Face Recognition/Detection by Probabilistic Decision-Based Neural Network

Shang-Hung Lin, Sun-Yuan Kung, *Fellow, IEEE,* and Long-Ji Lin

***Abstract*—This paper proposes a face recognition system based on probabilistic decision-based neural networks (PDBNN). With technological advance on microelectronic and vision system, high performance automatic techniques on biometric recognition are now becoming economically feasible. Among all the biometric identification methods, face recognition has attracted much attention in recent years because it has potential to be most *non-intrusive* and *user-friendly*. The PDBNN face recognition system consists of three modules: First, a *face detector* finds the location of a human face in an image. Then an *eye localizer* determines the positions of both eyes in order to generate meaningful feature vectors. The facial region proposed contains eyebrows, eyes, and nose, but excluding mouth. (Eye-glasses will be allowed.) Lastly, the third module is a *face recognizer*. The PDBNN can be effectively applied to all the three modules. It adopts a hierarchical network structures with nonlinear basis functions and a competitive credit-assignment scheme. The paper demonstrates a successful application of PDBNN to face recognition applications on two public (FERET and ORL) and one in-house (SCR) databases. Regarding the *performance*, experimental results on three different databases such as recognition accuracies as well as false rejection and false acceptance rates are elaborated in Section IV-D and V. As to the *processing speed*, the whole recognition process (including PDBNN processing for eye localization, feature extraction, and classification) consumes approximately one second on Sparc10, without using hardware accelerator or co-processor.**

***Index Terms*— Decision-based neural network (DBNN), probabilistic DBNN, face detection, eye localization, virtual pattern generation, positive/negative training sets, hierarchical fusion, face recognition system.**

I. Introduction

With its emerging applications to secure access control, financial transactions, etc., biometric recognition systems (e.g., face, palm, finger print) have recently taken on a new importance. With technological advance on microelectronic and vision system, high performance automatic techniques on biometric recognition are now becoming economically feasible [3]. Among all the biometric identification methods, face recognition has attracted much attention in recent years because it has potential to be most *nonintrusive* and *user-friendly*. In this paper we propose an integrated face recognition system for security/surveillance purposes. This system involves three major tasks: 1) human face detection from still images and video sequences; 2) eye localization; and 3) face recognition/rejection. Several different techniques have been proposed over the last 20 years to tackle the above three problems. A brief review of those research efforts is as follows.

The key issue and difficulty in face detection is to account for a wide range of variations in facial images. There exist several approaches for dealing with such variations, including: 1) spatial image invariants; 2) correlation template(s); and 3) view-based eigen-spaces, etc. Some schemes exploit some common and unique spatial image relationships existing among all face patterns, called *image invariants*. One such image-invariant is the local ordinal structure of brightness distribution between different parts of a human face [4]. The schemes check these invariants for positive occurrences at all image locations. Yet the correlation template approach computes a similarity measurement between a fixed target pattern and candidate image locations. If the correlation exceeds a certain threshold, then a face is confirmed, i.e., detected. Some face detection approaches use a bank of correlation templates to detect major facial subfeatures in the image [5], [6]. However, the set of allowable facial patterns is probably too large to be adequately represented by fixed template or templates. A closely related approach is that by means of view-based eigenspaces [7]. The approach assumes that the set of all possible face patterns occupies a small and parameterized subspace, derived from the original high-dimensional input image space. Typically, the approach approximates the subspace of face patterns using data clusters and their principal components from one or more example sets of face images. An image pattern is classified as "a face" if its distance to the clusters is below a certain threshold, according to an appropriate distance metric. An advantage of eigenspace approaches is that it is possible to deal with occluded situations. Several image reconstruction and pose estimation examples based on eigenface method are included in [7]. However, there is no experimental result showing the detection rate of occluded faces in the paper. Also, this approach has so far only demonstrated to be working in uncluttered background. Another eigenspace method is reported in [8]. this method estimates the likelihood function of the face class from the principal face subspace and then decide whether the input image pattern is a face or not based on its likelihood function value. Again, [8] only demonstrates its performance on the database containing faces in uncluttered background (the ARPA FERET database). This method is similar to our approach, whose objective is also to estimate

Manuscript received February 15, 1996; revised June 19, 1996.

S.-H. Lin and S.-Y. Kung are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08540 USA.

L.-J. Lin is with Siemens SCR Inc., Princeton, NJ 08540 USA.

Publisher Item Identifier S 1045-9227(97)00233-6.

the likelihood density of the face class. However, instead of estimating the probability distribution from the principal subspace, we perform maximum likelihood estimation on the subspace that consists of the high frequency components of the image (the edge information, see Section II-B). Another difference between our approach and the eigenface approach is that after density estimation step, we apply a reinforced and anti-reinforced learning procedure, to further fine-tune the decision boundary. According to the experimental result, this process of "training by positive and negative examples" significantly improve the detection accuracy (cf. Section III).

There are some existing techniques for eye localization based on Hough transform, geometry and symmetry check, and deformable models. Some cannot cope with shape changes, and others are too time consuming. Furthermore, none of them can locate eyes when the eyes are closed. In [9], a method was proposed for fast eye localization based on a novel filter. The basic idea is to use the relatively high horizontal-contrast density determination, facial geometry reasoning, and eye position determination. This scheme can locate eyes even when the eyes are closed.

As to previous research works on face recognition, we recommend a recent paper [10] and references therein for a more complete survey. Kelly [11] pioneered experiments using various kinds of facial features, including width of head, distance between eyes, top of head to eyes, between eye and nose and the distance from eye to mouth. Various statistical approaches have been developed: for examples, mixture distance [12], KL Transform [13], SVD [14], and eigenfaces [7] techniques. The eigenface technique can work with a large database, with 7562 images for about 3000 persons. It reached a very remarkable recognition rate (95%) in a 200 images test set. However, the performance degraded significantly when the face size changes [15]. Recently, [8] proposed a local-view method for eigenface approach to deal with scaling variation.

Neural network techniques are very suitable for the face detection and recognition systems. Instead of recognizing a face by following a set of human-designed rules, neural networks learn the underlying rules from the given collection of representative examples. This ability to automatically *learn from examples* makes neural network approaches attractive and exciting. Moreover, it is well known that neural networks are very robust and adaptive. Therefore, for the applications which undergo many variation factors (e.g., face recognition), neural networks seem to be a good remedy to solve the problems. How to construct a neural model structure is crucial for successful recognition. It is very much dependent on the intended application, e.g., [6] and [16], etc., e.g., gender [17], and facial expression classification [18]. For face detection, multilayer perceptron [19] and convolutional neural network [20] have been applied. For face verification, the dynamic link architecture [16] uses Gabor wavelets as features. Another is the Cresceptron [21], which is a multiresolution pyramid structure, similar to the classic Fukushima's neocognitron.

In this paper, a probabilistic decision based neural network (PDBNN) [1], [2] which has the merits of both neural networks and statistical approaches is proposed to attack face detection, eye localization, and face recognition altogether. Two features of the PDBNN make itself suitable implementation for not only the face recognition system, but also other biometric identification systems. These features are:

- *Architectural Feature*: The PDBNN inherits the modular structure from its predecessor, Decision Based Neural Network (DBNN) [22]. For each person to be recognized, PDBNN devotes one of its subnets to the representation of that particular person. This kind of structure is beneficial not only for training/recognition performance, but also for hardware implementation. More specifically, we have the following.
  1) The system will be easy to maintain. Take company security system as an example. The updating of a PDBNN-based security system is relatively straightforward. An addition or deletion of one or more subnets (using localized training process) is sufficient to take care of any change of personnel. A centralized system, in contrast, would have to involve a global updating.
  2) A distributed computing principle is adopted. When the number of persons increases, the computing hardware will become more demanding. Due to its modular architecture, a PDBNN-based biometric identification system is relatively easy to implement on parallel computer.
  3) It leads to a portable ID device. It is possible to implement a PDBNN-based biometric identification system on a wallet-size magnet card system. The user may carry a "smart card", which need to record only the parameters of the subnet in the PDBNN corresponding to the user him/herself.
- *Performance Feature*: The discriminant function of PDBNN is in a form of probability density. This yields low false acceptance and false rejection rates, as discussed in Section IV-B. This characteristic is very critical in preventing illegal entrance by an intruder.

The organization of the paper is as follows. In Section II, an overview of a total system for automatic face recognition is presented. The first two modules of the system, face detector and eye localizer, are discussed in Section III, after the structural properties and learning rules of PDBNN are described in great details. The third (face recognition) module implementation by PDBNN is discussed in Section IV. To further improve the performance, a hierarchical face recognition system is also proposed in Section V. Experimental results are provided in each section.

II. PDBNN Face Recognition System

A PDBNN-based face recognition system [1], [2], [23], [24] is being developed under a collaboration between Siemens Corporate Research, Princeton, and Princeton University. The

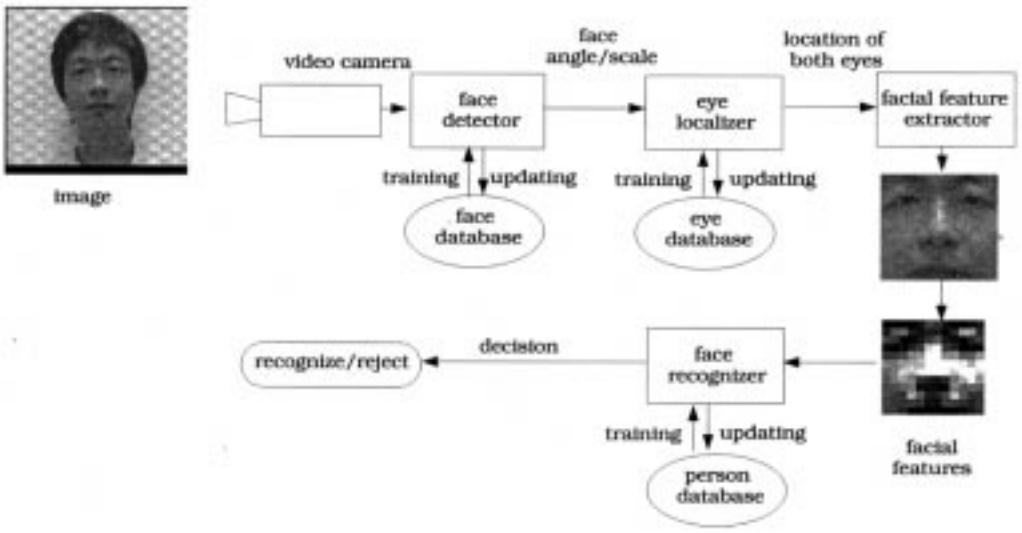


Fig. 1. System configuration of the face recognition system. Face recognition system acquires images from video camera. Face detector determines if there are faces inside images. Eye localizer indicates the exact positions of both eyes. It then passes their coordinates to facial feature extractor to extract low-resolution facial features as input of face recognizer.

total system diagram is depicted in Fig. 1. All the four main modules, face detector, eye localizer, feature extractor, and face recognizer are implemented on a SUN Sparc10 workstation. An RS-170 format camera with 16 mm, F1.6 lens is used to acquire image sequences. The S1V digitizer board digitizes the incoming image stream into 640 $\times$ 480 eight-bit grayscale images and stores them into the frame buffer. The image acquisition rate is on the order of four to six frames/s. The acquired images are then down sized to 320 $\times$ 240 for the following processing.

As shown in Fig. 1, the processing modules are executed sequentially. A module will be activated only when the incoming pattern passes the preceding module (with an agreeable confidence). After a scene is obtained by the image acquisition system, a quick detection algorithm based on binary template matching is applied to detect the presence of a proper sized moving object. A PDBNN face detector is then activated to determine whether there is a human face. If positive, a PDBNN eye localizer is activated to locate both eyes. A subimage (approximately 140 $\times$ 100) corresponding to the face region will then be extracted. Finally, the feature vector is fed into a PDBNN face recognizer for recognition and subsequent verification.

The system built upon the proposed has been demonstrated to be applicable under reasonable variations of orientation and/or lighting, and with possibility of eye glasses. This method has been shown to be very robust against large variation of face features, eye shapes and cluttered background [25]. The algorithm takes only 200 ms to find human faces in an image with 320 $\times$ 240 pixels on a SUN Sparc10 workstation. For a facial image with 320 $\times$ 240 pixels, the algorithm takes 500 ms to locate two eyes. In the face recognition stage, the computation time is linearly proportional to the number of persons in the database. For a 200 people database, it takes less than 100 ms to recognize a face. Furthermore, because of the inherent parallel and distributed processing nature of DBNN, the technique can be easily implemented via specialized hardware for real time performance.

