ROMAI J., v.10, no.1(2014), 93–104

A COMPARATIVE STUDY OF SOME ALGORITHMS FOR FACE RECOGNITION

Lăcrămioara Liţă, Elena Pelican
"Ovidius" University of Constanţa, Romania
{epelican, lgrecu}@univ-ovidius.ro

Abstract Face recognition is one of the most prevalent problems of pattern recognition and a current issue in the context of nowadays technology progress. This real-life problem needs real-time answers, so a variety of algorithms have been developed to address this issue. In this paper we present a comparative study (in terms of recognition rate) for some numerical linear algebra standard algorithms and an own method. The algorithms are tested on different datasets considered as individual items and joined together.

Keywords: pattern recognition, dimension reduction, eigenvalue, eigenvector, singular value.
2010 MSC: 68T10, 15A18.

1. INTRODUCTION

The face recognition problem comes down to identifying a specific image of a person by analyzing and comparing patterns from a face dataset. Facial recognition systems are commonly used for security purposes but are becoming more and more used in a variety of other applications. There have been developed different types of algorithms for solving the face recognition problem (see [5], [11]).

The problem under consideration is the following. Given a dataset of images belonging to P persons, all images are transformed into vectors $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_N\}$ of the same size $M \times 1$ and all vectors become columns of a matrix, $A = [\Gamma_1\, \Gamma_2\, \ldots\, \Gamma_N]$. The matrix A will be our (face/image) dataset. We split the dataset into two nonoverlapping subsets: training subset and testing subset. Given an image Γ (the image of a person) from the testing set, we want to find out if the algorithm classifies correctly that person.

In this paper we present a comparative study of some algorithms used to solve the face recognition problem. The algorithms in question are the following: the Eigenfaces (PCA) algorithm (see [15] and [16]), an own algorithm, named COD algorithm (see [9]), and Lanczos and Block Lanczos algorithms (see [1], [2], [6], [10], [17]). We compare the results of our COD algorithm with those obtained with an algorithm on blocks, namely the Block Lanczos algorithm. The conclusions are encouraging since the results closely resemble.

The paper is structured as follows. In Section 2 we present some algorithms for face recognition: the Eigenfaces (PCA) algorithm, Lanczos and Block Lanczos algorithms and an own algorithm, COD algorithm. In the next section, Section 3, we

present experiments and comparison between the results obtained with all the algorithms described in this paper. All algorithms are tested on three datasets: the well known ORL dataset (also known as AT&T dataset), an own face dataset CTOVF, and a bigger dataset obtained by putting together the previous two datasets. We use two performance indicators: recognition rate and average query time. In the last section we draw conclusions regarding the obtained results.

As a novelty, the experiments were performed from two different viewpoints: first, we take each dataset (ORL and CTOVF) as a different entity in comparing the recognition rate obtained with each of them, latter, we consider a larger dataset, composed of the two datasets mentioned above.

In what follows we present a comparative study (in terms of recognition rate and average query time) for the mentioned above numerical linear algebra standard algorithms and an own method, the COD algorithm.

2. SOME ALGORITHMS FOR FACE RECOGNITION

2.1. THE EIGENFACES (PCA) ALGORITHM

The eigenfaces are the principal components of a set of faces, namely eigenvectors of the covariance matrix of a dataset (see [15] and [16]). They are called eigenfaces because when represented, they resemble human faces. The principal components are given by the eigenvectors of the covariance matrix. The first principal component is the eigenvector corresponding to the largest eigenvalue, the second principal component is the eigenvector corresponding to the next largest eigenvalue and so on.

All N images (with $M = n_1 \times n_2$ resolution) are transformed into vectors $\Gamma_i : M \times 1$. We compute the average face vector $\Psi = \frac{1}{N} \sum_{i=1}^{N} \Gamma_i$, and substract Ψ from all vectors $\varphi_i = \Gamma_i - \Psi$.

We seek a set of orthonormal vectors $u_1,\ u_2,\ \ldots,\ u_N$ which best describes the patterns that appear in the dataset. Vectors u_k are eigenvectors of the covariance matrix $C = AA^T$ for $A = [\varphi_1\ \varphi_2 \ldots \varphi_N]$.

