FACE RECOGNITION USING HOUGH-KLT AND A FEED-FORWARD BACK-PROPAGATION NEURAL NETWORK

[1]Pablo Rivas Perea, [2]Mario I. Chacón Murguía
Digital Signal Processing & Vision Laboratory,
Chihuahua Institute of Technology
Ave. Tecnológico # 2909 Colonia 10 de Mayo
C.P. 31310 Chihuahua, Chih.
[1]pablorp80@ieee.org, [2]mchacon@itchihuahua.edu.mx

ABSTRACT

In this paper we describe the Hough-KLT algorithm for facial feature extraction based on facial feature lines, and the Euclidean distance classifier approach is utilized. In this paper we present the Feedforward Backpropagation neural network, FFBP, as classifier for face recognition using Hough-KLT. When the FFBP algorithm is compared, FFBP show results comparable with the Euclidean distance classifier.

1. INTRODUCTION

Face recognition is one of the most interesting and challenging areas in computer vision and pattern recognition. Current face recognition systems have high recognition rates, however various changes in face images also present a great challenge, and a face recognition system must be robust with respect to the many variabilities of face images such as viewpoint, pose, illumination, and facial expression [1]-[4]. The popular approaches for face recognition are the eigenface and Fisherface method. The eigenface method, or principal component analysis (PCA) [5], is the most well known method for face recognition [6]. Each of them comes with some advantages but is not free from limitations and drawbacks when cast in the setting of face recognition. PCA is a popular approach in image processing and communication theory that is quite often referred to as a Karhunen–Loeve Transformation (KLT). The PCA approach exhibits optimality when it comes to dimensionality reduction. However, it is not ideal for classification purposes as it retains unwanted variations occurring due to diversified lighting and facial expression [7]. At the DSP & Vision Laboratory novel feature extraction methods for face recognition have been developed. One of these methods is *Hough-KLT*, based on face lines in combination with KLT. This method utilizes the nearest-neighbor classifier. Therefore this method requires to be tested with other classifiers in order to compare its performance.

The theory of artificial neural networks, ANN, and fuzzy logic, FL, has been used to resolve several pattern recognition tasks having very good results [8]. As new pattern recognition methods are discovered or improved, new techniques of neural networks are proposed in which special systems can learn and generalize, also new systems based on fuzzy logic are proposed when a system needs to deal with uncertainty. This makes the pattern recognition models be smart and powerful [8]. ANN classifier is suggested to be used for the feature extraction method *Hough-KLT* created at DSP and Vision Laboratory.

In this paper we describe at Section II the Hough-KLT algorithm for facial feature extraction and the Euclidean distance classifier approach. The Feedforward Backpropagation Neural Network is presented for face recognition in Section III. And finally the general conclusions of this work are presented in Section IV.

2. HOUGH-KLT FEATURES FOR FACE RECOGNITION

After a deep study regarding the perception of faces, we observed that the lines are important features for recognition, especially on newborns. Therefore, we have decided to extract these features on every face. Hough transform is a useful tool for extracting lines based on the edges of an image.

2.1. Hough Transform.

In the domain of the Hough transform, HT, any line is defined as

$$\rho = x\cos\theta + y\sin\theta \tag{1}$$

where the HT of that line describes a sinusoidal wave varying its amplitude on the (ρ,θ) space, as shown in Fig. 1.

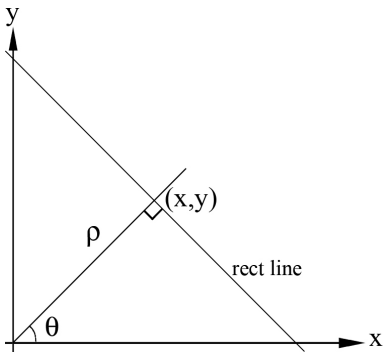

Fig. 1. Representation of a line in the (ρ,θ) space.

