

Image-based Pattern Recognition Principles

Ke-Jie Liao

E-mail: cajetp@yahoo.com.tw

Graduate Institute of Communication Engineering

National Taiwan University, Taipei, Taiwan, ROC

Aug. 2008

Abstract

The objective of pattern recognition is to assign an object to one of several predefined categories. The methods to solve the problems in pattern recognition field can be roughly divided into two parts. One is the decision-theoretic method dealing with the pattern which is represented as quantitative values. The other is structural method dealing with the pattern constructed by using its structural relations rather than numerical values. In this paper, we discuss four basic models for pattern recognition, that is, template matching, statistical, neural networks, and syntactic approach. Then we introduce eigenspace-based face recognition in the final section.

1 Introduction

Human can easily recognize things based on the past learning experiences. For examples, recognizing a people's nation from what kinds of languages he speaks, recognizing a friend by looking his face, recognizing a specific area in a map, etc. After computer are created and has been widely used all over the world, we human seek to find a way that best mimics human's recognition system for computers. Applications related to pattern recognition has been developed such as face recognition, handwriting recognition, iris recognition, language recognition, etc. Even though different applications are designed for different purposes, the main block diagram of different applications is the same. Fig.1 shows a typical diagram of a pattern recognition system.

The block diagram shown in Fig.1 can be viewed as two parts. One is classification phase which is in the left hand side of Fig.1. The other is training phase shown in the right hand side of Fig.1. The goal of training phase is to learn the structure of classifier via training data. Usually we can improve the system performance by refining classifier from different training data.

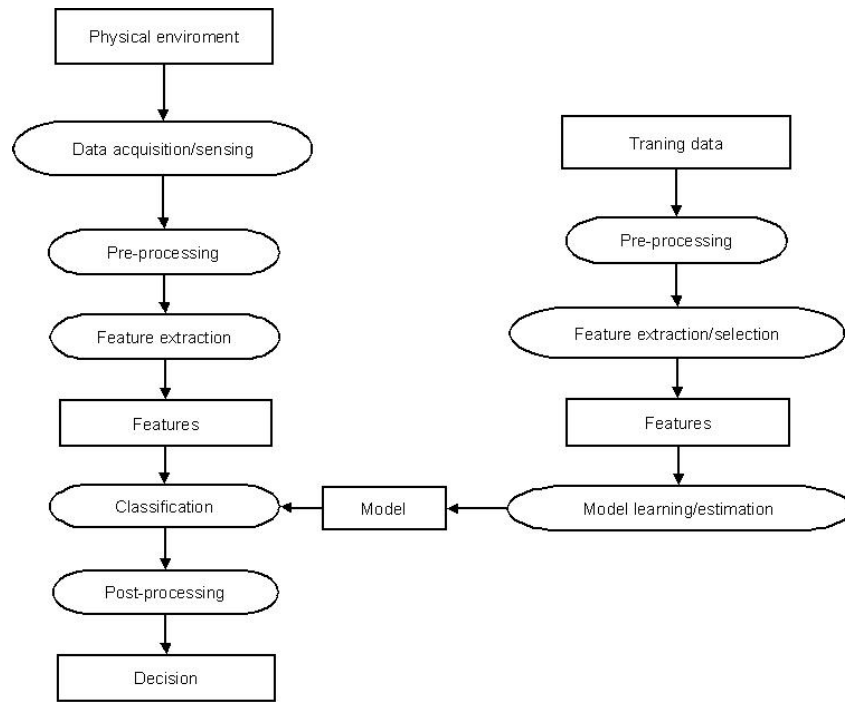


Fig.1 Process diagram of a pattern recognition system

After the classifier is defined, we can assign an unknown pattern to one of several predefined categories, which is the objective of classification phase.

In this paper, we briefly introduce four kinds of recognition models, which is shown in Table 1. The four models can roughly divided into two parts based on the difference between input's properties. One is the decision-theoretic method dealing with the pattern which is represented as quantitative values. The other is structural method dealing with the pattern constructed by using its structural relations rather than numerical values. Only the syntactic model employs structural method. Besides the four models, we also present a briefly introduction of face recognition in the section 6, which is one of the most user-friendly application among all pattern recognition applications.

Table 1: Pattern recognition models

Approach	Representation	Recognition function	Typical criterion
Template matching	Samples,pixels,curves	Correlation,distance measure	Classification error
Statistical	Features	Discriminant function	Classification Error
Neural network	Samples,pixels,features	Network function	Acceptance error
Syntactic or structure	Primitives	Rules,grammar	Mean square error

2 Two Dimensional Matched Filter

Matched filter is widely used for degrading the noise effect on our desired pattern and comparing the similarity of two objects. In the context of image processing, it is used for template matching to find the best location between reference image and template, or finding the correlation measure between two images. Given reference image $I(m, n)$ and template $H(m, n)$, where m and n are the row and column indices respectively, the basic form of 2-dimensional matched filter output is

$$Y(m, n) = I(m, n) * H^*(-m, -n) \quad (2-1)$$

, where $*$ represents discrete two dimensional convolution. Because Eq. (2-1) without normalization process results in the nondistinctive phenomenon, the normalized discrete 2D matched filter is required. It has the form

$$Y(m, n) = \frac{\sum_{m1} \sum_{n1} I(m+m1, n+n1) H^*(m1, n1)}{\sqrt{\sum_{m1} \sum_{n1} |H(m1, n1)|^2} \sqrt{\sum_{m1} \sum_{n1} |I(m+m1, n+n1)|^2}} \quad (2-2)$$

.Eq. (2-2) can also be viewed as 2D normalized cross correlation function between $I(m, n)$ and $H(m, n)$.

There is an example for template matching between binary image I and binary template image H . We want to find the location of the panda's left eye shown in Fig. 2-2 from the panda shown in Fig. 2-1. The distinctive effect shown in Fig. 2-3. The lightest point in Fig. 2-4 which is the peak value of Y reveals the location of the panda's eye. It also confirms the presence of Fig. 2-2 in Fig. 2-1.



Fig. 2-1 Input image I



Fig. 2-2 Template image H



Fig. 2-3 Output without normalization Fig.2-4 Output with normalization

The drawbacks of template matching are (1) the poor discriminative ability on template shape, (2) an enormous number of templates must often be test matched against an image field to account for changes in rotation and magnification of template objects. For this reason, template matching is usually limited to smaller local features, which are more invariant to size and shape variations of an object.