A. Face Detection and Eye Localization

The training patterns of the PDBNN face detector and eye localizer are from a set of images with predefined coordinates (i.e., annotation). Based on these coordinates, the facial feature vector of size 12 $\times$ 12 is extracted from the original facial image by the similar method shown in Section II-B. To detect a face in an input image, each of the possible subimages is processed to see if it represents a face. The input images are preprocessed before inputing to PDBNN for face detection. A confidence score is produced by the neural network, indicating the system's confidence on this detection result. If the score is below some threshold, then no object is detected. The network has consistently and reliably determined actual face positions, based on experiments performed on more than one thousand testing patterns.

After the face detection phase, a PDBNN eye localizer is applied to a face image to locate the left and right eyes. Fig. 2 shows an example of a detected face and the search windows for eye localization. This is a key feature of our approach, since the eye positions provide a very effective means to normalize the face size and reorient the face image. The pattern resolution used for eyes is much higher than that used for faces. The proposed technique is insensitive to small change of the head size, face orientation (up to approximately 30%), and the style of eye glasses. It is also insensitive to partial occlusion by specular reflection of the lenses.

B. Face Recognition

The face recognizer is also based on PDBNN. The pattern acquisition procedure for PDBNN is described as follows.

Fig. 2. The thin white box represents the rough location found by face detector. Based on the location of face, the searching windows for locating eyes are assigned, as illustrated by the two thick white boxes.

- *Facial Region*

  Based on the given location of left and right eyes, a facial region can be extracted. The facial region contains eyebrows, eyes, and nose. An example is shown in Fig. 3(a). Such a facial region—consisting of eyes and nose (excluding mouth)—can yield very high confidence. It can provide distinctive facial features and yet offer a relative invariance against different facial expressions, hair styles, and mouth movement. As for the secondary facial feature, cf. Section V, hairline and mouth features are very good candidates.
- *Feature Normalization and Resolution Reduction*

  Two kinds of feature are used for face recognition; intensity and edge. These two different feature vectors are fed into two PDBNN's. The final recognition result is the fusion of the outputs of these two PDBNN's. The intensity and edge values in the facial region are normalized (to a range between zero and one) to compensate for changing illumination. Edge filtering and histogram modification techniques can be applied to recondition the facial images. The normalized and reconditioned (e.g., 140×100) images are then reduced to coarser (e.g., 14×10) feature vectors. Advantages for adopting lower resolution facial features are: 1) reducing computational cost and storage space; and 2) increasing tolerance on location errors incurred by preceded face detection and eye localization steps. One example of extracted features is shown in Fig. 3.

1) Face Verification System: The trained system can be easily adapted to face verification application. Due to the distributed structure of PDBNN, any individual person's database may be individually retrieved for verification of his/her identity as proclaimed.

C. Training Pattern Generation

There are three main aspects of the training pattern generation scheme for PDBNN.

1) *Virtual training patterns*. In order to assure sufficient diversity of real facial images in the training set, the algorithm takes the acquired sensor image and transforms it to create additional training exemplars, i.e., *virtual training patterns*. Up to 200 virtual training patterns can be generated from one original pattern by applying various affine transformations (e.g., rotation, scaling, shifting) and mirroring process. Robustness of the trained network has consistently improved with the usage of virtual training set.
2) *Positive/negative training patterns*. Not all virtual training patterns should be considered as good face or eye patterns. If a virtual pattern is slightly perturbed from the original exemplar pattern, it will be included in the "positive" training set. On the other hand, if the perturbation exceeds certain threshold (empirically established by trial-and-error), the virtual pattern will be included in the "negative" training set. When training PDBNN, positive patterns are used for reinforced learning, and negative patterns for anti-reinforced learning. The presence of the virtual patterns usually enhances the robustness of the neural network.
3) *Run-time negative pattern generation*. During the training phase, the PDBNN—while still under training—can be used to examine the whole image database every k epochs. If the network falsely detects a face (eye) somewhere in an image, then that particular subimage will be included into the "negative" training set.

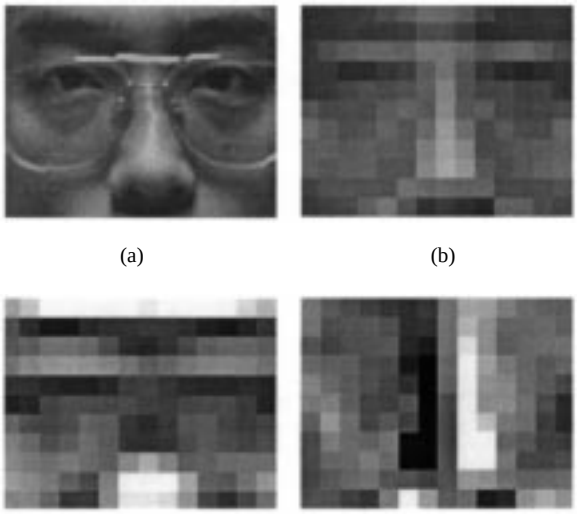


Fig. 3. (a) Facial region used for face recognition. (b) Intensity feature extracted from (a). (c) X-directional gradient feature. (d) Y-directional gradient feature.

III. PDBNN For Face Detection and Eye Localization

Face detection and eye localization can be viewed as a two-class classification problem. Take face detection for example. Given an input pattern, a face detector decides whether it is a face (ω_1) or not (ω_0). A *falsely rejected* pattern is a face pattern which is misclassified to nonface class. Similarly, a *falsely accepted* pattern is a nonface pattern which is accidentally misclassified to face class. Therefore, by reducing the misclassification errors, we are reducing the summation of false acceptance rate and false rejection rate. In this paper, we

apply the PDBNN classifier to implement face detector and eye localizer.

A. Probabilistic Decision-Based Neural Network

PDBNN is a probabilistic variant of its predecessor, DBNN [22]. DBNN is an efficient classification neural network. It has a modular network structure. One subnet is designated to represent one object class (this "One-Class-One-Network" property will be further explored in Section IV-A). PDBNN inherits this structural property. For the face detection problem, since the "non"-face class can be considered as the complement of face class, PDBNN detector uses only one, instead of using two, subnet. This subnet is used to represent the face class.

There are two properties of the DBNN learning rules. The first one is *decision based learning rules*. Unlike the approximation neural networks, where exact target values are required, the teacher in DBNN only tells the correctness of the classification for each training pattern. Based on the teacher information, DBNN performs a distributed and localized updating rule. There are three main aspects of this training rule.

1) *When to update?* A selective training scheme can be adopted, e.g., weight updating only when misclassification.
2) *What to update?* The learning rule is distributive and localized. It applies *reinforced learning* to the subnet corresponding to the correct class and *antireinforced learning* to the (unduly) winning subnet.
3) *How to update?* Adjust the boundary by updating the weight vector $\boldsymbol{w}$ either in the direction of the gradient of the discriminant function (i.e., reinforced learning) or opposite to that direction (i.e., antireinforced learning).

The second property of the DBNN learning rules is *hybrid locally unsupervised and globally supervised learning*. The training scheme of DBNN is based on the so-called *LUGS* (Locally Unsupervised Globally Supervised) learning. There are two phases in this scheme: during the locally-unsupervised (LU) phase, each subnet is trained individually, and no mutual information across the classes may be utilized. After the LU phase is completed, the training enters the Globally-Supervised (GS) phase. In GS phase teacher information is introduced to reinforce or anti-reinforce the decision boundaries obtained during LU phase. The discriminant functions in all clusters will be trained by the two-phase learning.

PDBNN follows the principles of these learning properties. In the LU phase, PDBNN uses the positive training patterns to adjust the subnet parameters by some unsupervised learning algorithm, and in the GS phase it only uses the misclassified patterns for reinforced and antireinforced learning. The negative patterns are only used for the antireinforced training of the subnet. The decision boundaries are determined by a threshold, which can also be trained by reinforced and antireinforced learning. The detailed description of PDBNN detector is given in the following.

Discriminant Functions of PDBNN

One major difference between the prototypical DBNN and PDBNN is that PDBNN follows probabilistic constraint. That is, the subnet discriminant functions of PDBNN are designed to model the log-likelihood functions. The reinforced and antireinforced learning is applied to *all* the clusters of the global winner and the supposed (i.e., the correct) winner, with a weighting distribution proportional to the degree of possible involvement (measured by the likelihood) by each cluster.

Given a set of iid patterns $\boldsymbol{X}^+ = \{\boldsymbol{x}(t); t = 1, 2, \cdots, N\}$, we assume that the class likelihood function $p(\boldsymbol{x}(t)|\omega)$ for class ω (i.e., face class or eye class) is a mixture of Gaussian distributions. Define $p(\boldsymbol{x}(t)|\omega, \Theta_r)$ to be one of the Gaussian distributions which comprise $p(\boldsymbol{x}(t)|\omega)$ $(p(\boldsymbol{x}(t)|\omega, \Theta_r) \equiv N(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r))$

$$p(\boldsymbol{x}(t)|\omega) = \sum_{r=1}^{R} P(\Theta_r|\omega) p(\boldsymbol{x}(t)|\omega, \Theta_r)$$

where Θ_r represents the rth cluster in the subnet. $P(\Theta_r|\omega)$ denotes the prior probability of cluster r. By definition $\Sigma_{r=1}^{R} P(\Theta_r|\omega) = 1$.

The discriminant function of one-subnet PDBNN models the log-likelihood function

$$\begin{aligned} \phi(\boldsymbol{x}(t), \boldsymbol{w}) &= \log p(\boldsymbol{x}(t)|\omega) \\ &= \log \left[\sum_r P(\Theta_r|\omega) p(\boldsymbol{x}(t)|\Theta_r, \omega) \right] \end{aligned} \tag{1}$$

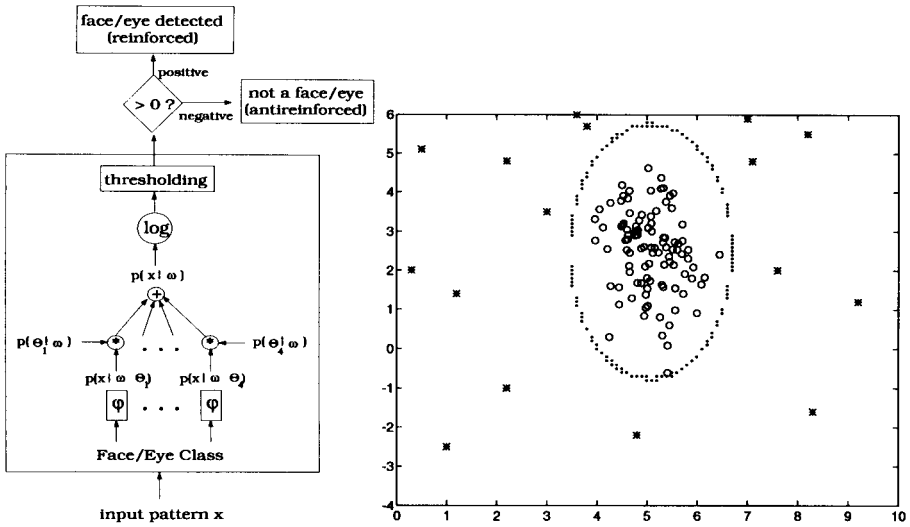
where

$$\boldsymbol{w} \equiv \{\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r, P(\Theta_r|\omega), T\}. \tag{2}$$

T is the threshold of the subnet. It will be used in the GS learning phase. The overall diagram of such discriminant function is depicted in Fig. 4(a). Again, an explicit teacher value would not be required, just like the original DBNN.

