Neural Networks 69 (2015) 111–125

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Optimized face recognition algorithm using radial basis function neural networks and its practical applications

Sung-Hoon Yoo [a], Sung-Kwun Oh [a,*], Witold Pedrycz [b,c,d]

[a] Department of Electrical Engineering, The University of Suwon, San 2-2 Wau-ri, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 445-743, South Korea

[b] Department of Electrical & Computer Engineering, University of Alberta, Edmonton T6R 2V4 AB, Canada

[c] Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah, 21589, Saudi Arabia

[d] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

ARTICLE INFO

Article history:
Received 14 July 2014
Received in revised form 14 May 2015
Accepted 17 May 2015
Available online 5 June 2015

Keywords:
P-RBF NNs (Polynomial based Radial Basis Function Neural Networks)
PCA (Principal Component Analysis)
ASM (Active Shape Model)
FCM (Fuzzy C-means Method)
DE (Differential Evolution)

ABSTRACT

In this study, we propose a hybrid method of face recognition by using face region information extracted from the detected face region. In the preprocessing part, we develop a hybrid approach based on the Active Shape Model (ASM) and the Principal Component Analysis (PCA) algorithm. At this step, we use a CCD (Charge Coupled Device) camera to acquire a facial image by using AdaBoost and then Histogram Equalization (HE) is employed to improve the quality of the image. ASM extracts the face contour and image shape to produce a personal profile. Then we use a PCA method to reduce dimensionality of face images. In the recognition part, we consider the improved Radial Basis Function Neural Networks (RBF NNs) to identify a unique pattern associated with each person. The proposed RBF NN architecture consists of three functional modules realizing the condition phase, the conclusion phase, and the inference phase completed with the help of fuzzy rules coming in the standard 'if-then' format. In the formation of the condition part of the fuzzy rules, the input space is partitioned with the use of Fuzzy C-Means (FCM) clustering. In the conclusion part of the fuzzy rules, the connections (weights) of the RBF NNs are represented by four kinds of polynomials such as constant, linear, quadratic, and reduced quadratic. The values of the coefficients are determined by running a gradient descent method. The output of the RBF NNs model is obtained by running a fuzzy inference method. The essential design parameters of the network (including learning rate, momentum coefficient and fuzzification coefficient used by the FCM) are optimized by means of Differential Evolution (DE). The proposed P-RBF NNs (Polynomial based RBF NNs) are applied to facial recognition and its performance is quantified from the viewpoint of the output performance and recognition rate.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Biometrics delivers technologies that identify individuals by measuring physical or behavioral characteristics of humans. A password or PIN (Personal Identification Number) type recently used is the means of personal authentication that requires to be memorized. They could be compromised relatively easily. In more advanced biometric scenarios, memorization is not required (Chellappa, Wilson, & Sirohey, 1995). The existing face recognition algorithms were studied by using 2D image. Besides the face, local eye or template matching-based methods were used. The issues of overhead of computing time as well as the memory requirements were raised that concerned an acquisition of image data or the ensuing learning. PCA transformation that enables to decrease processing time by reducing the dimensionality of the data has been proposed to solve such a problem. Boehnen and Russ (2005) used facial color present in 2D images while Colombo, Cusano, and Schettini (2005) used curvature, position and shape of the face. Colbry, Stockman, and Jain (2005) and Lu and Jain (2005) generated a statistical model of the eyes, nose and mouth and eyes. Recently, the more effective applications were supported by the use of ASM (Cootes, Cooper, Taylor, & Graham, 1995).

Most 2D feature-based biometrics algorithms typically require high quality images in order to achieve high performance (Mohammed, Minhas, Jonathan Wu, & Sid-Ahmed, 2011). Therefore in order to properly extract the feature candidates of face image, all unnecessary information existing in the face image must be

* Corresponding author. Tel.: +82 31 229 6342; fax: +82 31 220 2667.
E-mail addresses: shyoo@suwon.ac.kr (S.-H. Yoo), ohsk@suwon.ac.kr (S.-K. Oh), wpedrycz@ualberta.ca (W. Pedrycz).

http://dx.doi.org/10.1016/j.neunet.2015.05.001
0893-6080/© 2015 Elsevier Ltd. All rights reserved.

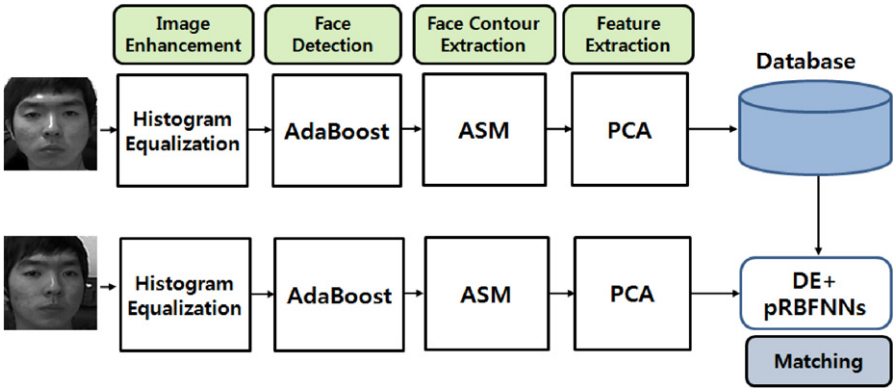


Fig. 1. An overall architecture of the face recognition system.

removed. The facial feature image can be thus obtained after removing the unnecessary features by augmenting the "conventional" recognition system by the ASM process. That is, facial features are extracted by removing the background and obstacles from input images by using the ASM.

In the sequel, face extraction and recognition are carried out by running a series of algorithms such as ASM, PCA, and DE-based P-RBF NNs. The main objective of this study is to improve the face recognition rate by handling the image of enhanced facial features through the multi-dimensional data preprocessing technologies (PCA combined with ASM) and DE-based P-RBF NNs. We realize this objective within the context of a practical recognition environment. Along with this new design approach we provide a thorough comparative analysis contrasting the performance of the recognition scheme proposed here with the performance of other approaches existing in the literature.

This paper is organized as follows: in Section 2, we introduce the proposed overall face recognition system and the design method. Section 3, Histogram equalization, AdaBoost, ASM, and PCA forming the preprocessing part of face recognition. Optimization techniques and a design method of a pattern classifier for face recognition are covered in Section 4. In Section 5, we analyze the performance of the proposed system by using input images data coming from a CCD camera. Finally, the conclusions are covered in Section 6.

2. Structure of face recognition system

In this section, an overall structure of proposed face recognition system and design method is described. Face recognition system is constructed to data preprocessing and RBF NNs pattern classifier applying optimization techniques. Histogram equalization, AdaBoost, ASM, and PCA realized a phase of data preprocessing. Histogram equalization compensates for illumination-distortion of images. The face area is detected by using AdaBoost. By using the ASM, face information is extracted and eventual obstacles are removed. The features of face images are extracted with the use of the PCA method. The RBF NN pattern classifier is constructed. The classifier comprises three functional modules. These modules realize the condition, conclusion and aggregation phase. The input space of the condition phase is represented by fuzzy sets formed by running the FCM clustering algorithm (Tsekourasa, Sarimveisb, Kavaklia, & Bafasb, 2005). The conclusion phase involves a certain polynomial. The output of the network is determined by running fuzzy inference. The proposed RBF NNs come with the fuzzy inference mechanism constructed with the use of the fuzzy rule-based network. In the construction of the classifier, Differential Evolution (DE) is used to optimize the momentum coefficient, learning rate, and the fuzzification coefficient used in the FCM algorithm. Fig. 1 portrays an overall architecture of the system.

3. Data reprocessing for facial feature extraction

In this section, we briefly elaborate on the histogram equalization, AdaBoost algorithm, ASM and PCA as they are being used at the phase of data preprocessing.

3.1. Histogram equalization

Histogram equalization (HE) is a commonly used technique for enhancing image contrast (Gonzalez & Woods, 2002).

Consider a facial image W with N pixels, and a total number of k gray levels, e.g., 256 gray levels in the dynamic range of $[0, L-1]$. The generic idea is to map the gray levels based on the probability distribution of the image input gray levels. For a given image W, the probability density function (PDF) of $PDF(W_k)$ is defined as

$$PDF(W_k) = n_k/n. \tag{1}$$

For $k = 0, 1, \ldots, L-1$, where n_k represents the number of times that the level, W_k appears in the input image W, and n is the total number of samples in the input image. Note that $PDF(W_k)$ is associated with the histogram of the input image which represents the number of pixels that have a specific input intensity X_k. A plot of n_k versus W_k is known as the histogram of image $W(I, J)$. Based on the PDF, the cumulative density function (CDF) is defined as

$$CDF(w) = \sum_{j=0}^{k} PDF(W_j) \tag{2}$$

where $W_k = w$, for $k = 0, 1, \ldots, L-1$. By definition, $PDF(W_{L-1}) = 1$. Through histogram equalization we map the input image into the entire dynamic range (W_0, W_{L-1}).

Fig. 2(a) and (b) show a face image along with its equivalent histogram. The output image produced after histogram equalization is given in Fig. 2(c) and (d). This result demonstrates the performance of the histogram equalization method in enhancing the contrast of an image through dynamic range expansion.