3 Image Registration

Image registration is a common issue in image processing applications because images should be aligned correctly so that making systems have better performance. It is necessary to spatially register the images, and thereby, to correct for relative translation shifts, rotational differences, scale differences and even perspective view difference. However, we need to detect the misregistration parameters before further alignment process between two images. In the rest of this section, we only focus on how to detect the misregistration parameters rather than alignment process.

Among all kinds of misregistration such as rotational, and scale differences, translational differences is the central problem because the methods employed in detecting translational parameters can be extended to other problems. The classical technique for detecting translational parameters is normalized correlation function which is identical to normalized 2D matched filter. The peak value of normalized correlation function's output is the translational offset coordinates between the input image pairs.

Instead of finding the translational parameters via spatial domain, we can indirectly obtain the parameters via frequency domain which is called *Phase Correlation Method*. Given two images $F_1(x, y)$, and $F_2(x, y) = F_1(x-x_0, y-y_0)$, where (x_0, y_0) denotes an offset with respect to one another, the Fourier transform of the images are related by

$$\mathcal{F}_2(w_x, w_y) = \mathcal{F}_1(w_x, w_y) \exp\{-i(w_x x_0 + w_y y_0)\} \quad (3-1)$$

The exponential phase shift factor can be computed by the *cross-power spectrum* of the two images given by

$$\mathcal{G}(w_x, w_y) \equiv \frac{\mathcal{F}_1(w_x, w_y)\mathcal{F}_2^*(w_x, w_y)}{|\mathcal{F}_1(w_x, w_y)\mathcal{F}_2(w_x, w_y)|} = \exp\{i(w_x x_0 + w_y y_0)\} \quad (3-2)$$

Taking the inverse Fourier transform of Eq.3-2 yields the spatial offset

$$G(x, y) = \delta(x - x_0, y - y_0) \quad (3-3)$$

in the space domain. Therefore, we can easily obtain offset parameters from Eq.3-3.

The phase correlation method for translational misregistration detection can be extended to scale and rotation misregistration detection. Consider a pair of images in which a second image is translated by an offset and rotated by an angle with respect to the first image. Then

$$F_2(x, y) = F_1(x \cos \theta_0 + y \sin \theta_0 - x_0, -x \sin \theta_0 + y \cos \theta_0 - y_0) \quad (3-4)$$

Taking Fourier transform of both sides of Eq.3-4, one obtains the relationship

$$\mathcal{F}_2(w_x, w_y) = \mathcal{F}_1(w_x \cos \theta_0 + w_y \sin \theta_0, -w_x \sin \theta_0 + w_y \cos \theta_0) \exp\{-i(w_x x_0 + w_y y_0)\} \quad (3-5)$$

The rotation component can be isolated by taking magnitudes $\mathcal{M}_1(w_x, w_y)$ and $\mathcal{M}_2(w_x, w_y)$ of both sides of Eq.3-1. By representing the frequency variables in polar form,

$$\mathcal{M}_2(\rho, \theta) = \mathcal{M}_1(\rho, \theta - \theta_0) \quad (3-6)$$

the phase correlation method can be used to determine the rotation angle θ_0 .

If a second image is a size-scaled version of a first image with scale factors (a, b) such that $F_2(x, y) = F_1(ax, by)$, then its Fourier transform is

$$F_2(w_x, w_y) = \frac{1}{|ab|} F_1\left(\frac{w_x}{a}, \frac{w_y}{b}\right) \quad (3-7)$$

By converting the frequency variables to a logarithmic scale, scaling can be converted to a translational movement. Then

$$\mathcal{F}_2(\log w_x, \log w_y) = \frac{1}{|ab|} \mathcal{F}_1(\log w_x - \log a, \log w_y - \log b) \quad (3-8)$$

The phase correlation method can be applied to determine the unknown scale factors (a, b).

4 Decision-Theoretic Methods

Decision-theoretic approaches to recognition are based on the use of decision functions, $d_i(\mathbf{x})$, $1 \leq i \leq W$, where W is the number of pattern classes and $\mathbf{x} = (x_1, x_2, \dots, x_n)$ represents an n -dimensional pattern vector which each component in \mathbf{x} represents different feature measurement. The pattern \mathbf{x} is classified to w_k if $d_k(\mathbf{x})$ generates the largest value among all $d_i(\mathbf{x})$.

The decision boundary $d_{ij}(\mathbf{x})$ separating class w_i from w_j is given by values of \mathbf{x} for which $d_i(\mathbf{x}) = d_j(\mathbf{x})$ or, equivalently, by values of \mathbf{x} for which

$$d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0. \quad (4-1)$$

We can simply classify \mathbf{x} to w_i by checking $\text{sign}(d_{ij}(\mathbf{x}))$ if all the boundary exists.

4.1 Bayes Statistical Classifiers

Because of the randomness under which pattern classes normally are generated, a probabilistic approach to recognition is important. Bayes statistical classifier which is optimal in the sense that it minimizes the total average loss in misclassification provides a probabilistic model for recognition. It assigns an unknown pattern \mathbf{x} to a class w_i if

$$\sum_{k=1}^W L_{ki} p(\mathbf{x} / w_k) P(w_k) < \sum_{q=1}^W L_{qi} p(\mathbf{x} / w_q) P(w_q) \quad (4.1-1)$$

for all $j ; j \neq i$, where L_{ij} is a misclassification loss function represented the pattern classifier decides that \mathbf{x} came from w_j , when it actually came from w_i , $p(\mathbf{x}/w_i)$ denotes the probability density function that a particular pattern x comes from class w_i , and $P(w_i)$ is the probability of occurrence of class w_i . Given the misclassification loss function L_{ij} is symmetrical function, Bayes classifier assigns an unknown pattern \mathbf{x} to a class w_i if

$$p(x / w_i)P(w_i) > p(x / w_j)P(w_j) \quad (4.1-2)$$

for $j = 1, 2, \dots, W; j \neq i$. Eq. 4.1-2 is the posterior probability decision rule. Also, the decision function for symmetrical loss function is of the form

$$d_j(x) = p(x / w_j)P(w_j) = P(w_j / x) \quad (4.1-3)$$

where a pattern vector \mathbf{x} is assigned to the class whose decision function yields the largest numerical value. However, the probability density functions of the patterns in each class, as well as the probability of occurrence of each class, must be known. The latter requirement usually is not a problem. For instance, if all classes are equally likely to occur, then $P(w_j) = 1/M$. Even if this condition is not true, these probabilities generally can be inferred from knowledge of the problem. Estimation of the probability density functions $p(\mathbf{x}/w_j)$ is another matter. If the pattern vectors, \mathbf{x} , are n dimensional, then $p(\mathbf{x}/w_j)$ is a function of n variables, which, if its form is not known, requires methods from multivariate probability theory for its estimation. These methods are difficult to apply in practice, especially if the number of representative patterns from each class is not large or if the underlying form of the probability density functions is not well behaved. For these reasons, use of the Bayes classifier generally is based on the assumption of an analytic expression for the various density functions and then an estimation of the necessary parameters from sample patterns from each class. By far the most prevalent form assumed for $p(\mathbf{x}/w_j)$ is the Gaussian probability density function. The closer this assumption is to reality, the closer the Bayes classifier approaches the minimum average loss in classification.