In practice, the HT algorithm requires to have a binary image as input, which represents the borders of the image [9]. The borders image is very important in order to the HT performs well.
In this case the border detection algorithm used is Canny [9]. Canny has proved to be efficient in the borders detection task. Once the border image is obtained, the HT is calculated, and the result is a representation of all the lines in the space ρ, and θ. Given this result we can extract the main lines or characteristic lines just obtaining the maximum points in the result of the HT. Obtaining the coordinates of those points we can obtain the lines through ρ, and θ information.

To obtain the characteristic lines on a face, we calculate the four maximum points contained on HT. We consider that four lines are enough to represent a face, based on the newborns vision system. The information of these 4 lines will be introduced in the feature vector to be defined in detail on further subsections.

Given a grayscale facial image $I(x,y)$ Canny is applied to it, $I_{BW} = T_{Canny}(I)$, and returns a binary image containing the borders of the facial image I_{BW}. Applying HT we get a matrix containing the sinusoidal representation of the lines of the original image, this matrix is also known as *accumulator* [9], $\{\mathbf{P}_{Acumulator}, \mathbf{P}_{\theta}, \mathbf{P}_{\rho}\} = T_{Hough}(I_{BW})$, where $\mathbf{P}_{Acumulator}$ is the accumulator, $\mathbf{P}_{\theta}$ (degrees) and $\mathbf{P}_{\rho}$ are the vector for the values ρ and θ on which the matrix $\mathbf{P}_{Acumulator}$ was generated. In these terms, the longest line or characteristic line can be defined as the peak value in the accumulator as follows

$$\mathbf{N}_{Peaks}(\theta_i,\rho_i) = T_{HoughPeaks}(\mathbf{P}_{Acumulator}, i) \qquad (2)$$

where $\mathbf{N}_{Peaks}(\theta_i,\rho_i)$ is a matrix of $2\times i$ elements where i denotes the number of peaks to extract. (θ_i,ρ_i) represents the coordinates in the space θ,ρ for the i-th peak value of $\mathbf{P}_{Acumulator}$.
To obtain the coordinates of the lines we can use the following definition

$$\mathbf{R}_{Lines}(x_{1i}, y_{1i}, x_{2i}, y_{2i}) = T_{Lines}(I_{BW}, \mathbf{P}_{\theta}, \mathbf{P}_{\rho}, \mathbf{N}_{Peaks}) \qquad (3)$$

where $\mathbf{R}_{Lines}(x_{1i}, y_{1i}, x_{2i}, y_{2i})$ denotes a matrix containing all the coordinates x_{1i}, y_{1i} where the i-th line begins and x_{2i}, y_{2i} where the line detected by the HT ends.
The result of apply the HT to a face, locating its 4 maximum points and plotting the main lines obtained, produces the result shown in Fig. 2. , where a) is the original image, b) is the binary image obtained trough Canny borders algorithm, d) is the spectrum representing the result of apply the HT to the image at b), finally c) is the original image plus its four characteristic lines.
For the generation of the first part of the features vector from the coordinates of its four characteristic lines, the following method was designed:

Step 1. Get the four maximum peak values with (2), for $i = 4$.

Step 2. Implement (3) and get the four characteristic lines original coordinates, stored at $\mathbf{R}_{Lines}(x_{1i}, y_{1i}, x_{2i}, y_{2i}) \quad \forall \quad i = 1...4$.

Step 3. Concatenate the values of the lines coordinates to obtain a singleton introduced at vector $\mathbf{z}_i$, such that the reverse process can be always possible. Therefore $\mathbf{z}_i = [l_{i_1} \quad l_{i_2}]$.

Step 4. Take the value of x_{1i} and add it to $\frac{y_{1i}}{1000}$, and introduce the result to l_{i_1}. The division by 1000 is performed in order to do not loose information and to do not increase the size of the feature vector.

Step 5. Take the value of x_{2i} and add it to $\frac{y_{2i}}{1000}$, and introduce the result to l_{i_2}. We have only two points, because one point is the beginning of the line and the second one is the end of it.

