Representing Image Matrices: Eigenimages vs. Eigenvectors

Daoqiang Zhang[1,2], Songcan Chen[1] and Jun Liu[1]

[1] Department of Computer Science and Engineering
Nanjing University of Aeronautics and Astronautics
Nanjing 210016, P.R. China
{dqzhang, s.chen}@nuaa.edu.cn
[2] National Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, P.R. China

Abstract. We consider the problem of representing image matrices with a set of basis functions. One common solution for that problem is to first transform the 2D image matrices into 1D image vectors and then to represent those 1D image vectors with *eigenvectors*, as done in classical principal component analysis. In this paper, we adopt a natural representation for the 2D image matrices using *eigenimages*, which are 2D matrices with the same size of original images and can be directly computed from original 2D image matrices. We discuss how to compute those eigenimages effectively. Experimental result on ORL image database shows the advantages of eigenimages method in representing the 2D images.

1 Introduction

Principal component analysis (PCA) [3] is a well-known data representation technique widely used in pattern recognition and signal processing [2], [4], [5]. When using PCA to represent 2D images, we have to first transform the 2D image matrices into 1D image vectors, then compute the corresponding 1D *eigenvectors* from the sample covariance matrix to represent any image vector as a weighted sum of a set of eigenvectors, and finally retransform the 1D sum vector back to a 2D matrix to obtain the reconstructed image. However, the vectorized representation of image has the following disadvantages. Firstly, concatenating a 2D image often leads to a corresponding high-dimensional 1D vector, which makes it very difficult and time-consuming to compute the corresponding eigenvectors. Secondly, such a concatenation for 2D image may cause the loss of some structure information hiding in original 2D image.

To overcome those problems, Yang *et al*. proposed two-dimensional PCA (2DPCA) [6] which directly compute eigenvectors of the so-called *image covariance matrix* without matrix-to-vector conversion. However, 2DPCA still use 1D eigenvectors to represent image matrices which will be detailed in the next section, if we view each row in the 2D image matrices as an image vector, then 2DPCA can be approximately seen

as conventional PCA operated on those row vectors. Thus the obtained 1D eigenvectors only reflect the row information in original images, and we call them as *row eigenvectors*. And each row of original 2D images is represented as a weighted sum of a set of those row eigenvectors. That is, 2DPCA represents 2D image matrices using 1D eigenvector in essence. A more recent development on that aspect is the generalized low rank approximations of matrices (GLRAM) [7] proposed by Ye. In GLRAM, two groups of 1D eigenvectors $L=\{l_1, l_2, \ldots, l_p\}$ and $R=\{r_1, r_2, \ldots, r_q\}$ are jointly used to represent 2D images. However, unlike in PCA and 2DPCA, GLRAM cannot find the global optimal solutions for L and R, and instead an iterative algorithm is used to find locally optimal solutions. Thus GLRAM needs a relatively more computational cost than 2DPCA.

In this paper, we make further step along with 2DPCA. As in 2DPCA, we compute the row eigenvectors r_j ($1=j=p$) by viewing each row as an image vector. Similarly, we can obtain the *column eigenvectors* l_i ($1=i=q$) by viewing each column as an image vector. Then we define an *eigenimage* as the outer-product between l_i and r_j and represent original images with those 2D eigenimages. Although the proposed method is very similar to GLRAM, there do exist at least two differences. First, although GLRAM also compute the two groups of eigenvectors L and R, there is no explicit definition for the eigenimage and thus no discussion on the interesting characteristics on eigenimages, as shown in the next section. Moreover, the eigenvectors L and R in GLRAM are computed with iterative steps and a good initial value is needed, while in our method L and R are computed both in closed-form and in a parallel manner. Experimental result on ORL image database shows the advantages of the proposed method in representing the 2D images compared with PCA, 2DPCA and GLRAM.