3.2. AdaBoost-based face detection

The purpose of face detection is to locate faces present in still images. This has long been a focus of computer vision research and has achieved a great deal of success (Gao, Pan, Ji, & Yang, 2012; Lopez-Molina, Baets, Bustince, Sanz, & Barrenechea, 2013; Rowley, Baluja, & Kanade, 1998; Sung & Poggio, 1998; Viola & Jones, 2004). Comprehensive reviews are given by Yang, Kriegman, and Ahuja (2002), and Zhao, Chellappa, and Phillips (2003). Among face detection algorithms, the AdaBoost (Freund & Schapire, 1995) based method proposed by Viola and Jones (2001) has gained great popularity due to a high detection rate, low complexity,

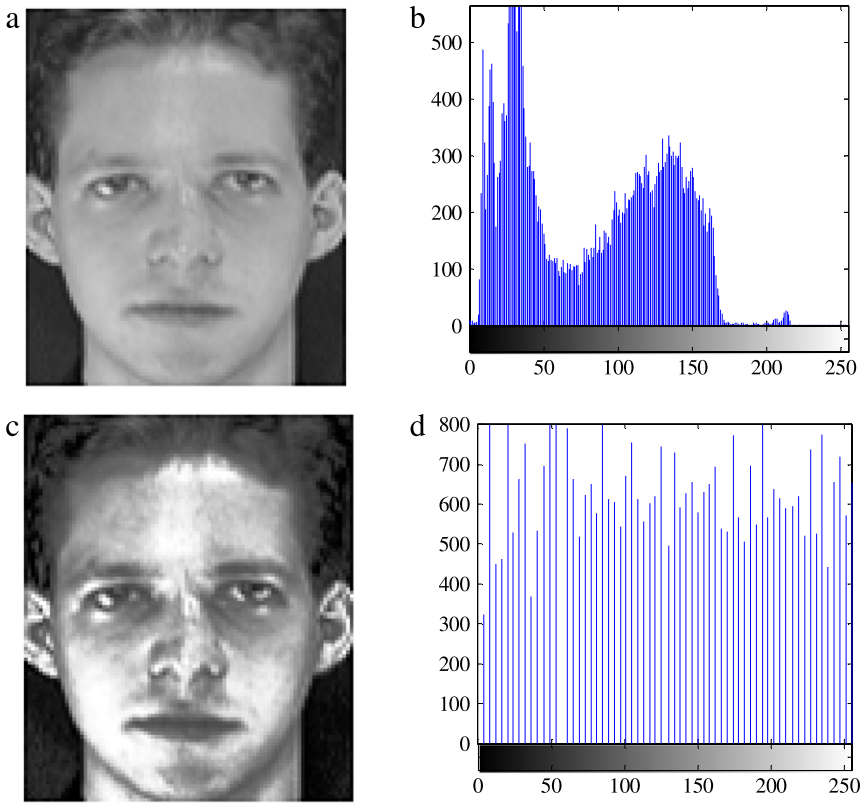


Fig. 2. Face images and histogram.

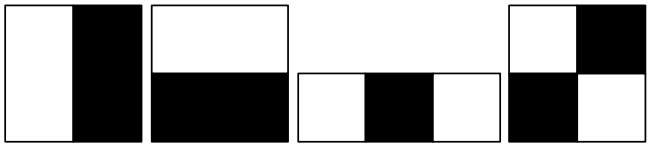


Fig. 3. Rectangle feature of Haar-like.

and solid theoretical foundations. The high speed of AdaBoost method is mainly due to the use of simple Haar-like features and a cascaded classifier structure, which excludes quickly most of the image window hypotheses. Fig. 3 shows several types of Haar-like features.

At a preprocessing stage, an auxiliary image I_i, called the integral image or summed-area table (Lienhart & Maydt, 2002) is formed on a basis of the original image I_0, where the value $I_i(i, j)$ is the sum of pixels above or to the left of position (i, j) in I_0. Using I_i, the sum of pixel intensities in I_0 any rectangle in can be calculated in constant time. Afterwards, each stage classifier was trained using the AdaBoost.

AdaBoost constructs the strong classifier as a combination of weak classifier with proper coefficients. This is an iterative learning process. Each candidate image window w, at all positions and all scales, is fed into a cascaded classifier. At each stage, the classifier response $h(w)$ comes as the sum of a series of feature responses $h_j(w)$ as described below

$$h(w) = \sum_{j=1}^{n_i} h_j(w) = \begin{cases} \alpha_{j1} & f_j(w) < t_j \\ \alpha_{j2} & \text{otherwise} \end{cases} \tag{3}$$

where $f_j(w)$ is the feature response of the jth Haar feature and α_{j1} and α_{j2} are the feature weight coefficients. If $h(w)$ is lower than a threshold t, the candidate window w is regarded as non-face and thrown away, otherwise it is sent to the next classifier. Multiple detections for a single face are pruned by non-maximum suppression realized at the last step. Fig. 4 shows the cascade of classifier with N stages. Each classifier in cascade AdaBoost detector works independently, and the minimum true positive and the maximum false alarm rates of these stages are the same. Only these sub-windows accepted as true positive by all stages of the detector are regarded as targets. 'T' means that true candidates of these sub-windows passed the verification of each classifier, and 'F' means that these false candidates are rejected by the corresponding classifier.

3.3. Face shape extraction using ASM

ASM is a commonly used technique for facial feature extraction. This method is similar to the Active Contour Model, or snakes (Huang, Hzu, & Cheng, 2010), but exhibits some advantage such that the instances of an ASM can only deform in the ways found in its training set. ASM also allows for a considerable level of variability in shape modeling, but the model is specific to the class of target objects or structures that it intends to represent.

3.3.1. The shape model

A shape model is described by n landmark points that represent the important positions in the object to be represented. These points are generated based on a set of training shapes. Each training shape $\boldsymbol{x}$ is represented as a shape vector, which is a collection of landmark points called a point distribution model (Wang, Xie, Zhu, Yang, & Zheng, 2013),

$$x = (x_0, y_0, x_1, y_1, \ldots, x_k, y_k, \ldots, x_{n-1}, y_{n-1})^T \tag{4}$$

where T denotes the transpose operation, and (x_k, y_k) are the coordinates of the kth landmark point.

Fig. 5 shows a training image with its landmark points being marked.

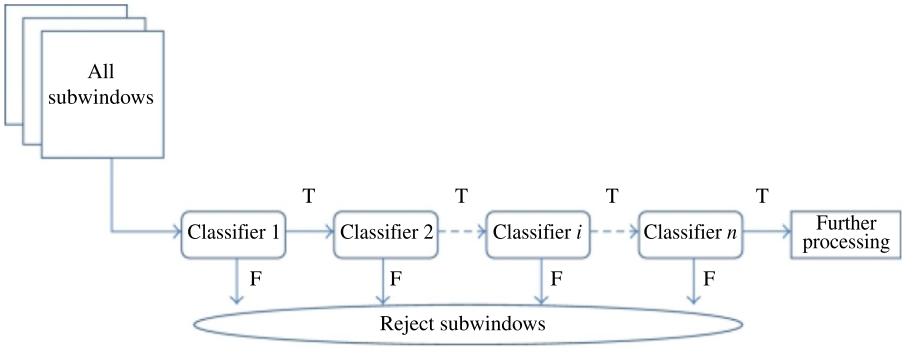


Fig. 4. Flowchart of the cascade AdaBoost detector.

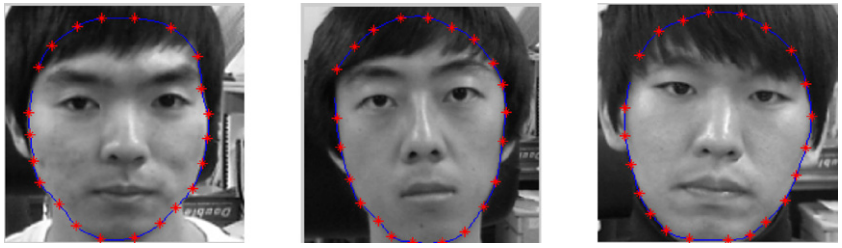


Fig. 5. Boundary of the face area.

The training shapes are all aligned by translation, rotation and scaling for minimizing the sum of squared distances between their corresponding landmark points. Then, the mean shape $\bar{x}$ and the deviation of each training shape from the mean are calculated. Principal component analysis (PCA) is then applied to capture most of the shape variations. Therefore, a shape model can be approximated as follows:

$$x \approx \bar{x} + Pb \tag{5}$$

where the $P = (p_1, p_2, \ldots, p_t)$ is the matrix whose columns are the first t eigenvectors with the largest eigenvalues arranged in a descending order, and $b = (b_1, b_1, \ldots, b_t)^T$ is a weight vector for the t eigenvectors, referred to as the shape parameters. When fitting the shape model to an object, the value of b_i is constrained to lie within the range ± 3 standard deviations. This can ensure that this range of the shape parameters can represent most of the shape variations in the training set. The number of eigenvectors t to be used is determined such that the eigenvectors can represent a certain amount of the shape variations in the training shapes, usually ranging from 90% to 95%. The desired number of eigenvectors t is given as the smallest t which satisfies the following condition

$$\sum_{i=1}^{t} \lambda_i \geq 0.95 \sum_{i=1}^{N} \lambda_i \tag{6}$$

where N is the overall number of eigenvectors available.

3.3.2. Modeling the gray-level appearance

The gray-level appearance (Cootes, Taylor, Lanitis, Cooper, & Graham, 1993), which describes the local texture feature around each landmark, is the normalized derivative of the profiles sampled perpendicular to the landmark contour and centered at the landmark. This gray-level information is used to estimate the best position of the landmarks in the searching process. The normalized derivative of the profiles is invariant to the offsets of the gray levels. The gray-level profile, g_{ij}, of the landmark j in the image i is a $(2n + 1)$-D vector, in which n pixels are sampled on either side of the landmark under consideration,

$$g_{ij} = [g_{ij0}, g_{ij1}, \ldots, g_{ij(2n-1)}] \tag{7}$$

where g_{ij}, $k = 0, \ldots, 2n + 1$, is the gray-level intensity of a corresponding pixel. The derivative profile of g_{ij} of length of $2n$ is given as follows:

$$dg_{ij} = [g_{ij1} - g_{ij0}, g_{ij2} - g_{ij1}, \ldots, g_{ij(2n+1)} - g_{ij(2n)}]. \tag{8}$$

The normalized derivative profile is given by

$$y_{ij} = \frac{dg_{ij}}{\sum_{k=0}^{2n} |dg_{ijk}|} \tag{9}$$

where $dg_{ijk} = g_{ij(k+1)} - g_{ijk}$. The covariance matrix of the normalized derivative profile for N training images comes in the form

$$C_{yj} = \frac{1}{N} \sum_{i=1}^{N} (y_{ij} - \bar{y}_j)(y_{ij} - \bar{y}_j) \tag{10}$$

where $\bar{y}_j$ is the mean profile. The ASM employs the information coming as a result of modeling the gray-level statistics around each landmark to determine the desired movement or adjustment of each landmark such that a face shape model can fit into the target object accurately. To determine the movement of a landmark, a search profile, which is a line passing through the landmark under consideration and perpendicular to the contour formed by the landmark and its neighbors is extracted. A number of sub-profiles will be generated when the best set of shape parameters is being searched. These sub-profiles are matched to the corresponding profiles obtained from the training set. The difference between a sub-profile y and the training profile is computed using the Mahalanobis distance:

$$f(y) = (y - \bar{y}_j)^T C_{yj} (y - \bar{y}_j). \tag{11}$$

Minimizing $f(y)$ is equivalent to the maximization of the probability of y matched to $\bar{y}$ according to a Gaussian distribution. Fig. 6 shows each movement of singular points of model point (y) and nearest edge $(\bar{y})$ by the ASM.

