PCA and kernel PCA using polynomial filtering: a case study on face recognition*

E. Kokiopoulou and Y. Saad
Comp. Sci. & Eng. Dept
University of Minnesota
{kokiopou,saad}@cs.umn.edu

November 16, 2004

Abstract

Principal component analysis (PCA) is an extensively used dimensionality reduction technique, with important applications in many fields such as pattern recognition, computer vision and statistics. It employs the eigenvectors of the covariance matrix of the data to project it on a lower dimensional subspace. Kernel PCA, a generalized version of PCA, performs PCA implicitly in a nonlinearly transformed feature space. In many cases, experiments show that kernel PCA is more effective than conventional PCA.

However, the requirement of PCA eigenvectors is a computational bottleneck which poses serious challenges and limits the applicability of PCA-based methods, especially for real-time computations. This paper proposes an alternative framework, relying on polynomial filtering which enables efficient implementations of both PCA and its kernelized version. Further improvements are achieved when polynomial filtering is combined with wavelet transforms to obtain sparse representations of images. We showcase the applicability of the proposed scheme on face recognition. In particular, we consider the eigenfaces and kernel eigenfaces methods which employ PCA and kernel PCA respectively. The numerical experiments reported indicate that the proposed technique competes with the PCA - based methods in terms of recognition rate, while being much more efficient in terms of computational and storage cost.

Keywords Principal Component Analysis, Polynomial Filtering, Kernels.

1 Introduction

Principal component analysis (PCA) [8] is one of the most popular dimensionality reduction techniques. It has numerous applications in many areas such as pattern recognition, computer vision, statistics and data analysis. PCA has been successfully applied in automated face recognition [22], resulting in the so called method of *eigenfaces* introduced by Kirby and Sirovich [10], Sirovich and Kirby [19] and Turk and Pentland [16], [20]. The eigenfaces method is one of the most popular appearance-based holistic approaches (see e.g., [1],[20]) which employs PCA on the covariance matrix C, constructed by the training data.

Kernel eigenfaces [9], [21] is a nonlinear generalized version of the eigenfaces method which employs kernel PCA [18] for face recognition. The input data points are nonlinearly transformed into a high dimensional feature space F. Then, linear PCA is implicitly performed in the feature space F via the "kernel trick": using Mercer kernels ([14]) one can efficiently compute the dot product between two vectors $x, y \in F$, as a function of the dot product of their corresponding data points in the input space. Kernel PCA (KPCA) is able to capture the nonlinear correlations among data points, and in some cases has been more successful than conventional PCA.

Typical implementations of the eigenfaces and kernel eigenfaces methods rely upon eigendecomposition of the covariance matrix. However, when the datasets are dynamic and of large scale, the applicability of the above methods is limited due to their high computational cost (which is $O(n^3)$ for dense matrices). This is even more evident in the case of real-time and adaptive algorithms (see e.g. [15]). In these cases, the eigendecomposition must be updated frequently and the time constraints are very strict. To that end, a lot of research efforts have been devoted to efficient eigenspace update schemes such as the one proposed in [7].

In this paper we propose a new implementation scheme which approximates directly the similarity score without computing the eigendecomposition of C or any other matrix decomposition. Denoting by A the data matrix in the input space, the new method relies on

*Work supported by the Minnesota Supercomputing Institute

polynomial filtering, where a well defined polynomial ψ of the matrix $AA^\top$ or $A^\top A$ is applied on the new face image and yields an approximation to the similarity score that is very close to the one obtained using eigendecomposition. The polynomial ψ is chosen appropriately such that it is a good approximation of the step function. The quality of the recognition compared with PCA methods, depends on how closely the polynomial ψ approximates the step function.

The polynomial filtering framework was applied successfully in [11] for dimensionality reduction in information retrieval. In this paper we showcase the applicability of this technique in a different context, that of face recognition. We claim that the proposed framework can be applied in any method employing PCA or kernel PCA to estimate similarities among data vectors.

Numerical experiments indicate that the proposed framework is quite close to the PCA methods in terms of recognition rate without suffering from their computational and storage limitations. The efficiency of the method can be further improved if we use sparse representations of image data obtained from orthogonal wavelet decompositions.

The remaining sections of this paper are organized as follows: Section 2 provides an overview of the eigenfaces method using eigenvalue decomposition. In Section 3 the eigenfaces method is interpreted in terms of Singular Value Decomposition (SVD). Next, in Section 4 the implementation of face recognition using eigenfaces, via polynomial filtering is described. Section 5 analyzes the kernel eigenfaces method and introduces the implementation of kernel eigenfaces using polynomial filtering in the transformed feature space. Section 7 describes the combination of polynomial filtering with sparse representations of the images using wavelets. Finally, Section 8 provides a series of numerical results verifying the practical advantages of the proposed scheme.

2 The method of eigenfaces

2.1 Construction of the face space Suppose that a face image consists of N pixels, so it can be represented lexicographically by a vector x of dimension N. Let $\{x_i | i = 1, \ldots, M\}$ be the training set of face images. The mean face is given by

$$\mu = \frac{1}{M} \sum_{i=1}^{M} x_i. \tag{2.1}$$

The covariance matrix of the translated training data is

$$C = \frac{1}{M} AA^\top \in R^{N \times N}, \tag{2.2}$$

where $A = [\tilde{x}_1, ..., \tilde{x}_M] \in R^{N \times M}$ is the matrix of the translated data points

$$\tilde{x}_i = x_i - \mu,\ i = 1, \ldots, M. \tag{2.3}$$

The eigenvectors $u_l,\ l = 1, \ldots, M$ of the covariance matrix C are usually called "eigenfaces", since they resemble faces when reshaped and illustrated in a pictorial fashion. In practice only a small number, say k, of eigenvectors corresponding to the largest eigenvalues are computed and then used for performing Principal Component Analysis (PCA) for face identification. The subspace spanned by the eigenfaces is called face space.

2.2 Face recognition using eigenfaces The face recognition procedure consists of two stages; the training stage and the recognition stage. In the training stage each face image x_i of the known individuals is projected on the face space and a k-dimensional vector P_i is obtained

$$P_i = U_k^\top (x_i - \mu),\ i = 1, \ldots, M, \tag{2.4}$$

where $U_k = [u_1, \ldots, u_k]$ is the matrix with orthonormal columns, which are the eigenvectors associated with the k largest eigenvalues.

