Face Recognition Using Kernel Principal Component Analysis

Kwang In Kim, Keechul Jung, and Hang Joon Kim

***Abstract*—A kernel principal component analysis (PCA) was recently proposed as a nonlinear extension of a PCA. The basic idea is to first map the input space into a feature space via nonlinear mapping and then compute the principal components in that feature space. This letter adopts the kernel PCA as a mechanism for extracting facial features. Through adopting a polynomial kernel, the principal components can be computed within the space spanned by high-order correlations of input pixels making up a facial image, thereby producing a good performance.**

***Index Terms*—Eigenface, face recognition, kernel principal component analysis, machine learning.**

I. Introduction

A PRINCIPAL component analysis (PCA) is a powerful technique for extracting a structure from potentially high-dimensional data sets, which corresponds to extracting the q eigenvectors that are associated with the largest q eigenvalues from the input distribution. This eigenvector analysis has already been widely used in face processing [1], [2]. A kernel PCA, recently proposed as a nonlinear extension of a PCA [3]–[5] computes the principal components in a high-dimensional *feature space* F, which is nonlinearly related to the input space. A kernel PCA is based on the principle that since a PCA in F can be formulated in terms of the dot products in F, this same formulation can also be performed using *kernel* functions (the dot product of two data in F) without explicitly working in F. A kernel PCA has already been shown to provide a better performance than a linear PCA in several applications [3], [5].

This letter adopts a kernel PCA as a mechanism for extracting facial information. Through the use of a polynomial kernel, higher order correlations can be utilized between input pixels in the analysis of facial images. This amounts to identifying the principal components within the product space of the input pixels making up a facial image. Based on these features, face recognition can then be performed using linear support vector machines (SVMs). Experimental results and comparisons with other face recognition methods including linear PCA show the effectiveness of the proposed method.

II. Face Feature Extraction

The basic idea of kernel PCA is to first map the input data $\mathbf{x}$ into a feature space F via a nonlinear mapping Φ and then perform a linear PCA in F. Assuming that the mapped data are centered, i.e., $\sum_{i=1}^{M} \Phi(\mathbf{x}_i) = 0$, where M is the number of input data (the centering method in F can be found in [3] and [5]), kernel PCA diagonalizes the estimate of the covariance matrix of the mapped data $\Phi(\mathbf{x}_i)$, defined as

$$C = \frac{1}{M} \sum_{i=1}^{M} \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i). \tag{1}$$

To do this, the eigenvalue equation $\lambda \mathbf{v} = C\mathbf{v}$ must be solved for eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{v} \in F \backslash \{0\}$. As $C\mathbf{v} = (1/M) \sum_{i=1}^{M} (\Phi(\mathbf{x}_i) \cdot \mathbf{v}) \Phi(\mathbf{x}_i)$, all solutions $\mathbf{v}$ with $\lambda \neq 0$ lie within the span of $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_M)$, i.e., the coefficients $\alpha_i (i = 1, \ldots, M)$ exist such that

$$\mathbf{v} = \sum_{i=1}^{M} \alpha_i \Phi(\mathbf{x}_i). \tag{2}$$

Then the following set of equations can be considered:

$$\lambda(\Phi(\mathbf{x}_i) \cdot \mathbf{v}) = (\Phi(\mathbf{x}_i) \cdot C\mathbf{v}) \quad \text{for all } i = 1, \ldots, M. \tag{3}$$

The substitution of (1) and (2) into (3) and the definition of an $M \times M$ matrix K by $K_{ij} \equiv k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$ produces an eigenvalue problem which can be expressed in terms of the dot products of two mappings

Solve

$$M\lambda \boldsymbol{\alpha} = K\boldsymbol{\alpha}$$

for nonzero eigenvalues λ_l and eigenvectors $\boldsymbol{\alpha}^l = (\alpha_1^l, \ldots, \alpha_M^l)^T$ subject to the normalization condition $\lambda_l(\boldsymbol{\alpha}^l \cdot \boldsymbol{\alpha}^l) = 1$.