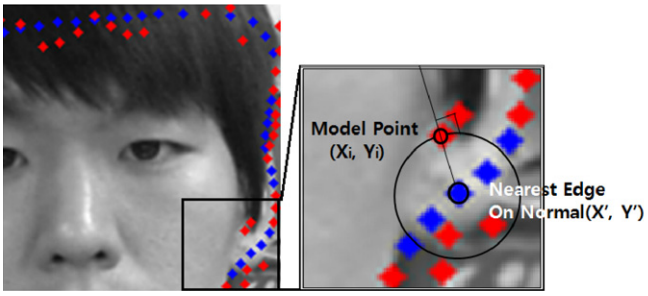


Fig. 6. Movement of singular point.

3.3.3. The optimization algorithm

Modifying the point distribution model (PDM) to fit the object in an image is an iterative optimization process. Starting from the mean shape representation of the model, each point of the model is allowed to move dynamically until it fits the object. At each model point or landmark, a profile perpendicular to the contour is extracted and a new and better position of that point is estimated along this profile. Different approaches (Yong, Zhang, Xiaoguang, & Zhang, 2000) can be used to search for a better position for the points. The simplest way is to find the strongest edge along the searching profile. Another approach is to create the gray-level appearance model or profile of each point, which will maximize the probability of the gray-level profile, as described in the last section. After searching, the shape parameters $b = (b_1, b_1, \ldots, b_t)^T$ and the pose parameters (i.e., rotation, scale, and translation of the model) are adjusted in such a way as to minimize the overall distance between the new position of the points and the position of the original points. The adjustment process is repeated until no significant changes in the model points have been observed.

3.4. Feature extraction using PCA

Principal component analysis is a standard technique used in statistical pattern recognition and signal processing for data reduction and feature extraction. As the pattern often contains redundant information, we map it to the feature space of lower dimensionality (Mohammed et al., 2011).

A face image of size $N \times N$ *pixels* can be considered as a one-dimensional vector of dimensionality N^2. For example, face image from the AT&T (formerly the ORL database of faces) database of size 112×92 can be considered as a vector of dimension 10,304, or equivalently points in a 10,304 dimensional space. An ensemble of images maps to a collection of points in this highly dimensional space. The main idea of the principal component is to find the vectors that best account for the distribution of face images within the entire image space. Because these vectors are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are face like in appearance, we refer to them as 'eigenfaces'.

Let the training set of face images be $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$, the average face of the set is expressed in the form in (12).

Fig. 7 shows the average face coming from the AT&T database.

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n. \tag{12}$$

Each face differs from the average by the vector

$$\Phi_i = \Gamma_i - \Psi. \tag{13}$$

This set of highly-dimensional vectors is then subject to principal component analysis, which seeks for a set of M orthonormal vectors, U_m, which the best describe the distribution of the data.

Fig. 7. Average face of AT&T database.

The kth vector, U_k, is selected in a way such that the expression

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(U_k^T\Phi_n)^2 \tag{14}$$

attains maximum, subject to the constraint

$$U_l^T U_k = \delta_{lk} = \begin{cases} 1, & \text{if } l = k \\ 0, & \text{otherwise} \end{cases}. \tag{15}$$

The vectors U_k are the eigenvectors of the covariance matrix

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T. \tag{16}$$

The covariance matrix C, however, is an $N \times N$ real symmetric matrix therefore calculating the N^2 eigenvectors and eigenvalues is an intractable task for image of typical sizes. Therefore, if we use the approximate equation derived from (14), the resultant covariance matrix C will be of dimensionality $N^2 \times N^2$. This result makes the task of determining eigenvectors and eigenvalues intractable and computationally unfeasible. Recall that $A = [\Phi_1, \Phi_2, \ldots, \Phi_M]$, the matrix multiplication of A^TA results in an $M \times M$ matrix. Since M is the number of faces in the database, the eigenvectors analysis is reduced 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).

Consider the eigenvectors v_i of A^TA such that

$$A^TAv_i = \mu_i v_i. \tag{17}$$

Pre-multiplying both sides by A and using (14), we obtain

$$AA^TAv_i = \mu_i Av_i. \tag{18}$$

We see that Av_is are the eigenvectors and μ_i s are the eigenvalues of $C = AA^T$.

Following the analysis shown above, we construct the $M \times M$ matrix $L = A^TA$, where $L_{mn} = \Phi_m^T\Phi_n$, and find the M eigenvectors, v_i, of L. These vectors determine linear combinations of the M training set face images to form the eigenfaces U_l. Fig. 8 shows the eigenface of the PCA.

$$U_l = \sum_{k=1}^{M} v_{lk}\Phi_k, \quad l = 1, \ldots, M. \tag{19}$$

With this analysis, the calculations are greatly reduced, 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, the training set of face images will be relatively small ($M \ll N^2$), and the calculations become quite manageable. The associated eigenvalues allow us to rank the eigenvectors according to their usefulness

Fig. 8. Eigenface.

when characterizing the variation among the images. The eigenface images calculated from the eigenvectors of L span a basis set that can be used to describe face images (Slavkovic & Jevtic, 2012). We evaluated a limited version of this framework on an ensemble of 115 images ($M = 115$) images of Caucasian males digitized in a controlled manner, and found that 40 eigenfaces ($M' = 40$) were sufficient to realize a very good description of face images. In practice, a lower value of M' can be sufficient for identification, since accurate reconstruction of the image is not an absolute requirement. In the framework of face recognition, we are concerned with a pattern recognition task rather than image reconstruction. The eigenfaces span an M' dimensional subspace of the original N^2 image space and hence, the M' significant eigenvectors of the L matrix with the largest associated eigenvalues, are sufficient for reliable representation of the faces in the face space characterized by the eigenfaces.

Then a new face image (Γ) is transformed into its eigenface components (projected onto "face space") in the following manner

$$w_k = U_k^T(\Gamma - \Psi) \tag{20}$$

for $k = 1, \ldots, M'$. The weights form a projection vector $\Omega^{\mathrm{T}} = [w_1, w_2, \ldots, w_{M'}]$ describing the contribution of each eigenface in the representation of the input face image, treating the eigenfaces as a basis set for face images. The projection vector is then used in a standard pattern recognition algorithm to identify which out of the number of predefined face classes, if any, describes the face to the highest extent.

4. Designing pattern classifier using radial basis function neural networks

In this section, we outline a design of fuzzy inference mechanism-based Radial Basis Function Neural Networks. A network structure of the proposed RBF NNs is divided into three modules involving the condition, conclusion, and inference phases. The input space of condition phases is divided by using FCM and Local Area of conclusion phases is represented to polynomial function. The final output of network is obtained through fuzzy inference of inference phases. Performance of proposed RBF NNs is improved by generating nonlinear discriminant function in the output space owing to fuzzy inference mechanism of polynomial-based structure (Balasubramanian, Palanivel, & Ramalingam, 2009; Connolly, Granger, & Sabourin, 2012; Han, Chen, & Qiao, 2011; Oh, Kim, Pedrycz, & Park, 2014; Park, Oh, & Kim, 2008).

4.1. Architecture of polynomial-based RBF networks

The general architecture of the radial basis function neural networks (RBF NNs) consists of the three layers as shown in Fig. 9.

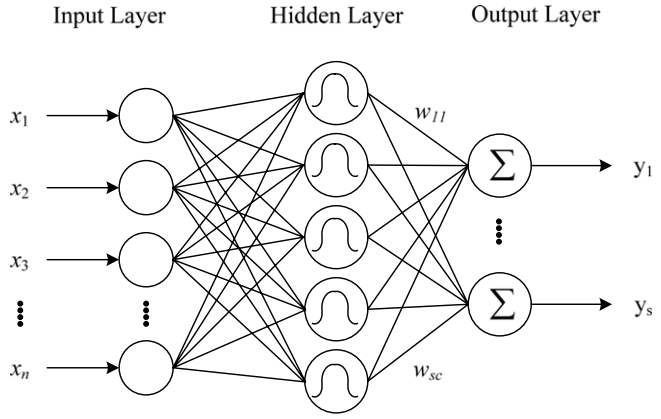


Fig. 9. General architecture of RBF neural networks.

The network exhibits a single hidden layer. Each node in the hidden layer determines a level of activation of the receptive field (radial basis function) $\Theta(\mathbf{x})$ given some input $\mathbf{x}$. The jth output $y_j(\mathbf{x})$ is a weighted linear combination of the activation levels of the receptive fields:

$$y_j(\mathbf{x}) = \sum_{i=1}^{c} w_{ji}\Theta_i(\mathbf{x}) \tag{21}$$

$j = 1, \ldots, s$ where s stands for the number of outputs (being equal to the number of classes encountered in a given classification problem). In the case of the Gaussian type of RBFs, we have

$$\Theta_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{v}_i\|^2}{2\sigma_i^2}\right) \tag{22}$$

where $\mathbf{x}$ is the n-dimensional input vector $[x_1, \ldots, x_n]^{\mathrm{T}}$, and $\mathbf{v}_i = [v_{i1}, \ldots, v_{in}]^{\mathrm{T}}$ is the center of the ith basis function $\Theta_i(\mathbf{x})$ while c is the number of the nodes in the hidden layer. Typically the distance $\|.\|$ used in (22) is the Euclidean one (Staiano, Tagliaferri, & Pedrycz, 2006).