In the recognition stage, the new image $x \in R^N$ to be processed, is translated and then projected into the face space to obtain the vector

$$P_x = U_k^\top (x - \mu). \tag{2.5}$$

The distance between P_x and each face image is defined by

$$\begin{aligned} d_i^2 &= \|P_x - P_i\|_2^2 \\ &= \|P_x\|_2^2 + \|P_i\|_2^2 - 2P_x^\top P_i,\ i = 1, \ldots, M, \end{aligned} \tag{2.6}$$

where $\|.\|_2$ is the Euclidean norm. Furthermore, in order to discriminate between face images and non-face images, the distance ϵ between the original image x and its reconstructed image from the face space, $x_f = U_k P_x + \mu$, is also computed:

$$\epsilon = \|x - x_f\|_2. \tag{2.7}$$

Note in passing that

$$\begin{aligned} \epsilon &= \|x - \mu - U_k P_x\|_2 \\ &= \|(x - \mu) - U_k U_k^\top (x - \mu)\|_2, \end{aligned}$$

and therefore ϵ represents simply the distance between $x - \mu$ and its orthogonal projection onto span$\{\mathrm{U_k}\}$, i.e.,

$$\epsilon^2 = \|(I - U_k U_k^\top)(x - \mu)\|_2^2 \tag{2.8}$$

$$= \|x - \mu\|_2^2 - \|P_x\|_2^2. \tag{2.9}$$

This metric is used to decide whether or not a given image is a face.

3 Eigenfaces in terms of the SVD

In this section we interpret the above training and recognition stages in terms of the truncated singular value decomposition of A. The SVD [6] of a rectangular $N \times M$ matrix A of rank r, is defined as

$$A = U\Sigma V^\top, \tag{3.10}$$
$$U^\top U = I_N \in R^{N\times N}, \tag{3.11}$$
$$V^\top V = I_M \in R^{M\times M}, \tag{3.12}$$

where $U = [u_1, \ldots, u_N]$ and $V = [v_1, \ldots, v_M]$ are unitary matrices and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_M)$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_M = 0$. The σ_i's are the *singular values* of A and the u_i's and v_i's are respectively *the left and right singular vectors* associated with σ_i, $i = 1, \ldots, r$. We define the i-th singular triplet of A as $\{u_i, \sigma_i, v_i\}$. It follows from the SVD that the matrix A can be expressed as a sum of r rank-one matrices,

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^\top .$$

Additionally, it is well known that

$$\min_{\mathrm{rank}(B)\leq k} \|A - B\|_F = \|A - A_k\|_F$$

where $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^\top$ and $\|.\|_F$ is the Frobenius norm. It is helpful for what follows to rewrite the matrix A_k as

$$A_k = U_k \Sigma_k V_k^\top, \tag{3.13}$$

where U_k (resp. V_k), consists of the first k columns of U (resp. V), and Σ_k is a diagonal matrix of size $k \times k$. Thus, if we truncate the SVD to keep only the k largest singular triplets we obtain the closest (in a least-squares sense) approximation to A.

Observe that the matrix U_k containing the k largest left singular vectors of $\tilde{A} = \frac{1}{\sqrt{M}} A$, is exactly the matrix computed by PCA containing the largest eigenvectors of the covariance matrix. This follows from the fact that

$$C = \tilde{A}\tilde{A}^\top = U\Sigma V^\top V \Sigma^\top U^\top = U\Sigma\Sigma^\top U^\top,$$

is the eigendecomposition of the covariance matrix. Using this observation, equation (2.4) can be written in the form

$$\begin{aligned} P_i &= U_k^\top \tilde{x}_i = U_k^\top \tilde{A} e_i \\ &= U_k^\top [U_k\ U_{N-k}] \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_{M-k} \end{bmatrix} \begin{bmatrix} V_k^\top \\ V_{M-k}^\top \end{bmatrix} e_i \\ &= [I_k\ 0] \begin{bmatrix} \Sigma_k V_k^\top \\ \Sigma_{M-k} V_{M-k}^\top \end{bmatrix} e_i \\ &= \Sigma_k V_k^\top e_i,\ i = 1, \ldots, M. \end{aligned}$$

Denote by $P = \Sigma_k V_k^\top$ the matrix whose columns are the projections P_i, $i = 1, \ldots, M$, of every known face image to the face space. Assuming that all vectors are normalized, the similarity measurement (2.6) among the new image x and all known images, can be equivalently computed by the similarity vector s_k,

$$\begin{aligned} s_k &= P^\top P_x = V_k \Sigma_k^\top U_k^\top (x - \mu) \\ &= \tilde{A}_k^\top (x - \mu), \end{aligned} \tag{3.14}$$

containing a similarity score between the new face image and each of the known images. Thus, the computation of the similarity vector s_k employs a rank k approximation of the translated matrix A. We discuss the assumption of normalized projected vectors in the following section.

Note also that using the SVD, equation (2.8) expresses the metric ϵ as the distance from $x - \mu$ to the space span$\{\mathrm{U_k}\}$ of the dominant left singular space.

In the sequel, we show how to approximate the similarity vector s_k in (3.14), as well as the distance ϵ in (2.8) without using any eigendecomposition. The proposed scheme relies on polynomial filtering.

4 Eigenfaces using polynomial filtering

Polynomial filtering allows to closely approximate the effect of reduced rank approximation used in PCA models. Denote by $\psi(A)$ a matrix polynomial of degree d on the matrix A, i.e.,

$$\psi(A) = \xi_d A^d + \xi_{d-1} A^{d-1} + \ldots + \xi_1 A + \xi_0 I.$$

Assuming that A is normal (i.e., $A^\top A = A A^\top$) and letting $A = Q\Lambda Q^\top$ be its eigendecomposition, observe that

$$\psi(A) = \psi(Q\Lambda Q^\top) = Q\psi(\Lambda)Q^\top.$$

Therefore, the polynomial on A is translated to a polynomial on its eigenvalues. We are now ready to describe how one can use polynomial filtering to approximate the similarity vector directly, avoiding completely eigenvalue computations.