2 Image representation with eigenimages

Let $A^k \in R^{r \times c}, k = 1,\ldots,n$ denote original image matrices, compute the mean image as $\overline{A} = \frac{1}{n}\sum_{k=1}^{n} A^k$, then we get the corresponding centered images $\overline{A}^k = A^k - \overline{A}$. We first compute the row eigenvectors as follows.

Concatenating the n image matrices $\overline{A}^k$ into an nr by c single matrix according to the following order: $\overline{A}_{row} = \left((\overline{A}^1)^T, (\overline{A}^2)^T, \ldots, (\overline{A}^n)^T\right)^T$, then

$$\left(\overline{A}_{row}\right)^T \left(\overline{A}_{row}\right) = \sum_{k=1}^{n} \left(\overline{A}^k\right)^T \left(\overline{A}^k\right) = \sum_{k=1}^{n}\sum_{j=1}^{r} \left(\overline{a}_j^k\right)^T \left(\overline{a}_j^k\right) = \sum_{i=1}^{nr} b_j^T b_j \,. \qquad (1)$$

Here $\overline{a}_j^k$ is the *j*-th row of image matrix $\overline{A}^k$ and b_j is the *j*-th row of $\overline{A}_{row}$. Note that Eq. (1) is in fact *n* times of the *image covariance matrix* $\Sigma = \frac{1}{n}\sum_{k=1}^{n}\left(\overline{A}^k\right)^T\left(\overline{A}^k\right)$ used in 2DPCA and nr times of the *covariance matrix* $\Sigma' = \frac{1}{nr}\sum_{i=1}^{nr} b_j^T b_j$ when viewing each row b_j as an image vector. However, according to matrix theory multiplying a matrix by a constant does not change its eigenvectors [1], so Σ has the same eigenvectors as Σ'. That is, what 2DPCA really computes is only the eigenvectors corresponding to nr row vectors. We name those *c* eigenvectors as *row eigenvectors* and denote them as $r_1, r_2, ..., r_c$. Here r_1 is the eigenvector corresponding to the largest eigenvalue and r_c the eigenvector corresponding to the smallest eigenvalue.

Following a similar procedure, we can obtain the column eigenvectors. By concatenating the *n* image matrices $\overline{A}^k$ into an *r* by *nc* single matrix according to the following order: $\overline{A}_{col} = \left(\overline{A}^1, \overline{A}^2, ..., \overline{A}^n\right)$, we have

$$\left(\overline{A}_{col}\right)\left(\overline{A}_{col}\right)^T = \sum_{k=1}^{n}\left(\overline{A}^k\right)\left(\overline{A}^k\right)^T = \sum_{k=1}^{n}\sum_{j=1}^{c}\left(\overline{a}_j^k\right)\left(\overline{a}_j^k\right)^T = \sum_{i=1}^{nc} b_j b_j^T . \quad (2)$$

Here $\overline{a}_j^k$ is the *j*-th column of image matrix $\overline{A}^k$ and b_j is the *j*-th column of $\overline{A}_{col}$. By computing the eigenvectors of Eq. (2), we obtain the *column eigenvectors*. And we denote the *r* column eigenvectors as $l_1, l_2, ..., l_r$. Here l_1 is the eigenvector corresponding to the largest eigenvalue and l_r the eigenvector corresponding to the smallest eigenvalue.

Now we are in a position to introduce the concept of *eigenimage*. Given *c* row eigenvectors $r_1, r_2, ..., r_c$ and *r* column eigenvectors $l_1, l_2, ..., l_r$, define *eigenimage* as

$$E_{ij} = l_i \cdot r_j^T, (1 \le i \le r, 1 \le j \le c) . \quad (3)$$

It is easy to verify that the *eigenimage* E_{ij} has the following characters:

(1) E_{ij} is a 2D matrix with the same size of original image, i.e. $r \times c$.

(2) The *intrinsic dimensionality* or *rank* of E_{ij} is no more than 1.