The covariance matrix $C = AA^T$ is $C \in \boldsymbol{R}^{M \times M}$, where $M = n_1 \times n_2$ is the resolution of an image. Because in practice, the number M is very large, the computational effort to determine the M eigenvalues and M eigenvectors for matrix C is huge. In this case, the idea is to reduce the size and therefore the amount of calculations. Then, we consider $L = A^T A,\ L \in \boldsymbol{R}^{N \times N}$. Usually N, the number of images in the dataset is much smaller then the size of a vector, M, and in this case it is much easier to compute N eigenvectors and N eigenvalues for a matrix of size $N \times N$. From among the eigenvectors v_i of matrix L we obtain the eigenvectors Av_i of matrix C. From all N eigenvectors of L, are kept only the first K vectors corresponding to the largest K eigenvalues.

For a new image, Γ, we project $\Gamma - \Psi$ onto subspace $\{u_1,\ u_2,\ \ldots,\ u_K\}$, hence we have $\omega_i = u_i^T(\Gamma - \Psi),\ i = 1 : K$. The vector $\Omega^T = [\omega_1,\ \omega_2, \ldots, \omega_K]$ describes the contribution of each eigenface in representing the image Γ and is used to classify the new image Γ.

Eigenfaces algorithm (see [15]):

1. All images are transformed into vectors: $\Gamma_1,\ \Gamma_2,\ \ldots,\ \Gamma_N$.

2. Compute the average face vector: $\Psi = \frac{1}{N}\sum_{i=1}^{N}\Gamma_i$.

3. Substract the average face vector from all vectors: $\varphi_i = \Gamma_i - \Psi,\ i = \overline{1,N}$.

4. Compute the covariance matrix: $C = \frac{1}{N}\sum_{i=1}^{N}\varphi_i\varphi_i^T = \frac{1}{N}AA^T$.

5. Compute the eigenvectors $u_i,\ i = \overline{1,N}$ *of matrix C and keep only K vectors corresponding to the K largest eigenvalues.*

6. Obtain the vector: $\Omega_i^T = \left[\omega_1^i,\ \omega_2^i, \ldots, \omega_K^i\right]$ *where* $\varphi_i \approx \hat{\varphi}_i = \sum_{j=1}^{K}\omega_j^i u_j$.

7. Given a image Γ*, normalize it:* $\varphi = \Gamma - \Psi$.

8. Project Γ *on the eigenvectors space:* $\hat{\varphi} = \sum_{j=1}^{K}\omega_j u_j$.

9. Represent $\hat{\varphi}$ *as* $\Omega^T = [\omega_1,\ \omega_2, \ldots, \omega_K]$.

10. Find $i_0 \in \{1, \ldots, N\}$ *satisfying* $\left\|\Omega - \Omega_{i_0}\right\| = \min_{1\le i\le N}\|\Omega - \Omega_i\|$.

2.2. COD ALGORITHM

In order to obtain better results (as recognition rate) than with the PCA algorithm (a truncated SVD algorithm), we have to find a good low-rank approximation for matrix A. Another option for computing a rank k truncated SVD is using orthogonalization via deflation algorithm proposed in [3] and [4]. For a matrix $A \in \boldsymbol{R}^{M\times N},\ M \ge N$, is generated a sequence of matrices $A_1,\ A_2,\ \ldots, A_{k+1}$, for which

$$A_{k+1} = A_k - \tilde{\sigma}_k\tilde{u}_k\tilde{v}_k^T = A - \sum_{j=1}^{k}\tilde{\sigma}_j\tilde{u}_j\tilde{v}_j^T = A - \tilde{U}_k\tilde{D}_k\tilde{V}_k^T = A - \tilde{B}_k \tag{1}$$

where $\tilde{u}_k$, $\tilde{v}_k$ and $\tilde{\sigma}_k$ are given by formula (2), (3), and respectively (4), $\tilde{U}_k = [\tilde{u}_1\ \tilde{u}_2 \ldots\ \tilde{u}_k]$, $\tilde{V}_k = [\tilde{v}_1\ \tilde{v}_2 \ldots\ \tilde{v}_k]$, $\tilde{D}_k = diag\,(\tilde{\sigma}_1,\ \tilde{\sigma}_2, \ldots,\ \tilde{\sigma}_k)$, and $\tilde{B}_k = \tilde{U}_k\tilde{D}_k\tilde{V}_k^T$. Hence, matrix $\tilde{B}_k$ serves as a low-rank approximation of matrix A.