Let $\tilde{x} = x - \mu$ be the translated new image. In order to estimate the similarity measurement, we use a polynomial ψ of $\tilde{A}^\top \tilde{A}$ such that

$$\begin{aligned} s &= \psi(\tilde{A}^\top \tilde{A})\tilde{A}^\top \tilde{x} \\ &= \psi(V\Sigma^\top \Sigma V^\top) V \Sigma^\top U^\top \tilde{x} \\ &= V\psi(\Sigma^\top\Sigma)V^\top V \Sigma^\top U^\top \tilde{x} \\ &= V\psi(\Sigma^\top\Sigma)\Sigma^\top U^\top \tilde{x}. \end{aligned} \tag{4.15}$$

Compare the last expression above with (3.14). Choosing the polynomial $\psi(t)$ appropriately will allow us to interpretate this approach as a compromise between the

correlation [2] and the PCA approaches. Assume now that ψ is not restricted to being a polynomial but can be any function (even discontinuous). When $\psi(t) = 1\ \forall x$, then $\psi(\Sigma^\top\Sigma)$ becomes the identity operator and the above scheme would be equivalent to the correlation method. On the other hand, taking ψ to be the step function

$$\psi(t) = \begin{cases} 0, & 0 \le t \le \sigma_k^2 \\ 1, & \sigma_k^2 \le t \le \sigma_1^2 \end{cases} \tag{4.16}$$

results in $\psi(\Sigma^\top\Sigma) = \begin{bmatrix} I_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ where I_k is the identity matrix of size k and $\mathbf{0}$ is a zero matrix of an appropriate size. Then, equation (4.15) may be re-written as:

$$\begin{aligned} s &= V\psi(\Sigma^\top\Sigma)\Sigma^\top U^\top\tilde{x} \\ &= \begin{bmatrix} V_k & V_{n-k} \end{bmatrix}\begin{bmatrix} \Sigma_k^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\begin{bmatrix} U_k^\top \\ U_{m-k}^\top \end{bmatrix}\tilde{x} \\ &= \begin{bmatrix} V_k\Sigma_k^\top & \mathbf{0} \end{bmatrix}\begin{bmatrix} U_k^\top \\ U_{m-k}^\top \end{bmatrix}\tilde{x} \\ &= V_k\Sigma_k^\top U_k^\top\tilde{x} \\ &= \tilde{A}_k^\top\tilde{x} \end{aligned} \tag{4.17}$$

which is precisely the rank-k approximation provided in equation (3.14).

Using polynomial filtering we can also approximate the "faceness" (i.e., whether or not a given image contains a face) of an image as it is expressed by equation (2.8). Using the SVD, observe that

$$\begin{aligned} \psi(C)(x-\mu) &= \psi(\tilde{A}\tilde{A}^\top)(x-\mu) \\ &= \psi(U\Sigma V^\top V\Sigma^\top U^\top)(x-\mu) \\ &= U\psi(\Sigma\Sigma^\top)U^\top(x-\mu). \end{aligned} \tag{4.18}$$

Note that if ψ is exactly the step function (4.16), then $\|\psi(C)(x-\mu)\|_2 = \|U_kU_k^\top(x-\mu)\|_2 = \|P_x\|_2$ which would allow to obtain ϵ from (2.8). If the polynomial ψ is an approximation of the step function, this will provide an estimate of the distance metric ϵ, needed to decide on the faceness of an image, without the availability of U or U_k.

Therefore, the approach of polynomial filtering in PCA models can give virtually the same result as eigendecomposition, without resorting to the costly eigenvalue decomposition or any other matrix decomposition. Furthermore, the need to store additional (dense or sparse) matrices as is the case in PCA, is completely avoided as is the need to update these matrices, when the subspace used for learning changes dynamically. The selection of the cut-off point is somewhat similar to the issue of choosing the parameter k in the PCA method. However, there is a salient difference between the two: choosing a large k in PCA may render the method much more expensive, while selecting a high cut-off in polynomial filtering does not affect cost significantly.

Recall that in the computation of the similarity vector we assumed that the projected vectors P_i have unity norm. Since this is not the case, we need to normalize the similarity vector by the corresponding norms $\|P_i\|_2$. These norms are computable using a polynomial filter on $AA^\top$. Using equations (2.4) and (4.18) observe that $\|P_i\|_2 = \|U_kU_k^\top(x_i-\mu)\|_2 = \|\psi(C)(x_i-\mu)\|_2$, where again ψ is the step function (4.16). The filter does not need to be accurate enough before it is useful. Usually, a rough approximating polynomial of degree 2 or 3 suffices. This computation can be accomplished off-line at the training stage, before the recognition stage. It is fairly expensive, but its cost is amortized during the recognition phase, especially when the number of face images very large.

In order to overcome this problem we suggest two solutions. Before applying the proposed scheme we normalize all input data vectors x_i. Next, we compute the similarity score and sort the samples in descending order. Then we have two options. Using the first $k \ll M$ samples, either we can employ PCA or we can use k-nearest neighbor classification. Observe that since $k \ll M$, the cost of exact PCA will be very limited, and certainly orders of magnitude smaller than PCA on the original data matrix. Similarly, applying k-nearest neighbor classification on a very small set of data points will have very limited cost. We observed empirically that the first option yields slightly better results and this is the option that we included in our experiments (Section 8) with $k = 30$. Thus, using polynomial filtering one can reject very efficiently the irrelevant images and then use a more accurate and expensive algorithm on the first few highly ranked samples.

5 The kernel eigenfaces method

Kernel Principal Component Analysis (KPCA) [18] is a generalization of PCA that has been successfully applied in many pattern recognition problems. In [21] and [9] the authors investigate the applicability of kernel PCA in face recognition. Kernel methods employ a nonlinear mapping $\Phi : R^N \to F$ from the input space R^N to another product space F which is called *feature space*. The dimension of the feature space can be arbitrarily large, possibly even infinite.

Denote by $\langle . \rangle$ the Euclidean dot product. Dot products among data points in F are computed efficiently using the so called "kernel trick". This is achieved using Mercer kernels where a kernel function

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle,$$

with $x_i, x_j \in R^N$ and $\Phi(x_i), \Phi(x_j) \in F$, computes the dot product in F. Therefore, any algorithm that can be expressed solely in terms of dot products, can be extended in a nonlinear kernelized version of itself. There are various types of kernels. The most extensively used in the community are tabulated in Table 1.

Kernel PCA is the straightforward application of PCA on the feature space. We now describe briefly how kernel PCA can be performed implicitly in F, while doing computations in the input space R^N. Assume that the data points in F are centered i.e., their mean is equal to zero. The reader is referred to [18] for a detailed analysis of kernel PCA and also for details about the process of centering the data points in the feature space. The covariance matrix is

$$C_\Phi = \frac{1}{M}\sum_{j=1}^{M}\Phi(x_j)\Phi(x_j)^\top = \frac{1}{M}A_\Phi A_\Phi^\top, \tag{5.19}$$

where $A_\Phi = [\Phi(x_1), \Phi(x_2), \ldots, \Phi(x_M)]$ is the data matrix of the translated data points in the feature space. The corresponding eigenvalue problem for performing PCA in F, is

$$C_\Phi w_\Phi = \lambda w_\Phi. \tag{5.20}$$

Expanding the eigenvectors w_Φ in the span of the columns of A_Φ we get

$$w_\Phi = \sum_{j=1}^{M} z_i \Phi(x_j) = A_\Phi v_z, \tag{5.21}$$

where $v_z = [z_1, z_2, \ldots, z_N]^\top$. In order to compute the z's we project the eigenvalue problem (5.20) on each column of A_Φ resulting in

$$\langle \Phi(x_k), C_\Phi w_\Phi \rangle = \lambda \langle \Phi(x_k), w_\Phi \rangle,\ k = 1, \ldots, M.$$