1) Elliptic Basis Function (EBF): In most general formulation, the basis function of a cluster should be able to approximate the Gaussian distribution with full-rank covariance matrix. A hyper-basis function (HyperBF) is meant for this [26]. However, for those applications which deal with high-dimensional data but finite number of training patterns, the training performance and storage space discourage such matrix modeling. A natural simplifying assumption is to assume uncorrelated features of unequal importance. That is, suppose that $p(x|\omega, \Theta_r)$ is a D-dimensional Gaussian distribution with uncorrelated features

$$\begin{aligned} p(\boldsymbol{x}(t)|\omega, \Theta_r) &= \frac{1}{(2\pi)^{D/2} \prod_d^D \sigma_{rd}} \\ &\quad \cdot \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d(t) - w_{rd})^2}{\sigma_{rd}^2} \right) \\ &\sim N(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r) \end{aligned} \tag{3}$$



(a) (b)

Fig. 4. (a) Schematic diagram of PDBNN face (or eye) detector. (b) An example of pattern detection by PDBNN. '$\circ$' stands for patterns belonging to the target object, and '$*$' stands for "nonobject" patterns.

where $\boldsymbol{x}(t) = [x_1(t), x_2(t), \cdots, x_D(t)]^T$ is the input pattern, $\boldsymbol{\mu}_r = [w_{r1}, w_{r2}, \cdots, w_{rD}]^T$ is the mean vector, and diagonal matrix $\boldsymbol{\Sigma}_r = \mathrm{diag}\,[\sigma_{r1}^2, \sigma_{r2}^2, \cdots, \sigma_{rD}^2]$ is the covariance matrix.

To approximate the density function in (3), we apply the elliptic basis functions (EBF's) to serve as the basis function for each cluster

$$\psi(\boldsymbol{x}(t), \omega, \Theta_r) = -\tfrac{1}{2}\sum_{d=1}^{D} \beta_{rd}(x_d(t) - w_{rd})^2 + \theta_r \quad (4)$$

where

$$\theta_r = -\frac{D}{2}\ln 2\pi - \sum_{d=1}^{D} \ln \sigma_{rd}.$$

After passing an exponential activation function, $\exp\{\psi(\boldsymbol{x}(t), \omega, \Theta_r)\}$ can be viewed the same Gaussian distribution as described in (3), except a minor notational change: $1/\beta_{rd} = \sigma_{rd}^2$.

Learning Rules for PDBNN

Recall that the training scheme for PDBNN follows the LUGS principle. The locally unsupervised (LU) phase for the PDBNN can adopt one of the unsupervised learning schemes (e.g., VQ, k-mean, EM, ...). As for the globally supervised (GS) learning, the decision-based learning rule is adopted. Both training phases need several epochs to converge. The network enters the GS phase after the LU training is converged.

1) Unsupervised Training for LU Learning: The values of the parameters in the network are initialized in the LU learning phase. The following lists two unsupervised clustering algorithms. Either one of the two can be applied to the LU learning.

- *K-mean and Vector Quantization*: The k-mean method adjusts the center of a cluster based on the distance $||x - \mu_r||^2$ of its neighboring patterns x. Basically k-mean algorithm assumes that the covariance matrix of a cluster i is $\sigma_i^2 I$. An iteration of the k-mean algorithm in the LU phase contains two steps: first, all the input patterns are examined to find out their closest cluster centers. Then, each cluster center are moved to the mean of its neighboring patterns.

  One limitation of the k-mean algorithm is that the number of clusters needs to be decided before training. An alternative is to use the vector quantization (VQ) algorithm. VQ can adaptively create a new neuron for an incoming input pattern if it is determined (by a vigilance test) to be sufficiently different from the existing clusters. Therefore, the number of clusters may vary from class to class. Notice that one must carefully select the threshold value for the vigilance test so that the clusters will not be too large to incur high reproduction error, or too small to lose generalization accuracy.

  Like k-mean, the VQ algorithm assumes that the covariance matrix of a cluster i is $\sigma_i^2 I$.
- *EM*: The EM algorithm [27] is a special kind of quasi-Newton algorithm with a searching direction having a positive projection on the gradient of the log likelihood. In each EM iteration, there are two steps: Estimation (E) step and Maximization (M) step. The M step maximizes a likelihood function which is further refined in each iteration by the E step. The EM algorithm in the LU phase is as follows [28]. Use the data set $\boldsymbol{X}^+ = \{\boldsymbol{x}(t); \boldsymbol{x}^{(t)} \in \omega, t = 1, 2, \cdots, N\}$. The goal of the EM learning is to maximize the log likelihood of data set $\boldsymbol{X}^+$

$$l(\boldsymbol{w}; \boldsymbol{X}^+) = \sum_{t=1}^{N} \log p(\boldsymbol{x}(t)|\omega) = \sum_{t=1}^{N} \log \left[\sum_r P(\Theta_r|\omega) p(\boldsymbol{x}(t)|\Theta_r, \omega)\right]. \quad (5)$$

The EM algorithm begins with the observation that *the optimization of the likelihood function $l(\boldsymbol{w};\boldsymbol{X}^+)$ would be simplified if only a set of additional variables, called "missing" or "hidden" variables, were known*. In order to include the missing information into the optimization formula, here we define indicator variables $z_r(t)$ to specify which cluster generate the pattern

$$z_r(t)=\begin{cases}1, & \text{if pattern } \boldsymbol{x}(t) \text{ is from cluster } r\\ 0, & \text{otherwise.}\end{cases} \tag{6}$$

Now we refer to the observable data $\boldsymbol{X}^+$ as the "incomplete data" and posit a "complete data" set $\boldsymbol{Y}$ that includes the missing variables $\boldsymbol{Z}$. The probability model $p(\boldsymbol{y}|\boldsymbol{x},\omega)$ is used to link the missing variables to the actual data. The logarithm of this density p defines the "complete-data likelihood"

$$l_c(\boldsymbol{w};\boldsymbol{X}^+)=\sum_{t=1}^{N}\sum_{r} z_r(t)\log\left[P(\Theta_r|\omega)p(\boldsymbol{x}(t)|\Theta_r,\omega)\right] \tag{7}$$

$$=\sum_{t=1}^{N}\sum_{r} z_r(t)[\log P(\Theta_r|\omega) + \log p(\boldsymbol{x}(t)|\Theta_r,\omega)]. \tag{8}$$

Since $z_r(t)$ indicates which cluster the input pattern belongs to, the log operator and the second summation can be exchanged. We can see that the optimization has been decomposed into several subproblems.

Notice that since the indicator variable $z_r(t)$ is actually unknown, in the E step of the jth iteration of the EM algorithm we take the expectation of the complete-data likelihood

$$Q(\boldsymbol{w},\boldsymbol{w}^{(j)})=E[l_c(\boldsymbol{w};\boldsymbol{X}^+)|\boldsymbol{X}^+,\omega,\boldsymbol{w}^{(j)}] \tag{9}$$

for simplicity sake, we change the notation of the expectation to the following:

$$Q(\boldsymbol{w},\boldsymbol{w}^{(j)}) = E^{(j)}\left[\sum_{t=1}^{N}\sum_{r} z_r(t)[\log P(\Theta_r|\omega) + \log p(\boldsymbol{x}(t)|\Theta_r,\omega)]\right] \tag{10}$$

$$=\sum_{t=1}^{N}\sum_{r} h_r^{(j)}(t)[\log P(\Theta_r|\omega)+\log p(\boldsymbol{x}(t)|\Theta_r,\omega)] \tag{11}$$

where we define $p^{(j)}(\boldsymbol{x}(t)|\Theta_r,\omega)\equiv N(\boldsymbol{\mu}_r^{(j)},\boldsymbol{\Sigma}_r^{(j)})$ and

$$h_r^{(j)}(t)\equiv E^{(j)}[z_r(t)] = P^{(j)}(z_r(t)=1|\boldsymbol{x}(t),\omega) = \frac{P^{(j)}(z_r(t)=1|\omega)p^{(j)}(\boldsymbol{x}|\omega,z_r(t)=1)}{p^{(j)}(\boldsymbol{x}(t)|\omega)} = \frac{P^{(j)}(\Theta_r|\omega)p^{(j)}(\boldsymbol{x}(t)|\omega,\Theta_r)}{\sum_k P^{(j)}(\Theta_k|\omega)p^{(j)}(\boldsymbol{x}(t)|\omega,\Theta_k).} \tag{12}$$

$\boldsymbol{w}^{(j)}$ represents the parameter set in the PDBNN at epoch j. After the Expectation step, in the M step of the EM algorithm we maximize $Q(\boldsymbol{w},\boldsymbol{w}^{(j)})$ with respect to the weighting parameters.

The followings are the operations taken in a iteration of the EM algorithm. At iteration j. 1) *E-step:* we first compute the conditional posterior probabilities $h_r^{(j)}(t),\forall r$

$$h_r^{(j)}(t)=\frac{P^{(j)}(\Theta_r|\omega)p^{(j)}(\boldsymbol{x}(t)|\omega,\Theta_r)}{\sum_k P^{(j)}(\Theta_k|\omega)p^{(j)}(\boldsymbol{x}(t)|\omega,\Theta_k).} \tag{13}$$

2) *M-step:* maximizing $Q(\boldsymbol{w},\boldsymbol{w}^{(j)})$ with respect to $\boldsymbol{w}$ [cf. (2)], we have:

$$P^{(j+1)}(\Theta_r|\omega)=(1/N)\sum_{t=1}^{N}h_r^{(j)}(t)$$

$$\boldsymbol{\mu}_r^{(j+1)}=\left(1\Big/\sum_{t=1}^{N}h_r^{(j)}(t)\right)\sum_{t=1}^{N}h_r^{(j)}(t)\boldsymbol{x}(t)$$

$$\boldsymbol{\Sigma}_r^{(j+1)}=\left(1\Big/\sum_{t=1}^{N}h_r^{(j)}(t)\right)\sum_{t=1}^{N}h_r^{(j)}(t)[\boldsymbol{x}(t)-\boldsymbol{\mu}_r^{(j)}]\cdot[\boldsymbol{x}(t)-\boldsymbol{\mu}_r^{(j)}]^T. \tag{14}$$

Notice that since the threshold T will not affect the likelihood value, it is not updated here. When the EM iteration converges, it should ideally obtain maximum likelihood estimation (MLE) of the data distribution. EM has been reported to deliver excellent performance in several data clustering problems [28].

In the LU learning phase of PDBNN, we usually use k-mean or VQ to determine initial positions of the cluster centroids, and then use EM to learn the cluster parameters of each class.

2) Supervised Training for GS Learning: In the global supervised (GS) training phase, teacher information is utilized to fine-tune decision boundaries. When a training pattern is misclassified, the reinforced or antireinforced learning technique [22] is applied

$$\begin{array}{ll}\text{Reinforced learning:} & \boldsymbol{w}^{(j+1)}=\boldsymbol{w}^{(j)}+\eta\nabla\phi(\boldsymbol{x},\boldsymbol{w})\\ \text{Antireinforced learning:} & \boldsymbol{w}^{(j+1)}=\boldsymbol{w}^{(j)}-\eta\nabla\phi(\boldsymbol{x},\boldsymbol{w}).\end{array} \tag{15}$$

In this training phase, two sets of patterns are involved. If the misclassified training pattern is from the positive training (i.e., face or eye) set (the data set $\boldsymbol{X}^+$ in the LU phase), reinforced learning will be applied. If the training pattern belongs to the so-called negative training (i.e., "nonface(or eye)") set, then only the anti-reinforced learning rule will be executed—since there is no "correct" class to be reinforced.