4.2 Neural Networks

The ideas of neural networks stem from the operation of human neural networks. It uses a multitude of elemental nonlinear computing elements (called neurons) organized as networks reminiscent of the way in which neurons are believed to be interconnected in the brain. In its most basic form, the perceptron learns a linear decision function that dichotomizes two linearly separable training sets. Figure 4.2-1 shows schematically the perceptron model for two pattern classes, where w_i is undetermined coefficient. By using appropriate training algorithms for different cases, w_i can be obtained for each case respectively. In other words, the structure of neural networks is more flexible than Bayes classifier.

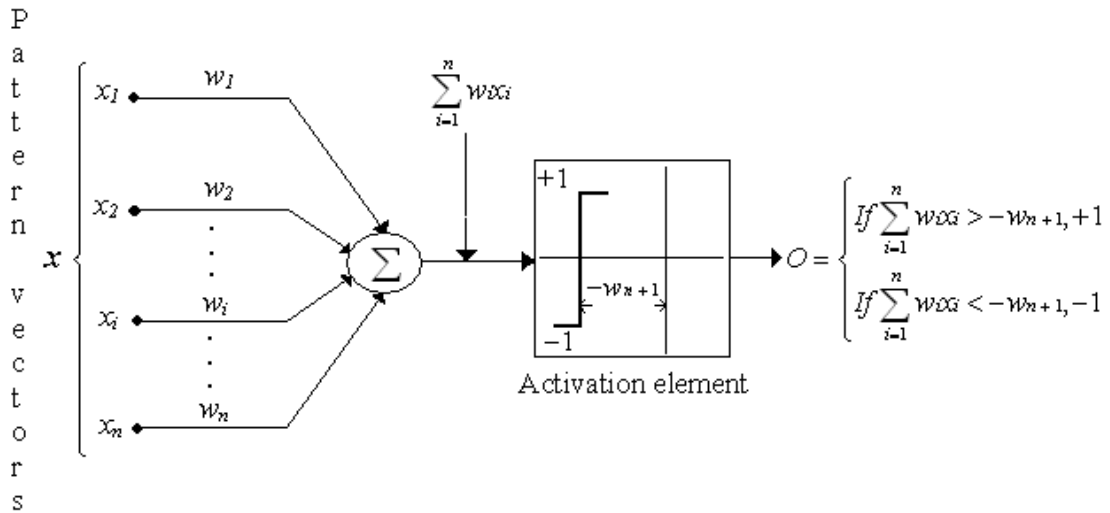


Fig.4.2-1 The two pattern classes model for neural networks

4.2.1 Multilayer Feedforward Neural Networks

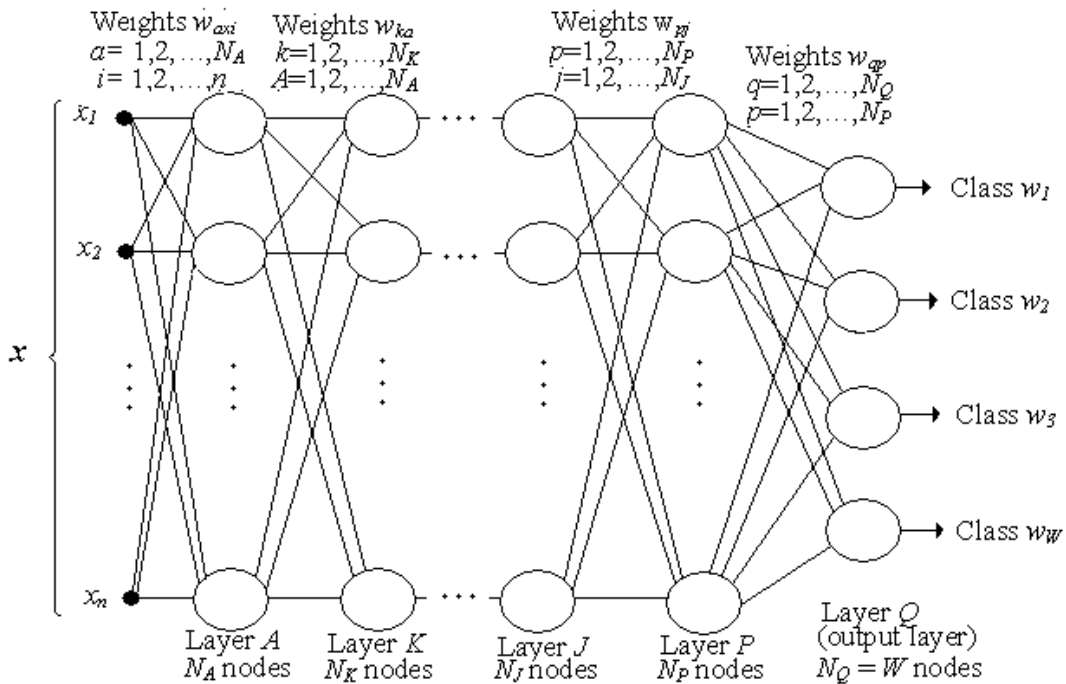


Fig.4.2.1-1 The basic structure of multilayer feedforward neural networks

The basic structure of multilayer feedforward neural networks is shown in Fig.4.2.1-1. It recognizes a pattern vector x as belonging to class w_i , if the i th output of the network is "high" while all other outputs are "low". Each neuron has the same form as the perceptron model shown in Fig.4.2-1 except the activation function is differentiable function called sigmoid function. The sigmoid activation function has

the following form with layer K denote the layer preceding layer J , I_j is the input to the layer J , O_k is the output of layer K , and w_{jk} denote the coefficients in the layer J ,

$$h_j(I_j) = \frac{I}{1 + e^{-\left(\sum_{k=1}^{N_k} w_{jk} O_k + \theta_j\right) / \theta_0}} \quad (4.2.1-1)$$

where

$$I_j = \sum_{k=1}^{N_k} w_{jk} O_k \quad (4.2.1-2)$$

,and

$$O_k = h_k(I_k) \quad (4.2.1-3)$$

In order to train the multilayer feedforward neural networks, we must solve the main problem in training a multilayer network, that is , adjusting the weights in the so called hidden layers. That is, in those other than the output layer. Therefore, we first adjust the coefficient of the output layer because the desired output of each node is known and the total coefficients can be obtained by a back propagation way. The training steps for multilayer feedforward neural network are summarized as follows.

