Proceedings of the 2003 IEEE
International Conference on Robotics,Intelligent Systems and Signal Processing
Changsha, China - October 2003

Facial Expression Analysis By Using KPCA

Zhong Jin
Department of Computer Science
Nanjing University of Sci. and Tech.
Nanjing, People's Republic of China
Jinzhong@mail.njust.edu.cn

Franck Davoine
Laboratoire Heudiasyc, UMR CNRS 6599
Universite de Technologie de Compiegne
BP 20529, 60205 Compiegne, France
Franck.Davoine@hds.utc.fr

Zhen Lou
Department of Computer Science
Nanjing University of Sci. and Tech.
Nanjing, People's Republic of China

Abstract

This paper discussed a problem of robustness of existing kernel principal component analysis (KPCA) and proposed a new approach to do facial expression analysis by using KPCA. Experimental results on CMU facial expression image database and Yale database are encouraging.

1 Introduction

Facial expression is one of the most powerful, natural, and immediate means for human beings to communicate their emotions and intentions. Automatic facial expression analysis has attracted the interest of many computer vision researchers for its potential applications to human behavior interpretation and multimodal human-computer interface. An overview of the early work was given by Samel and Iyengar [1] in 1992. Pantic and Rothkrantz gave a survey of the recent work on automatic analysis of facial expression [2].

The Facial Action Coding System (FACS) [3] is probably the most known study on facial activity. It provides the descriptive power necessary to describe the details of facial expression [4]. In a general way, a facial expression is a combination of action units (AUs) of FACS. Automatic recognition of AUs is a difficult problem [5], and relatively little work has been reported.

Most of the studies on automatic facial expression analysis perform an emotional classification of the following six basic emotions [6]: happiness, sadness, surprise, fear, anger and disgust. Principal Component Analysis (PCA) has played a fundamental role in feature extraction and dimensionality reduction while Linear Discriminant Analysis (LDA) is powerful to extract discriminant information [7, 8, 9]. Independent Component Analysis (ICA) can utilize the high-order dependencies in addition to the second-order dependencies among the pixels [10].

Recently, Scholkopf et al. [11, 12] proposed a novel approach so called Kernel Principal Component Analysis (KPCA). It can be used to extract nonlinear principal components efficiently instead of carrying out the nonlinear mapping explicitly.

In this paper, we discussed a problem of robustness of KPCA and proposed a new approach to do facial expression analysis by using KPCA. The rest of this paper is organized as follows. In Section 2, background on PCA and KPCA is introduced. A new approach using KPCA is proposed in Section 3. In Section 4, experiments on CMU database and Yale database are performed and discussed. Lastly, we draw our conclusion and future work in Section 5.

2 Background Review

2.1 Principal Component Analysis

Principal component analysis (PCA) is a popular technique for feature extraction and dimensionality reduction. Suppose that x_i $(i = 1, 2, \cdots, N)$ is a set of centered observations of an m-dimensional zero-mean variable. Let

$$\sum_{i=1}^{N} x_i = 0 \qquad (1)$$

0-7803-7925-X/03/$17.00 ©2003 IEEE

The covariance matrix of the variable can be estimated as follows:

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T \tag{2}$$

PCA aims at making the covariance matrix Σ in (2) be diagonal. It leads to an eigenvector problem:

$$\lambda v = \Sigma v \tag{3}$$

Since

$$\Sigma v = \frac{1}{N}\sum_{i=1}^{N}(x_i \cdot v)x_i, \tag{4}$$

all solutions v in Eq. (3) must lie in the span of $x_1, \cdots, x_N$. Hence, Eq. (3) can be shown to be equivalent to

$$\lambda(x_i \cdot v) = (x_i \cdot \Sigma v) \tag{5}$$

for all $i = 1, 2, \cdots, N$.

2.2 Kernel PCA

The basic idea of kernel principal component analysis (KPCA) is to map the input data into some high dimensional feature space via a nonlinear function and then perform PCA on the high dimensional space. We map the input x to a high dimensional feature space F via a nonlinear function Φ:

$$\Phi : x \longrightarrow \Phi(x) \in F \tag{6}$$

Assume that the mapped observations are centered, i.e.,

$$\sum_{i=1}^{N}\Phi(x_i) = 0 \tag{7}$$

in the feature space F. Otherwise, see [11].