The gradient vectors in (15) are computed as follows:

$$\left.\frac{\partial\phi(\boldsymbol{x}(t),\boldsymbol{w})}{\partial w_{rd}}\right|_{\boldsymbol{w}=\boldsymbol{w}^{(j)}} = h_r^{(j)}(t)\cdot\beta_{rd}^{(j)}(x_d(t)-w_{rd}^{(j)})$$

$$\left.\frac{\partial\phi(\boldsymbol{x}(t),\boldsymbol{w})}{\partial\beta_{rd}}\right|_{\boldsymbol{w}=\boldsymbol{w}^{(j)}} = h_r^{(j)}(t)\cdot\frac{1}{2}\left(\frac{1}{\beta_{rd}^{(j)}}-(x_d(t)-w_{rd}^{(j)})^2\right) \quad (16)$$

where $h_r^{(j)}(t)$ is the conditional posterior probability as shown in (13), and $w_{rd}^{(j)}$ and $\beta_{rd}^{(j)}$ are defined in (3) and (4), respectively. As to the conditional prior probability $P(\Theta_r|\omega)$, since the EM algorithm can automatically satisfy the probabilistic constraints $\Sigma_r P(\Theta_r|\omega)=1$ and $P(\Theta_r|\omega)\geq 0$, it is applied to update the $P(\Theta_r|\omega)$ values in the GS phase so that the influences of different clusters are regulated: At the end of the epoch j

$$P^{(j+1)}(\Theta_r|\omega) = (1/N)\sum_{t=1}^{N} h_r^{(j)}(t). \quad (17)$$

3) Threshold Updating: The threshold value of PDBNN detector can also be learned by the reinforced and antireinforced learning rules. Since the increment of the discriminant function $\phi(\boldsymbol{x}(t),\boldsymbol{w})$ and the decrement of the threshold T have the same effect on the decision making process, the direction of the reinforced and anti-reinforced learning for the threshold is the opposite of the one for the discriminant function. For example, given an input $\boldsymbol{x}(t)$, if $\boldsymbol{x}(t)\in\omega$ but $\phi(\boldsymbol{x}(t),\boldsymbol{w})<T$, then T should reduce its value. On the other hand, if $\boldsymbol{x}(t)\notin\omega$ but $\phi(\boldsymbol{x}(t),\boldsymbol{w})>T$, then T should increase. An adaptive learning rule to train the threshold T is proposed in the following: Define $d(t)\equiv T-\phi(\boldsymbol{x}(t),\boldsymbol{w})$. Also define a penalty function $l(d(t))$. $l(d(t))$ can be either a step function, a linear function, or a fuzzy-decision sigmoidal function. Once the network finishes the training, the threshold values can be trained as follows: Given a positive learning parameter η, at step j

$$T^{(j+1)} = \begin{cases} T^{(j)}-\eta l'(d(t)) & \text{if } \boldsymbol{x}(t)\in\omega \\ & \text{(reinforced learning)} \\ T^{(j)}+\eta l'(d(t)) & \text{if } \boldsymbol{x}(t)\notin\omega \\ & \text{(antireinforced learning).} \end{cases} \quad (18)$$

One simple example of PDBNN detector is shown in Fig. 4(b).

B. Experimental Result

The following paragraphs summarize the empirical results of the PDBNN face detector and eye localizer.

1) Face Detector: For face detection, 92 annotated images were used for training, and 473 images for testing. Since our face detector is used mainly for surveillance purpose, all the images were taken in the normal indoor lighting condition with cluttered background (see Fig. 2). The image size is 320 $\times$ 240 pixels, and the face size is about 140 $\times$ 100 pixels. The variation of head orientation is about 15 degrees toward the four directions (up, down, right, left). The performance was measured by the error (in terms of pixels) between the detected face location and the true location. To be fair to different sizes of faces, the errors were normalized with the assumption that the distance between the two eyes is 40 pixels, which is the average distance in our annotated images. In order to reduce the searching time, the images (320 $\times$ 240 pixels) are normally down-sized by a factor of 7. Working with the low resolution images (search range approximately 46 $\times$ 35 pixels, search step 1 pixel, and block size 12 $\times$ 12 pixels (equivalent to the size of facial feature vector), our pattern detection system detects a face within 200 ms on a SUN Sparc 10 machine. For our face recognition application, a 10-pixel error is acceptable, since the main purpose of face detection is merely to restrict the searching areas for eye localization. Among all the testing images, 98.5% of the errors are within five pixels and 100% are within 10 pixels in the original high-resolution image (which is less than 1 pixel error in the low resolution images).

We compared the PDBNN face detector with two of the leading face detection algorithms [19], [20]. A database consisting of 23 images provided by Sung and Poggio was used for comparison. The pictures in this database are from a wide variety of preexisting sources. There are 155 faces in this database. Under the similar false acceptance performance (three nonface subimages were falsely accepted as faces by [20], five in [19], and six by PDBNN. The false acceptance rates of all the three are below 10^{-6}), [20] missed 34 faces, [19] missed 36 faces, and the PDBNN face detector missed 43 faces. Fig. 5 shows some faces that are detected by the PDBNN face detector. Two reasons explain why PDBNN has inferior performance. First, compared to the huge number of training images used by both groups (4000 in [19] and 16 000 in [20]), our training set only consists of 92 images. A more interesting comparison would be made if a training database for PDBNN with comparable number of samples were available. Second, we observed that PDBNN cannot detect the "artificial faces" in the database (e.g., faces on poker cards, hand-drawn faces) (cf. Fig. 6). Since the PDBNN face detector is mainly used in surveillance and security applications, we think that this characteristic is actually beneficial.

2) Eye Localization: Eye detection is a much harder problem than face detection, because 1) the eye areas are usually very smal; 2) eyes may be open, closed, or semiclosed; and 3) eye glasses often blur the eye areas. Therefore, it requires more training images than face detection. In our experiments, we used 250 annotated images for training and 323 images for testing. The conditions on the images are the same as the database for face detection experiment. We only train a left eye detector. To detect the right eye, we generate the mirror image of the original one, and try to detect a left eye in the mirror image. Like the face detector experiment, the errors are normalized with the assumption that the eye-to-eye distance is 40 pixels. For our face recognition system, three-pixel errors (or less) are well within the tolerance, and even five-pixel errors are often still acceptable. The experimental result shows that 96.4% of the errors are within three pixels, and 98.9% are within five pixels.

3) Assisting Real-Time Face Recognition: In this paragraph, we would like to demonstrate how the PDBNN face

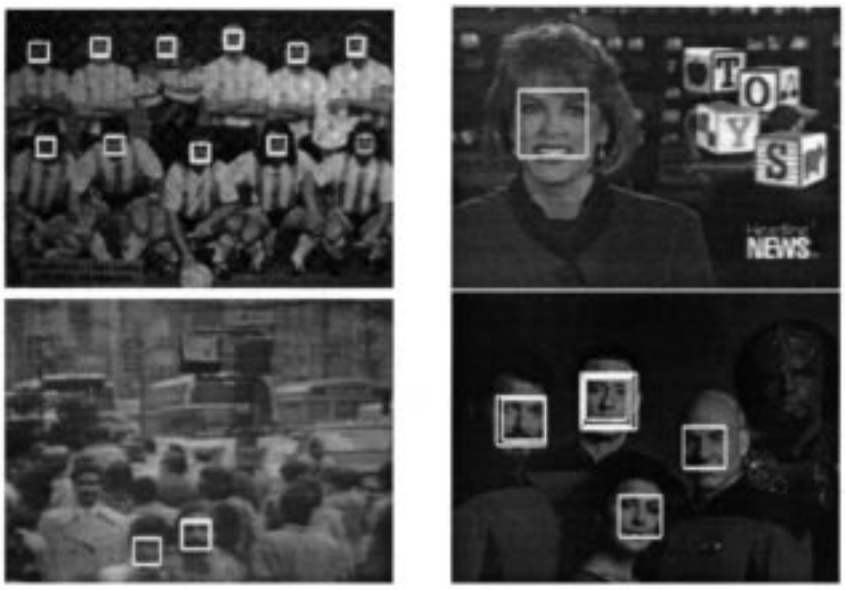


Fig. 5. Some faces detected by DBNN face detector.

(a)

(b)

Fig. 6. (a) Faces detected by Sung and Poggio's algorithm. (b) Face detected by PDBNN. Note that the artificial face drawing (on the board) is determined by (a) as a face—but not by (b).

detector and eye localizer in our system can help us choose useful training images and good test images for the following face recognizer so that a good final recognition result can be achieved. Generally speaking, there are two kinds of sources for a face recognition application to obtain its training images. The first one is by photographing (e.g., mugshots in police database). The number of the training images obtained by this source is usually small and additional images are not easy to get if more training images are needed. The second source to acquire images is by video camera (e.g., gateway security control). The number of the available images can be very large. For example, a video camera with rate 15 frames/s can produce 300 images in 20 s. The system developer therefore has the luxury to choose "good" face images from the large image set to train the recognizer. (Here "a good face image" means an image that contains a face which is clear and distinguishable for the system to determine who he or she is.) However, since it is not easy to decide which image is "good" enough for training and since the number of images is large, the task of choosing training images is difficult and tiresome for human operators. Moreover, when the system is on duty, it must have the ability to automatically choose a good image for recognition from the video stream. Otherwise it just wastes computation power and may get inferior recognition performance. Ideally speaking, if we have a very powerful face recognizer that can recognize faces perfectly under all kinds of size, orientation, deformity, and lighting variations, then all images that contain faces are "good" images for recognition. Unfortunately, so far no existing face recognition techniques can achieve that level of perfection. Therefore, the procedure for selecting images of good quality is still necessary. Notice that we *do not* claim that our system can recognize faces with large variations; instead, we claim that when facing a sequence of video stream, our system can cleverly select the frames that are good enough to represent the identity of the person.

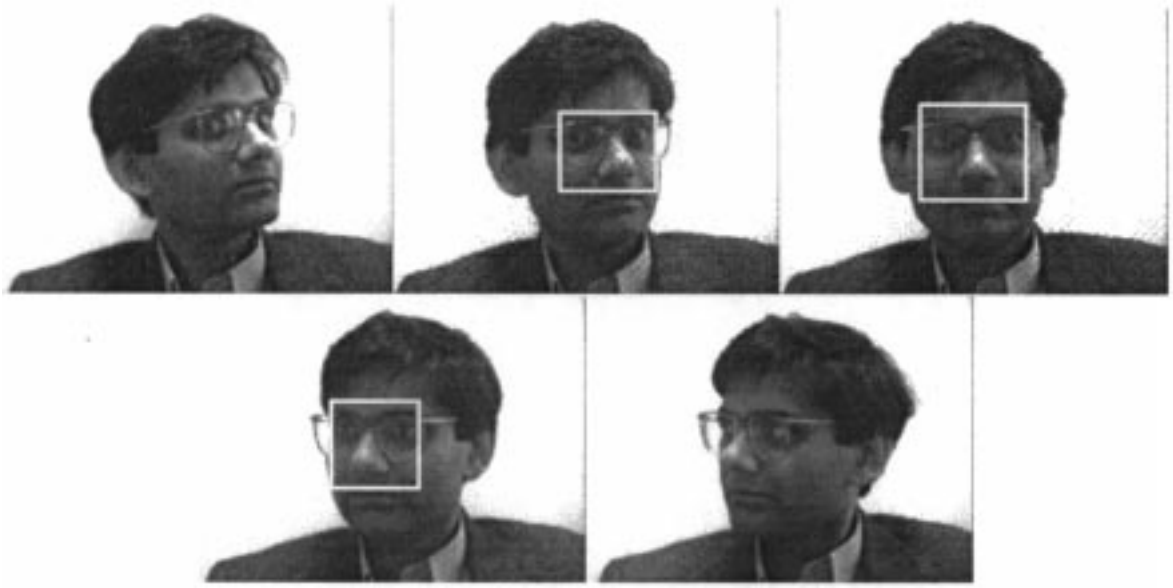


Fig. 7. Images in SCR40 × 150 database. The system rejects the faces in the first and the last images because of low confidence scores on the PDBNN face detector. The faces in the middle three pictures are correctly detected with high confidence scores. The detected faces are depicted by the white bounding boxes.

The confidence score of the PDBNN provides a convenient and accurate criterion for the classification purpose. The experimental results in previous paragraph support the above statement. In this section we show another use of confidence scores: *selecting "useful" face images from large amount of image data*. The confidence scores of the face detector and eye localizer faithfully reflect the correctness of the detected position of the pattern. Therefore an image with high confidence scores of the two modules can almost always generate qualified facial pattern for face recognition.

In this experiment, we have used an image database of 40 people. The SCR 40 × 150 database contains 150 images for each person, which were taken continuously while the person was slowly moving/rotating his head (see Fig. 7). All the images have clean background. Heads rotate not only in wide angle (up to 45 degree) but also along various axes (i.e., left-right, up-down, and tilted rotations). The face detector and eye localizer worked correctly for 75% of the 6000 images in this database. (They are considered a *valid data set*.) Note that the current face detector and eye localizer were trained only on frontal view faces. They nevertheless can handle faces within 30 degrees reasonably reliably. Indeed, most of the failures occurred for faces with large rotation/tilt angle (45 degrees or more). Moreover, since for all the failure cases, the confidence scores of face detector and eye localizer are very low, we can automatically screen out the images with large head rotation angle or in ill condition by thresholding.

We have selected 20% of the valid data set as the training set for the face recognizer. As to the test images, we have used 60 images per person, or 2400 images in total, from the original (40 people, 150 images/person) database. To ensure fairness, the 2400 images were randomly picked from the entire database (excluding those used in the training set). The face detector and eye localizer have automatically selected 2176 out of 2400 images to serve as "valid test set."