The proposed P-RBF NNs (Polynomial based RBF NNs) exhibit a similar topology as the one encountered in RBF NNs. However the functionality and the associated design process exhibit several evident differences. In particular, the receptive fields do not assume any explicit functional form (say, Gaussian, ellipsoidal, etc.), but are directly reflective of the nature of the data and come as the result of fuzzy clustering. Given the prototypes formed by the FCM method, the receptive fields are described in the following way

$$\Theta_i(\mathbf{x}) = A_i(\mathbf{x}) = \frac{1}{\sum_{j=1}^{c}\left(\frac{\|\mathbf{x}-\mathbf{v}_i\|^2}{\|\mathbf{x}-\mathbf{v}_j\|^2}\right)}. \tag{23}$$

In addition, the weights between the output layer and the hidden layer are not single numeric values but come in the form of polynomials of the input variables (hence the term of functional links used in this architecture)

$$w_{ji} = f_{ji}(\mathbf{x}). \tag{24}$$

The neuron located at the output layer realizes a linear combination of the activation levels of the corresponding receptive fields hence (21) can be rewritten as follows,

$$y_j(\mathbf{x}) = \sum_{i=1}^{c} f_{ji}(\mathbf{x})A_i(\mathbf{x}). \tag{25}$$

The above structure of the classifier can be represented through a collection of fuzzy rules.

$$\text{If } \mathbf{x} \text{ is } A_i, \text{ then } f_{ji}(\mathbf{x}) \tag{26}$$

where the fuzzy set A_i is the i-cluster (membership function) of the ith fuzzy rule, $f_{ji}(\mathbf{x})$ is a polynomial function generalizing a numeric weight used in the standard form of the RBF NNs, and c is the number of fuzzy rules (clusters), and $j = 1, \ldots, s$.

4.2. Three processing phase of polynomial-based radial basis function neural networks

The proposed P-RBF NNs are implemented by realizing three processing phases that is, condition, conclusion and aggregation phases. Condition and conclusion phases relate to the formation of the fuzzy rules and their ensuing analysis. Aggregation phase is concerned with a fuzzy inference (mapping procedure).

4.2.1. Condition phase of networks

The condition phase of the P-RBF NNs is formed by means of the Fuzzy C-Means. In this section, we briefly review the objective function-based fuzzy clustering with intent of highlighting its key features pertinent to the architecture of the network (Roh, Oh, & Pedrycz, 2010). The FCM algorithm is aimed at the formation of 'c' fuzzy sets (relations) in $\mathbf{R}^n$. The objective function Q guiding the clustering is expressed as a sum of the distances of individual data from the prototypes $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots,$ and $\boldsymbol{v}_c$,

$$Q = \sum_{i=1}^{c} \sum_{k=1}^{N} u_{ik}^m \|\mathbf{x}_k - \mathbf{V}_i\|^2. \tag{27}$$

Here, $\|\ \|$ denotes a certain distance function; 'm' stands for a fuzzification factor (coefficient), $m > 1.0$. N is the number of patterns (data). The resulting partition matrix is denoted by $U = [u_{ik}]$, $i = 1, 2, \ldots, c$; $k = 1, 2, \ldots, N$. While there is a substantial diversity as far as distance functions are concerned, here we adhere to a weighted Euclidean distance taking on the following form

$$\|\mathbf{x}_k - \mathbf{V}_i\|^2 = \sum_{j=1}^{n} \frac{(x_{kj} - v_{ij})^2}{\sigma_j^2} \tag{28}$$

with σ_j being a standard deviation of the jth variable. While not being computationally demanding, this type of distance is still quite flexible and commonly used.

Consider the set X consisting of N patterns $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_k \in \mathbf{R}^n$, $1 \le k \le N$. In clustering we assign patterns $\mathbf{x}_k \in X$ into c clusters, which are represented by its prototypes $\mathbf{v}_i \in \mathbf{R}^n$, $1 \le i \le c$. The assignment to individual clusters is expressed in terms of the partition matrix $U = [u_{ik}]$ where

$$\sum_{i=1}^{c} u_{ik} = 1, \quad 1 \le k \le N \tag{29}$$

and

$$0 < \sum_{k=1}^{N} u_{ik} < N, \quad 1 \le i \le c. \tag{30}$$

The minimization of Q is realized in successive iterations by adjusting both the prototypes and the partition matrix, that is min Q $(U, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c)$. The corresponding formulas used in an iterative fashion read as follows

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_k - \mathbf{v}_i\|}{\|\mathbf{x}_k - \mathbf{v}_j\|} \right)^{\frac{2}{m-1}}}, \quad 1 \le k \le N,\ 1 \le i \le c \tag{31}$$

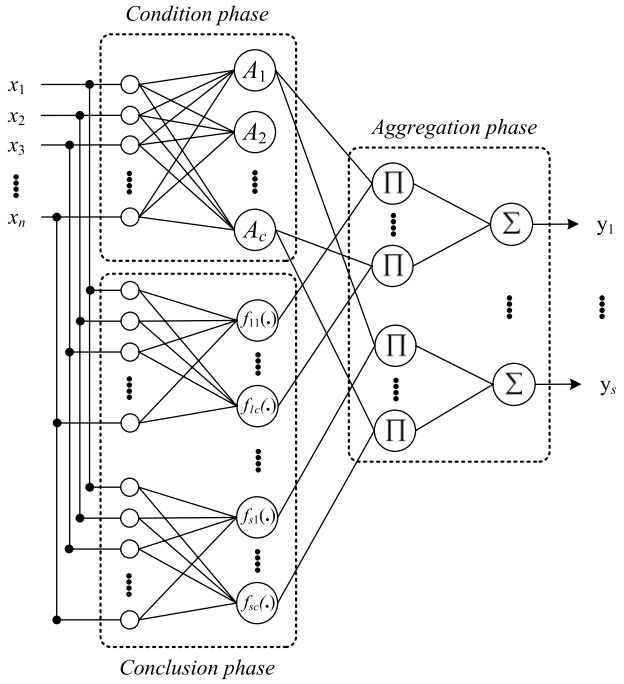


Fig. 10. Topology of P-RBF NNs exhibiting three functional modules of condition, conclusion and aggregation phases.

and

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^m}, \quad 1 \le i \le c. \tag{32}$$

The properties of the optimization algorithm are well documented in the literature, cf. Bezdek (1981). In the context of our investigations, we note that the resulting partition matrix produces 'c' fuzzy relations (multivariable fuzzy sets) with the membership functions $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_c$ forming the corresponding rows of the partition matrix U, that is $U = [\boldsymbol{u}_1^{\mathrm{T}} \quad \boldsymbol{u}_2^{\mathrm{T}} \quad \ldots \quad \boldsymbol{u}_c^{\mathrm{T}}]$. From the design standpoint, there are several essential parameters of the FCM that impacts the usage of the produced results. These parameters concern the number of clusters, the values of the fuzzification coefficient and a form of the distance function. The fuzzification coefficient exhibits a significant impact on the form (shape) of the developed clusters. The commonly used value of m is equal to 2. Lower values of the fuzzification coefficient produce more Boolean-like shapes of the fuzzy sets where the regions of intermediate membership values are very much reduced. When we increase the values of "m" above 2, the resulting membership functions start to become "spiky" with the values close to 1 in a very close vicinity of the prototypes. We anticipate that the adjustment of the values of m will substantially impact the performance of the network.

4.2.2. Conclusion phase of the network

Polynomial functions are dealt with in the conclusion phase. For convenience, we omit the suffix j from the original notation $f_{ji}(\mathbf{x})$ shown in Fig. 10 and described by (26). Several classes of polynomials are worth noting

$$\text{Constant; } f_i(\mathbf{x}) = a_{i0} \tag{33}$$

$$\text{Linear; } f_i(\mathbf{x}) = a_{i0} + \sum_{j=1}^{n} a_{ij} x_j \tag{34}$$

$$\text{Quadratic; } f_i(\mathbf{x}) = a_{i0} + \sum_{j=1}^{n} a_{ij}x_j + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk}x_j x_k. \tag{35}$$

These functions are activated by the corresponding entries of the partition matrix and lead to local regression models located at the condition phase of the individual rules.

In case of the quadratic function, the dimensionality of the problem increases quite quickly especially when dealing with problems of high dimensionality. The reduced quadratic function is also discussed with intent of reducing computational burden.

$$\text{Reduced Quadratic; } f_i(\mathbf{x}) = a_{i0} + \sum_{j=1}^{n} a_{ij}x_j + \sum_{k=1}^{n} a_{ijk}x_k^2. \tag{36}$$

The use of some constant in (33) (which are special cases of the polynomial functions) reduces the P-RBF NNs to the standard RBF neural networks as illustrated in Fig. 9.

4.2.3. Aggregation phase of networks

Let us consider the P-RBF NNs structure by considering the fuzzy partition realized in terms of FCM as shown in Fig. 10. The node denoted by Π is realized as a product of the corresponding fuzzy set and the polynomial function. The family of fuzzy sets A_i forms a partition (so that the sum of membership grades sum up to one at each point of the input space). The "$\sum$" neuron realizes a sum as shown in (25). The output of P-RBF NNs can be obtained by following a standard inference mechanism used in rule-based systems (Oh, Pedrycz, & Park, 2004),

$$y_j = g_j(\mathbf{x}) = \sum_{i=1}^{c} \frac{u_i f_{ji}(\mathbf{x})}{\sum_{k=1}^{c} u_k} = \sum_{i=1}^{c} u_i f_{ji}(\mathbf{x}) \tag{37}$$

where, $u_i = A_i(\mathbf{x})$. All the entries sum up to 1 as indicated by (9). $g_j(\mathbf{x})$ describes here the discriminant function for discerning jth class.

Based on the local polynomial-like representation, the global characteristics of the P-RBF NNs result through the composition of their local relationships.

4.3. The discriminant function polynomial-based radial basis function neural networks classifiers

There are many different ways to describe pattern classifiers. One of the most useful ways is the one realized in terms of a set of discriminant functions $g_i(\mathbf{x})$, $i = 1, \ldots, m$ (where m stands for the number of classes). The classifier is said to assign an input vector $\boldsymbol{x}$ to class ω_i if

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \text{for all } j \neq i. \tag{38}$$

Thus, the classifiers are viewed as networks that compute m discriminant functions and select the category corresponding to the largest value produced by these functions.