Then the covariance matrix in F is:

$$\bar{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}\Phi(x_i)\Phi(x_i)^T \tag{8}$$

The corresponding eigenvalue problem is

$$\bar{\lambda}u = \bar{\Sigma}u \tag{9}$$

By the same argument as in (4), all solutions u lie in the span of $\Phi(x_1), \cdots, \Phi(x_N)$. In other words, there exist coefficients $\alpha_i (i = 1, \cdots, N)$ such that

$$u = \sum_{i=1}^{N}\alpha_i \Phi(x_i). \tag{10}$$

Moreover, as Eq. (3) is equivalent to Eq. (5), Eq. (9) is equivalent to

$$\bar{\lambda}(\Phi(x_j) \cdot u) = (\Phi(x_j) \cdot \bar{\Sigma}u) \tag{11}$$

for all $j = 1, 2, \cdots, N$.

By combining Eqs. (10) and (11), we have

$$\bar{\lambda}\sum_{i=1}^{N}\alpha_i(\Phi(x_k) \cdot \Phi(x_i)) =$$
$$\frac{1}{N}\sum_{i'=1}^{N}\sum_{i=1}^{N}\alpha_i(\Phi(x_{i'}) \cdot \Phi(x_i))(\Phi(x_j) \cdot \Phi(x_{i'}))$$

for all $j = 1, 2, \cdots, N$.

Define an $N \times N$ matrix K by

$$K = (K_{ij})_{N\times N}, \tag{12}$$

where

$$K_{ij} := (\Phi(x_i) \cdot \Phi(x_j)) \tag{13}$$

for all $i, j = 1, \cdots, N$. Then, we get

$$N\bar{\lambda}K\alpha = K^2\alpha, \tag{14}$$

where α denotes the column vector with entries $\alpha_1, \cdots, \alpha_N$. It is shown in [11] that the following eigenvalue problem

$$\lambda\alpha = K\alpha, \tag{15}$$

gives us all solutions α of the eigenvalue problem of Eq. (14), where $\lambda = N\bar{\lambda}$ is an eigenvalue of the matrix K.

By the mapping Φ in Eq. (6), we assume that an original nonlinear problem in the input space can be transformed to a linear problem in the high dimensional feature space F. However, it is impossible to compute the matrix K directly without carrying out the mapping Φ. Fortunately, for certain mappings Φ and corresponding feature spaces F, there is a highly effective trick for computing the dot product $(\Phi(x) \cdot \Phi(y))$ in feature spaces using kernel functions. If the mapping Φ satisfies the Mercer's condition [13], the dot product can be replaced by a kernel function as follows:

$$k(x, y) = (\Phi(x) \cdot \Phi(y)), \tag{16}$$

which allow us to compute the value of the dot product in the high dimensional feature space F without having to carry out the mapping Φ explicitly.

The following polynomial is a common used kernel function:

$$k(x, y) = (x \cdot y)^d, \tag{17}$$

where d is any positive integer.

2.3 The KPCA Algorithm

Here is the KPCA algorithm:

1. Select a kernel function $k(x,y)$. Calculate the dot product matrix

$$K = (k(x_i, x_j))_{N\times N}. \tag{18}$$

2. Solve the eigen-value problem of Eq. (15) to get n ($n \leq N$) non-zero eigen-values:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0 \tag{19}$$

and the corresponding eigen-vectors $\alpha^1, \alpha^2, \cdots, \alpha^n$.

3. For any given test sample x, compute the projections of $\Phi(x)$ on the eigen-vectors $u^1, \cdots, u^n$ in feature space F as follows:

$$z = \begin{bmatrix} \frac{1}{\sqrt{\lambda_1}} \sum_{i=1}^{N} \alpha_i^1 k(x_i, x) \\ \frac{1}{\sqrt{\lambda_2}} \sum_{i=1}^{N} \alpha_i^2 k(x_i, x) \\ \cdots\cdots \\ \frac{1}{\sqrt{\lambda_n}} \sum_{i=1}^{N} \alpha_i^n k(x_i, x) \end{bmatrix} \tag{20}$$