TABLE I
PERFORMANCE OF FACE RECOGNITION SYSTEM USING DATABASE WITH LARGE HEAD ORIENTATION. WITH THE ASSISTANCE OF THE CONFIDENCE SCORES FROM PDBNN FACE DETECTOR AND EYE LOCALIZER, USEFUL TRAINING PATTERNS CAN BE AUTOMATICALLY SELECTED (THE VALID SET). THE IDENTIFICATION ACCURACY IS HIGHLY IMPROVED BY USING VALID TRAINING SET

PDBNN	Trained by original set	Trained by valid set
Recognition	84.64%	98.34%
False rejection	10.03%	0.97%
Misclassification	5.33%	0.69%

Table I shows the performance of PDBNN face recognizer on the valid test set of 40 people. Here a false rejection pattern means that its confidence score generated by the face recognizer is below the threshold, and a misclassified pattern means that its confidence score is higher than the threshold but it is classified to a wrong person. For the sake of comparison, we have also trained a PDBNN by a set of training images which were selected from the original data set. That is, bad face images are also included in the training set. We can see that *the recognition rate is deteriorated because of bad training patterns*.

IV. PDBNN FOR FACE RECOGNITION

The PDBNN face recognizer can be considered as an extension of the PDBNN face detector. For a K-people recognition problem, a PDBNN face recognizer consists of K different subnets. Analogous to the PDBNN detector, a subnet i in the PDBNN recognizer estimates the distribution of the patterns of person i only, and treats those patterns which do not belong to person i as the "non i" patterns.

This one-class-one-network (OCON) structure is beneficial in many aspects. Discussion is given in Sections IV-A and IV-B. The extended structure and learning rules for multisubnet PDBNN are presented in Section IV-C.

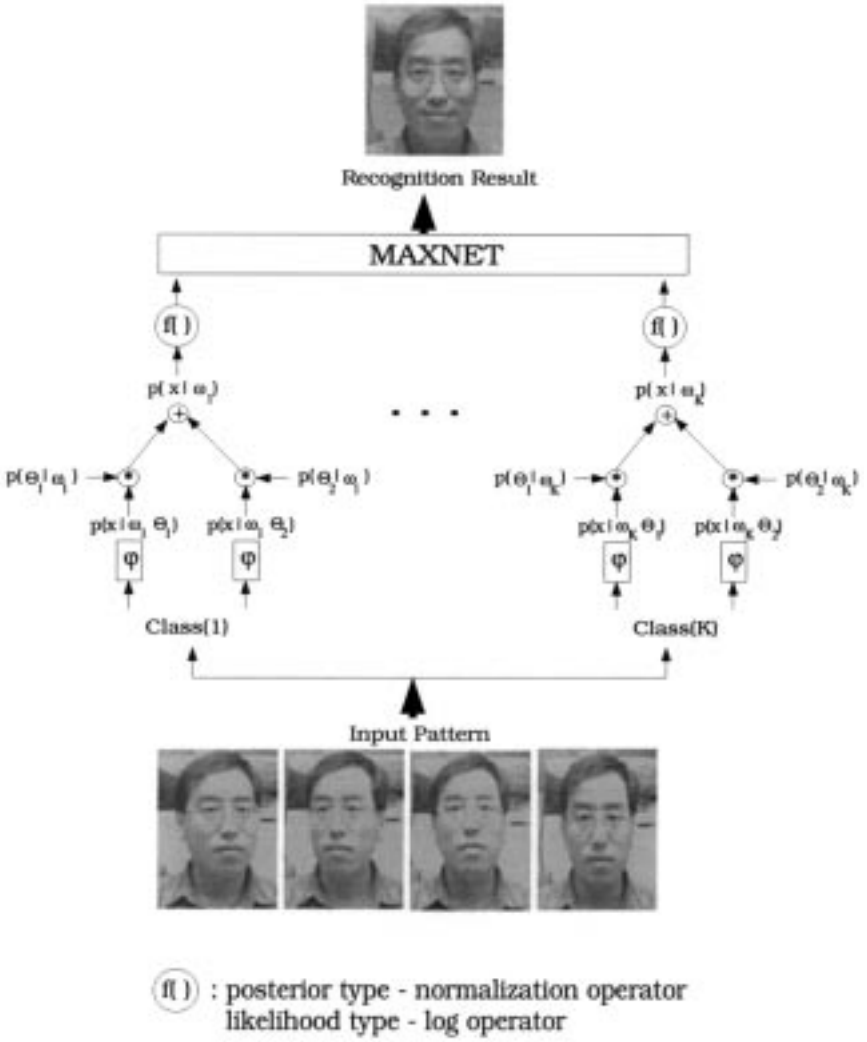


Fig. 8. Structure of PDBNN face recognizer. Each class subnet is designated to recognize one person. All the network weightings are in probabilistic format.

A. ACON Versus OCON Structures

Typically, one output node is designated to represent one class. The all-class-in-one-network (ACON) structure is adopted by the conventional multilayer perceptron (MLP), where all the classes are lumped into one super-network. The supernet has the burden of having to simultaneously satisfy all the teachers, so the number of hidden units K tends to be large. *Empirical results confirm that the convergence rate of ACON degrades drastically with respect to the network size because the training of hidden units is influenced by (potentially conflicting) signals from different teachers.* Inherited from the prototypical DBNN, the PDBNN adopts a one-class-in-one-network (OCON) structure, where one subnet is designated to one class only. The structure of the PDBNN recognizer is depicted in Fig. 8. Each subnet specializes in distinguishing its own class from the others, so the number of hidden units is usually small. Pandya and Macy [29] compare the performance between the ACON and OCON structures on handwritten character recognition. They observe that OCON model achieves better training (99.5% vs 94%) and generalization (87% vs 82%) accuracies and yet requires only 1/4 of ACON's training time.

The OCON structure of PDBNN makes it most suitable for *incremental training*, i.e., *network upgrading upon adding/removing memberships*. Moreover, due to the OCON structure, the trained PDBNN face recognizer can be easily adapted to a face verification system. Any individual person's database can be individually stored, either in computer or in user's magnet card, and individually retrieved for verification of his/her identity as proclaimed.

One may argue that compared to ACON structure, the OCON structure is slow in retrieving time when the number of classes is very large. This is not true because, as we have mentioned earlier, when the number of classes is large, the number of hidden neurons in the ACON structure also tends to be very large. Therefore ACON is also slow. Since the computation time of both OCON and ACON increases as number of classes grows, a linear increasing of computation time (i.e., OCON) is the least we can expect. Besides, the OCON structure is suitable for distributed computing, while ACON is not. Therefore, greater speedup can be anticipated

for the OCON structure if distributed hardware or software implementation is allowed. Moreover, since *the retrieving time of the PDBNN system is actually just a very small portion of the whole system time*, it will not affect the system performance too much even if the class number grows very large. For example, in a 200-people PDBNN recognition system, it takes about 1 s for an input image to go through the whole recognition process (including preprocessing, detection, localization, feature extraction, and recognition). In this 1-s system time, recognition (the retrieving) only takes 100 ms, which is only one tenth of the system time. The retrieving time will become 50% of the system time when the number of people in the database grows over 2000. In this case the system time becomes 2 s, which is still fast enough for security applications.

B. False Acceptance and False Rejection

False rejection and false acceptance rates are important not only for detection problems but also for several recognition problems. For example, it is more dangerous for a face recognition security system to falsely give entry permit to an intruder than mistakenly recognize one member in personnel to another member. As will be illustrated in the following paragraphs, due to the OCON structure and probabilistic discriminant functions, PDBNN is capable of generating low false rejection and false acceptance rates.

Statistical theory of pattern classification shows that the decision boundaries generated by Bayes posterior probabilities produce minimum classification error. For k-class classification problems, the Bayes decision boundaries divide the feature space into k different regions. One simple example is shown in Fig. 9(a). There are two Gaussian distributions in the figure, and the decision boundary is depicted by dotted line. In many data classification applications (e.g., OCR), the data classifiers are designed, either implicitly or explicitly, to approach the Bayes decision boundaries so that the minimum classification error can be reached [30], [31]. However, since the Bayes decision theory is derived based on the assumption that all the data classes are "known," the Bayes decision rule may not be suitable for those object recognition problems which deal with "unknown" data classes. For this type of recognition problems, the false rejection and false acceptance rates should be taken care of. Take Fig. 9(b) as an example. The asteroid points represent for unknown data points. The job of the classifier for this problem is now not only to determine the data class of this input pattern, but also to fire alarm if this input pattern is not from either one of the data classes. Obviously the Bayes classifier in the figure fails to provide the capability of firing alarm for unknown data.

People have tried two approaches to tackle this type of recognition problems. One approach is by thresholding. The input pattern is considered as class i not only when the Bayes conditional posterior of class i for the pattern is the highest among all the classes, but also when the value of the posterior is higher than a certain threshold. With this thresholding mechanism, certain amount of outlier data can be rejected. However, as we can see in Fig. 9(c), due to the property that all the conditional posterior probabilities sum to one, some data points, though very far away from data clusters, still possess very high posterior values. Thresholding is unable to reject those points.

The second approach is by directly approximating the distribution of unknown data. That is, we categorize all the unknown data into a big object class called "unknown class," and then we try to estimate its distribution, just like what we did for the regular object classes. This approach is feasible, but one needs to keep in mind that the distribution of the unknown data is usually much more complicated than that of a regular class data. One example is face recognition for security system. The objective of the system is to reject all the people who are not in the database. Therefore, the size of the unknown class, or intruder class, is close to the population of the whole world. Intuitively we can say that the distribution of the unknown face class is very complicated.

The PDBNN is very suitable for tackling false rejection and false acceptance problem. The reason is that PDBNN focuses only on density distributions in individual classes (rather than global partitioning), no additional intruder class subnet is needed. Also, since the Gaussian density value drops as the distance to the centers increases, the decision regions tend to be more locally conserved, and thus a lower false acceptance rate will be expected. Numerous experiments indicate that this is the case. Fig. 9(d) shows the decision boundaries made by PDBNN.

C. Extension of PDBNN to Multiclass Recognition

Similar to the one-subnet PDBNN, we use mixture of Gaussian as the class likelihood function $p(\boldsymbol{x}(t)|\omega_i)$ for the multiclass PDBNN. The discriminant function of each subnet in PDBNN is as follows:

$$\begin{aligned}\phi(\boldsymbol{x}(t), \boldsymbol{w}_i) &= \log p(\boldsymbol{x}(t)|\omega_i) \\ &= \log\left[\sum_r P(\Theta_{r|i}|\omega_i) p(\boldsymbol{x}(t)|\Theta_{r|i}, \omega_i)\right] \end{aligned} \quad (19)$$

where $\boldsymbol{w_i} \equiv \{\boldsymbol{\mu}_{r|i}, \boldsymbol{\Sigma}_{r|i}, P(\Theta_{r|i}|\omega_i), T_i\}$ is the parameter set for subnet i. The EBF can also be applied to serve as cluster basis function $p(\boldsymbol{x}(t)|\Theta_r, \omega_i)$. The overall diagram of such discriminant function is depicted in Fig. 8. Notice that since the output of PDBNN is of probability form, it can be used as an indicator of the confidence score of the recognition result.