Substituting equation (5.21) into the above expression and defining the $M \times M$ Gram matrix $K_{i,j} = k(x_i, x_j)$, it turns out that the z's are determined by the solution of the eigenvalue problem

$$\frac{1}{M}Kz = \lambda z. \tag{5.22}$$

After solving the above eigenvalue problem, we need to normalize the eigenvectors w_Φ of C_Φ in the feature space. Imposing the condition $\langle w_\Phi^k, w_\Phi^k \rangle = 1$, gives

$$1 = \sum_{i,j=1}^{M} z_i^k z_j^k \langle \Phi(x_i), \Phi(x_j) \rangle \tag{5.23}$$

$$= \sum_{i,j=1}^{M} z_i^k z_j^k K_{ij} \tag{5.24}$$

$$= \langle z^k, K z^k \rangle \tag{5.25}$$

$$= \lambda_k \langle z^k, z^k \rangle, \tag{5.26}$$

nonlinearity	$k(x, y)$
polynomial	$(x^\top y)^d$
Gaussian	$\exp(-\|\mathrm{x} - \mathrm{y}\|^2/\sigma^2)$
sigmoid	$\tanh(\kappa \mathrm{x}^\top \mathrm{y} - \delta)$

Table 1: Various kernel functions.

where the sum is taken on all values of i, j. In other words, normalizing each z^k by the square root of the corresponding $\sqrt{\lambda_k}$ eigenvalue, we ensure that the eigenvectors w_Φ of the covariance matrix have unit norm in the feature space.

Let x be a test point whose mapping on the feature space is $\Phi(x)$. Then, the projection of $\Phi(x)$ on the eigenvectors w_Φ provides the nonlinear principal components

$$\langle w_\Phi, \Phi(x) \rangle = \sum_{i=1}^{M} z_i \langle \Phi(x_i), \Phi(x) \rangle = \sum_{i=1}^{M} z_i k(x_i, x).$$

Therefore, we can extract the k nonlinear principal components by working implicitly in the feature space.

5.1 Kernel eigenfaces in terms of the SVD In this section we provide an interpretation of kernel PCA in terms of SVD that will facilitate the description of the polynomial filtering technique in the transformed feature space. Let

$$A_\Phi = \hat{U}\hat{\Sigma}\hat{V}^\top \tag{5.27}$$

be the singular value decomposition of the data matrix A_Φ in F. We will show that the computed eigenvectors w_Φ in kernel PCA, are essentially the left singular vectors u_Φ corresponding to the largest singular values of A_Φ. The eigendecomposition of matrices C_Φ and $\hat{C}_\Phi$ is correspondingly

$$C_\Phi = A_\Phi A_\Phi^\top = \hat{U}\hat{\Sigma}\hat{\Sigma}^\top\hat{U}^\top, \tag{5.28}$$

$$\hat{C}_\Phi = A_\Phi^\top A_\Phi = \hat{V}\hat{\Sigma}^\top\hat{\Sigma}\hat{V}^\top. \tag{5.29}$$

Observe now that $\hat{C}_\Phi = \frac{1}{M}K$. Therefore, the eigenproblem (5.22) that is solved in kernel PCA, computes the principal eigenvectors $\hat{V}$ of $\hat{C}_\Phi$. The computed z's are essentially the eigenvectors of $\hat{C}_\Phi$ and the λ's are the squares of the singular values $\hat{\sigma}$ of A_Φ. Using (5.27) we have

$$A_\Phi \hat{V} = \hat{U}\hat{\Sigma}.$$

In kernel PCA, we need to compute the principal eigenvectors $\hat{U}$ of C_Φ. Denote by w_Φ^i the i-th principal eigenvector of C_Φ. Essentially, what kernel PCA computes is (see also equation (5.21))

$$w_\Phi^i = A_\Phi \hat{v}_i = \hat{\sigma}_i \hat{u}_i.$$

In order to obtain the $\hat{u}_i$'s we need to normalize the computed w_Φ^i by the corresponding $\hat{\sigma}_i$. This is exactly the normalization step taking place in (5.26).

In the training stage, we project all the known data points on the first k columns of $\hat{U}$ in order to get the nonlinear principal components

$$\hat{P}_i = \hat{U}_k^\top \Phi(x_i) = \hat{U}_k^\top A_\Phi e_i = \hat{\Sigma}_k \hat{V}_k^\top e_i,\ i = 1, \ldots, M.$$

In other words, the columns of $\hat{P} = \hat{\Sigma}_k \hat{V}_k^\top$ hold the nonlinear principal components of the known data points. Assume now that a new test point x needs to be identified. First, its nonlinear principal components will be computed

$$\hat{P}_x = \hat{U}_k^\top \Phi(x),$$

and then the similarity score will be computed to perform the recognition step:

$$(5.30)\quad \hat{s} = \hat{P}^\top \hat{P}_x = \hat{V}_k \hat{\Sigma}_k^\top \hat{U}_k^\top \Phi(x) = (A_k)_\Phi^\top \Phi(x).$$

Following a similar analysis as in Section 4, we conclude that we can approximate the similarity score that would be obtained by kernel PCA, using polynomial filtering. The following section describes this process.

5.2 Kernel eigenfaces with polynomial filtering

Consider the computation of the similarity vector in the feature space. Similarly to equation (4.15), the similarity vector induced by the transformed data points $\Phi(x_i),\ i = 1, \ldots, M$ is given by

$$\begin{aligned} s_\Phi &= \psi(A_\Phi^\top A_\Phi) A_\Phi^\top \Phi(x) \\ &= \psi(\hat{C}_\Phi) A_\Phi^\top \Phi(x) \\ &= \frac{1}{M} \psi(K) A_\Phi^\top \Phi(x) \\ &= \frac{1}{M} \psi(K) t_\Phi, \end{aligned}$$

where $\Phi(x)$ is the centered mapping of x in F and $t_\Phi = A_\Phi^\top \Phi(x)$. As was mentioned above, we can compute the $\hat{C}_\Phi$ matrix using dot products in the feature space and thus via kernel function evaluations. In particular, this matrix is a scaled version of the Gram matrix $K_{ij} = k(x_i, x_j)$ used in kernel PCA. In addition, we can compute the vector t_Φ using again dot products in the feature space. Each component $t_\Phi(i)$ of this vector is the dot product of the new face image $\Phi(x)$ with $\Phi(x_i)$ in F, and can be computed via the kernel evaluation i.e., $t_\Phi(i) = k(x_i, x)$.