It is noted that in Step 3, the eigen-vectors u^j ($j = 1, \cdots, n$) in feature space F are known in form as follows according to Eq. (10):

$$u^j = \frac{1}{\sqrt{\lambda_j}} \sum_{i=1}^{N} \alpha_i^j \Phi(x_i), \quad (j = 1, \cdots, n) \tag{21}$$

where $\frac{1}{\sqrt{\lambda_j}}$ $(j = 1, \cdots, n)$ are the normalizing factors. Thus, the projection of $\Phi(x)$ on u^j is

$$\begin{aligned} (u^j \cdot \Phi(x)) &= \frac{1}{\sqrt{\lambda_j}} \sum_{i=1}^{N} \alpha_i^j (\Phi(x_i) \cdot \Phi(x)) \\ &= \frac{1}{\sqrt{\lambda_j}} \sum_{i=1}^{N} \alpha_i^j k(x_i, x) \end{aligned} \tag{22}$$

3 A new approach using KPCA

According to statistical theory, the covariance matrix estimator $\bar{\Sigma}$ of Eq. (8) will converge to the true covariance matrix of the input variable as the number of samples, N, becomes larger enough. Thus, as N becomes larger, we can obtain more robust solutions with the eigenvalue problem of Eq. (9).

On the other hand, the dot product matrix K of Eq. (18) will not have any convergence as the number of samples, N, becomes larger. Besides, as N increases, the size of the dot product matrix K increases linearly. Therefore, as N increases, we can not obtain robust solutions with the eigenvalue problem of Eq. (15).

Moreover, the application of Kernel PCA to a data set of thousand samples may create problem of numerical analysis. In fact, it is not feasible to compute eigenvectors of a matrix of a rank greater than one thousand with classical algorithms as Gauss-Jordan.

Our idea is to use L mean vectors instead of N samples for an L-class problem so that the dimension of the dot product matrix K decreases from $N \times N$ to $L \times L$. Suppose $\omega_1, \omega_2, \cdots, \omega_L$ are L known pattern class. We can compute the mean vector of each class ω_j as follows:

$$\bar{x}_j = \frac{1}{N_j} \sum_{x_i \in \omega_j} x_i \tag{23}$$

for $j = 1, \cdots, L$, where N_j is the number of training samples which belong to the class ω_j. According to statistical theory, $\bar{x}_j$ will converge to the expectation vector of the class ω_j as N becomes large enough.

We propose a new approach to do KPCA by using the mean vectors $\bar{x}_j$ ($j = 1, \cdots, L$) instead of using the samples x_i ($i = 1, \cdots, N$) directly. Here is the proposed algorithm by using KPCA:

a. Calculate the dot product matrix

$$K = (k(\bar{x}_i, \bar{x}_j))_{L\times L}. \tag{24}$$

b. Solve the eigen-value problem of Eq. (15) to get n ($n \leq L$) non-zero eigen-values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0$ and the corresponding eigen-vectors $\alpha^1, \alpha^2, \cdots, \alpha^n$.

c. For any given test sample x, compute the projections of $\Phi(x)$ on the eigen-vectors $u^1, \cdots, u^n$ in feature space F as follows:

$$z = \begin{bmatrix} \frac{1}{\sqrt{\lambda_1}} \sum_{j=1}^{L} \alpha_j^1 k(\bar{x}_j, x) \\ \frac{1}{\sqrt{\lambda_2}} \sum_{j=1}^{L} \alpha_j^2 k(\bar{x}_j, x) \\ \cdots\cdots \\ \frac{1}{\sqrt{\lambda_n}} \sum_{j=1}^{L} \alpha_j^n k(\bar{x}_j, x) \end{bmatrix} \tag{25}$$

In this way, the size of the dot product matrix K of Eq.(24) is fixed for an L class problem. Moreover, the matrix K will have a tendency to converge as the number of samples becomes large enough. Therefore, we can extract more robust features.

4 Experiments

From CMU-Pittsburgh AU-Coded Face Expression Database [14], 463 facial expression mask images can be obtained by using a spatial adaptive triangulation technique based on local Gabor filters [9]. Six facial expressions are concerned as follows: surprise, anger, sadness, joy, fear and disgust. The mask images are grayscale with a resolution of 60×70. 60 mask images in CMU database are shown in Fig. 1.

Yale database consists of 15 subjects. Each subject has 3 images with different facial expressions: surprise, joy and sadness. The background in the Yale images is removed. 45 facial expression mask images can be obtained by using a spatial adaptive triangulation technique based on local Gabor filters [9]. 30 mask images in Yale database are shown in Fig. 2.