For the purpose of principal component extraction, the projections of $\mathbf{x}$ are computed onto the eigenvectors $\mathbf{v}^l$ in F. Fig. 1 shows the architecture of a kernel PCA for face feature extraction, which involves three layers with entirely different roles. The input layer is made up of source nodes that connect the kernel PCA to its environment. Its activation $\mathbf{x}$ comes from the gray level values of the face image. The hidden layer applies a nonlinear mapping Φ from the input space to the feature space F, where the inner products are computed. These two operations are in practice performed in one single step using the kernel k. The outputs are then linearly combined using weights α_i^l resulting in an lth nonlinear principal component corresponding to Φ. Thereafter, the first q principal components (assuming that

Manuscript received February 28, 2001; revised November 15, 2001. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. John M. Anderson.

K. I. Kim is with Artificial Intelligence Laboratory, Computer Science Department, Korea Advanced Institute of Science and Technology, Taejon 305-701, Korea (e-mail: kimki@ai.kaist.ac.kr).

K. Jung is with PRIP Laboratory, Computer Science and Engineering Department, Michigan State University, East Lansing, MI 48824 USA.

H. J. Kim is with the Artificial Intelligence Laboratory, Compute Engineering Department, Kyungpook National University, Taegu, Korea.

Publisher Item Identifier S 1070-9908(02)03400-4.

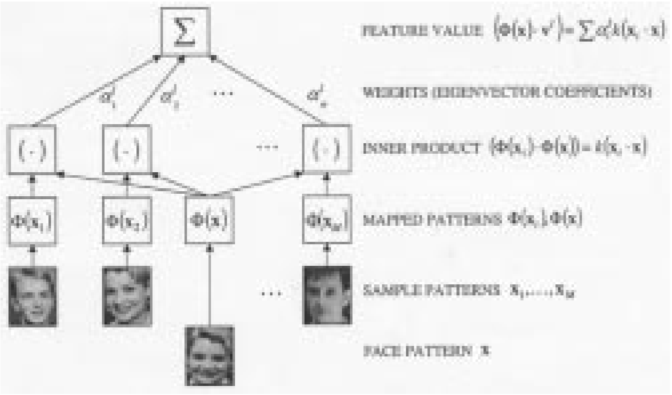
Σ
FEATURE VALUE $(\Phi(\mathbf{x})\cdot v^j)=\sum \alpha_i^j k(\mathbf{x}_i\cdot\mathbf{x})$
α_1^j
α_2^j
α_M^j
WEIGHTS (EIGENVECTOR COEFFICIENTS)
$(\cdot)$
INNER PRODUCT $(\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}))=k(\mathbf{x}_i\cdot\mathbf{x})$
$\Phi(\mathbf{x}_1)$
$\Phi(\mathbf{x}_2)$
$\Phi(\mathbf{x})$
$\Phi(\mathbf{x}_M)$
MAPPED PATTERNS $\Phi(\mathbf{x}_i),\Phi(\mathbf{x})$
SAMPLE PATTERNS $\mathbf{x}_1,\ldots,\mathbf{x}_M$
FACE PATTERN $\mathbf{x}$

Fig. 1. Face feature extraction architecture with kernel PCA.

the eigenvectors are sorted in a descending order of their eigenvalue size) constitute the q-dimensional feature vector for a face pattern.

By selecting the proper kernels, k, various mappings, Φ, can be indirectly induced. One of these mappings can be achieved by taking the d-order correlations between the entries, x_i, of the input vector $\mathbf{x}$. Since $\mathbf{x}$ represents a face pattern with x_i as a pixel value, a PCA in F computes the dth order correlations of the input pixels, and more precisely the most important q of the dth order cumulants. Note that these features cannot be extracted by simply computing all the correlations and performing a PCA on such preprocessed data, since the required computation is prohibitive when d is not small ($d > 2$): for N-dimensional input patterns, the dimensionality of the feature space F is $(N+d-1)!/d!(N-1)!$. However, this is facilitated by the introduction of a polynomial kernel, as a polynomial kernel with degree d ($k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d$) corresponds to the dot product of two monomial mappings, Φ_d [3], [4]

$$\begin{aligned}(\Phi_d(\mathbf{x})\cdot\Phi_d(\mathbf{y})) &= \sum_{i_1,\,\ldots,\,i_d=1}^{N} x_{i_1}\cdot\cdots\cdot x_{i_d}\cdot y_{i_1}\cdot\cdots\cdot y_{i_d}\\ &= \left(\sum_{i=1}^{N} x_i\cdot y_i\right)^d = (\mathbf{x}\cdot\mathbf{y})^d.\end{aligned}$$

III. Face Recognition

Linear SVMs are used as the face recognizer. SVM is a natural choice because of its robustness even in the absence of a rich set of training examples. The success of SVMs in face recognition [6] and other related problems [7], [8] provides us with further motivation to rely on SVMs as the recognizer. Since SVMs have originally been proposed for two-class classification, their basic scheme is extended to multiclass face recognition by adopting *one-per-class* decomposition. This works by constructing a SVM ω_r for each class r that first separates that class from all the other classes and then uses an expert F to arbitrate between each SVM output in order to produce the final decision.