The procedure based on polynomial filters is therefore clear. First, the vector t_Φ is computed and then the filter $\psi(K)$ is applied to it. This last computation simplifies to a series of matrix vector products with the Gram matrix K.

6 Approximating the step function

We now consider ways of approximating the ideal step function ψ given in (4.16) using a polynomial $\hat{\psi}$. One approach would be to discretize the step function in the interval $[0, b]$, with $b \equiv \sigma_1^2$, and then to find the coefficients of the polynomial which interpolates the produced data points in a least-squares sense. As is well-known this approach will produce a polynomial with potentially large fluctuations between the data points resulting in a poor recognition performance (see also below).

Another approach is to rely on Hermite interpolation by imposing smoothness conditions at both endpoints of the interval. Assume that we enforce the following conditions at endpoints 0 and b,

$$\begin{aligned} &\hat{\psi}(0) = 0, \quad \hat{\psi}^{(1)}(0) = \hat{\psi}^{(2)}(0) = \cdots = \hat{\psi}^{(i)}(0) = 0 \\ &\hat{\psi}(b) = 1, \quad \hat{\psi}^{(1)}(b) = \hat{\psi}^{(2)}(b) = \cdots = \hat{\psi}^{(j)}(b) = 0 \end{aligned}$$

Using the above $i+j+2$ conditions, we can employ Hermite interpolation in order to determine the coefficients of a polynomial of degree $i + j + 1$ that will satisfy the given constraints. The derived polynomial $\hat{\psi}(t)$ moves from 0 to 1, as t moves from 0 to b. It can be shown [5] that the critical point, called *inflexion point*, where $\hat{\psi}$ moves rapidly from 0 to 1 is at:

$$(6.31)\qquad t_{\text{infl}} = \frac{b}{1 + j/i} \quad \text{or} \quad \frac{j}{i} = \frac{b}{t_{\text{infl}}} - 1.$$

Therefore, the ratio $\frac{j}{i}$ determines the localization of the inflexion point. This approach has the disadvantage that the degree of the polynomial needs to become adequately large in order for the approximation to be qualitative.

The most successful approach when approximating a function with polynomials, is the piecewise polynomial approximation where instead of using only one large degree polynomial on the whole interval, we use several smaller degree polynomials at appropriate subintervals of the original interval. The difficulty with the piecewise polynomials is that they cannot be easily evaluated when their argument is a matrix. Erhel et al in [5] suggest a new technique, called PPF hereafter, which approximates any piecewise matrix polynomial by a matrix polynomial in some least-squares sense. This technique is used [5] for solving ill-conditioned linear systems in image restoration where the problem matrix is symmetric positive semidefinite with a large number of singular values close to zero and the right-hand side vector is perturbed with noise. In this paper we apply a similar technique in the totally different context of face recognition.

Figure 1 illustrates the approximations to the step function ψ obtained by the above methods. We compare

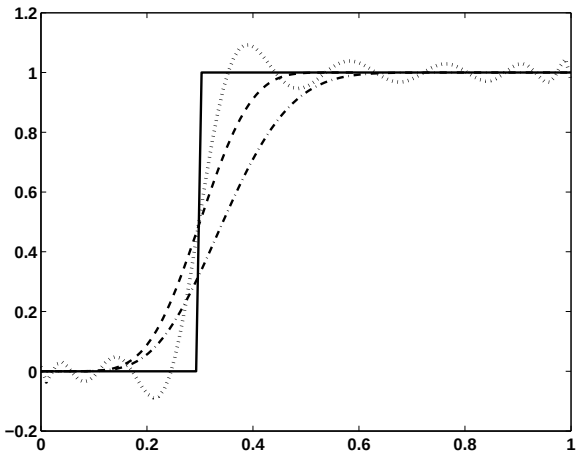

Figure 1: Approximating the step function ψ (solid line). Dotted line: least-squares, dash-dotted line: Hermite and dashed line: PPF.

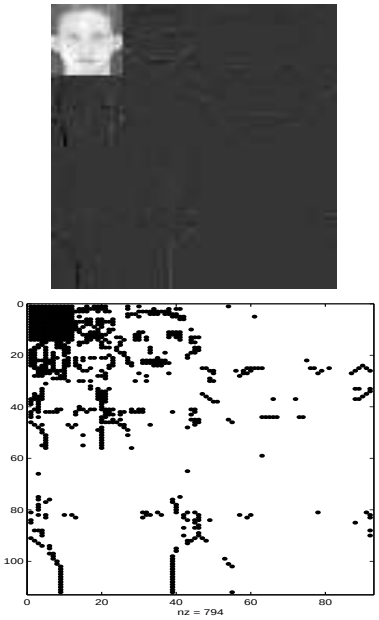

Figure 2: Upper plot: wavelet decomposition of level 3 of a sample face from the ORL database. Bottom plot: sparsity pattern after numerical dropping.

Figure 3: Sample face images from the YALE database. Facial expressions from left to right: 'centerlight', 'glasses', 'happy', 'leftlight', 'noglasses', 'normal', 'rightlight', 'sad', 'sleepy', 'surprised' and 'wink'.

the least-squares, Hermite and PPF techniques. For the PPF method we used two intervals with Hermite interpolation for the polynomial of the first interval and 1 for the second interval. The degree of the resulting polynomial was 14 for all methods. The cut off point was located at 0.3. Notice the large fluctuations of the least squares approximation. Also observe that PPF offers a more qualitative approximation to the step function, compared to Hermite. In our experiments, we employed PPF as our polynomial filtering method for face recognition.

7 Sparse representation using wavelets

The proposed polynomial filtering framework becomes even more efficient if the data matrix is sparse. We can obtain sparse representation of images using wavelet decompositions [3],[4],[12]. Wavelet decompositions have been a very useful tool for lossy image compression with high quality. Using wavelets the image data is transformed to different frequency components that can be processed individually. Usually, the high frequency components are numerically very small and can be dropped without any significant impact on the quality of the reconstruction from the wavelet space.

The upper plot in Figure 2 depicts the resulting image of a sample face from the ORL dataset, using Haar wavelet decomposition of level three. We give more information on the ORL database in the next section. Before applying the wavelet transformation we normalized the image vector to unit norm. The bottom plot illustrates the sparsity pattern of the image in the wavelet space after numerical dropping with tolerance $2e-3$. The resulting matrix is very sparse with only 7.7 % of the matrix elements being non-zero.