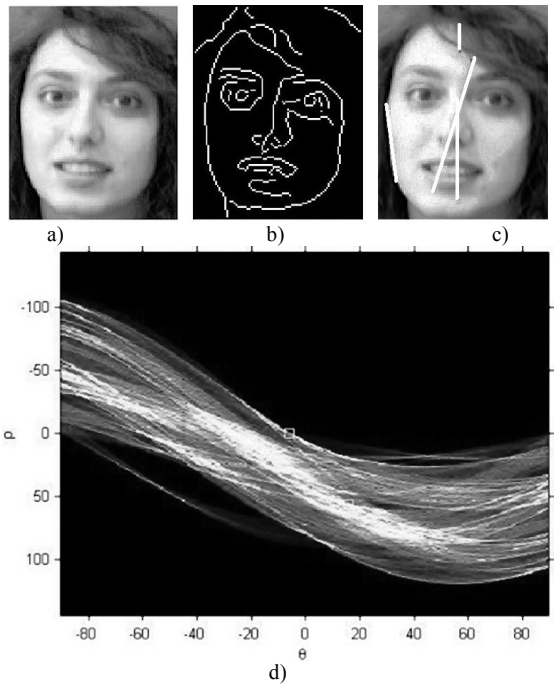

Fig. 2. Hough transform of a face: a) original image, b) face borders trough Canny, c) original image plus its 4 characteristic lines, and d) spectrum obtained by applying Hough transform.

The feature vector equation can be defined as follows

$$\mathbf{z}_i = \begin{bmatrix} x_{11} + \frac{y_{11}}{1000}, x_{21} + \frac{y_{21}}{1000} \ldots \\ x_{1i} + \frac{y_{1i}}{1000}, x_{2i} + \frac{y_{2i}}{1000} \end{bmatrix} \quad (4)$$

Once we got the first part of the feature vector $\mathbf{z}_i$, this vector must be concatenated with the original image $\mathbf{i}_{xy}$ in a canonical form (vector column), to construct the final feature vector $\mathbf{d}_{i+xy}$. $\mathbf{i}_{xy}$ is defined as

$$\mathbf{i}_{xy} = I(x, y) \quad (5)$$

where $I(x, y)$ is the original image of size $x \times y$, in other words, if the image *I* is an 90×80 element vector, then $\mathbf{i}_{xy}$ has 7200 elements. The final feature vector is defined as

$$\mathbf{d}_{i+xy} = [\mathbf{z}_i \quad \mathbf{i}_{xy}] \quad (6)$$

where $\mathbf{d}_{i+xy}$ has $i+xy$ elements. Example, if $\mathbf{z}_i$ has 8 elements (the start and end points of 4 lines) and $\mathbf{i}_{xy}$ has 7200 elements, then $\mathbf{d}_{i+xy}$ has 7208 elements. The vector $\mathbf{z}_i$ is linked to the information of the original image in order to contribute and complement to the feature vector before the transformation via KLT.

2.2. Principal Component Analysis

Principal Component Analysis, PCA, is a very widely used technique for dimensionality reduction. The objective of PCA is to transform the representation space **X** into a new space **Y**, in which the data are uncorrelated. The covariance matrix in this space is diagonal. PCA aims to find the new set of orthogonal axis to maximize the variance of the data. The final objective is dimensionality reduction of the problem [8].

The steps needed for PCA are the following.

Step 1. The covariance matrix $\mathrm{Cov}_{\mathbf{X}}$ is calculated over the input vectors set $\mathbf{x}_i$ that represents i facial images represented as vectors $\mathbf{x}$. The covariance is defined as

$$\mathrm{Cov}_{\mathbf{X}} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \quad (7)$$

where $\bar{\mathbf{x}}$ denotes the mean of each variable of the vector $\mathbf{x}$, and n is the amount of input vectors.