Let $\hat{u}_k \in \text{Range}\,(A_k)$ and $\hat{v}_k \in \text{Range}\left(A_k^T\right)$ be an arbitrary pair of unit vectors that satisfy $\hat{u}_k^T A_k\hat{v}_k > 0$. We have

$$\tilde{u}_k = \frac{A_k\hat{v}_k}{\|A_k\hat{v}_k\|_2}, \tag{2}$$

$$\tilde{v}_k = \frac{A_k^T \hat{u}_k}{\left\| A_k^T \hat{u}_k \right\|_2}, \tag{3}$$

and

$$\tilde{\sigma}_k = \frac{\left(\|A_k \hat{v}_k\|_2 \left\| A_k^T \hat{u}_k \right\|_2 \right)}{\left(\hat{u}_k^T A_k \hat{v}_k \right)}. \tag{4}$$

Theorem 2.1. *(see [3] and [4]) Let the matrices* $\hat{U}_k \in \mathbb{R}^{M\times k}$, $\hat{V}_k \in \mathbb{R}^{N\times k}$, *and* $\hat{D}_k \in \mathbb{R}^{k\times k}$ *be defined by the equalities*

$$\hat{U}_k = [\hat{u}_1\ \hat{u}_2\ \ldots\ \hat{u}_k],\ \hat{V}_k = [\hat{v}_1\ \hat{v}_2\ \ldots\ \hat{v}_k] \text{ and } \hat{D}_k = (\hat{\sigma}_1,\ \hat{\sigma}_2,\ \ldots,\ \hat{\sigma}_k), \tag{5}$$

where

$$\hat{\sigma}_j = \hat{u}_j^T A \hat{v}_j, \text{ for } j = 1,\ \ldots,\ r = rank(A). \tag{6}$$

Then, in exact arithmetic, the following relations hold for $k = 1,\ \ldots,\ r$*:*

$$Range\,(A_{k+1}) \subseteq Range\,(A_k),\ Range\left(A_{k+1}^T\right) \subseteq Range\left(A_k^T\right), \tag{7}$$

$$Range\left(\hat{U}_k\right) \subseteq Range\,(A),\ Range\left(\hat{V}_k\right) \subseteq Range\left(A^T\right), \tag{8}$$

$$\hat{U}_k^T A_{k+1} = 0,\ A_{k+1}\hat{V}_k = 0, \tag{9}$$

$$\hat{U}_k^T \hat{U}_k = I,\ \hat{V}_k^T \hat{V}_k = I. \tag{10}$$

Thus, for $k = r$ *the columns of* $\hat{U}_r$ *and* $\hat{V}_r$ *constitute orthonormal basis for Range*(A) *and Range*(A^T)*, respectively. Consequently,*

$$A_{r+1} = 0 \tag{11}$$

and

$$A = \tilde{U}_r \tilde{D}_r \tilde{V}_r^T. \tag{12}$$

The customization (see [9]) consists in a proper choice at each iteration for $\hat{u}_{i+1} \in \text{Range}\,(A_i)$ and $\hat{v}_{i+1} \in \text{Range}\left(A_i^T\right)$. We have tested several variants of choices for these initializations (each iteration), but the obtained results were not satisfactory. These customizations are presented in a submitted paper.

The COD algorithm is the following.