(3) Any two eigenimages E_{kl} and E_{mn}, satisfying

$$trace\left(\left(E_{kl}\right)^T\left(E_{mn}\right)\right) = \begin{cases} 1, & k = m \text{ and } l = n \\ 0, & otherwise \end{cases} . \quad (4)$$

(4) Any image A^k $(1 \le k \le n)$ can be represented by a weighted sum of eigenimages plus the mean image $\overline{A}$ as

$$A^k = \sum_{i=1}^{r}\sum_{j=1}^{c} D_{ij}^k E_{ij} + \overline{A}. \quad (5)$$

(5) The coefficients D_{ij}^k in Eq. (5) is computed using

$$D_{ij}^k = l_i^T \overline{A}^k r_j, (1 \le i \le r, 1 \le j \le c). \quad (6)$$

From Eqs. (5) and (6), we accurately represent the original image A^k with rc eigenimages. In fact, we can approximately represent image A^k with partial ($< rc$) eigenimages. If we choose the $q(<c)$ row eigenvectors corresponding to the largest q eigenvalues of Eq. (1) and the $p(<r)$ column eigenvectors corresponding to the largest p eigenvalues of Eq. (2). Then we can approximately represent image A^k as $\hat{A}^k = \sum_{i=1}^{p}\sum_{j=1}^{q} D_{ij}^k E_{ij} + \overline{A}$ with only pq eigenimages. And $\hat{A}^k$ is also called reconstructed image in image compression.

We conclude this section by giving the detailed description of algorithm Eigenimage as follows.

Algorithm 'Eigenimage':

Step 1: Input n $r \times c$ image matrices $A^1, A^2, ..., A^n$, and fix $p(\le r)$ and $q(\le c)$.

Step 2: Compute the mean image $\overline{A} = \frac{1}{n}\sum_{k=1}^{n} A^k$ and generate $\overline{A}^k = A^k - \overline{A}, (1 \le k \le n)$.

Step 3: Concatenate $\overline{A}^k (1 \le k \le n)$ into $\overline{A}_{row} = \left((\overline{A}^1)^T, (\overline{A}^2)^T, ..., (\overline{A}^n)^T\right)^T$, and compute the q row eigenvectors $\{r_1, r_2, ..., r_q\}$ corresponding to the q largest eigenvalues of $\left(\overline{A}_{row}\right)^T \left(\overline{A}_{row}\right)$.

Step 4: Concatenate $\overline{A}^k (1 \le k \le n)$ into $\overline{A}_{col} = \left(\overline{A}^1, \overline{A}^2, ..., \overline{A}^n\right)$, and compute the p row eigenvectors $\{l_1, l_2, ..., l_p\}$ corresponding the p largest eigenvalues of $\left(\overline{A}_{col}\right)\left(\overline{A}_{col}\right)^T$.

Step 5: Compute the eigenimages $E_{ij} = l_i \cdot r_j^T, (1 \le i \le p, 1 \le j \le q)$.

Step 6: Compute the coefficients

$D_{ij}^k = l_i^T \bar{A}^k r_j, (1 \le i \le p, 1 \le j \le q, 1 \le k \le n)$.

Step 7: Compute the reconstructed image

$$\hat{A}^k = \sum_{i=1}^{p} \sum_{j=1}^{q} D_{ij}^k E_{ij} + \bar{A}, (1 \le k \le n) .$$

3 Experimental results

In this section, we compare the performances of the proposed Eigenimage method with those of PCA, 2DPCA, and GLRAM on ORL face database. We are more concerned on the image reconstruction quality measured by PSNR and the time required by algorithms. All of our experiments are carried out on a PC machine with P4 1.7GHz CPU and 256MB memory.