D. Learning Rules for Multisubnet PDBNN

The multisubnet PDBNN follows the same learning rules as the one-subnet PDBNN in the LU learning phase. Each subnet performs k-mean, VQ, or EM to adjust its parameters. In the GS learning phase, only the misclassified patterns are used. The GS learning rules for the multiclass PDBNN is the straightforward extension of the one-class version of the GS learning mentioned in Section III-A2. Notice that there are

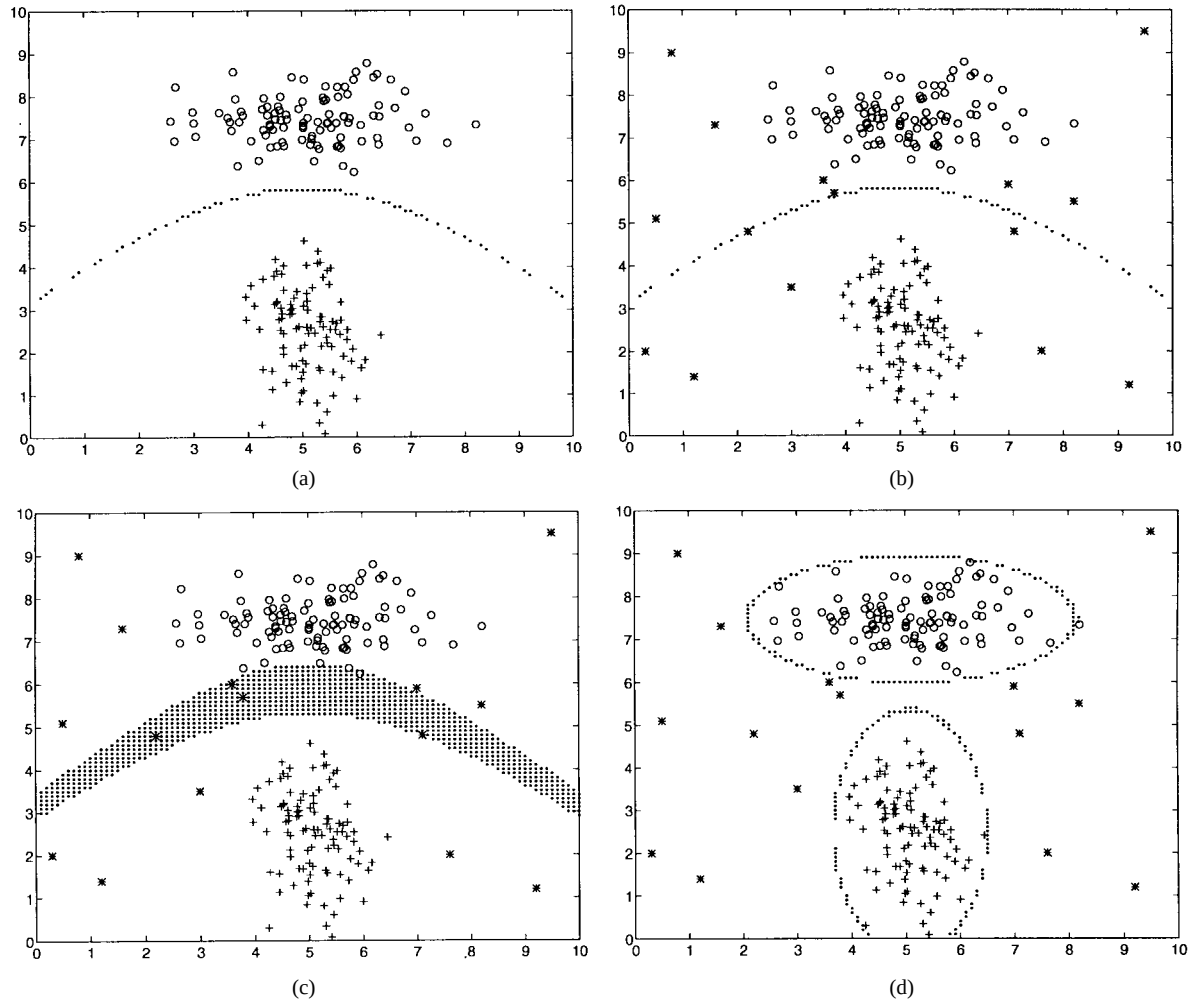


Fig. 9. An example of false acceptance/rejection issue in pattern recognition problems. Three types of patterns appear in this figure. "$\circ$" and "$+$" represent for patterns from two different classes. "$*$" represents for intruder patterns. The dotted lines represent the decision boundaries. (a) A Bayes decision boundary between two classes. (b) Same as (a), but intruder patterns appear. We can see that those intruder patterns are classified to either class "$\circ$" or class "$+$". (c) Rejection region is formed by increasing the decision threshold. Five intruder patterns are rejected by threshold, but ten are still misclassified. (d) Decision boundaries generated by PDBNN. The decision regions are locally preserved, and most intruder patterns are rejected. For detailed description please refer to Section IV-B.

two types of misclassified patterns. The first type is from the positive dataset. A pattern of this type will be used for 1) reinforced learning of the threshold and discriminant function of the class which it actually belongs to, and 2) anti-reinforced learning of the thresholds and discriminant functions of the classes which have higher discriminant function values than the true class has. The second type of the misclassified patterns is from the negative dataset. A negative training pattern is either a intruder face pattern or a nonface pattern. Since there is no subnet representing the negative data, a pattern of this type is used only for anti-reinforced learning of the threshold and discriminant function of the class which it is misclassified to.

The data adaptive scheme of the GS learning for the multiclass PDBNN is the extension of the learning scheme shown in Section III-A2. Here we show the block adaptive gradient version of the GS learning. At the beginning of each GS iteration, use the still-under-training PDBNN to classify all input patterns $\boldsymbol{x}(t)$. $\boldsymbol{x}(t)$ is classified to class ω_j if $\phi(\boldsymbol{x}(t), \boldsymbol{w}_j) > \phi(\boldsymbol{x}(t), \boldsymbol{w}_k), \forall k \neq j$ and $\phi(\boldsymbol{x}(t), \boldsymbol{w}_j) \geq T_i$, where T_i is the threshold for the subnet i. Create a data set for each class ω_i according to the classification results: $D^i = \{\boldsymbol{x}(t); (\boldsymbol{x}(t)$ belongs to class $\omega_i)$ or (is classified to ω_i by current PDBNN). Divide this data set into three subsets:

- $D_1^i = \{\boldsymbol{x}(t); \boldsymbol{x}(t) \in \omega_i, \boldsymbol{x}(t)$ is classified to $\omega_i\}$ (correctly classified set);
- $D_2^i = \{\boldsymbol{x}(t); \boldsymbol{x}(t) \in \omega_i, \boldsymbol{x}(t)$ is misclassified to other class $\omega_j\}$ (false rejection set);
- $D_3^i = \{\boldsymbol{x}(t); \boldsymbol{x}(t) \notin \omega_i, \boldsymbol{x}(t)$ is misclassified to $\omega_i\}$ (false acceptance set).

We only use the patterns which are in the false rejection set (D_2^i) and the false acceptance set (D_3^i) to train the subnet ω_i. In the beginning of the GS learning, the intermediate parameter $h_{r|i}^{(j)}(t)$ is computed as in (13). Also, the cluster prior probabilities $P(\Theta_{r|i}|\omega_i)$ is updated by the same rule as in (17). To update the mean $\boldsymbol{\mu}_{r|i}$ and the diagonal covariance matrix $\boldsymbol{\Sigma}_{r|i}$, we apply gradient ascent approach. At GS iteration j, $\boldsymbol{\mu}_{r|i}$

and $\Sigma_{r|i}$ are updated by the following:

$$\begin{aligned}\mu_{r|i}^{(j+1)} &= \mu_{r|i}^{(j)} + \eta_\mu \sum_{t,\boldsymbol{x}(t)\in D_2^i} h_{r|i}^{(j)}(t)\Sigma_{r|i}^{-1(j)}[\boldsymbol{x}(t) - \mu_{r|i}^{(j)}] \\ &\quad - \eta_\mu \sum_{t,\boldsymbol{x}(t)\in D_3^i} h_{r|i}^{(j)}(t)\Sigma_{r|i}^{-1(j)}[\boldsymbol{x}(t) - \mu_{r|i}^{(j)}] \\ \Sigma_{r|i}^{(j+1)} &= \Sigma_{r|i}^{(j)} + \tfrac{1}{2}\eta_\sigma \sum_{t,\boldsymbol{x}(t)\in D_2^i} h_{r|i}^{(j)}(t)(\boldsymbol{H}_{r|i}^{(j)}(t) - \Sigma_{r|i}^{-1(j)}) \\ &\quad - \tfrac{1}{2}\eta_\sigma \sum_{t,\boldsymbol{x}(t)\in D_3^i} h_{r|i}^{(j)}(t)(\boldsymbol{H}_{r|i}^{(j)}(t) - \Sigma_{r|i}^{-1(j)})\end{aligned} \tag{20}$$

where

$$\boldsymbol{H}_{r|i}^{(j)}(t) = \Sigma_{r|i}^{-1(j)}[\boldsymbol{x}(t) - \mu_{r|i}^{(j)}][\boldsymbol{x}(t) - \mu_{r|i}^{(j)}]^T\Sigma_{r|i}^{-1(j)}$$

and η_μ, η_σ are user-defined positive learning rates.

The thresholds T_i in the multisubnet PDBNN are also trained by the reinforced and anti-reinforced rules, just like the one-subnet case.

E. Experimental Results

In this section, two experimental results for face recognition will be discussed. In the first part we use three face databases consisting of frontal view face images. Frontal view face databases are used in most face recognition research groups. The second experiment explores the ability of PDBNN in solving a problem which is seldom discussed in the face recognition literature. We will show how PDBNN can "reject" intruders. This is an important issue, especially if the recognition system is for biometric identification applications.

1) Experiment 1—Frontal View Faces: We have conducted the experiment on three image database: the SCR 80 × 20 database, the ARPA/ARL FERET database, and the ORL database. The SCR 80 × 20 database consists of 80 people of different races, ages, and genders. The database contains 20 images for each person. (If a person wears glasses, ten of the image are with glasses and ten without.) All of the images were taken under natural indoor illumination condition. All of the images has clean background. The facial orientations in our image database are roughly between −15 and 15 degrees. In many of those images, the person's head is tilted up to 15 degrees. We have created a training data set by using four images per person (two with eye glasses and two without, if the person wears eye glasses). The testing image set includes 16 images per person, 1280 images in total. For all of the images, the face detector always correctly detects the center of the face (i.e., 100% success rate). Eye localization is a more difficult task than face detection, in particular when eye glasses are present. The eye is typically 20 pixels wide and 10 pixels high. Among the 1280 images, eye localizer mis-detects the eyes in five images by errors of more than five pixels. For the remaining 1275 images, the DBNN face recognizer can achieve 100% recognition rate, no matter it was implemented by traditional DBNN or probabilistic DBNN.

TABLE II
PERFORMANCE OF DIFFERENT FACE RECOGNIZERS ON 200 PEOPLE IN THE FERET DATABASE

	Training Accuracy	Testing Accuracy
Probabilistic DBNN	97%	99%
Traditional DBNN	100%	96%
Multi-Layer Perceptron	99.5%	87.5%

A front-view experiment is conducted on the ARPA/ARL FERET database. There are 304 persons, and each of them has two frontal view images. The variation between the two images is much larger than SCR 80 × 20 and 40 × 150 database, in terms of illumination, size, and facial expression. Phillips [32] achieves a 97% top-one identification rate on a preselected 172 faces from FERET database. Moghaddam and Pentland [8] report 99% recognition rate on preselected 155 faces. In this experiment, we take the advantage that *the confidence scores on the face detector and eye localizer can reflect the accuracy of the detected pattern positions*. Therefore, we selected the images whose confidence scores on both the face detector and eye localizer are above the threshold. (For more details about this prescreening scheme, please refer to Experiment 2.) Among the 304 $\times 2 = 608$ images, 491 images have passed both the face detector and eye localizer (success rate = 80.8%). There are 200 persons whose both frontal view images have passed and therefore the images of these 200 people are used for our face recognizer experiment. One image per person is used for training and the other for testing. The face recognizer experimental results are shown in Table II. We can see that under reasonably high training accuracy (97%), PDBNN achieved higher recognition rate (99%) than traditional DBNN (96%). We also used MLP to implement face recognizer. The performance is inferior to both types of DBNN. Notice that the recognition result of this experiment does not imply that the PDBNN face recognizer has superior performance to the eigenspace approach [8] (99% for 155 people), since some of the images they chose may not be selected by the PDBNN face detector and eye localizer. However, this experimental result tells us several things. They are: 1) the PDBNN face recognizer has the capability of recognizing up to 200 people; 2) the face detector and eye localizer can help choosing recognizable images automatically; and 3) PDBNN has better recognition performance than the multilayer perceptron does.