A *max-selector* is the simplest form of arbitrator. If $\mathbf{g} = (g^1, \ldots, g^R)^T$ denotes the output of a system of R one-per-class SVMs, the *max-selector* picks class r for the input $\mathbf{x}$, which then maximizes g^r as defined by

$$F = \arg\max_r(\mathbf{g}).$$

However, a max-selector suffers from a scaling problem, because it assumes that all the gs are on the same scale, which is not the case for SVMs. If the SVM is trained to produce outputs for the SVs as ±1, the scale is not robust as it only depends on a few data, often including outliers [9]. In a max-selector, the output class is determined by choosing the maximum of all the SVM outputs. However, the outputs of the remaining SVMs, other than the winner, also carry certain information. Moreover, the mean of g can vary significantly according to the class of input [9]. This knowledge can be used to improve the overall recognition performance. In [9], a stacking technique based on linear mapping was applied for normalizing $\mathbf{g}$. While this technique shows significant improvements over the bare one-per-class decomposition, preliminary experiments have indicated that a linear normalization is often insufficient for face recognition as the relation among gs becomes nonlinear. In this letter, after uniformly scaling $\mathbf{g}$ by applying a tangent hyperbolic function $\mathbf{h} = (h, \ldots, h)^T$, a nonlinear mapping M: $\mathbf{R}^R \rightarrow \mathbf{R}^R$ is used to aggregate the answers of all the SVMs into a score for each class. Thus, the arbitrator can be defined by

$$F = \arg\max_r(M(\mathbf{h}(\mathbf{g}))).$$

A two-layer neural network, composed of a hidden layer of size three with a tangent hyperbolic activation function, is adopted for mapping M. The network is designed to minimize the mean square error between $\mathbf{h}(\mathbf{g}(\mathbf{x}))$ and the desired output $\mathbf{y} = (-1, \ldots, +1, \ldots -1)^T$, and trained using a backpropagation algorithm.

The recognition is then performed by extracting a facial feature vector using a kernel PCA and classifying it using the SVMs.

IV. Experimental Results

To assess the proposed method, experiments were performed using the ORL (Olivetti Research Laboratory) database. This database includes ten different images of 40 distinct subjects. For some of the subjects, the images were taken at different times, plus there are variations in facial expression (open/closed eyes, smiling/nonsmiling) and facial details (glasses/no glasses).

The original face images were all sized 92 × 112 with a 256-level gray scale. The gray scale was linearly normalized to lie within $[-1, 1]$. The experiments were performed with five training images and five test images per person for a total of 200 training images and 200 test images. There was no overlap between the training and test sets. Since the recognition performance is affected by the selection of the training images, the reported results were obtained by training 20 recognizers with different training examples (random selection of five images from ten per subject, out of a total of $10!/5! = 30\,240$ selections) and selecting the average error over all the results.