With three different distance metrics, i.e., $L1$ norm, $L2$ norm and cosine angle, the following six classifiers are used in our experiments:

- L1 Min: The minimum classifier using L1 distance.
- L1 1NN: The nearest neighbor classifier using L1 distance.
- L2 Min: The minimum classifier using L2 distance
- L2 1NN: The nearest neighbor classifier using L2 distance .
- Cosine Max: The maximum classifier using cosine angle (correlation).
- Cosine 1NN: The nearest neighbor classifier using cosine angle (correlation).

In experiments, the first 210 mask images in CMU database are for training. There are 35 mask images for training for each facial expression.

The other 253 mask images in CMU database and all the 45 mask images in Yale database are for test.

Polynomial in Eq. (17) is used as kernel function. Experimental results of number of erroneous classification samples by using the proposed KPCA and the existing KPCA are listed in Table 1 and Table 2 respectively. Moreover, average recognition rates for any given d in Eq. (17) are calculated and listed in the last column in Table 1 and Table 2.

From Table 1 and Table 2, we have the following facts and discussions:

- There exists a problem of robustness with the existing KPCA for a larger d. From the average recognition rates, the proposed KPCA has a better performances than the existing KPCA when d becomes larger.
- For L1 Min, L2 Min, Cosine Max and Cosine 1NN, the numbers of erroneous classification samples by the proposed KPCA are similar to those by the existing KPCA. For L1 1NN and L2 1NN, the numbers of erroneous classification samples by the proposed KPCA are much smaller than those by the existing KPCA.

5 Conclusion

In this paper, we discussed a problem of robustness of the dot product matrix K in the existing KPCA and proposed a KPCA approach using the mean vectors of each class for an L class problem. Experimental results on CMU facial expression image database and Yale database are encouraging.

References

[1] A. Samel and P. A. Iyengar. Automatic recognition and analysis of human faces and facial expression: A survey. *Pattern Recognition*, 25(1):65–77, 1992.

[2] Maja Pantic and Leon J. M. Rothkrantz. Automatic analysis of facial expression: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1424–1445, 2000.

[3] P. Ekman and W. V. Friesen. *Facial Action Coding System (FACS): Manual.* : Consulting Psychologists Press, Palo Alto, 1978.

[4] Gianluca Donato, Marian Stewart Bartlett, Joseph C. Hager, Paul Ekman, and Terrence J. Sejnowski. Classifying facial actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):974–989, 1999.

[5] Ying li Tian, Takeo Kanade, and Jeffrey F. Cohn. Recognizing action units for facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):97–114, 2001.

[6] P. Ekman and W. V. Friesen. *Unmasking the Face.* Prentice Hall, New Jerdey, 1975.