1. Initialization

Assigning an arbitrary set of weights throughout the network (not equally).

2. Iterative step

- a. Computing O_j for each node by using training vector, then generating the error terms for output δ_q , where $\delta_q = (r_q - O_q)h_q'(I_q)$, r_q is the desired response.
- b. Backward passing appropriate error signal is passed to each node and the corresponding weight changes are made.

In a successful training session, the network error decreases with the number of iterations and the procedure converges to a stable set of weights that exhibit only small fluctuations with additional training. After the system has been trained, it classifies patterns using the parameters established during the training phase. In normal operation, all feedback paths are disconnected. Then any input pattern is allowed to propagate through the various layers, and the pattern is classified as belonging to the class of the output node that was high, while all the others were low.

The complexity of decision surfaces implemented by multilayer networks is

shown in Fig. 4.2.1-2. As in the third row of Fig.4.2.1-2 presents, the complexity of decision regions implemented by a three-layer network is, in principle, arbitrary. In practice, a serious difficulty usually arises in structuring the second layer to respond correctly to the various combinations associated with particular classes. The reason is that patterns of the same class may occur on both sides of lines in the pattern space.


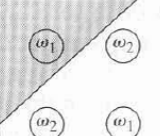
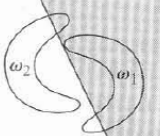
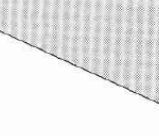
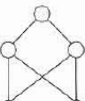
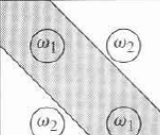
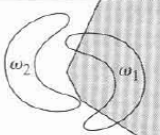
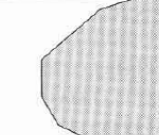
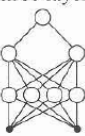
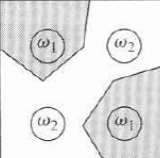
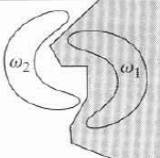
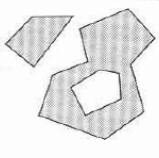
Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer 	Single hyperplane			
Two layers 	Open or closed convex regions			
Three layers 	Arbitrary (complexity limited by the number of nodes)			

Fig. 4.2.1-2 Decision surfaces complexity implemented by multilayer networks

5 Structural Methods

Decision-theoretic methods deal with patterns quantitatively and largely ignore any structural relationships inherent in a pattern's shape. In the contrast, structural methods uses the representatives of structural relationship as the input. In the following subsections, we introduce the syntactic recognition.

5.2 Syntactic Recognition

Syntactic recognition provides a unified methodology for handling structural recognition methods. The basic idea of syntactic recognition is similar to the language system of human. The recognizer of syntactic recognition called automaton receives $L(G)$ as input which can be viewed as some kinds of language and decides whether it is valid or not. This concept is shown in Fig. 5.2-1. We introduce two different types of syntactic recognition respectively in the following two subsections.

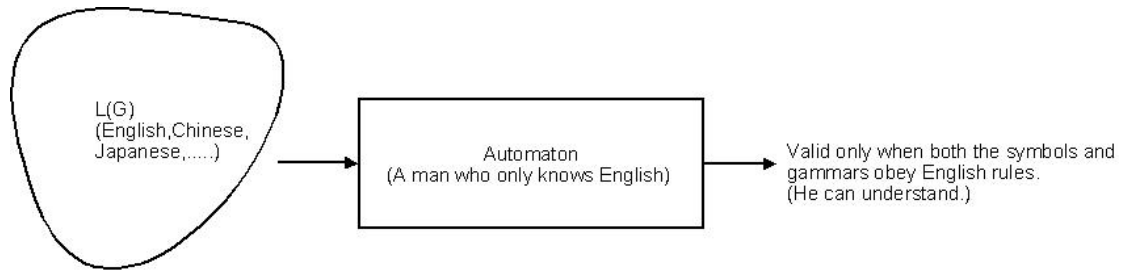


Fig. 5.2-1 Conceptual diagram of syntactic recognition

5.2.1 Syntactic Recognition for Strings

The input to the automata are unknown sentences, $L_i(G_i), 1 \leq i \leq W$, where W is the number of automaton (It can be viewed as the number of languages in use for the system). The grammar is defined as the 4-tuple

$$G = (N, \Sigma, P, S) \quad (5.2.1-1)$$

Where

- N is a finite set of variables called *nonterminals*,
- Σ is a finite set of constants called *terminals*,
- P is a set of rewriting rules called *productions*, and
- S in N is called the *starting symbol*.

It is required that N and Σ be disjoint sets.

Also, the *finite automaton* is defined as the 5-tuple

$$A_f = (Q, \Sigma, \delta, q_0, F) \quad (5.2.1-2)$$

Where

- Q is a finite, nonempty set of *states*,
- Σ is a finite input *alphabet*,
- δ is a *mapping* from $Q \times \Sigma$ into the collection of all subsets of Q ,
- q_0 is the *starting state*, and
- F is a set of *final states*.

The conversion between regular grammar and corresponding automaton states as follow.

Given $G = (N, \Sigma, P, S)$, where $X_0 \equiv S$, and suppose that N is composed of X_0 plus n additional nonterminals X_1, X_2, \dots, X_n . The set Q for the automaton is formed by introducing $n + 2$ states $\{q_0, q_1, \dots, q_n, q_{n+1}\}$ such that q_i corresponds to X_i for $0 \leq i \leq n$, and q_{n+1} is the final state. The set of input symbols is identical to the set of terminals in G . The mappings in δ are obtained by using the following two rules, for $\forall a$ in

Σ , and each i and j , with $0 \leq i \leq n$, $0 \leq j \leq n$,

1. If $X_i \rightarrow aX_j$ is in P , then $\delta(q_i, a)$ contains q_j .
2. If $X_i \rightarrow a$ is in P , then $\delta(q_i, a)$ contains q_{n+1} .

For the case of known grammars in advance, we can learn automata directly from the corresponding grammar by using the above algorithm. Conversely, if the grammars are not known in advance, the syntactic recognition required specification of the appropriate automata for each class under consideration. Therefore, It is necessary to learn the automata from sample patterns (such as strings or trees).