Step 2. The *d* eigenvalues of $\mathrm{Cov}_{\mathbf{X}}$ are extracted and defined as $\lambda_1, \lambda_2, \ldots \lambda_d$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$.

Step 3. The *d* eigen-vectors are calculated in $\Phi_1, \Phi_2, \ldots \Phi_d$ and associated to $\lambda_1, \lambda_2, \ldots \lambda_d$.

Step 4. A transformation matrix is created $\mathbf{W}_{PCA} = [\Phi_1, \Phi_2, \ldots \Phi_d]$.

Step 5. The new vectors $\mathbf{y}$ are calculated using the following equation

$$\mathbf{Y} = \mathbf{W}_{PCA}^T \mathbf{X} \quad (8)$$

where T denotes the transpose of $\mathbf{W}_{PCA}$, and $\mathbf{X}$ denotes the matrix containing all the input vectors. $\mathbf{Y}$ denotes the matrix containing all the new vectors.

2.3. Karhunen-Loeve Transformation

The KLT [6] is also known as principal component analysis. As a difference from PCA, with KLT the

input vectors $\{\mathbf{x}_i\}$ are normalized to the interval [0 1]. So, when the normalization is done, the PCA algorithm can be followed at no change.
At this point we can get the KLT transformation matrix, but it should be clear that the input vectors $\{\mathbf{x}_i\}$ are the training set, in other words, a set of faces selected specially for training purposes. These images are represented as vectors and then they can follow the normal process of eigenvectors, etc. This means that if a facial pattern image has size of 100x20, it should be introduced as a 2000 element vector. If the # of faces is 10 per individual, with 8 people, the matrix $\{\mathbf{x}_i\}$ generated should be of 2000 x 80 elements.

2.4. Hough-KLT Implementation

The vector $\mathbf{d}_{i+xy}$ (input to the KLT) is composed by 8 coefficients of the TH $\mathbf{z}_i$, and the remaining coefficients are composed by the original image $\mathbf{i}_{xy}$, as denoted in (6).

This feature vector is transformed to the KLT domain in order to get the transformation matrix. Following these steps can perform the recognition process:

Step 1. Of a given facial image I_{face} belonging to a class "*C*", $I_{face} \in C$, it is adjusted to set it on a vectorial form. This means, obtain $\mathbf{i}_{xy}$.

Step 2. Compute the 8 elements of the vector of coordinates $\mathbf{z}_i$ with (4).

Step 3. The TH coefficients and the image on vectorial form are concatenated in order to get $\mathbf{d}_{i+xy}$.

Step 4. The vector $\mathbf{d}_{i+xy}$ is projected on the transformation matrix $\mathbf{W}_{KLT}$ obtained with (8), in order to compute **Y**.

Step 5. Compute the Euclidean distance between **Y** and ever single existing elements on the hyperplane. The nearest neighbor classifies **Y** as a member of the same class.

The Euclidean distance between the new face image **Y**, projected over the transformation matrix, $\mathbf{W}_{KLT}$, and other image $\mathbf{Z}_C = \mathbf{d}_{i+xy_C} \cdot \mathbf{W}_{KLT}$ belonging to a class *C*, is redefined from (4.4) as follows

$$d(\mathbf{Y}, \mathbf{Z}_C) = \|\mathbf{Y} - \mathbf{Z}_C\| \quad (9)$$

In this way, **Y** is assigned to the class *C* where $d(\mathbf{Y}, \mathbf{Z}_C)$ is minimum.

2.5. Yale Face Database

The face database "Yale Face Database" contains images of subjects in a variety of conditions included with-without glasses, illumination and expression variations [6]. In Fig. 3 are presented samples of two different subjects under the conditions described above.

Fig. 3. Sample faces of the YALE database.