In this paper, the proposed P-RBF NNs classifier is used for two-class or multi-class problems. If a classification problem is multi-class one, then we use (38) as the discriminant function, otherwise, we consider the following decision rule defined commonly as a single discriminant function $g(\mathbf{x})$ in a two-class problem.

$$\text{Decide } \omega_1 \text{ if } g(\boldsymbol{x}) > 0; \text{ otherwise decide } \omega_2. \tag{39}$$

The final output of networks, (37) is used as a discriminant function $g(\boldsymbol{x})$ and can be rewritten in a form of the linear combination

$$g(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{fx} \tag{40}$$

where $\boldsymbol{a}$ is a vector of coefficients of polynomial functions used in the conclusion phase of the rules in (33)–(36) and $\mathbf{fx}$ is a matrix of u. These can be defined for each of polynomials as follows.

(i) Constant;

$$\boldsymbol{a}^{\mathrm{T}} = [a_{10}, \ldots, a_{c0}], \qquad \mathbf{fx} = [u_1, \ldots, u_c]^{\mathrm{T}}.$$

(ii) Linear;

$$\boldsymbol{a}^{\mathrm{T}} = [a_{10}, \ldots, a_{c0}, a_{11}, \ldots, a_{c1}, \ldots, a_{cn}]$$
$$\boldsymbol{fx} = [u_1, \ldots, u_c, u_1x_1, \ldots, u_cx_1, \ldots, u_cx_n]^{\mathrm{T}}.$$

(iii) Quadratic;

$$\boldsymbol{a}^{\mathrm{T}} = [a_{10}, \ldots, a_{c0}, a_{11}, \ldots, a_{c1}, \ldots, a_{cn}, \ldots, a_{cnn}]$$
$$\boldsymbol{fx} = [u_1, \ldots, u_c, u_1x_1, \ldots, u_cx_1, \ldots, u_cx_n, \ldots, u_cx_nx_n]^{\mathrm{T}}.$$

(iv) Reduced quadratic;

$$\boldsymbol{a}^{\mathrm{T}} = [a_{10}, \ldots, a_{c0}, a_{11}, \ldots, a_{c1}, \ldots, a_{cn}, \ldots, a_{cnn}]$$
$$\boldsymbol{fx} = [u_1, \ldots, u_c, u_1x_1, \ldots, u_cx_1, \ldots, u_cx_n, \ldots, u_cx_n^2]^{\mathrm{T}}.$$

For the discriminant function coming in the form of (40), a two-class classifier implements the decision rule expressed by (39). Namely, $\mathbf{x}$ is assigned to ω_1 if the inner product $\boldsymbol{a}^T\boldsymbol{fx}$ is greater than zero and to ω_2 otherwise. The equation $g(\boldsymbol{x}) = 0$ defines the decision surface that separates the two classes. Otherwise, a multi-class classifier implements the decision rule expressed by (38). Each output node generates a discriminant function corresponding to each class. If the ith output is larger than all remaining outputs, the pattern $\boldsymbol{x}$ is assigned to ith class.

4.4. Optimized process of P-RBF NNs by using differential evolution

The algorithm of Differential Evolution (DE) (Dervis & Selcuk, 2004; Storn, 1997) introduced by Storn (1997) is a parallel direct search method, which utilizes NP parameter vectors as a population for each generation G. DE can be categorized into a class of floating-point encoded, evolutionary optimization algorithms. Currently, there are several variants of DE. The particular variant used throughout this investigation is the DE/rand/1/bin scheme. This scheme will be discussed here and more detailed descriptions are provided. Since the DE algorithm was originally designed to work with continuous variables, the optimization of continuous problems is discussed first.

The DE algorithm is a population-based algorithm using three operators: crossover, mutation, and selection. Several optimization parameters require tuning. These parameters are put together under the common name of control parameters. In fact, there are only three real control parameters of the algorithm, namely differentiation (or mutation) constant F, crossover constant CR, and size of population NP. The rest of the parameters are the dimension of problem D that scales the difficulty of the optimization task; maximum number of generations (iterations) GEN, which may serve as a stopping condition; and the boundary constraints imposed on the variables that limit the feasible area.

Regarding DE variants Price et al. present a notation to identify different ways to generate new vectors on DE.

The most popular of them is called *DE/rand/1/bin*, where the first term means differential evolution, the second term indicates how the base vector is chosen (in this case, vector chosen at random). The number in the third term expresses how many vector differences will contribute in the differential mutation (one pair in this case). Finally, the fourth term shows the type of crossover utilized (bin from binomial in this variant).

The DE algorithm can be outlined shortly as the following sequence of steps:

Step 1: Set up the values of the parameters of the method such as a crossover rate (CR), scaling factor (SF), and mutation type (one out of the 5 types available) and then randomly generate "NP" population in search space. Each variable (or vector) in the n-dimensional search space will be denoted by the

$D_i(t) = [x_1(t), x_2(t), \ldots, x_n(t)]$ and the population $\mathrm{NP}(t) = \{D_1(t), D_2(t), \ldots, D_s(t)\}$ is composed of the elements $D(t)$. Evaluate each individual using the objective function.

Step 2: Perform mutation to generate a mutant vector $D_{mutant}(t+1)$. For the target vector, randomly choose distinct vectors $D_a(t), D_b(t), D_c(t)$, etc. ($a, b, c \in \{1, 2, \ldots, \mathrm{NP}\}$).

There are five types of mutation:

$$DE/Rand/1/\beta : D_{mutant}(t+1) = D_c(t) + \beta(D_a(t) - D_b(t)) \quad (41)$$

$$DE/Best/1/\beta : D_{mutant}(t+1) = D_{best}(t) + \beta(D_a(t) - D_b(t)) \quad (42)$$

$$DE/Rand/2/\beta : D_{mutant}(t+1) = D_e(t) + \beta(D_a(t) - D_b(t) - D_c(t) - D_d(t)) \quad (43)$$

$$DE/best/2/\beta : D_{mutant}(t+1) = D_{best}(t) + \beta(D_a(t) - D_b(t) - D_c(t) - D_d(t)) \quad (44)$$

$$DE/RandToBest/1 : D_{mutant}(t+1) = D_c(t) + \beta(D_{best}(t) - D_c(t)) + \beta(D_a(t) - D_b(t)). \quad (45)$$

Produce a mutant vector using one of the mutation methods shown above. Generally, *DE/R and/1/β* is used.

Step 3: Perform crossover to obtain a trial vector for each target vector using its mutant vector in the following equation:

$$D^j_{trial}(t+1) = \begin{cases} D^j_{mutant}(t+1) & \text{if}(rand < CR) \text{ or } j = j^{rand\ index} \\ D^j_{target}(t). & \text{Otherwise} \end{cases} \quad (46)$$

$j = 1, 2, \ldots, n$; r and $\in [0, 1]$ is a random number drawn from a uniform distribution over [0, 1]; CR is the crossover rate $\in [0, 1]$; and $j^{rand\ index} \in (1, 2, \ldots, n)$ is randomly selected index.

Step 4: Evaluate the trial vectors $D_{trial}(t+1)$. If a trial vector comes with the better fitness than that of individual in the current generation, it is transferred to the next generation.

Step 5: Until the termination criterion has been satisfied, repeat Steps 2–4.

In this study, essential design parameters are optimized *by DE/rand/1/bin* method.

4.4.1. Initialization

As with all evolutionary optimization algorithms, DE works with a population of solutions, rather than a single solution. The population P at generation G contains NP solution vectors called individuals of the population where each vector represents a potential solution for the optimization problem

$$P^{(G)} = X_i^{(G)} = x_i^{(G)}, \quad i = 1, \ldots, NP;\ j = 1, \ldots, D;\ G = 1, \ldots, G_{\max}. \quad (47)$$

In order to form a starting point for optimum seeking, the population must be initialized. Often there is no specific knowledge available about the location of a global optimum. Typically, we might have knowledge about the boundaries of the problem's variables. In this case, a natural way to initialize the population $P^{(0)}$ (initial population) is to seed it with random values within the given boundary constraints:

$$P^{(0)} = x_{j,i}^{(0)} = x_j^{(L)} + rand_j[0, 1] \times \left(x_j^{(U)} - x_j^{(L)}\right) \quad \forall i \in [1, NP];\ \forall j \in [1, D], \quad (48)$$

where $rand_j[0, 1]$ represents a uniformly distributed random variable assuming values in [0, 1].

4.4.2. Mutation

The self-referential population recombination scheme of DE is different from the other evolutionary algorithms. From the first generation onward, the population of the subsequent generation $P^{(G+1)}$ is obtained on the basis of the current population $P^{(G)}$. First a temporary or trial population of candidate vectors for the subsequent generation, $P^{(G+1)} = V^{(G+1)} = v_{j,i}^{(G+1)}$, is generated as follows:

$$v_{j,i}^{(G+1)} = \begin{cases} x_{j,r3}^{(G)} + F \times (x_{j,r1}^{(G)} - x_{j,r2}^{(G)}), & rand_j[0, 1] < CR \vee j = k, \\ x_{ij}^{(G)}, & \text{otherwise}, \end{cases} \quad (49)$$

where $i \in [1, \mathrm{NP}]$; $j \in [1, D]$, $r1, r2, r3 \in [1, \mathrm{NP}]$, randomly selected, except for $r1 \neq r2 \neq r3 \neq i$, $k = (\mathrm{int}(\mathrm{rand}_i[0, 1] \times D) + 1)$, and $\mathrm{CR} \in [0, 1]$, $F \in (0, 1]$.

Three randomly chosen indexes, $r1$, $r2$, and $r3$ refer to three randomly chosen vectors of the population. They are different from each other and also different from the running index i. New random values for $r1$, $r2$, and $r3$ are assigned for each value of index i (for each vector). A new value for the random number rand[0, 1] is assigned for each value of index j (for each vector parameter).