Using orthogonal wavelet decompositions, the data matrix A is transformed to A_w via a change of basis

$$A_w = QA,\ Q \in R^{M\times M}, \tag{7.32}$$

where Q is a unitary matrix and A_w is very sparse. Exploitation of the sparsity of A_w renders the polynomial filtering framework very efficient, since the evaluation of the similarity vector simplifies to a series of (sparse) matrix-vector products. Another advantage of the proposed scheme is that it makes no explicit use of matrix Q, since it holds that

$$\begin{aligned} s_w &= \psi(A_w^\top A_w)A_w^\top q_w \\ &= \psi(A^\top Q^\top QA)A^\top Q^\top Qq \\ &= \psi(A^\top A)A^\top q = s_k, \end{aligned}$$

Figure 4: Sample face images from the ORL database. There are 10 available facial expressions and poses for each subject.

Figure 5: Sample face images from the AR database. Facial expressions from left to right: 'natural expression', 'smile', 'anger' and 'scream'.

which is exactly the similarity vector which would be obtained, if we performed the computation in the input space. Note that this is not the case for PCA, because it works on $C = \tilde{A}\tilde{A}^\top$. Now, the combination of polynomial filtering with wavelets is very clear. We first normalize each column of A and next transform it to the wavelet space and obtain its sparse representation. Then, we compute the similarity vector efficiently using sparse matrix-vector products.

Therefore, using the polynomial filtering framework, the estimation of the similarity measurement simplifies to a series of matrix vector products. In particular, when the data matrix is dense, the cost of our technique is $\mathrm{O}(dN^2)$, where d is the degree of the polynomial ψ. However, note that the cost of the eigendecomposition for dense matrices is $\mathrm{O}(N^3)$ [6]. On the other hand, when the matrix is sparse, the cost of our scheme is $\mathrm{O}(d \cdot \mathrm{nnz})$ and the cost of PCA (with k eigenvectors) is approximately $\mathrm{O}(k(\mathrm{nnz}+\mathrm{N})+\mathrm{k}^3)$, where nnz is the number of nonzero elements of the sparse matrix. Clearly, the storage cost is negligible, since there is no need to store the dimensionality reduction matrix.

8 Numerical results

In this section we experiment with the above methods in terms of recognition accuracy. All experiments were implemented in `MATLAB` 6.5 on a Xeon@2.4GHz. We used three datasets that are publically available; YALE, ORL and a subset of AR. The YALE database [1] contains 165 images of 15 individuals that include variation in both facial expression and lighting. In the preprocessing phase, each face image was closely cropped, and the size of images after the cropping phase was decreased to 112×92. Figure 3 illustrates a small sample from the YALE database.

dataset	size	sparsity (%)
ORL	10304×400	21.9
YALE	10304×165	23.4
AR	17864×504	17.8

Table 2: The datasets and their characteristics.

The ORL (formerly Olivetti) database [17] contains 40 individuals and 10 different images for each individual. Figure 4 illustrates a small sample from the ORL database. In this case no preprocessing was done. Finally, the AR face database [13] contains 126 subjects with 4 different facial expressions for each individual. Figure 5 illustrates two sample subjects from the AR database. The characteristics of the datasets are tabulated in Table 2 where the *sparsity* denotes the sparsity of the data matrix when transformed in the wavelet space (using $tol = 2e-3$).

In what follows, error rates are estimated using a cross validation "leave-one-out" strategy. In order to compute the error rate with respect to a certain facial expression, the image associated with it was used as a test image. In order to recognize the test image, all images, excluding the test one, are projected to the reduced subspace. Then, the test image is projected as well and recognition is performed using a nearest neighbor rule. Note that in this case every individual is represented by all of its different images except for the test individual which is represented by one image less.

In the following examples, we compute two different types of error rates. Denote by s a certain subject in the database and $\tilde{s}$ the recognized subject returned by the system. The first type of error, e_i, is the number of misses counted across the subjects for a given facial expression i. In other words, e_i returns the error rate when the facial expression i is used as test image for all individuals. Denote also by N_f the number of different facial expressions/poses associated with each individual in the database. The second type of error is

$$e = \frac{1}{N_f}\sum_{i=1}^{N_f} e_i,\ i = 1, ..., N_f. \tag{8.33}$$

Figure 6: Reconstruction experiments on an ORL sample subject (upper left image) using eigenfaces. Upper right image: $k = 30$. Lower left image: $k = 60$ and lower right image: $k = 90$.

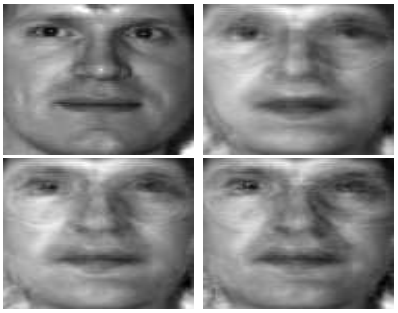

Figure 8: Reconstruction experiments on an YALE sample subject (upper left image) using eigenfaces. Upper right image: $k = 30$. Lower left image: $k = 60$ and lower right image: $k = 90$.

Figure 7: Reconstruction experiments on an ORL sample subject (upper left image) using polynomial filtering. Upper right image: $\eta = 0.0248$. Lower left image: $\eta = 0.0109$ and lower right image: $\eta = 0.0065$.

Figure 9: Reconstruction experiments on an YALE sample subject (upper left image) using polynomial filtering. Upper right image: $\eta = 0.0118$. Lower left image: $\eta = 0.0043$ and lower right image: $\eta = 0.0022$.

Thus, e is the mean error rate averaged across all different facial expressions. In what follows, denote by PCA the "eigenfaces" method and by KPCA the "kernel eigenfaces" method.

In the implementation of PPF we used two subintervals $[0,a]$ and $[a,b]$. We model the cut-off point as a percentage ηb of the right endpoint of the whole interval $[0,b]$, where $\eta \in (0,1)$. The right endpoint of the left interval was set to $a = 2\eta b$. In addition, as equation (6.31) indicates, the cut off point where ψ moves rapidly from 0 to 1, is determined by the ratio

$$\frac{j}{i} = \frac{b}{t_{\text{infl}}} - 1.$$

Therefore, keeping the ratio j/i constant, we are free to choose the parameters i and j. Let γ be a multiplicative factor that affects both i and j. Then, the degree of the polynomial in the general case will be $\gamma i + \gamma j + 1$. Thus, in the implementation of PPF we need two parameters; η and γ. The former determines the localization of the inflexion point and the latter determines the quality of the approximation.