We also conduct an experiment on the face database from the Olivetti Research Laboratory in Cambridge, UK (the ORL database). There are ten different images of 40 different persons. There are variations in facial expression (open/close eyes, smiling/nonsmiling), facial details (glasses/no glasses), scale (up to 10%), and orientation (up to 20 degrees). A HMM-based approach is applied to this database and achieves 13% error rate [33]. The popular eigenface algorithm [15] reports the error rate around 10% [33], [34]. In [35], a pseudo 2-D HMM method is used and achieves 5% at the expense of long computation time (4 minutes/pattern on Sun Sparc II). In

TABLE III
PERFORMANCE OF DIFFERENT FACE RECOGNIZERS ON THE ORL DATABASE. PART OF THIS TABLE IS ADAPTED FROM LAWRENCE ET AL. [34]

System	Error rate	Classification time	Training Time
PDBNN	4%	< 0.1 seconds	20 minutes
SOM + CN	3.8%	< 0.5 seconds	4 hours
Pseudo 2D-HMM	5%	240 seconds	n/a
Eigenface	10%	n/a	n/a
HMM	13%	n/a	n/a

TABLE IV
FALSE ACCEPTANCE RATES OF FACE PATTERNS ON VARIOUS TYPES OF NEURAL NETWORK. (FALSE REJECTION RATE = 0% AND MISCLASSIFICATION RATE = 0%)

	w/o negative examples	w/ negative examples
Probabilistic DBNN	13.75%	8.13%
Traditional DBNN	33.75%	22.5%
Multi-layer Perceptron	33.75%	12.5%

[34] Lawrence *et al*. use the same training and test set size as Samaria did and a combined neural network (self organizing map and convolutional neural network) to do the recognition. This scheme spent four hours to train the network and less than 1 s for recognizing one facial image. The error rate for ORL database is 3.8%. Our PDBNN-based system reaches similar performance (4%) but has much faster training and recognition speed (20 minutes for training and less than 0.1 s for recognition). Both approaches run on SGI Indy. Table III summarizes the performance numbers on ORL database.

2) Experiment 2—Presence of Intruder Patterns: It is more crucial for a recognition system to have the ability to reject intruders. Thus the false acceptance rate and false rejection rate of the system are important performance metrics. We have conducted an experiment for false acceptance/rejection. Among the 80-person frontal-view database, we have chosen 20 persons as our "known person" database and picked up 40 persons from the rest 60 to be the "intruder" database. The remaining 20 persons serve as "negative examples" for the network training. The training datasets are formed by randomly choosing two images per person from database and then generating 50 *virtual training patterns* (cf. Section II-C) from each images. Both "known person" and "negative" training sets have 20 $\times 2 \times$ 50 training patterns. The test sets are formed by selecting four images from each person. There are 20 $\times$ 4 test patterns from known person database and 40 $\times$ 4 patterns from intruder database. If a test pattern is from known person database but is rejected by the network, then the pattern is *falsely rejected*. On the other hand, if the pattern is from intruder database but is misrecognized to one of the person in the known person database, then this pattern is *falsely accepted*.

We have compared the performance of probabilistic DBNN, traditional DBNN [22] and multilayer perceptron MLP. MLP is trained by backpropagation algorithm. We have chosen the number of clusters (or hidden neurons) which can generate the best performance. There are two clusters for each subnet in either probabilistic or traditional DBNN, and there are 30 hidden neurons in MLP. The experiment results are as follows. For the sake of comparison, we adjust the thresholds so that all three networks reach 0% false rejection rate. The false acceptance rates are shown in Table IV. Notice that since there are only 20 people in the known person database, the misclassification rates of all the three models are 0%. As mentioned in Section IV-B, likelihood density generates more locally conserved decision regions than posterior probability. Therefore, the probabilistic DBNN can achieve lower false acceptance rate than the traditional DBNN and MLP, whose outputs can be proven to converge to posterior probabilities [30], [31]. We also note that using negative training examples can significantly improve false acceptance rate. This effect can be observed in all of the three networks.

V. HIERARCHICAL FACE RECOGNITION SYSTEM

Yet another face database containing images of 66 persons, called IM66, is used for the experiment. There are more variations in this database than in SCR 80 $\times$ 20 and SCR 40 $\times$ 150. The database has, for each person, images of five different head orientations, four illumination conditions, and three expressions (smile, frown, and surprised; each expression has two images). There are a total of $4 \times 5 + 3 \times 2 =$ 26 images per person. Fig. 11 illustrates the variation of those facial images. By applying the PDBNN face recognizer on this database, we found that although PDBNN can handle size, orientation, and expression variations, it does not work well under lighting variation. The final performance is shown in the first column of Table V. In order to increase the discriminant power of the face recognition system, a hierarchical PDBNN recognition system is proposed in this section. This system cascades two processing stages, as shown in Fig. 10. A *face verifier* stage is cascaded after the (original) *face recognizer* stage. Possible candidates of the second hierarchy sensor are hairline and mouth. In this system we applied forehead and hairline region for the face verifier. The face verifier is itself another PDBNN classifier. Its function is to verify or reject the decision of the primary recognizer. The verifier can be trained by the decision-based learning rule. The input vector is a 12 $\times$ 8 feature vector obtained by down-sampling the forehead and hairline region of the face image. Since the hairline (forehead) region is smoother than the eye-nose region, it is easier to normalize the lighting effect on this area. Therefore the influence of the lighting variation on final recognition result can be reduced with the presence of the hairline verifier.

The verification scheme is described as follows: We first construct several "similarity lists" for the verification process. Every class has its own similarity list. We make the lists after the PDBNN face verifier is trained. The list's length varies from person to person. Initially, the similarity list of class j contains only class j itself. If a training pattern (originally from class j) is classified into another class by verifier, say, class k, then the class k will be added to the similarity list of class j. This process repeats until all the training patterns in the known person database are tested.

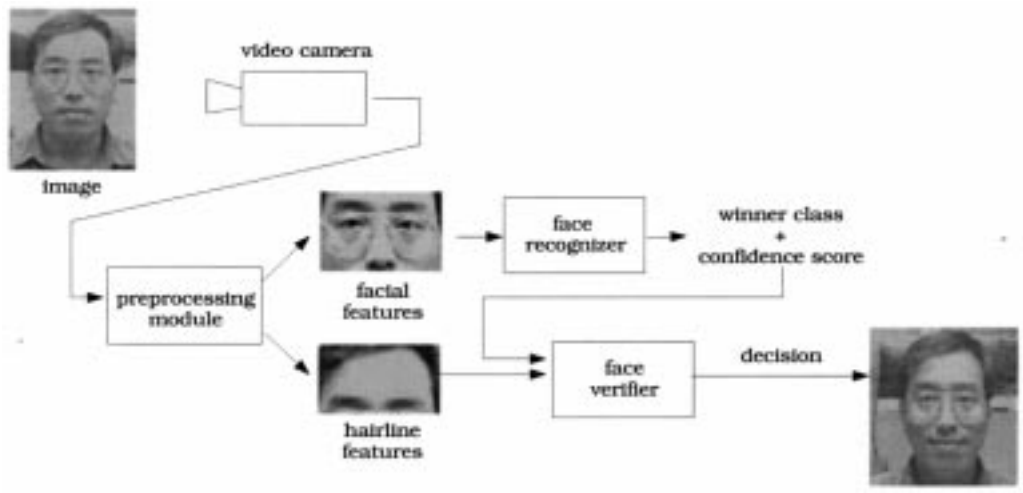


Fig. 10. Hierarchical Information Processing System based on PDBNN. Primary features are from the facial region, while the hairline features are used as supporting verifier.

Fig. 11. Variations in IM66 face database. First row: faces of different persons. Second row: lighting variation. Third row: orientation variation. Fourth row: expression variation.

When the similarity lists are available, we use the following rules to verify the result of face recognizer.

- If the highest confidence score of the face recognizer is from subnet i and it is below the recognizer threshold, the input pattern is recognized as class i if the highest score of the face verifier is also from subnet i *and* its confidence score exceeds the threshold of the verifier. Otherwise this pattern will be rejected.
- If the highest confidence score of the face recognizer is from subnet i and it is above the recognizer threshold, the input pattern is recognized as class i if the highest score of the face verifier is from one of the classes on the

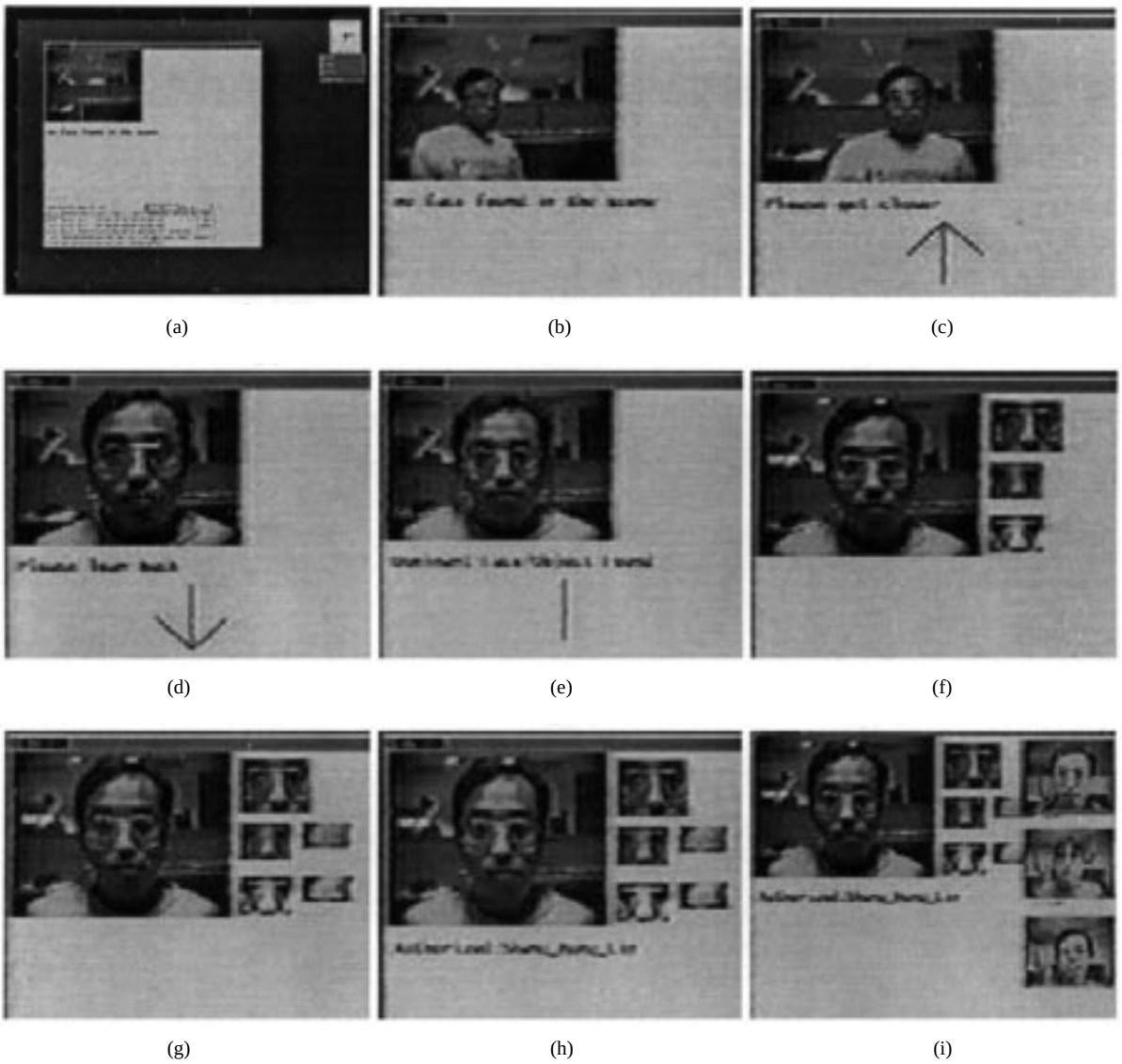


Fig. 12. Live sequence of face recognition system. (a) X window interface. The image on upper-left corner is acquired from a video camera in real time. (b) A person comes into the camera range (time index: 0 s). (c) The system detects the presence of a face. However, the face in the image is too small. The system then asks the person to stand closer. (text on picture: "Please step closer") (time index: 0.33 s) (d) The person stands too close to the camera. The system asks the person to step back. (text on picture: "Please lean back") (time index: 4.7 s) (e) The system locates the face. ("text on picture: "Dominant Face/Object Found") (time index: 5 s) (f) Facial features are extracted. (time index: 6 s) (g) Hairline feature is extracted. (time index: 6.33 s) (h) The person is recognized. ("text on picture: "Authorized: Shang Hung Lin") (time index: 9.1 s) Note: Special sound effects (a doorbell sound and a human voice calling the name of the authorized person) are executed between time 6.33 s and time 9.1 s. The actual recognition time takes less than half second. (i) The pictures of the three persons most similar to the person under test. (time index: 9.2 s).

similarity list of class i *and* its confidence score exceeds the threshold of the verifier. Otherwise this pattern will be rejected.