Suppose that all patterns of a class are generated by an unknown grammar G and that a finite sets of samples R^+ with the property

$$R^+ \subseteq \{v | v \text{ in } L(G)\} \quad (5.2.1-3)$$

is available. For a positive integer k , we define the k tail of z with respect to R^+ as the set $h(z, R^+, k)$, where

$$h(z, R^+, k) = \{w | zw \text{ in } R^+, |w| \leq k\}. \quad (5.2.1-4)$$

A procedure for learning an automaton $A_f(R^+, k) = (Q, \Sigma, \delta, q_0, F)$ consists of letting

$$Q = \{q | q = h(z, R^+, k) \text{ for } z \text{ in } \Sigma^*\} \quad (5.2.1-5)$$

and, for each a in Σ ,

$$\delta(q, a) = \{q' \text{ in } Q | q' = h(za, R^+, k), \text{ with } q = h(z, R^+, k)\}. \quad (5.2.1-6)$$

In addition, we let

$$q_0 = h(\lambda, R^+, k) \quad (5.2.1-7)$$

and

$$F = \{q | q \text{ in } Q, \lambda \text{ in } q\} \quad (5.2.1-8)$$

where λ is the empty string. Therefore, we can obtain $A_f(R^+, k)$ from the sample

patterns and a particular value of k .

There are a relations between k and automaton's performance , that is , the value of k controls the nature of the resulting automaton. Given the R^+ is fixed , the following properties exemplify the dependence of $A_f(R^+, k)$ on k .

Property 1. $R^+ \subseteq L[A_f(R^+, k)]$ for all $k \geq 0$,where $L[A_f(R^+, k)]$ is the language accepted by $A_f(R^+, k)$.

Property 2. $L[A_f(R^+, k)] = R^+$ if k is equal to ,or greater than, the length of the longest string in R^+ ; $L[A_f(R^+, k)] = \Sigma^*$ if $k=0$.

Property 3. $L[A_f(R^+, k+1)] \subseteq L[A_f(R^+, k)]$.

This three properties is graphically shown in Fig. 5.2.1-2.

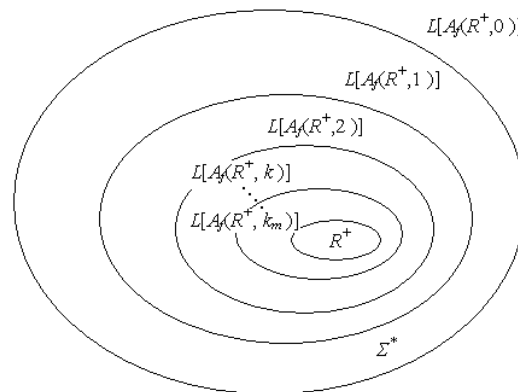


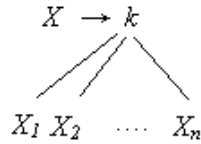
Fig. 5.2.1-2 Graphic relation between k and $L[A_f(R^+, k+1)]$

5.2.2 Syntactic Recognition for trees

A tree grammar is defined as the 5-tuple

$$G = (N, \Sigma, P, r, S) \tag{5.2.2-1}$$

where, as before, N and Σ are sets of nonterminals and terminals, respectively; S , contained in N , is the start symbol, which in general can be a tree; P is a set of productions of the form $T_i \rightarrow T_j$, where T_i and T_j are trees ;and r is a *ranking function* that denotes the number of direct descendants (offspring) of a node whose label is a terminal in the grammar. Of particular relevance to our discussion are *expansive* tree grammars having productions of the form



where X_1, X_2, \dots, X_n are nonterminals and k is a terminal.

A tree automata begin simultaneously at each node on the frontier (the leaves taken in order from left to right) of an input tree and proceed along parallel paths toward the root. Specifically, a frontier-to-root automaton is defined as

$$A_t = (Q, F, \{f_k \mid k \in \Sigma\}) \tag{5.2.2-2}$$

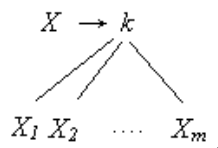
where

Q is a finite set of states,

F , a subset of Q , is a set of final states, and

f_k is a relation in $Q^m \times Q$ such that m is a rank of k .

For an expansive tree grammar, $G = (N, \Sigma, P, r, S)$, we construct the corresponding tree automaton by letting $Q = N$, with $F = \{S\}$ and, for each symbol a in Σ , defining a relation f_k such that $(X_1, X_2, \dots, X_m, X)$ is in f_k if and only if there is in G a production



6 Face Recognition

Face recognition is one of the most user-friendly techniques among all pattern recognition applications such as iris, fingerprint, and retina, etc. The reason is evidence that we human recognize others by his face rather than his iris, fingerprint, or retina. As a result, face recognition has been developed to many applications. Table 2 shows some applications of face recognition.

Table 2 : Some applications of face recognition

Areas	Specific Applications
Biometrics	Immigration, National ID, Passports, Voter Registration
	Drivers' Licenses, Entitlement Programs
	Welfare Fraud
Information Security	Desktop Logon(Windows NT,Windows95)
	Application Security, Database Security, File Encryption
	Internet Security, Internet Access, Medical Records
	Secure Trading Terminals
Law Enforcement And Surveillance	Advanced Video Surveillance, CCTV Control
	Portal Control, Post-Event Analysis
	Shoplifting and Security Tracking and Investigation
Smart Cards	Stored Value Security, User Authentication
Access Control	Facility Access, Vehicular Access

However, the weakness of face recognition should be noted. Either iris, fingerprint, or retina recognition method performs high recognition accuracy compared with face recognition. Also, face recognition has two unsolved fatal problems, that is, illumination and pose varied problems. Consequently, we can only perform good face recognition system under some constrained circumstance.

Before considering the actual design problems of a face recognition system, realizing a face recognition of human deeply may give us a direction to solve the problems. We list the nineteen results that reveals the characteristics of human as below.

Recognition as a function of available spatial resolution

Result 1: Humans can recognize familiar faces in very low-resolution images.

Result 2: The ability to tolerate degradations increases with familiarity.

Result 3: High-frequency information by itself is insufficient for good face recognition performance.

The nature of processing: Piecemeal versus holistic

Result 4: Facial features are processed holistically.

Result 5: Of the different facial features, eyebrows are among the most important for recognition.

Result 6: The important configural relationships appear to be independent across the width and height dimensions.

The nature of cues used: Pigmentation, shape and motion

Result 7: Face-shape appears to be encoded in a slightly caricatured manner.