2.6. Results with Euclidean distance

The experiments consists on vary the samples for training from 1 to 8 samples per subject. 10 classes were constructed according to the number of subjects to recognize. The samples picked for training and for testing where selected arbitrary. These experiments were realized on YALE. The highest performance obtained when testing, reaches the 78% for face recognition as presented on Table I. The average performance for the YALE database is 63% of recognition.

TABLE I
RECOGNITION RATE RESULTS FOR YALE

SUBJECT #	TRAINING SAMPLES							
	TS1	TS2	TS3	TS4	TS5	TS6	TS7	TS8
S1	0.2	0.4	0.7	0.7	0.8	0.8	0.8	0.8
S2	0.9	0.9	0.9	1	1	1	1	1
S3	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
S4	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3
S5	0.3	0.9	0.9	0.9	0.9	1	1	1
S6	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
S7	0.4	0.4	0.4	0.3	0.3	0.3	0.5	0.6
S8	0.6	0.8	0.9	0.9	0.9	0.9	0.9	0.9
S9	0.5	0.6	0.6	0.7	0.7	0.8	0.7	0.6
S10	0.6	0.7	0.8	0.8	0.8	0.8	0.8	0.8
TOTAL	**0.55**	**0.67**	**0.72**	**0.73**	**0.74**	**0.76**	**0.77**	**0.78**

Where S refers to the subject, and TS means Training Sample.

3. FFBP-HOUGH-KLT FOR FACE RECOGNITION

In this section we describe the experiments and the results for a Feedforward-Backpropagation Hough-KLT face recognition scheme.

3.1. Feedforward-Backpropagation Network

A *"Feedforward"* network is a computing device where the processing units are distributed on layers in a unidirectional way via weights [10]. The Fig. 4 shows in a) an example of a four-layer Feedforward network. With a Feedforward containing four perceptrons that are independent of each other in the hidden layer, a point is classified

into 4 pairs of linearly separable regions, each of which has a unique line separating the region as shown in b). The MALTAB software utilizes a singular notation for neural networks. For Feedforward on MATLAB we have the figure shown in c). We utilize MATLAB for the entire project reported on this document.

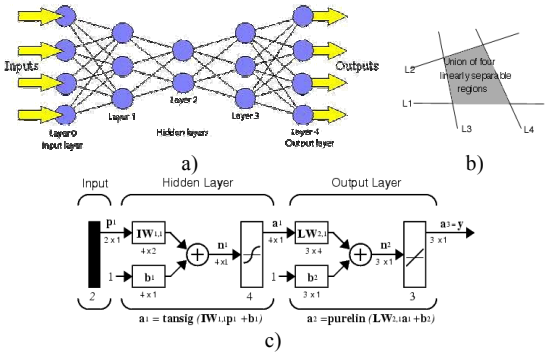

Fig. 4. Feedforward networks. a) a 4 layer Feedforward network. b) intersection of 4 linearly separable regions forms the center region. c) example of MATLAB notation for a Feedforward.

Multi-layer Perceptron, MLP consists on multiple perceptrons utilizing different activation functions that allows good class separation for real life problems. The training algorithm for MLP is the well-known Backpropagation, BP.

3.2. BP Algorithm

The BP training is one of the most important algorithms developed for ANN [11]. Given a set of training input output samples $\{\mathbf{x}^{(k)}, \mathbf{d}^{(k)}\}$, k denotes the number of the pairs; the algorithm adjusts the weights of the neurons to classify the input patterns to their corresponding classes [11].
This algorithm is realized in two phases. First, the input pattern $\mathbf{x}^{(k)}$ is propagated from the input layer to the output layer, and the result of this data flow produces the actual network output $\mathbf{y}^{(k)}$. Then the error is computed according to the difference between the desired output $\mathbf{d}^{(k)}$ and the current network output $\mathbf{y}^{(k)}$, this error is back propagated from the output layer to the previous layer to adjust the weights following the descendent gradient method [11]. The weight adjust process is performed until the input layer is reached.