Example 1 (Reconstruction experiments) In the first example, we illustrate the effectiveness of the eigenfaces method vis-a-vis the polynomial filtering method in reconstruction quality. We use a sample face image from both ORL and YALE datasets and try to reconstruct it from the learning subspace. Let x be the test image and x_f its reconstructed image. Using eigenfaces, the approximation to x_f is $x_f = U_k P_x + \mu = U_k U_k^\top (x - \mu) + \mu$, where we have used equation (2.5). On the other hand, using polynomial filtering, the x_f is approximated by $x_f = \psi(AA^\top)(x-\mu)+\mu$, according to equation (4.18). We experimented with various values of the dimension k of the learning subspace that gave rise to corresponding values for the parameter η used in PPF. We also used $\gamma = 10$, resulting in a polynomial of degree $d = i + j + 1 = 21$.

Figures 6 and 7 illustrate the experimental results

ORL dataset		
$k = 40$	PPF (%)	KPPF (%)
γ=2	2.5	3.25
γ=3	3.5	4.25
γ=4	2.75	3.5
γ=5	3	3.5
YALE dataset		
$k = 40$	PPF (%)	KPPF (%)
γ=2	26.06	29.09
γ=3	25.45	26.67
γ=4	26.06	26.06
γ=5	26.06	26.67
AR dataset		
$k = 40$	PPF (%)	KPPF (%)
γ=2	8.33	7.34
γ=3	8.53	7.74
γ=4	7.14	8.93
γ=5	6.15	8.33

Table 3: Error rates of PPF methods for various values of γ, on all face databases.

ORL dataset		
$k = 50$, $\gamma = 4$	KPCA (%)	KPPF (%)
d=2	3.25	3.75
d=3	3.25	3.5
d=4	3.25	4
d=10	3.25	3.25
YALE dataset		
$k = 50$, $\gamma = 4$	KPCA (%)	KPPF (%)
d=2	28.48	24.84
d=3	27.88	25.45
d=4	27.88	26.06
d=10	26.67	26.06
AR dataset		
$k = 50$, $\gamma = 4$	KPCA (%)	KPPF (%)
d=2	5.75	6.35
d=3	5.75	6.75
d=4	5.95	6.35
d=10	5.56	5.95

Table 4: Error rates of kernel methods using the polynomial kernel with degree d, on all datasets.

which compare the effectiveness of the PPF method, against that of eigenfaces, tested on a sample subject from the ORL dataset. Observe that the reconstruction quality is almost identical and this is achieved while the costly eigenvalue calculations were completely avoided. The same observations can be drawn from the reconstruction experiments performed on a sample subject from the YALE database. The results are illustrated in Figures 8 and 9. Observe that the largest the value of k, the better the reconstruction quality.

Example 2 (Find the best value for γ) In the second example we investigate the behavior of the PPF methods with respect to the degree of the polynomial ψ. Table 3 illustrates the error rate of both PPF and kernel PPF (KPPF) with respect to γ, tested on all datasets. In KPPF we used the polynomial kernel with $d = 3$ and for both methods we used the value of η that corresponds to $k = 40$. Observe that in most cases the value $\gamma = 4$ seems to give the most satisfactory results. To that end, in what follows, we use $\gamma = 4$ for both PPF methods.

Example 3 (Compare various kernels) In this example we study the behavior of the kernelized methods with respect to the choice of the kernel and its parameters. We experimented with the polynomial kernel and its degree d

$$k(x, y) = (x^\top y)^d,$$

as well as with the Gaussian kernel with variance σ

$$k(x, y) = \exp(-\|x - y\|^2/\sigma^2).$$

The purpose of this example is to compare the effectiveness of the two kernels and infer the best practical value for its parameter. The results for all datasets are tabulated in Tables 4 and 5. The dimension k of the reduced space and the appropriate value for γ, used in the kernel PPF method, are also depicted in each table.

Observe that the polynomial kernel yields slightly better results than the Gaussian kernel on both databases. Furthermore, the value $d = 4$ of the degree seems to be the most appropriate one. Therefore, in what follows, when experimenting with the kernelized methods, we are using the polynomial kernel with $d = 4$ for all datasets.

Example 4 (Compare all methods in terms of recognition rate and computational cost) In the fourth example we investigate the effect of the dimension k of the reduced space on the recognition performance of the methods. We used `MATLAB`'s `svd` builtin function since the matrix is dense and this way we avoid the explicit formulation of matrices $AA^\top$ or $A^\top A$. We experimented with $k = 20 : 20 : 100$ (in `MATLAB` notation) and measure the error rate (%) given by equation (8.33), for all face databases. In what follows, we used $\gamma = 4$ for both PPF methods.

Tables 6, 7 and 8 illustrate the error rate e, computed by equation (8.33), versus the dimension k mea-

ORL dataset		
$k = 50$, $\gamma = 4$	KPCA (%)	KPPF (%)
σ=0.3	3	3
σ=0.5	3	3.25
σ=0.7	3.25	3.75
σ=0.9	3.25	4
YALE dataset		
$k = 50$, $\gamma = 4$	KPCA (%)	KPPF (%)
σ=0.3	27.88	27.88
σ=0.5	27.28	26.66
σ=0.7	27.88	25.45
σ=0.9	27.88	24.84
AR dataset		
$k = 50$, $\gamma = 4$	KPCA (%)	KPPF (%)
σ=0.3	5.95	5.16
σ=0.5	5.56	5.95
σ=0.7	5.75	6.15
σ=0.9	5.95	6.75

Table 5: Error rates of kernel PCA and kernel PPF using the Gaussian kernel with variance σ, on all datasets.

sured on the ORL, YALE and AR datasets respectively. All tables contain the corresponding time measurements t (in sec) for each method. The timings for PCA methods measure the time needed to construct the subspace (i.e., computing the eigenvectors) and perform the recognition of the test image (i.e., one step of "leave-one-out" cross validation). The timings for PPF methods measure the time needed to recognize the test data point via polynomial filtering.

Concerning the ORL database, observe that PPF competes with PCA in terms of error rate. Furthermore, the PPF methods are much more efficient achieving significant speedups over their PCA counterparts. On the YALE dataset, the results are quite similar with PPF outperforming PCA not only in timings but in error rate as well. Finally, on the AR dataset, the results are similar to ORL, with the PPF methods being quite close to PCA in terms of error rate and being much more efficient in terms of computational cost.