- We have built a 38-class hierarchical PDBNN face recognition system. This system can successfully recognize the 38 persons in the IM66 database (the positive dataset), and reject the remaining 28 who are considered as "intruders."

The training scheme is as follows. Since the feature dimension is very high (13×9 for face recognizer, 12×8 for face verifier), EBF is recommended for both face recognizer and face verifier as discriminant function. And for the same reason, we have adopted the k-mean algorithm for LU learning. For both face recognizer and face verifier, we have used ten images per person in the 38-person group to form the training set, and the rest 16 images for the test set. We have also used ORL face database from UK (40 persons) to serve as "intruder training patterns" (for face recognizer only). The images of the other

TABLE V
PERFORMANCE OF HIERARCHICAL FACE RECOGNITION SYSTEM. THE RATES ON THE FIRST THREE ROWS ARE CALCULATED FROM THE KNOWN PERSON DATA SET, SO THEY SUM UP TO 100% IN EACH COLUMN. THE FALSE ACCEPTANCE RATES, WHICH ARE IN THE FOURTH ROW, ARE OBTAINED FROM THE INTRUDER DATA SET

	w/o face verifier	w/ face verifier
Recognition	92.65%	97.75%
False rejection	7.29%	2.25%
Misclassification	0.06%	0%
False acceptance	9.35%	0%

28 persons in the IM66 database are used as "intruder test patterns."

The recognition result of the face recognizer is summarized in Table V. Notice that since the recognition rate is the percentage of the patterns in the known people database that are correctly recognized by the system, the summation of recognition, false rejection and misclassification rates amount to 100%. The false acceptance rate is the only number that is computed using the patterns in the unknown database. In this experiment, we can see that with face recognizer along, the performance is worse than that on SCR 80 $\times$ 20. This is not surprising because the variation in IM66 database is larger than that in SCR 80 $\times$ 20. With the help of face verifier, the hierarchical PDBNN system successfully rejects all the intruder test patterns. For the test patterns in the 38-person database, PDBNN tends to reject doubtful patterns rather than to misrecognize them to wrong classes. This is very desirable for high security applications. Fig. 12 shows several frames from a live sequence, demonstrating the performance of this system with respect to both recognition accuracy and processing speed.

VI. Concluding Remarks

An automatic face recognition system is proposed in this paper. This system performs human face detection, eye localization, and face recognition in close-to-real-time speed. The PDBNN, a probabilistic variant of the decision based neural network, is applied to implement the major modules of this system. This modular neural network deploys one subnet to take care of one object class, and therefore it is able to approximate the decision region of each class locally and efficiently. This locality property is attractive especially for biometric identification applications (face recognition, for example). Moreover, because its discriminant function obeys probability constraint, PDBNN has more nice properties such as low false acceptance and false rejection rates.

Acknowledgment

The authors wish to gratefully acknowledge Drs. M. Y. Chiu, M. Fang of Siemens Corporate Research, Inc., and Mr. R. Bang of Princeton University for their invaluable contributions.

References

[1] S.-H. Lin, S. Y. Kung, and L.-J. Lin, "A probabilistic DBNN with applications to sensor fusion and object recognition," in *Proc. 5th IEEE Wkshp. Neural Networks Signal Processing*, Cambridge, MA, Aug 1995, pp. 333–342.

[2] S.-H. Lin, S. Y. Kung, and M. Fang, "A neural network approach for face/palm recognition," in *Proc. 5th IEEE Wkshp. Neural Networks Signal Processing*, Cambridge, MA, Aug 1995, pp. 323–332.

[3] B. Miller, "Vital signs of identity," *IEEE Spectrum*, pp. 22–30, Feb. 1994.

[4] P. Sinha, "Object recognition via image invariants: a case study," in *Investigative Ophthalmology Visual Sci.*, May 1994, pp. 1735–1740.

[5] M. Bichsel, "Strategies of robust objects recognition for automatic identification of human faces," Ph.D. dissertation, ETH, Zürich, Switzerland, 1991.

[6] R. Brunelli and T. Poggio, "Face recognition: features versus templates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1042–1052, 1993.

[7] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp. 84–91, June 1994.

[8] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object detection," in *Proc. 5th Int. Conf. Comput. Vision*, Cambridge, MA, June 1995.

[9] M. Fang, A. Singh, and M-Y. Chiu, "A fast method for eye localization," Siemens Corporate Res., Tech. Rep. SCR-94-TR-488, 1994.

[10] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: a survey," *Proc. EEE*, vol. 83, no. 5, May 1995.

[11] M. D. Kelly, "Visual identification of people by computer," Stanford AI Project, Stanford, CA, Tech. Rep. AI-130, 1970.

[12] I. J. Cox, J. Ghosn, and P. N. Yianilos, "Feature-based face recognition using mixture distance," NEC Res. Instit., Princeton, NJ, Tech. Rep. 95-09, 1995.

[13] S. Akamatsu, T. Sasaki, H. Fukamachi, and Y. Suenaga, "A robust face identification scheme-KL expansion of an invariant feature space," *SPIE Proc.: Intell. Robots Comput. Vision X: Algorithms Technol.*, vol. 1607, 1991, pp. 71–84.

[14] Y. Cheng, K. Liu, J. Yang, and H. Wang, "A robust algebraic method for face recognition," in *Proc. 11th Int. Conf. Pattern Recognition*, 1992, pp. 221–224.

[15] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, pp. 71–86, 1991.

[16] M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, and C. von der Malsburg, "Distortion invariant object recognition in dynamic link architecture," *IEEE Trans. Comput.*, vol. 42, pp. 300–311, 1993.

[17] B. A. Golomb and T. J. Sejnowski, "SEXNET: a neural network identifies sex from human faces," in *Advances in Neural Information Proceedings Systems 3*, D. S. Touretzky and R. P. Lipmann, Eds. San Mateo, CA: Morgan Kaufmann, 1991.

[18] A. Rahardja, A. Sowmya, and W. Wilson, "A neural network approach to component versus holistic recognition of facial expressions in images," *SPIE Proc.: Intell. Robots Comput. Vision X: Algorithms Technol.*, vol. 1607, 1991, pp. 62–70.

[19] K. K. Sung and T. Poggio, "Learning human face detection in cluttered scenes," in *Computer Analysis of Image and Patterns*. 1995, pp. 432–439.

[20] H. A. Rowley, S. Baluja, and T. Kanade, "Human face detection in visual scenes," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-95-158R, 1995.

[21] J. Weng, T. S. Huang, and N. Ahuja, "Learning recognition and segmentation of 3D objects from 2-D images," in *Proc. IEEE Int. Conf. Comput Vision*, 1993, pp. 121–128.

[22] S. Y. Kung and J. S. Taur, "Decision-based neural networks with signal/image classification applications," *IEEE Trans. Neural Networks*, vol. 6, pp. 170–181, Jan 1995.

[23] S. Y. Kung, M. Fang, S. P. Liou, and J. S. Taur, "Decision-based neural network for face recognition system," in *Proc. 1995 IEEE Int. Conf. Image Processing*, Washington, D.C., vol. I, Oct 1995, pp. 430–433.

[24] S.-H. Lin and S. Y. Kung, "Probabilistic DBNN via expectation-maximization with multisensor classification applications," in *Proc. 1995 IEEE Int. Conf. Image Processing*, Washington, D.C., vol. III, Oct 1995, pp. 236–239.

[25] S.-H. Lin, Y. Chan, and S. Y. Kung, "A probabilistic decision-based neural network for locating deformable objects and its applications to surveillance system and video browsing," in *tProc. Int. Conf. Acoust., Speech. Signal Processing*, Atlanta, GA, May 1996.

[26] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, pp. 1481–1497, Sept. 1989.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," in *J. Roy. Statist. Soc., B39*, 1976, pp. 1–38.

[28] L. Xu, M. I. Jordan, and G. E. Hinton, "A modified gating network for the mixture of experts architecture," in *Proc. World Congr. Neural Networks*, San Diego, CA, 1994, pp. II405–II410.

[29] A. S. Pandya and R. B. Macy, *Pattern Recognition with Neural Networks in C++*. Boca Raton, FL: CRC Press, 1996.

[30] M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate bayesian *a posteriori* probabilities," *Neural Computa.*, vol. 3, pp. 461–483, 1991.

[31] D. W. Ruck, S. K. Rogers, and *et al*, "The multilayer perceptron as an approximation to a bayes optimal discriminant function," *IEEE Trans. Neural Networks*, vol. 1, pp. 296–298, Dec 1990.

[32] P. J. Phillips, "Matching pursuit filters applied to face identification," in *SPIE Proc.: Automat. Syst. Identification Inspection of Humans*, vol. 2277, 1994, pp. 2–11.

[33] F. S. Samaria and A. C. Harter, "Parameterization of a stochastic model for human face identification," in *Proc. IEEE Wkshp. Applicat. Comput. Vision*, Sarasota, FL, 1994.
[34] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural network approach," NEC Res. Inst., Princeton, NJ, Tech. Rep. 1995.
[35] F. S. Saramia, "Face recognition using hidden Markov model," Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., 1994.

Shang-Hung Lin received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C. in 1991, and the M.S. and Ph.D. degrees in electrical engineering from Princeton University, NJ, in 1994 and 1996, respectively.

He is currently working with Epson Palo Alto Laboratory, Palo Alto, CA. His primary research interests include neural networks, pattern recognition, computer vision, and image processing.

Sun-Yuan Kung (S'74–M'78–SM'84–F'88) received the Ph.D. degree in electrical engineering from Stanford University, CA.

In 1974, he was an Associate Engineer of Amdahl Corporation, Sunnyvale, CA. From 1977 to 1987, he was a Professor of Electrical Engineering-Systems of the University of Southern California, L.A. In 1984, he was a Visiting Professor of the Stanford University and the Delft University of Technology. In 1994 he was a Toshiba Chair Professor at Waseda University, Tokyo, Japan. Since 1987, he has been a Professor of Electrical Engineering at the Princeton University. His research interests include spectrum estimations, digital signal/image processing, VLSI array processors, and neural networks. He has authored more than 300 technical publications. He authored three books, *VLSI Array Processors*, (Englewood Cliffs, NJ: Prentice-Hall, 1988) (with Russian and Chinese translations). He also authored two neural network books, *Digital Neural Networks*, (Englewood Cliffs, NJ: Preentice–Hall, 1993), and *Principal Component Neural Networks*, (New York: Wiley, 1996). He has also edited or coedited numerous reference and proceeding books, including "VLSI and Modern Signal Processing," (Englewood Cliffs, NJ: Prentice-Hall, 1985) (with Russian translation), "VLSI Signal Processing," Vols. I and II, (Piscataway, NJ: IEEE Press, 19XX), "Neural Networks for Signal Processing, Vols. I–III, (Piscataway, NJ: IEEE Press, 19XX), and "Systolic Arrays," (Washington, D.C.: IEEE Computer Society Press, 1988), and "Application-Specific Array Processors, (Washington, D.C.: IEEE Computer Society Press, 1991).

Dr. Kung has served as an Editor-in-Chief of the *Journal of VLSI Signal Processing* since 1990. He is now an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS. He was appointed as the first Associate Editor in the VLSI area in 1984 and as the first Associate Editor in the neural network area in 1991 of the IEEE TRANSACTIONS ON SIGNAL PROCESSING. He served as a Member of IEEE Signal Processing Society's Administration Committee from 1989 to 1991. He is presently serving on Technical Committees on VLSI signal processing and on neural networks. He has served as a Founding Member and General Chairman of various IEEE Conferences, including IEEE Workshops on VLSI Signal Processing in 1982 and 1986, the International Conference on Application Specific Array Processors in 1990 and 1991, and IEEE Workshops on Neural Networks and Signal Processing in 1991 and 1992. He received the 1992 IEEE Signal Processing Society's Technical Achievement Award for his contributions on "parallel processing and neural network algorithms for signal processing." He was appointed as an IEEE-SP Distinguished Lecturer in 1994. He was the Keynote Speaker for the First International Conference on Systolic Arrays, Oxford, 1986, and International Symposium on Computer Architecture and DSP in 1989, IEEE Workshops on VLSI Signal Processing in 1995, and IEEE Workshops on Neural Networks and Signal Processing in 1996.

Long-Ji Lin received the B.S. degree in 1983 and the M.S. degree in 1985, both in electrical engineering, from National Taiwan University. He received the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, in 1992.

Currently he works at Siemens Corporate Research in Princeton, NJ. His research interests include neural networks, image processing, and mobile robots.