4.4.3. Crossover

The index k refers to a randomly chosen vector of parameter and it is used to ensure that at least one vector parameter of each individual trial vector $V^{(G+1)}$ differs from its counterpart present in the previous generation $X^{(G)}$. A new random integer value is assigned to k for each value of the index i (prior to construction of each trial vector). F and CR are control parameters of DE. Both values remain constant during the search process. Both values as well as the third control parameter, NP (population size), remain constant during the search process. F is a real-valued factor in range [0.0, 1.0] that controls the amplification of differential variations. CR is a real-valued crossover factor taking on the values in the range [0.0, 1.0] that controls the probability that a trial vector will be selected from the randomly chosen, mutated vector, $V_{j,i}^{(G+1)}$ instead of from the current vector, $x_{j,i}^{(G)}$. Generally, both F and CR affect the convergence rate and the robustness of the search process. Their optimal values are dependent both on the characteristics of the objective function and on the population size, NP. Usually, suitable values for F, CR and NP can be found through experimentation after a number of tests using different values. Practical guidelines on how to select control parameters NP, F and CR can be found in Storn (1997).

4.4.4. Selection

The selection scheme of DE differs from the one encountered in other evolutionary algorithms. On the basis of the current population $P^{(G)}$ and the temporary population $P'^{(G+1)}$, the population of the next generation $P^{(G+1)}$ is formed as follows:

$$X_i^{(G+1)} = \left\{ \begin{matrix} V_i^{(G+1)}, & \text{if } \Im(V_i^{(G+1)}) \leq \Im(X_i^{(G)}) \\ X_i^{(G+1)}, & \text{otherwise} \end{matrix} \right\}. \quad (50)$$

Thus each individual of the temporary or trial population is compared with its counterpart in the current population. The one with the lower value of cost-function $\Im(X)$ to be minimized will propagate to the population of the next generation. As a result, all the individuals of the next generation are better than their counterparts in the current generation. The interesting point concerning the DE selection scheme is that a trial vector is only compared to one individual vector, not to all the vectors in the current population.

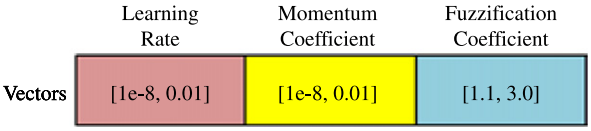


Fig. 11. A content of the vector of parameters.

4.4.5. Boundary constraints

It is important to notice that the recombination operation of DE is able to extend the search outside of the initialized range of the search space (refer to (4.2) and (4.3)). It is also worthwhile to notice that sometimes it is a beneficial property in problems with no boundary constraints because it is possible to find the optimum that is located outside of the initialized range. However, in the boundary-constrained problems, it is essential to ensure that parameter values lie inside their allowed ranges after recombination. A simple way to guarantee this is to replace the parameter values that violate boundary constraints with random values generated within the feasible range:

$$u_{j,i}^{(G+1)} = \begin{cases} x_j^{(L)} + rand_j[0,1] \times (x_j^{(U)} - x_j^{(L)}), \\ \quad \text{if } u_{j,i}^{(G+1)} < x_j^{(L)} \vee u_{j,i}^{(G+1)} > x_j^{(U)} \\ u_{i,j}^{(G+1)}, \quad \text{otherwise} \end{cases} \tag{51}$$

where $i \in [1, \text{NP}]$; $j \in [1, D]$.

This is the method used in this work. Another simple but less efficient method is to reproduce the boundary constraint violating values according to relationship (51) as many times as necessary to satisfy the boundary constraints. Yet another simple method that allows bounds to be approached asymptotically while minimizing the amount of disruption that results from resetting the bound values is expressed as follows

$$u_{j,i}^{(G+1)} = \begin{cases} (x_{j,i}^{(G)} + x_j^{(L)})/2, & \text{if } u_{j,i}^{(g+1)} < x_j^{(L)}, \\ (x_{j,i}^{(G)} + x_j^{(U)})/2, & \text{if } u_{j,i}^{(g+1)} < x_j^{(U)}, \\ u_{j,i}^{(G+1)}, & \text{otherwise.} \end{cases} \tag{52}$$

Through the optimization of these parameters such as learning rate, momentum coefficient and fuzzification coefficient by using DE, P-RBF NNs structure exhibits better convergence properties in the generation process of the networks from the viewpoint of performance. Fig. 11 shows a content of the vectors used in the optimization of the P-RBF NNs.

Individual particles of the P-RBF NNs include entries that represent optimized learning rate, momentum, and fuzzification coefficient. The learning rate and momentum of vectors are applied to optimize the connections (weights). The fuzzification coefficient changes the shape of the membership functions produced by the fuzzy C-means clustering. The value of the membership function depends on the center point and the fuzzification coefficient to be adjusted.

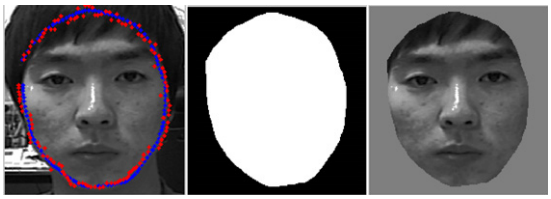


Fig. 13. Face contour extraction.

5. Application to the design of face recognition system

The proposed face recognition system is designed by using P-RBF NNs-based pattern recognition scheme. Histogram equalization, AdaBoost, ASM and PCA are used to data preprocessing (as discussed in Section 3). First, the image obtained from the camera is converted to a gray scale and its dimensionality is reduced by using the PCA (Section 3.4). The PCA-reduced data are then used as the inputs for the P-RBF NNs based classification scheme. Fig. 12 shows a flow of processing realized by the overall classification system.

Color images of 640 × 480 size are converted to gray images. The distorted images are improved through histogram equalization. We extracted images including the face area to squares each of the $N \times N$ size. After the completion of the extraction phase, a personal profile consists of the extracted face contour and the shape obtained by ASM. Finally, face images are stored in the JPG format where the images are of size 200 × 200 pixels. A Face area is extracted using the ASM that helps removing disturbances and reflecting the morphological characteristics of individuals faces. Fig. 13 visualizes the process of extracting the face contour from images using the ASM. A new face shape consists of by finding edge to the vertical direction in the boundary of the face shape. The process of face contour extraction is dependent on the size and position of the face shape. An incorrect initial position of the shape takes a long time to find a face and may also lead to incorrect results.

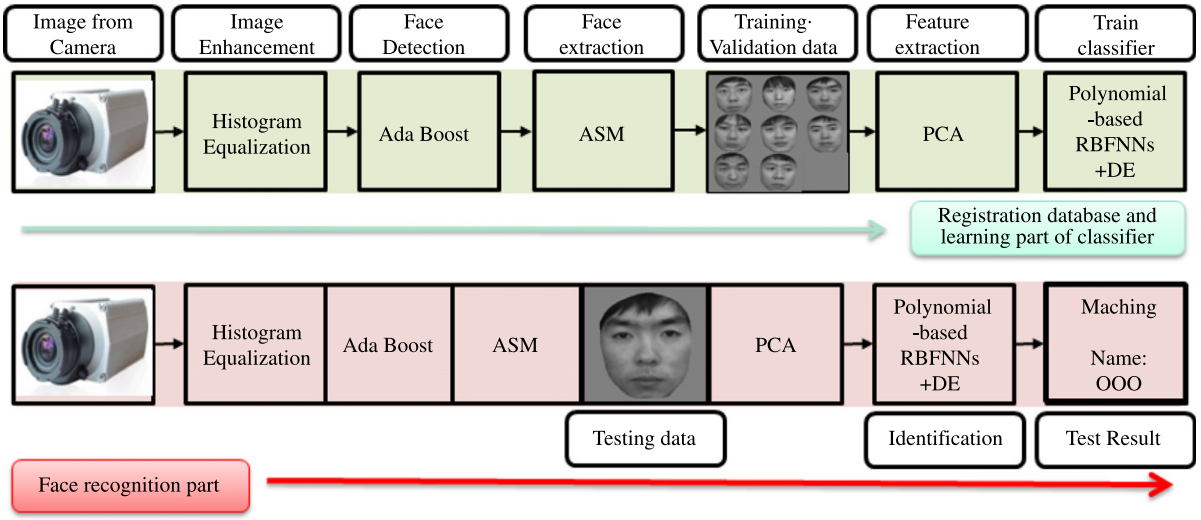


Fig. 12. An overall processing scheme of the classification system.

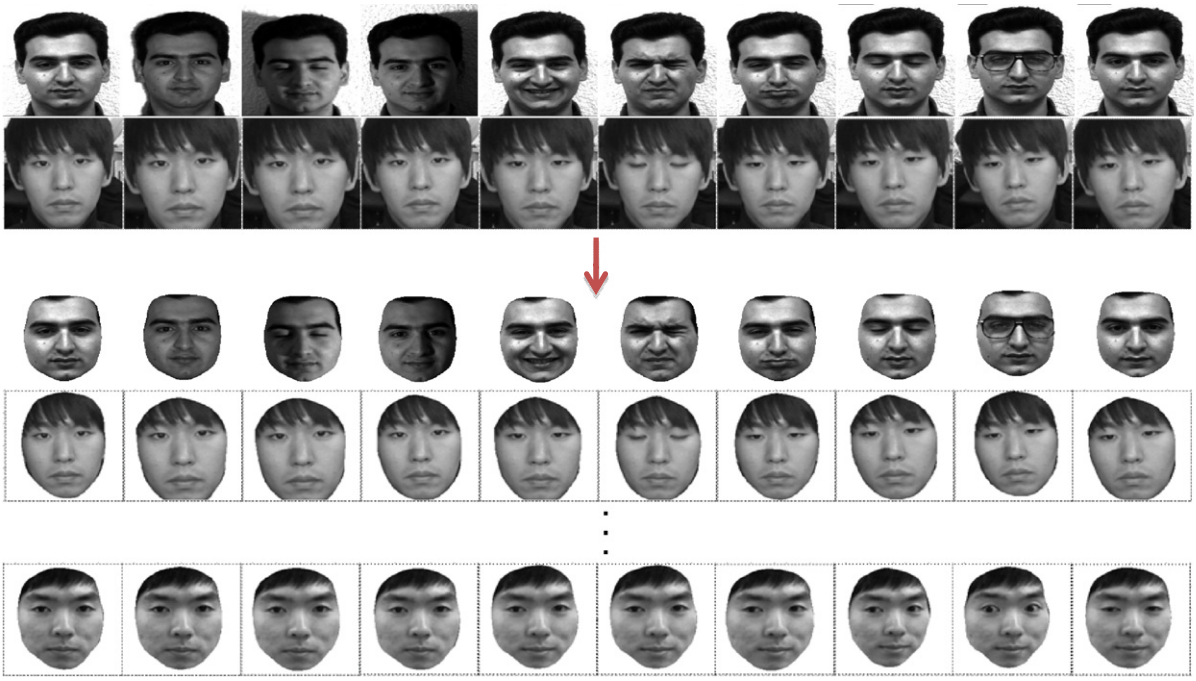


Fig. 14. Example of facial image dataset used in the experiment.