Result 8: Prolonged face viewing can lead to highlevel aftereffects, which suggest prototype-based encoding.

Result 9: Pigmentation cues are at least as important as shape cues.

Result 10: Color cues play a significant role, especially when shape cues are degraded.

Result 11: Contrast polarity inversion dramatically impairs recognition performance, possibly due to compromised ability to use pigmentation cues.

Result 12: Illumination changes influence generalization.

Result 13: View-generalization appears to be mediated by temporal association.

Result 14: Motion of faces appears to facilitate subsequent recognition.

Developmental progression

Result 15: The visual system starts with a rudimentary preference for face-like patterns.

Result 16: The visual system progresses from a piecemeal to a holistic strategy over the first several years of life.

Neural underpinnings

Result 17: The human visual system appears to devote specialized neural resources for face perception.

Result 18: Latency of responses to faces in inferotemporal (IT) cortex is about 120 ms, suggesting a largely feedforward computation.

Result 19: Facial identity and expression might be processed by separate systems.

A general statement of the face recognition can be formulated as follows: Given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces. Additional information such as race, age, gender, facial expression and speech may be used in narrowing the search to get a higher

recognition performance. The problem involves in detecting faces from a cluttered scenes, extracting our desired features from the face regions, identification or verification. In identification problems, an unknown input face can be labeled as one of the existing face in database, whereas in verification problems, the recognition system reports back if the input face is in stored database.

In the following subsections, we focus our attention on still image-based face recognition, and a holistic approach to solve the problems. We introduce eigenspace-based approach to face recognition.

6.1 Eigenspace-based approach

Face recognition is a high-dimensional pattern recognition problem. Even low-resolution face images generate huge dimensional feature spaces (20 000 dimensions in the case of a 100×200 pixels face image). In addition to the problems of large computational complexity and memory storage, this high dimensionality makes very difficult to obtain statistical models of the input space using well-defined parametric models. Moreover, this last aspect is further stressed given the fact that only few samples for each class (1–3) are normally available for the system training. However, the intrinsic dimensionality of the face space is much lower than the dimensionality of the image space, since faces are similar in appearance and contain significant statistical regularities. This fact is the starting point of the use of eigenspace-based methods for reducing the dimensionality of the input face space.

Standard Eigenspace-based face recognition

Standard Eigenspace-based face recognition is a PCA(Principal Component Analysis) method. The scheme is based on an information theory approach that decomposes face images into a small set of characteristic feature images called “eigenfaces.” These eigenfaces are just the eigenvectors of the distribution of face images within the entire image space. It consists a set of complete orthogonal basis. In the language of information theory, the method extract the relevant information in a face image, encode it as efficiently as possible, and compare one face encoding with a database of models encoded similarly. Recognition is performed by projecting new images to the subspace spanned by the eigenfaces and then classifying the face by comparing its position in face space with the positions of known individuals, which is also represented by projecting to eigenspace.

Let training set of intensity face images be $I_1, I_2, I_3, \dots, I_M$, its corresponding vector form denoted as $\Gamma_1, \Gamma_2, \Gamma_3, \dots, \Gamma_M$ with each length is N . The average face of

the set is defined by

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad (6.2-1)$$

Each face differs from the average by the vector $\Phi_i = \Gamma_i - \Psi$. An example of training set is shown in Fig. 6.2-1, with the average face shown in Fig.6.2-2.

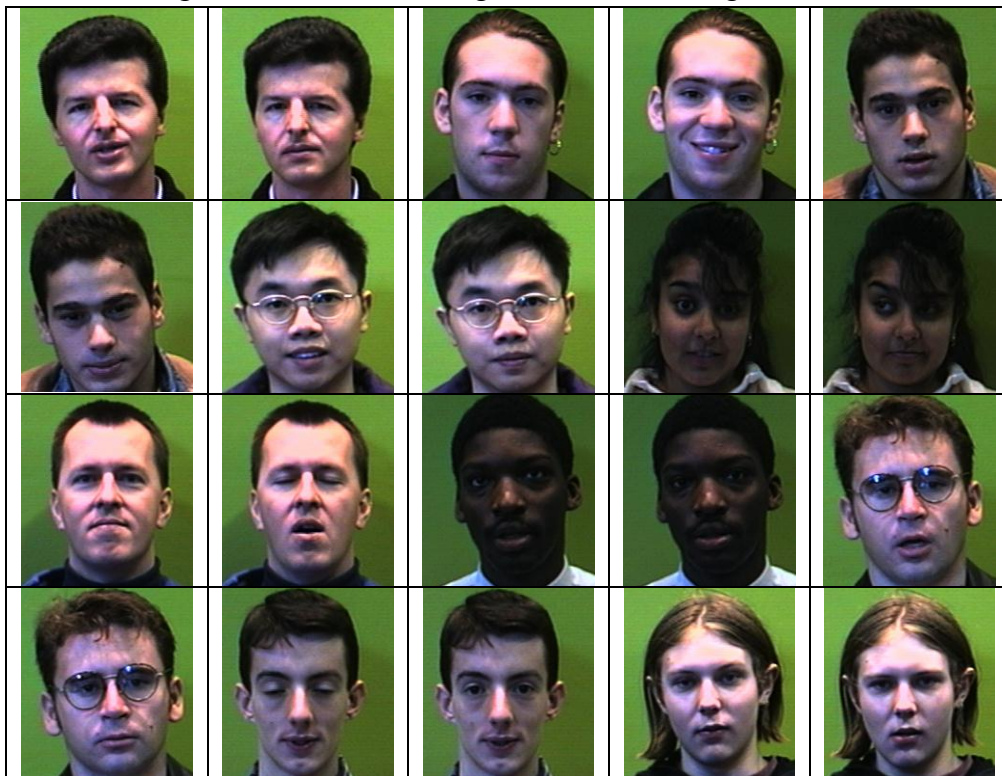


Fig. 6.2-1 A training set of face



Fig.6.2-2 Mean face

We seek a set of M orthonormal vectors, \mathbf{u}_n , which best describes the distribution of the data. The k th vector, \mathbf{u}_k , is chosen such that

$$\lambda_k = \frac{1}{M} \sum_{n=1}^M (\mathbf{u}_k^T \Phi_n)^2 \quad (6.2-2)$$

is a maximum. The vectors \mathbf{u} and scalars λ are the eigenvectors and eigenvalues, respectively, of the covariance matrix

$$C = \frac{1}{M} \sum_{n=1}^M \Phi_n \Phi_n^T = AA^T \quad (6.2-3)$$

where the matrix $A = [\Phi_1, \Phi_2, \dots, \Phi_M]$.