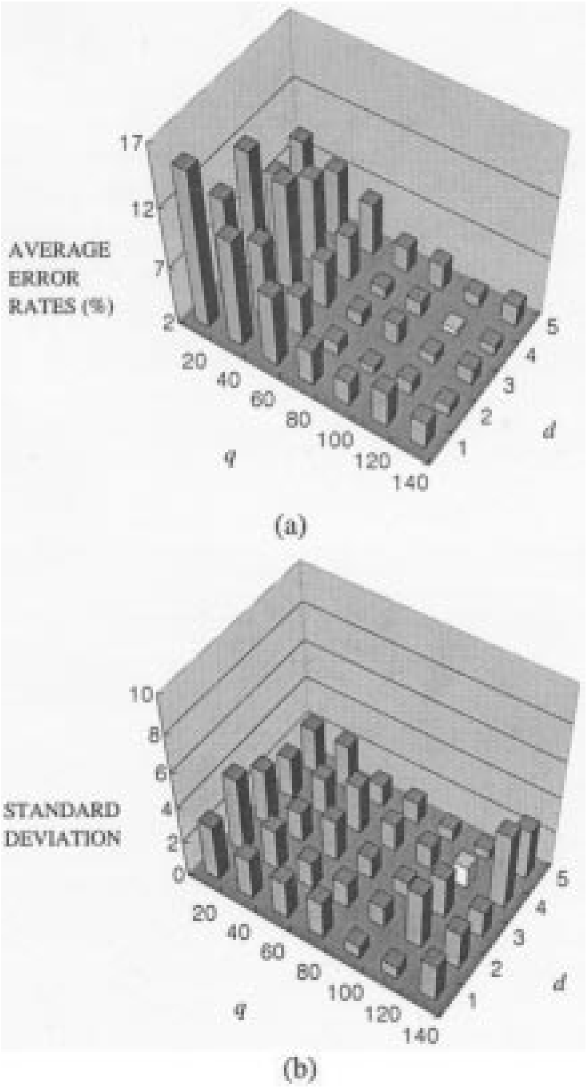

Fig. 2. Experimental results with different polynomial degrees d and number of eigenvectors q: (a) average error rates and (b) standard deviation.

The tuned parameters included the polynomial degree d and number of eigenvalues q as follows: $d = \{1, 2, 3, 4, 5\}$ and $q = \{20, 40, 60, 80, 100, 120, 140\}$. Fig. 2 shows the results: (a) average error rates and (b) standard deviation of error rates. A tendency of smaller error rates for the higher d and q was observed, while a saturation point was reached when $d \geq 4$ and $q \geq 100$. The best performance of 2.5% error rate was obtained when $(d, q) = (4, 120)$ (marked with a white bar), which clearly outperformed the linear PCA (equivalent to a first-degree polynomial kernel PCA: 4.1% error rate). Table I shows a summary of the performance of various systems for which results using the ORL database are available [6], [10]–[12]. The proposed method produced better results and a significant reduction in the error rate (16.7%) compared with the performances of the best existing system-linear SVMs [6]. The 2.5% error rate reported for the proposed method was an average of 20 simulations, however, the individual simulations had given error rates as low as 1.5%.

TABLE I
PERFORMANCES OF VARIOUS SYSTEMS

Systems	Error rates (%)
Eigenfaces [11]	10.0
Pseudo-2D HMM [11]	5.0
Probabilistic decision-based neural network [12]	4.0
Convolutional neural network [10]	3.8
Linear SVMs [6]	3.0
Kernel PCA	2.5

V. CONCLUSION

A kernel PCA-based face feature extraction method was presented, whereby the use of a polynomial kernel enables the principal components to be computed within the product space of the input pixels making up a facial pattern. Using SVMs as the recognizer, experimental results with the ORL database confirm the effectiveness of the proposed method.

REFERENCES

[1] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.

[2] J. Zhang, Y. Yan, and M. Lades, "Face recognition: Eigenface, elastic matching, and neural nets," *Proc. IEEE*, vol. 85, pp. 1423–1435, Sept. 1997.

[3] B. Schölkopf, A. Smola, and K. Müller, "Non-linear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, 1998.

[4] K. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Networks*, vol. 12, pp. 181–201, Mar. 2001.

[5] B. Schölkopf, A. Smola, and K. Müller, "Kernel principal component analysis," in *Advances in Kernel Methods-Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 327–352.

[6] G. D. Guo, S. Z. Li, and K. L. Chan, "Face recognition by support vector machines," in *Proc. Int. Conf. Automatic Face and Gesture Recognition*, 2000, pp. 196–201.

[7] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 1997, pp. 130–136.

[8] K. I. Kim, K. Jung, S. H. Park, and H. J. Kim, "Supervised texture segmentation using support vector machines," *Electron. Lett.*, vol. 35, no. 22, pp. 1935–1936, 1999.

[9] E. Mayoraz and E. Alpaydin, "Support vector machines for multi-class classification," Dalle Molle Inst. for Percept, Artif. Intell., Switzerland, Tech. Rep. IDIAP-PR 98-06, 1998.

[10] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Networks*, vol. 8, pp. 98–113, Jan. 1997.

[11] F. S. Samaria, "Face recognition using hidden Markov models," Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., 1994.

[12] S.-H. Lin, S.-Y. Kung, and L.-J. Lin, "Face recognition/detection by probabilistic decision-based neural network," *IEEE Trans. Neural Networks*, vol. 8, pp. 114–132, Jan. 1997.