Fig. 15. Average face obtained after running the PCA.

The dataset of faces obtained in this way consists of 350 images with 10 images per person (including in total 35 persons). Fig. 14 provides an example of the extracted facial image dataset.

The weights of new facial database are calculated through running the PCA algorithm. Fig. 15 presents an average face of 15 persons obtained in this way.

Fig. 16 shows reconstructed eigenfaces in face space by extracting eigenvectors corresponding to the largest eigenvalues coming from the overall 350 eigenvectors.

The PCA weights associated with each candidate image are used to form the discrimination functions associated with the RBF NNs. The weights of the candidate image are obtained in real-time. Fig. 17 is used as the inputs of the discriminant function.

Fig. 17 shows discrimination process of recognition candidate person based on real output by figure. As shown, we extracted the 5 dimensional feature by PCA. It means that we extracted 5-eigenvectors as shown in Fig. 16. Fig. 17 displays the polynomial function of the conclusion part connected of recognized candidate (A ∼ D).

In this study, we carry out two suites of experiments as described below:

(a) *Case* 1: Carry out AdaBoost and histogram equalization without ASM from real-time images.
(b) *Case* 2: Carry out using AdaBoost and histogram equalization with ASM from real-time images.

In this paper, we experiment with the Yale and IC&CI Lab datasets.

5.1. The Yale dataset

The Yale face dataset consists of 165 grayscale images of 15 individuals represented in the GIF format. There are 11 images per person, one per different facial expression or configuration: center-light, w/glasses, happy, left-light, wearing/no glasses, normal, right light, sad, sleepy, surprised, and wink. Because of the changes to extreme facial expressions, images concerning surprised facial expressions are not included in the experiments. As a result, we arrive at a dataset comprising 150 images of 15 persons with 10 images per person.

Table 1 summarizes the values of the parameters of the proposed classifier along with the parameters of the design environment.

Each experiment we split data into 50%–30%–20% for training, validation, and testing subsets, namely, 80% (50%–30% training and validation set) of the whole pattern is selected randomly for training and the remaining pattern is used for testing purpose.

Fig. 16. Eigenfaces formed in the face space.

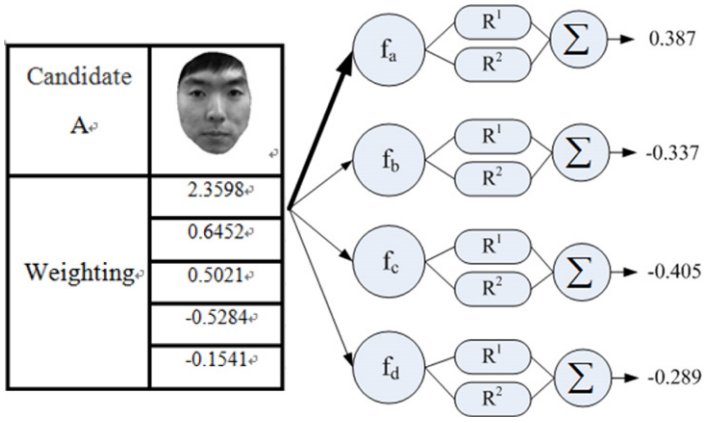


Fig. 17. Weights of the PCA used in the discrimination process.

Table 1
Values of the parameters used in the experiments.

RBF NNs		
Data split		Training: Validation: Testing = 50%: 30%: 20%
Optimization algorithm		DE
The number of evaluations of the objective function (Generations/swarms)		1000 (20 × 50)
Search space	The number of rules	2, 3, 4, 5
	Polynomial type	Linear, and reduced quadratic type
	Fuzzification coefficient	[1.1, 3.0]

Table 2
Classification rate obtained for the Yale dataset (without obstacle factors). The best results are shown in boldface.

Number of rules	*Case* 1 (without ASM)		*Case* 2 (with ASM)	
	L-RBF NNs	RQ-RBF NNs	L-RBF NNs	RQ-RBF NNs
2	86.00% ± 4.94	73.33% ± 4.35	82.00% ± 5.58	73.33% ± 5.84
3	86.00% ± 5.48	64.67% ± 12.61	86.67% ± 5.27	52.66% ± 5.96
4	**88.00% ± 2.98**	66.00% ± 8.63	83.33% ± 4.08	55.33% ± 6.06
5	86.00% ± 3.65	66.66% ± 10.54	79.33% ± 4.94	50.66% ± 9.83

Table 3
Results for the IC&CI Lab. data (without obstacle factors). The best results shown in boldface.

Number of rules	*Case* 1 (without ASM)		*Case* 2 (with ASM)	
	L-RBF NNs	RQ-RBF NNs	L-RBF NNs	RQ-RBF NNs
2	94.88% ± 1.19	93.33% ± 3.45	92.92% ± 2.98	94.67% ± 2.44
3	93.56% ± 2.35	92.78% ± 1.51	95.73% ± 1.97	**97.56% ± 3.94**
4	95.14% ± 2.13	95.67% ± 1.71	94.21% ± 1.34	97.22% ± 2.45
5	95.45% ± 4.34	94.64% ± 2.70	96.45% ± 2.61	96.12% ± 2.16

There was no overlap between the training, validation, and testing sets (Lewrence, Giles, Tsoi, & Back, 1997). For each combination of the parameters, the experiment was repeated five times. The results are reported by presenting the average and standard deviation of the classification rate obtained over these five repetitions of the experiment. The number of rules, polynomial type and the fuzzification coefficient are optimized by the DE.

The experimental results expressed in terms of the classification rate and its standard deviation are reported in Table 2. In Tables 2–5, the abbreviations L-RBF NNs and RQ-RBF NNS refer to polynomial types of each RBF NNs such as linear and reduced quadratic type.

When the polynomial type is linear and the number of rules is set to 4, the recognition rate is higher than 85% in *Case* 1 (without ASM). When the polynomial type is linear and the number of rules is 3, we obtain a similar recognition rate in *Case* 2 (with ASM). Notably, the recognition rate in *Case* 1 is slightly higher than the one obtained for *Case* 2.

5.2. IC&CI Laboratory dataset

IC&CI (Intelligent Control & Computational Intelligence) face database contains a set of face images taken by students in the University of Suwon. There are 10 different images of each of 35 distinct subjects. Each image was digitized and presented in the 200 × 200 pixel array with gray levels. Images feature frontal view faces with different facial expressions, and occlusions. The values of the parameters of the experimental setup are the same as used in the experiments with the Yale data.

The obtained results are included in Table 3.

Next, the performance of the classifier face recognition is reported in presence of various obstacle factors.

In this study, the classifier is trained by images used in the previous experiment (*Case* 1). The classification rate is evaluated by using test images that are affected by obstacles. The experiments are completed under the same conditions as set up in the previous experiments.

Table 4
Results for the IC&CI Lab. dataset (in case of wearing a cap). The best results shown in boldface.

Number of rules	Case 1 (without ASM)		Case 2 (with ASM)	
	L-RBF NNs	Q-RBF NNs	L-RBF NNs	Q-RBF NNs
2	53.18% ± 4.09	50.75% ± 3.98	55.51% ± 3.55	64.19% ± 5.38
3	56.16% ± 3.78	53.16% ± 4.34	63.75% ± 4.47	**74.72%** ± **6.34**
4	61.02% ± 5.46	59.57% ± 4.68	68.07% ± 6.61	73.14% ± 5.69
5	62.97% ± 4.51	**65.55%** ± **5.97**	66.22% ± 5.34	70.25% ± 4.56

Table 5
Results for the IC&CI Lab. dataset (in case of using a mobile phone).

Rules number	*Case* 1 (without ASM)		*Case* 2 (with ASM)	
	L-RBF NNs	RQ-RBF NNs	L-RBF NNs	RQ-RBF NNs
2	63.67% ± 2.09	67.02% ± 3.57	72.23% ± 3.71	76.83% ± 3.90
3	68.92% ± 3.60	70.21% ± 3.85	74.31% ± 2.61	80.40% ± 5.42
4	67.15% ± 4.86	70.32% ± 2.88	75.07% ± 4.77	**80.44%** ± **4.37**
5	**71.61%** ± **3.46**	63.22.% ± 4.16	77.88% ± 4.75	76.23% ± 3.26

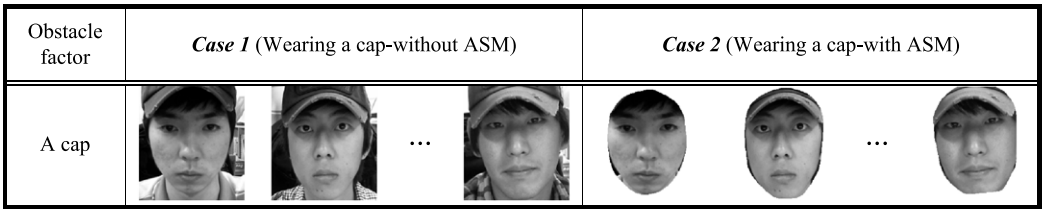


Fig. 18. A real-time acquired testing image for recognition (wearing a cap).

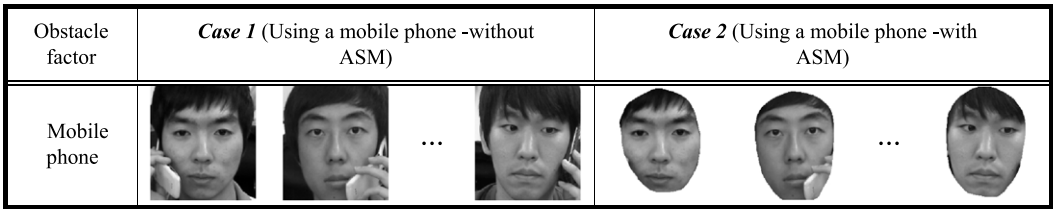


Fig. 19. A real-time acquired testing image for recognition (in case of using a mobile phone).