3.3. Feature vector

The feature vector was constructed with (6). The feature vector suffered a dimensionality reduction to a size of 34 elements. The samples picked for training were 8 samples per subject. The samples were picked randomly. We have designed the system for 10 people (10 classes). The training matrix size was 34x80. The face database utilized is YALE.

3.4. Design of the FFBP network

In this experiment we have constructed a 2-layered network. The FFBP network was constructed for 34 inputs at the input layer; 80 neurons in the hidden layer with *tangsig* activation functions; and 10 neurons at the output layer with *purelin* activation functions. This architecture is shown graphically in Fig. 5.

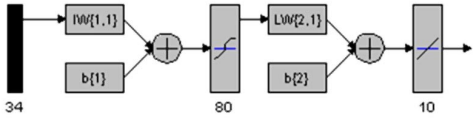

Fig. 5. FFBP network architecture, experiment 3.

For the Backpropagation algorithm we have decided to use the Bayesian regularization Backpropagation algorithm, *trainbr*. Regarding to the learning, gradient descent with momentum weight and bias learning function, *learngdm*, was utilized. The network in the training process has reached 7.2423e-28 of error rate. This has done in 6 epochs. The Fig. 6 shows the error curve.

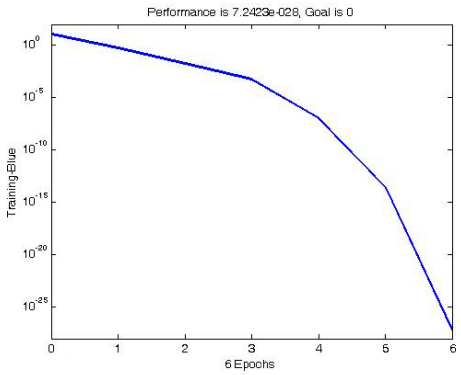

Fig. 6. Error curve for experiment 3.

3.5. Testing the FFBP network

For testing phase, we pick one of the two available samples for testing for each person.
The sample is picked randomly. Ideally the output of the neuron on training and testing should be the identity matrix. In practice (depending of the activation function) we have decimal values. The output of the network on testing is shown in Table II. For neuron N1, the output expected for subject S1 is one, but as can be seen we have 0.6338 that is close to one but with errors. However, if we

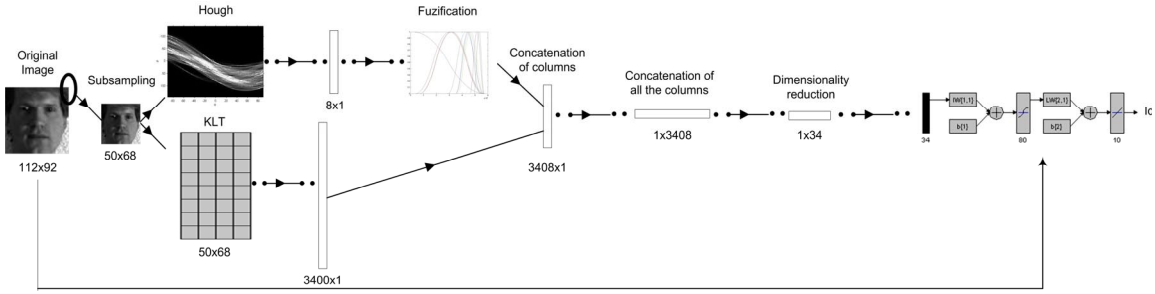

Fig. 7. General work scheme for FFBP-Hough-KLT face recognition.

calculate the maximum output of the neurons we can assign classification depending of the neuron. This means, if the neuron N8 has the maximum output, means that it is the subject S8. The results shown in Table II are also known as confusion matrix.

The performance reached with this experiment is 60%. Still being a low rate, for pattern recognition. However, this architecture results can be compared with the Euclidean distance classifier.