Observe that the PPF dense implementations are slightly more efficient than the wavelet ones. We expect that for much larger datasets the results will be different. Also notice that in the case of kernel PPF, the sparsity of the Gram matrix depends on the type of kernel. For example, using Gaussian kernels is not possible to get a sparse Gram matrix even if the data are sparse. But even in the case of the polynomial kernel, the Gram matrix will have in general more nonzero

	PCA		PPF		PPF-wvlt	
	e	t	e	t	e	t
k=20	3.5	32.74	3	2.52	3	6.79
k=40	2.75	30.68	2.75	2.49	3	6.79
k=60	3.25	30.93	3.25	2.48	3	7.13
k=80	3.25	32.96	3	2.52	3	6.78
k=100	3	32.03	3	2.49	2.75	6.78
	KPCA		KPPF		KPPF-wvlt	
	e	t	e	t	e	t
k=20	3.25	17.69	3.75	3.12	4.25	4.89
k=40	2.75	17.69	4.25	3.12	4	4.88
k=60	3	17.69	3.75	3.12	4.25	4.88
k=80	3.25	17.65	3.75	3.12	4.25	4.89
k=100	2.75	17.64	4	3.12	4	4.89

Table 6: Error rates e (%) and timings t (in sec) of all methods for various values of k, on the ORL database.

elements than the data matrix.

9 Conclusion

We have described a new framework for implementing PCA and kernel PCA without eigenvalue calculations. The new framework relies on polynomial filtering, in order to render the same effect as PCA, for dimensionality reduction. We illustrated the applicability of the proposed technique in the eigenfaces and kernel eigenfaces method for face recognition. The numerical experiments indicated that the new scheme has very close performance to the PCA methods, while being much more efficient in terms of computational cost and storage.

10 Acknowledgements

We would like to thank prof. N. Papanikolopoulos for his valuable advice concerning face recognition issues.

References

[1] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Trans. Pattern Analysis and Machine Intelligence, Special Issue on Face Recognition*, 19(7):711—20, July 1997.

[2] R. Brunelli and T. Poggio. Face recognition: Features vs Templates. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(10):1042–1053, 1993.

[3] C. K. Chui. *Wavelets: A Mathematical Tool for Signal Analysis*. SIAM, Philadelphia, PA, 1997.

[4] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41:909–996, 1988.

	PCA		PPF		PPF-wvlt	
	e	t	e	t	e	t
k=20	29.70	5.93	25.45	1.15	25.45	1.18
k=40	27.88	6.02	26.06	1.16	26.06	1.18
k=60	27.27	6.10	25.45	1.14	25.45	1.17
k=80	27.27	6.22	25.45	1.16	25.45	1.23
k=100	26.06	6.33	25.45	1.15	25.45	1.16
	KPCA		KPPF		KPPF-wvlt	
	e	t	e	t	e	t
k=20	27.27	1.79	26.06	0.64	27.27	1.02
k=40	27.88	1.80	25.45	0.63	25.45	1.01
k=60	27.88	1.82	25.45	0.63	26.67	1
k=80	27.88	1.80	26.06	0.63	25.45	1
k=100	27.27	1.80	25.45	0.64	25.45	1.01

Table 7: Error rates e (%) and timings t (in sec) of all methods for various values of k, on the YALE database.

	PCA		PPF		PPF-wvlt	
	e	t	e	t	e	t
k=20	8.34	82.02	6.35	5.71	6.55	13.29
k=40	6.75	82.02	7.34	5.71	6.94	13.34
k=60	6.15	83.12	7.14	5.71	6.94	13.27
k=80	6.15	83.67	6.75	5.70	6.75	13.27
k=100	5.75	83.64	6.35	5.71	6.35	13.3
	KPCA		KPPF		KPPF-wvlt	
	e	t	e	t	e	t
k=20	8.33	23.34	7.34	7.64	5.36	11.06
k=40	7.54	23.44	7.94	7.69	6.35	10.77
k=60	6.75	23.44	8.13	7.67	7.14	10.77
k=80	6.55	23.44	8.33	7.66	7.14	10.81
k=100	6.35	23.77	8.33	7.67	6.75	10.83

Table 8: Error rates e (%) and timings t (in sec) of all methods for various values of k, on the AR database.

[5] J. Erhel, F. Guyomarc, and Y. Saad. Least-squares polynomial filters for ill-conditioned linear systems. Technical Report umsi-2001-32, Minnesota Supercomputing Institute, 200 Union Street S.E. Minneapolis, MN 55455, 2001.

[6] G. H. Golub and C. Van Loan. *Matrix Computations, 3rd edn.* The John Hopkins University Press, Baltimore, 1996.

[7] L. Hoegaerts, L. De Lathauwer, J.A.K. Suykens, and J. Vanderwalle. Efficiently updating and tracking the dominant kernel eigenspace. *16th International Symposium on Mathematical Theory of Networks and Systems*, July 5-9 2004. Belgium.

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

[9] K. Kim, K. Jung, and H. Kim. Face Recognition using Kernel Principal Component Analysis. *IEEE Signal Processing Letters*, 9(2):40–42, February 2002.

[10] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12, 1990.

[11] E. Kokiopoulou and Y. Saad. Polynomial Filtering in Latent Semantic Indexing for Information Retrieval. In *ACM-SIGIR Conference on research and development in information retrieval*, Sheffield, UK, July 25th-29th 2004.

[12] R.-C. Li. Fast Partial Eigenvalue Decomposition with Wavelet Transformation for Large Images. Technical report, Department of Mathematics, University of Kentucky, 1999.

[13] A.M. Martinez and R. Benavente. The AR Face Database. Technical report, CVC no. 24, 1998.

[14] J. Mercer. Functions of Positive and Negative Type and Their Connection with the Theory of Integral Equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

[15] S.K. Nayar, S.A. Nene, and H. Murase. Subspace Methods for Robot Vision. *IEEE Transactions on Robotics and Automation*, 12(5):750–758, October 1996.

[16] A. Pentland, B. Moghaddam, and T. Starner. View-based and Modular Eigenspaces for Face Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[17] F. Samaria and A. Harter. Parameterisation of a Stochastic Model for Human Face Identification. In *2nd IEEE Workshop on Applications of Computer Vision*, Sarasota FL, December 1994.

[18] B. Scholkopf, A. Smola, and K. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural computation*, 10:1299–1319, 1998.

[19] L. Sirovich and M. Kirby. Low-dimensional Procedure for the Characterization of Human Faces. *J. Optical Soc. Am. A (JOSA A)*, 4(3):519–524, March 1987.

[20] M. Turk and A. Pentland. Face Recognition using Eigenfaces. In *Int. Conf. on Patt. Recog.*, pages 586–591, 1991.

[21] M. Yang, N. Ahuja, and D. Kriegman. Face Recognition Using Kernel Eigenfaces. In *Int. Conf. on Image Processing*, volume 1, pages 37–40, 2000.

[22] W. Zhao, R. Chellapa, P. Phillips, and A. Rosenfeld. Face Recognition: a Literature Survey. *ACM Computing Surveys*, 35(4):399–458, December 2003.