– In case of wearing a cap (see Fig. 18).
In case of wearing a cap, Table 4 shows the experimental results obtained for the two cases.
When wearing a cap, the decrease of overall performance was noticeable in comparison to the results produced in the previous experiment (without obstacles). The recognition rate of *Case* 2 is better than the one reported in *Case* 1 by using the effective facial features because the unnecessary image parts have been removed with the use of the ASM.
– In case of using a mobile phone (see Fig. 19).
The corresponding results are reported in Table 5.
In this case, the recognition rate for *Case* 2 is better than *Case* 1 because the unnecessary image parts have been removed with the aid of ASM.

5.3. Comparative analysis—PCA and RBF NNs

We now compare recognition performance of PCA and RBF NNs in order to evaluate the performance of the proposed model. The process of face recognition experimental using PCA is given as follows.

First, the data are divided into training data (50%), validation data (30%), and testing data (20%) exactly in the same conditions as for the RBF NNs. Database consists of PCA weights of training images. Next, we calculate Euclidean distance between test image weights and database image (Yale and IC&CI Lab. image) weights. Then the final recognized candidate is decided as the candidate with the fewest errors. Finally, recognition performance is obtained by using test image and determined database image. Recognition performance is represented to recognition rate (%) and the number of false recognition (false recognition/test data).

The summary of the final classification results is presented in Fig. 20.

When we used the non obstacle (general) images, the recognition rate of *Case* 1 (without ASM) is slightly better than *Case* 2 (with ASM). When obstacles are involved, the recognition rate of *Case* 2 is higher than the one reported for *Case* 1 by using the effective facial features because the unnecessary image parts are removed by using ASM. Also, the recognition rate of RBFNNs is better than the one reported for the PCA.

6. Conclusions

In this study, the proposed face recognition system comprises two main functional modules. In the preprocessing part, two-dimensional gray face images are obtained by using AdaBoost and then histogram equalization is used to improve the quality of image. The personal profile consists of the extracted face contour and shape by using the ASM. The features were extracted by the PCA algorithm. In the classifier part, we proposed the optimized RBF NNs for the problem of face recognition. In the recognition part, the proposed P-RBF NNs exhibit some unique and useful characteristics. The P-RBF NNs involve a partition module formed by the FCM clustering and used here as an activation function

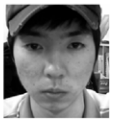
			Normal Image		Wearing a cap		Using a mobile phone	
								
	Basic 2D (*Case 1*)	Hybrid (*Case 2*)	Basic 2D (*Case 1*)	Hybrid (*Case 2*)	Basic 2D (*Case 1*)	Hybrid (*Case 2*)	Basic 2D (*Case 1*)	Hybrid (*Case 2*)
Recognition rate (%) for Basic PCA (Number of success time)	80.00% (12/15)	66.67% (10/15)	97.14% (34/35)	91.43% (32/35)	42.86% (15/35)	62.86% (22/35)	65.71% (23/35)	74.29% (26/35)
Recognition rate (%) using RBFNNs (Standard deviation)	88.00% (2.98)	86.67% (5.27)	95.67% (1.71)	97.56% (3.94)	65.55% (5.97)	74.72% (6.34)	71.61% (3.46)	80.44% (4.37)

Fig. 20. Comparison of face recognition results—shown are selected faces.

of the neurons located in the hidden layer. The P-RBF NNs are expressed as a collection of "if-then" fuzzy rules. The image data obtained from the CCD camera is used for preprocessing procedures including image stabilizer, face detection and feature extraction. The preprocessed image data are processed with the aid of RBF NNs. We have reported recognition rates of the ASM—processed images, generic images and facial images containing some obstacles. In the presence of obstacles, the recognition rate of *Case* 2 is better than the one reported in *Case* 1 by using the effective facial features. This improvement can be attributed to the fact that the unnecessary parts of the image have been removed with the use of the ASM.

Acknowledgment

This work was supported by the GRRC program of Gyeonggi province [GRRC Suwon 2015-B2, Center for U-city Security & Surveillance Technology].

References

Balasubramanian, M., Palanivel, S., & Ramalingam, V. (2009). Real time face and mouth recognition using radial basis function neural networks. *Expert Systems with Applications*, *36*, 6879–6888.

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. New York: Plenum Press.

Boehnen, C., & Russ, T. (2005). A fast multi-modal approach to facial feature detection. In *Proceedings of the seventh IEEE workshop on applications of computer vision, WACV/MOTION'05* (pp. 135–142).

Chellappa, R., Wilson, C. L., & Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, *83*(5), 704–740.

Colbry, D., Stockman, G., & Jain, A. (2005). Detection of anchor points for 3D face verification. In *Proc. of A3DISS, San Diego CA*.

Colombo, A., Cusano, C., & Schettini, R. (2005). 3D face detection using curvature analysis. In *4 Int'l symposium on. image and signal processing and analysis. ISPA*.

Connolly, J.-F., Granger, E., & Sabourin, R. (2012). Evolution of heterogeneous ensembles through dynamic particle swarm optimization for video-based face recognition. *Pattern Recognition*, *45*(7), 2460–2477.

Cootes, T., Cooper, D., Taylor, C., & Graham, J. (1995). Active shape models-their training and application. *Computer Vision and Image Understanding*, *61*(1), 38–39.

Cootes, T. F., Taylor, C. J., Lanitis, A., Cooper, D. H., & Graham, J. (1993). Building and using flexible models incorporating grey-level information. In *Fourth internat. conf. computer vision* (pp. 242–246).

Dervis, K., & Selcuk, O. (2004). A simple and global optimization algorithm for engineering problems: Differential evolution algorithm. *Turkish Journal of Electrical Engineering*, *12*, 53–60.

Freund, Y., & Schapire, R. E. (1995). A decision–theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23–27).

Gao, Y., Pan, J., Ji, G., & Yang, Z. (2012). A novel two-level nearest neighbor classification algorithm using an adaptive distance metric. *Knowledge-Based Systems*, *26*, 103–110.

Gonzalez, R. C., & Woods, R. E. (2002). *Digital image processing*. New Jersey: Prentice-Hall.

Han, H. G., Chen, Q. L., & Qiao, J. F. (2011). An efficient self-organizing RBF neural network for water quality prediction. *Neural Networks*, *24*, 717–725.

Huang, Y.-S., Hzu, T.-C., & Cheng, F.-H. (2010). Facial landmark detection by combining object detection and active shape model. In *Electronic commerce and security, ISECS* (pp. 381–386).

Lewrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural network approach. *IEEE Transactions on Neural Networks*, *8*(1), 98–113.

Lienhart, R., & Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. In *IEEE conference on image processing, ICIP, New York* (1) (pp. 900–903).

Lopez-Molina, C., Baets, B. D., Bustince, H., Sanz, J., & Barrenechea, E. (2013). Multiscale edge detection based on Gaussian smoothing and edge tracking. *Knowledge-Based Systems*, *50*, 101–111.

Lu, X., & Jain, A. K. (2005). *Multimodal facial feature extraction for automatic 3D face recognition, technical report MSU-CSE-05–22*. Computer Science and Engineering, Michigan State University.

Mohammed, A. A., Minhas, R., Jonathan Wu, Q. M., & Sid-Ahmed, M. A. (2011). Human face recognition based on multidimensional PCA and extreme learning machine. *Pattern Recognition*, *44*(10–11), 2588–2597.

Oh, S.-K., Kim, W.-D., Pedrycz, W., & Park, H.-S. (2014). Fuzzy radial basis function neural networks with information granulation and its parallel genetic optimization. *Fuzzy Sets and Systems*, *237*, 96–117.

Oh, S.-K., Pedrycz, W., & Park, B.-J. (2004). Self-organizing neurofuzzy networks in modeling software data. *Fuzzy Sets and Systems*, *145*, 165–181.

Park, B.-J., Oh, S.-K., & Kim, H.-K. (2008). Design of polynomial neural network classifier for pattern classification with two classes. *Journal of Electrical Engineering & Technology*, *3*(1), 108–114.

Roh, S.-B., Oh, S.-K., & Pedrycz, W. (2010). A fuzzy of parallel polynomial neural networks with information granules formed by fuzzy clustering. *Knowledge-Based Systems*, *23*(3), 202–219.

Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(1), 23–38.

Slavkovic, M., & Jevtic, D. (2012). Face recognition using eigenface approach. *Serbian Journal of Electrical Engineering*, *9*(6), 121–130.

Staiano, A., Tagliaferri, R., & Pedrycz, W. (2006). Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering. *Neurocomputing*, *69*, 1570–1581.

Storn, R. (1997). Differential evolution, a simple and efficient heuristic strategy for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359.
Sung, K.-K., & Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(1), 39–51.
Tsekourasa, G., Sarimveisb, H., Kavaklia, E., & Bafasb, G. (2005). A hierarchical fuzzy clustering approach to fuzzy modeling. *Fuzzy Sets and Systems*, *150*(2), 245–266.
Viola, P., & Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE conference on computer vision and pattern recognition, CVPR, Hawaii* (1) (pp. 511–518).
Viola, P., & Jones, M. J. (2004). Robust real-time object detection. *International Journal of Computer Vision*, *57*(2), 137–154.
Wang, Qihui, Xie, Lijun, Zhu, Bo, Yang, Tingjun, & Zheng, Yao (2013). Facial feature extraction based on active shape model. *Journal of Multimedia*, *8*(6), 747–754.
Yang, M.-H., Kriegman, D., & Ahuja, N. (2002). Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(1), 34–58.
Yong, L., Zhang, C. S., Xiaoguang, L., & Zhang, D. (2000). Face contour extraction with active shape models embedded knowledge. In *Internat. conf. signal processing* 2, (pp. 1347–1350).
Zhao, W., Chellappa, R., & Phillips, P. (2003). Face recognition: a literature survey. *ACM Computing Surveys*, *35*(4), 399–458.