection, 2001.

[95] H. Wang and N. Ahuja. Facial expression decomposition, 2003.

[96] J. M. Wang, D. J. Fleet, and A. Hertzmann. Multifactor gaussian process models for style-content separation. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 975–982, New York, NY, USA, 2007. ACM Press.

[97] J. Weston and C. Watkins. Multi-class support vector machines, 1998.

[98] Wikipedia. Delaunay triangulation — wikipedia, the free encyclopedia, 2007. [Online; accessed 28-November-2007].

[99] Wikipedia. Support vector machine — wikipedia, the free encyclopedia, 2007. [Online; accessed 20-November-2007].

[100] Y.-K. Wu and S.-H. Lai. Facial expression recognition based on supervised lle analysis of optical flow and ratio image. *International Computer Symposium*, Nov 2006.

[101] S. Xin and H. Ai. Face alignment under various poses and expressions. In Tao et al. [85], pages 40–47.

[102] Q. Xu, P. Zhang, W. Pei, L. Yang, and Z. He. A facial expression recognition approach based on confusion-crossed support vector machine tree. In *IIH-MSP*, pages 309–312. IEEE Computer Society, 2006.

[103] L. Yin and X. Wei. Multi-scale primal feature based facial expression modeling and identification. In *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 603–608, Washington, DC, USA, 2006. IEEE Computer Society.

[104] J. Yu and B. Bhanu. Evolutionary feature synthesis for facial expression recognition. *Pattern Recogn. Lett.*, 27(11):1289–1298, 2006.

[105] X. Zhang and Y. Jia. A linear discriminant analysis framework based on random subspace for face recognition. *Pattern Recogn.*, 40(9):2585–2591, 2007.

[106] Z. Zhang, M. Lyons, M. Schuster, and S. Akamatsu. Comparison between geometry-based and gaborwavelets -based facial expression recognition using multi-layer perceptron.

[107] W. Zheng, X. Zhou, C. Zou, and L. Zhao. Facial expression recognition using kernel canonical correlation analysis (kcca). *Neural Networks, IEEE Transactions on*, 17(1):233–238, Jan. 2006.