However, the matrix C is N^2 by N^2 , and determining the N^2 eigenvectors and eigenvalues is an intractable task for typical image sizes. Therefore, we use an indirect method to obtain the eigenvectors and eigenvalues. Consider the eigenvectors \mathbf{v}_i of $A^T A$ such that

$$A^T A \mathbf{v}_i = \mu_i \mathbf{v}_i \quad (6.2-4)$$

Premultiplying both sides by A , we have

$$AA^T A \mathbf{v}_i = C(A \mathbf{v}_i) = \mu_i (A \mathbf{v}_i) \quad (6.2-5)$$

from which we see that $A \mathbf{v}_i$ are the eigenvectors of C , and μ_i are the eigenvalues of C . Following this analysis, we can simplify the eigenvalue computation from the order of the number of pixels in the images (N^2) to the order of the number of images in the training set (M). In practice, training set of face images will be relatively small ($M \ll N^2$), and the calculations become quite manageable. Fig.6.2-3 shows the corresponding eigenfaces of the training set shown in Fig.6.2-1.

A new face image (Γ) is transformed into its eigenface components (projected into “eigenspace”) by a simple inner product,

$$w_k = \mathbf{u}_k^T (\Gamma - \Psi) \quad (6.2-6)$$

for $k = 1, \dots, M'$, where M' is the number of most significant eigenvalue to sufficiently represent the set of face images. Selecting the number M' to improve the dimensional reduction is recommendable to apply a given criterion for neglect the components with small projection variance. If we just ignore a number of components, the mean square error of the given representation is the sum of the eigenvalues not used in the representation. Therefore a good criterion would be to choose only M' components, obtained by the *Normalized Residual Mean Square Error*

$$RMSE(M') = \frac{\sum_{k=M'+1}^N \lambda_k}{\sum_{k=1}^N \lambda_k} \quad (6.2-7)$$

Considering M' given by $RMSE(M') < 5\%$ will be good for standard applications.

The weights form a vector $\Omega^T = [w_1, w_2, \dots, w_{M'}]$ that describes the contribution of each eigenface in representing the input face image, treating the eigenfaces as a basis for face images. The vector mat then be used in a standard pattern recognition algorithm to find which of a number of predefined face classes, if any, best describes the face. The simplest method for

determining which face class provides the best description of an input image face is to find the face class k that minimizes the Euclidian distance. We need to compare two metrics to obtain the final decision. One is ε_k , where

$$\varepsilon_k^2 = \|(\Omega - \Omega_k)\|^2 \quad (6.2-8)$$

measures the distance between input and the prototype weight of class k .

The other is ε , where

$$\varepsilon^2 = \|(\Phi - \Phi_f)\|^2 \quad (6.2-9)$$

measures the distance between the image and the face space, where $\Phi = \Gamma - \Psi$ and

$$\Phi_f = \sum_{i=1}^{M'} w_i u_i.$$

Fig. 6.2-4 shows the geometric relationship between ε_k and ε , where $\Omega_k = [\Lambda_k]_p$, and $p = \{u_i\}$.

There are four possibilities for an input image and its pattern vector :

- (1) Near face space and near a face class.
- (2) Near face space but not near a known face class.
- (3) Distant from face space and near a face class.
- (4) Distant from face space and not near a known face class.

In the first case, an individual is recognized and identified. In the second case, an unknown individual is present. The last two cases indicate that the image is not a face images.

Standard eigenspace-based methods uses PCA as projection approach ,Euclidean distance as metric function. However, there are other kinds of eigenspace-based approach which mostly differ from projection methods or metric functions. We introduce a so-called FLD projection method in the rest of the section.

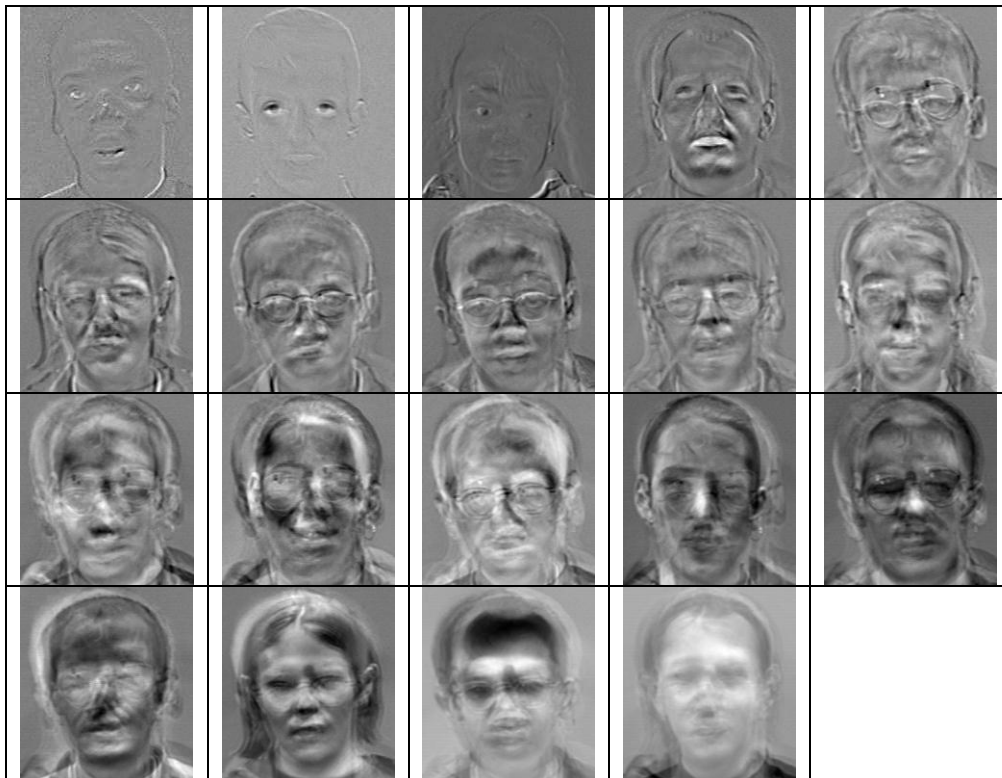


Fig. 6.2-3 eigenfaces

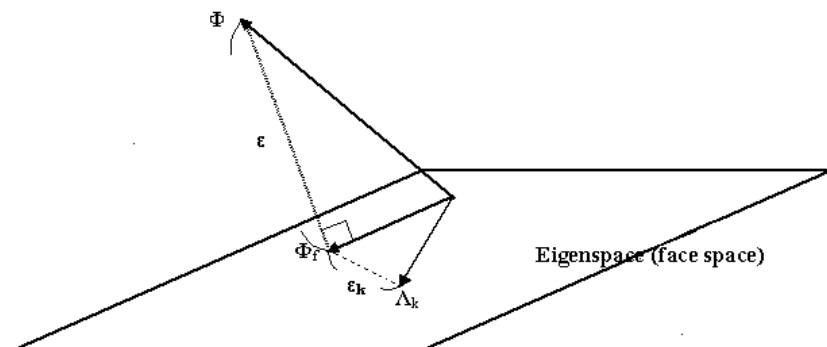


Fig.6.2-4 Geometric relationship between ϵ_k and ϵ .