We use the cropped ORL database in our first experiment. The cropped ORL face database consists of 400 different grey scale images of 40 different persons, with a resolution of 64×64. PCA first concatenating each 64×64 image to a 4094-dimensional vector, and then compute the corresponding eigenvectors, which is a very time-consuming procedure because the 4094×4094 covariance matrix is very huge. In our experiments, we adopt a trick introduced in [1] which need only computing the eigenvectors of an n×n matrix, and n is the size of face database. The left part of Fig. 1 shows the 36 eigenvectors of PCA corresponding to the largest 36 eigenvalues. Note these 1D eigenvectors are retransformed back to 2D images for visualization. We can see from the figure that these eigenvectors looks like the faces, i.e. they are dependent to the input data. On the other hand, the right part of Fig. 1 shows the 36 eigenimages of the Eigenimage method. We are surprised to see that, unlike PCA, these eigenimages do not look like faces any more, but show themselves with some 'regular' strip or block structures.

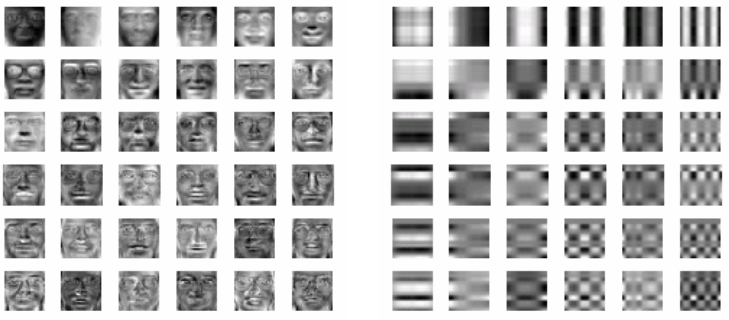

Fig. 1. Eigenvectors (Left) and Eigenimages (Right) on ORL database.

Table 1. Comparisons of performances of four methods on ORL database

Methods	PSNR	CR	Time (s)
PCA	25.19	11.39	8.75
2DPCA	24.96	10.64	0.28
GLRAM	28.60	12.42	6.14
Eigenimage	28.58	12.42	0.59

Table 1 gives the detailed comparisons of four methods concerning the image reconstruction quality (measured with PSNR), compression ration (CR) and the required time. Here PCA and 2DPCA use 32 and 6 eigenvectors respectively, while GLRAM and Eigenimage both use 18 row and column eigenvectors. From Table 1, 2DPCA has the fastest speed but its compressed image quality is the worst under the same compression ratio, because it needs too many coefficients to represent an image. On the opposite, GLRAM has the best compressed image quality but it takes relatively more time, because there exist iterative steps in its solving procedure. The proposed Eigenimage achieves a good tradeoff between the compressed image quality and the speed. As seen from Table 1, the compressed image quality of Eigenimage is only slightly (about 0.02 db) worse than that of GLRAM, but its speed is tens of times higher than that of GLRAM. In fact, the required time of Eigenimage is about 2 times of 2DPCA, because Eigenimage can be seen as performing 2 times of 2DPCA in row and column, respectively.

Acknowledgements

This work was supported by China Postdoctoral Science Foundation, Jiangsu Planned Projects for Postdoctoral Research Funds, and National Science Foundation of China under the Grant No. 60473035.

References

1. Golub, G.H., Van Loan, C.F.: Matrix Computations. The Johns Hopkins University Press, MD (1996)
2. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall (1999)
3. Jolliffe, I.T.: Principal Component Analysis. Springer-Verlag, New York (1986)
4. Kirby, M., Sirovich, L.: Application of the KL Procedure for the Characterization of Human Faces. IEEE Trans. PAMI 12 (1990) 103-108
5. Turk, M., Pentland, A.: Eigenfaces for Recognition. J. Cognitive Neurosci. 3 (1991) 71-86
6. Yang, J., Zhang, D., et al.: Two-dimensional PCA: a New Approach to Appearance-based Face Representation and Recognition. IEEE Trans. PAMI 26 (2004) 131-137
7. Ye, J.: Generalized Low Rank Approximation of Matrices. In: Proceedings of ICML (2004)