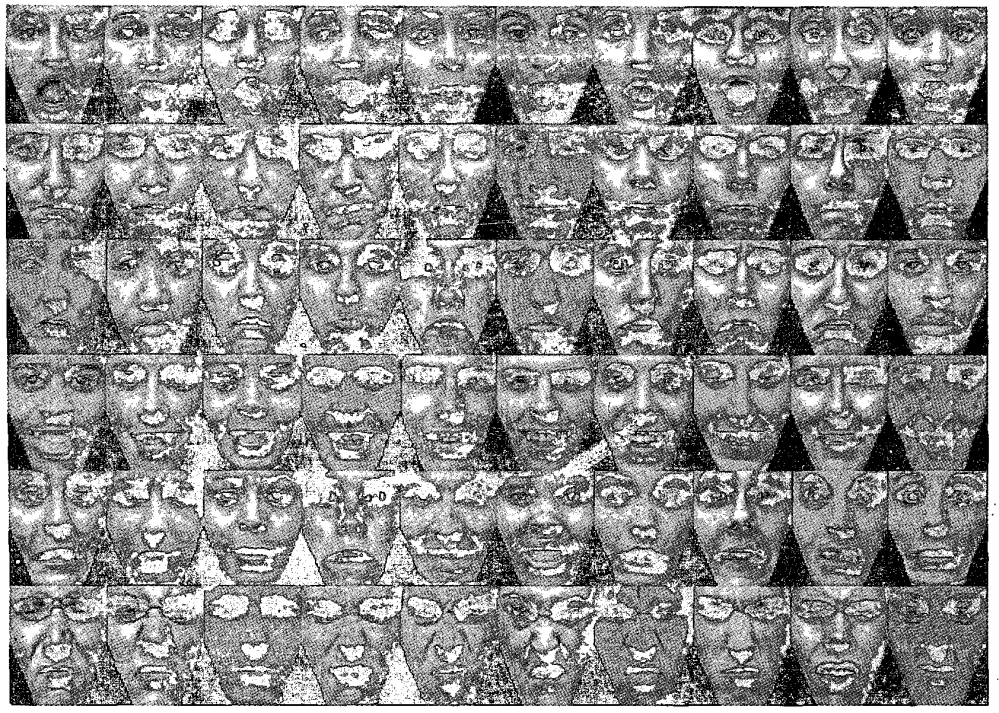

Figure 1. 60 mask images with 6 facial expressions: surprise, anger, sadness, joy, fear and disgust

[7] I. T. Jolliffe. *Principal Component Analysis.* Springer, New York, 1986.

[8] M. J. Lyons, J. Budynek, and S. Akamastu. Automatic classification of single facial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1357–1362, 1999.

[9] S. Dubuisson, F. Davoine, and M. Masson. A solution for facial expression representation and recognition. *Signal Processing: Image Communication*, 17(9):657–673, 2002.

[10] Marian Stewart Bartlett. *Face Image Analysis by Unsupervised Learning and Redundancy Reduction.* PhD thesis, University of California, San Diego, 1998.

[11] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[12] Klaus-Robert Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

[13] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, New York, 1995.

[14] Takeo Kanade, Jeffrey F. Cohn, and Yingli Tian. Comprehensive database for facial expression analysis. In *Proceeding of the Fourth International Conference of Face and Gesture Recognition*, pages 46–53, Grenoble, France, 2000.

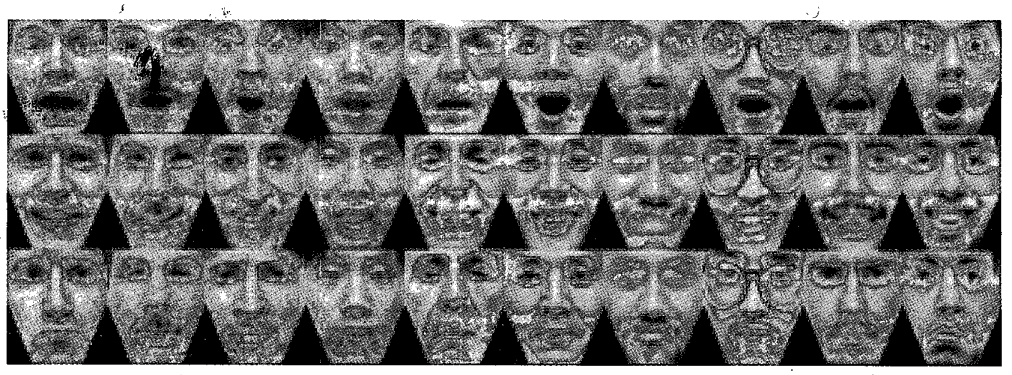

Figure 2. 30 mask images in Yale database with 3 facial expressions: surprise, joy and sadness

Table 1. Number of erroneous classification samples by using the proposed KPCA

Test Database	d	L1		L2		Cosine		Average Recognition Rate
		Min	*1NN*	*Min*	*1NN*	*Max*	*1NN*	
CMU	1	42	35	43	36	72	48	81.8%
	2	85	58	83	53	94	59	71.5%
	3	68	45	69	41	81	57	76.2%
	4	120	73	118	74	111	91	61.3%
Yale	1	13	14	12	16	17	23	64.8%
	2	20	21	20	21	26	26	50.4%
	3	17	16	16	16	21	22	60.0%
	4	25	21	21	20	32	29	45.2%

Table 2. Number of erroneous classification samples by using the existing KPCA

Test Database	d	L1		L2		Cosine		Average Recognition Rate
		Min	*1NN*	*Min*	*1NN*	*Max*	*1NN*	
CMU	1	42	72	43	60	57	56	75.0%
	2	78	89	100	75	107	61	66.4%
	3	61	92	90	72	86	65	69.3%
	4	253	253	210	95	126	77	33.2%
Yale	1	13	17	12	12	15	12	70.0%
	2	18	31	18	25	24	19	50.0%
	3	18	29	13	20	15	20	57.4%
	4	45	45	45	45	20	19	18.9%