FLD eigenspace-based approach

FLD(Fisher Linear Discriminant) is another projection method that searches for the projection axes on which the face images of different classes are far from each other (similar to PCA), and at the same time where the images of a same class are close from each other. The advantage of FLD against PCA is that the information kept in the dimensional reduction is better for recognition purposes. Fig.6.2-5 shows the comparison between PCA and FLD.

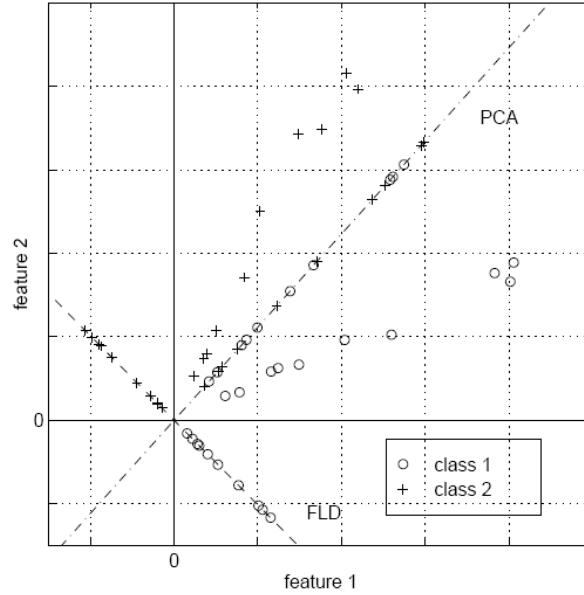


Fig. 6.2-5 Comparison between PCA and FLD

In order to define the mathematical structure under FLD, first we define the parameter $\Upsilon(\mathbf{u})$ to be maximized on the successive projection axes as:

$$\gamma(\mathbf{u}) = \frac{s_b(\mathbf{u})}{s_w(\mathbf{u})} \quad (6.2-10)$$

with \mathbf{u} represents any projection unitary vector in the image space, and $s_b(\mathbf{u})$ and $s_w(\mathbf{u})$ given by:

$$s_b(\mathbf{u}) = \sum_{i=1}^{NC} P(C_i) \{(m(i) - m)\mathbf{u}\}^2 \quad (6.2-11)$$

$$s_w(\mathbf{u}) = \sum_{i=1}^{NC} P(C_i) E[\{(x(i) - m(i))\mathbf{u}\}^2] \quad (6.2-12)$$

where \mathbf{m} is the global mean vector, $P(C_i)$ are the probabilities associated to each class C_i , NC is the number of class, $\mathbf{m}(i)$ are the average vectors of C_i , and $\mathbf{x}(i)$ are the vectors associated to C_i . $s_b(\mathbf{u})$ measures the separation between the individual class means respect to the global mean face, and $s_w(\mathbf{u})$ measures the separation between vectors of each class respect to their own class mean. Alternative we define the scatter matrices:

$$S_b = \sum_{i=1}^{NC} P(C_i) (\mathbf{m}(i) - \mathbf{m})(\mathbf{m}(i) - \mathbf{m})^T \quad (6.2-13)$$

$$S_w = \sum_{i=1}^{NC} P(C_i) E[(x(i) - m(i))(x(i) - m(i))^T] \quad (6.2-14)$$

then :

$$\gamma(u) = \frac{u^T S_b u}{u^T S_w u} \quad (6.2-15)$$

Therefore, the solution for our problem is the solution of the generalized eigensystem:

$$S_b u^k = \lambda_k S_w u^k \quad (6.2-16)$$

Then w^k would be the fisherfaces and λ_k are the successive γ parameters associated with each fisherface.

7 Conclusions

We have reviewed the main operation of four basic pattern recognition models and the principles of eigenspace-based face recognition approach. We concluded these four models' characteristics as follows, template matching is simple to implement but the template size must be small to decrease computational delay, statistical methods highly depends on the assumption of distribution, neural networks can adaptively refine the classifier and the decision surface in principle can be arbitrarily implemented, syntactic methods concerned structural sense to encode but additional process to define primitives are required. Different applications adopted different models, there was no prototype model adapted for all recognition problems.

In our future work, we would like to focus our attention on face recognition problems. Seeking to find a new way by using frequency domain rather than space domain, applying image compression method to face recognition, considering video-based face recognition problems, and adding color factor into face recognition problems where we only treated intensity face images in this paper.

References

- [1] J.T.Tou,R.C.Gonzalez,"Pattern Recognition Principles",Addison-Wesley Publishing Company,1974.
- [2] R.C.Gonzalez,R.E.Woods,"*Digital Image Processing(Second Edition)*", Prentice-Hall,Inc,2002,pp.693-750.
- [3] William K. Pratt,"*Digital Image Processing : PIKS Inside(Third Edition)*",

John Wiley & Sons, Inc, 2001, pp. 613-637.

- [4] M. Turk and A. Pentland, "Eigenfaces for Recognition", *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71-86, 1991.
- [5] Kah-Kay Sung and T. Poggio, "Example-Based Learning for View-Based Human Face Detection", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, pp. 39-51 January 1998.
- [6] W. Zhao, R. Chellappa, A. Rosenfeld, P.J. Phillips, "Face Recognition: A Literature Survey", *ACM Computing Surveys*, 2003, pp. 399-458.
- [7] J. Ruiz-del-Solar, P. Navarrete, "Eigenspace-Based Face Recognition: A Comparative Study of Different Approaches", *IEEE Trans. on Systems, Man and Cybernetics—Part C: Applications and Reviews*, Vol. 35, No. 3, pp. 315-325 August 2005
- [8] Pawan Sinha, Benjamin Balas, Yuri Ostrovsky, and Richard Russell, "Face Recognition by Humans: Nineteen Results All Computer Vision Researchers Should Know About", *Proceedings of the IEEE*, Vol. 94, No. 11, pp. 1948-1962 November 2006.