TABLE I
RESULTS FOR TESTING FFBP ON YALE

	TESTING VECTOR										
Neuron number	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Is Correct
N1	0.63	0.31	0.72	0.17	0.21	1.00	0.23	0.22	0.47	0.14	0
N2	-0.19	1.86	-0.96	0.61	-0.09	1.41	0.13	0.50	0.48	-0.55	0
N3	-0.13	-0.16	0.08	0.57	-0.01	0.28	0.40	0.58	0.37	-0.04	0
N4	-0.44	0.37	-0.14	0.70	0.12	-0.52	-0.22	0.21	0.29	0.11	1
N5	0.12	0.20	-0.64	0.33	0.52	-0.82	-0.75	0.28	-1.40	-0.55	1
N6	0.36	-0.39	-0.01	0.37	0.30	1.44	-0.54	0.29	0.47	-0.11	0
N7	-0.54	0.16	0.07	0.07	-0.43	-0.32	1.52	0.26	-0.30	-0.15	1
N8	1.02	0.80	-0.19	-0.08	0.22	0.28	-0.47	1.20	0.50	0.32	1
N9	0.16	0.22	-0.33	0.66	-0.09	0.51	0.01	0.13	0.53	-0.35	1
N10	-0.69	-0.69	0.64	-0.54	-0.36	1.44	-0.34	-0.86	0.52	0.97	1
Perfor-mance											60%

4. CONCLUSIONS

We have presented the Hough-KLT algorithm for feature extraction; using the Euclidean distance classifier the system present an average performance of 63% for YALE. We have presented a FFBP classifier experiments based on the scheme shown in Fig. 7. The higher recognition rate on testing reaches 60%. This result is comparable with the Euclidean distance classifier. This proves the power of the neural networks in face recognition applications. We have to remember that YALE is a face database containing variations in lighting conditions. Therefore the recognition rate is affected by this situation and low performance is expected as normal.

5. ACKNOWLEDGEMENTS

The authors appreciate the support of COSNET, and SEP-DGEST for the support of this research under grant 445.05-P. Also CONACyT for the support of this research under grant #:193324.

6. REFERENCES

H. Cevikalp, M. Neamtu, M. Wilkes, A. Barkana, "Discriminative Common Vectors for Face Recognition", IEEE Transactions on Pattern Analysis and Machinery Intelligence, Vol. 27, No. 1, Jan 2005.

X. He, S. Yan, Y. Hu, P. Niyogi, H. J. Zhang, "Face Recognition Using Laplacianfaces", IEEE Transactions on Pattern Analysis and Machinery Intelligence, Vol. 27, No. 3, Mar 2005.

M. J. Er, W. Chen, S. Wu, "High-Speed Face Recognition Based on Discrete Cosine Transform and RBF Neural Networks", IEEE Transactions on Neural Networks, Vol. 16, No. 3, May 2005.

T. Kim, J. Kittler, "Locally Linear Discriminant Analysis for Multimodally Distributed Classes for Face Recognition with a Single Model Image", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 3, Marzo 2005.

M. Turk and A. Pentland, "Face recognition using eigenfaces," in Proc. IEEE Conf. Computer Vision Pattern Recognition, 1991, pp. 586 – 591.

S. Z. Li and A. K. Jain, Handbook of face recognition, Springer, USA, 2004.

P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, July 1997, pp. 711–720.

G. M. J. Ramírez, "Estudio de métodos de reconocimiento de patrones basado en redes neurales y lógica difusa", M.S. dissertation, Grad. Studies & Research Div., Chihuahua Institute of Technology, Chihuahua, Mexico, 2005.

R. C. Gonzalez, R. E. Woods, S. L. Eddins, "*Digital image processing using Matlab"*, New Jersey, Pearson Prentice Hall, 2004.

L. H. Tsoukalas, R. E. Uhrig, "*Fuzzy and Neural Approaches in Enginnering",* Wiley Interscience, 1997.

Chin-Teng Lin and S.C. George Lee. "*Neural Fuzzy Systems"*. Prentice Hall. 1996.