COD algorithm

1. Initialize $\hat{u}_1 \in \text{Range}\,(A)$, $\hat{u}_1 = \hat{u}_1 / \|\hat{u}_1\|$ *and* $\hat{v}_1$,
$\hat{v}_1 = \hat{v}_1 / \|\hat{v}_1\|$ *and* $A_1 = A$
2. for $i = 1,\ 2,\ \ldots,\ k$ *compute*
$\tilde{u}_i = A_i \hat{v}_i / \|A_i \hat{v}_i\|_2$
$\tilde{v}_i = A_i^T \hat{u}_i / \left\| A_i^T \hat{u}_i \right\|_2$

$\tilde{\sigma}_i = \left(\|A_i * \hat{v}_i\|_2 \left\| A_i^T \hat{u}_i \right\|_2 \right) / \left(\hat{u}_i^T A_i \hat{v}_i \right)$
$A_{i+1} = A_i - \tilde{\sigma}_i \tilde{u}_i \tilde{v}_i^T$
choose $\hat{u}_{i+1} \in \text{Range}\,(A_{i+1})$, $\hat{u}_{i+1} = \hat{u}_{i+1} / \|\hat{u}_{i+1}\|$
choose $\hat{v}_{i+1} \in \text{Range}\left(A_{i+1}^T\right)$, $\hat{v}_{i+1} = \hat{v}_{i+1} / \|\hat{v}_{i+1}\|$
end
3. Let $U = [\tilde{u}_1\ \tilde{u}_2\ \ldots\ \tilde{u}_k]$ *and* $B = A_{k+1}$.
4. Obtain $\Omega_i^T = \left[\omega_1^i,\ \omega_2^i, \ldots, \omega_k^i\right]$ *where* $col_i A = \sum_{j=1}^{k} \omega_j^i \tilde{u}_j$.
5. Given a image Γ, *obtain* $\Gamma = \sum_{j=1}^{k} \omega_j \tilde{u}_j$.
6. Represent Γ *as* $\Omega^T = [\omega_1,\ \omega_2, \ldots, \omega_k]$.
7. Find $i_0 \in \{1, \ldots, k\}$ *satisfying* $\left\|\Omega - \Omega_{i_0}\right\| = \min_{1 \le i \le k} \|\Omega - \Omega_i\|$.

As mentioned before, after extensive tests, in which we have tried to give a proper initialization to $\hat{u}_{i+1}$ and $\hat{v}_{i+1}$, we obtained that the best choices we can make for this type of face recognition problem are the ones described in [9].

2.3. LANCZOS AND BLOCK LANCZOS ALGORITHMS

The Lanczos method tries to find the optimal eigen-subspace of a matrix A in a Krylov subspace (see [7]):

$$K(A, q_1, k) = span\left\{q_1, Aq_1, \ldots, A^{k-1} q_1\right\}. \tag{13}$$

The orthonormal basis Q_k of $K(A, q_1, k)$ can be efficiently computed via the Lanczos procedure. Accordingly, A can be approximated as $A \approx Q_k T_k Q_k^T$, where T_k is a tridiagonal matrix:

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{bmatrix} \tag{14}$$

The Lanczos method generates a sequence of tridiagonal matrices T_k with the property that the extremal eigenvalues of $T_k \in \boldsymbol{R}^{k \times k}$ are progressively better estimates of matrix A extremal eigenvalues (see [7]).

Lanczos algorithm:

Input: $m \times m$ symmetric matrix A, k.
1. Initialization: $r_0 = q_1;\ \beta_0 = 1;\ q_0 = 0;\ l = 0.$
2. for l = 1 : k-1 do
$q_{l+1} = \frac{r_l}{\beta_l}; l = l + 1; \alpha_l = q_l^T A q_l;$
$r_l = Aq_l - \alpha_l q_l - \beta_{l-1} q_{l-1};$
$\beta_l = \|r_l\|_2 ;$
end for
Output: $Q_k = (q_1, \ldots, q_k)$ *and* T_k *as (14).*

The block Lanczos procedure tries to find an approximation of $A:\ A \approx Q_k T_k Q_k^T$, where T_k is a block tridiagonal matrix (see [7]):

$$T_k = \begin{bmatrix} M_1 & B_1^T & & & \\ B_1 & M_2 & B_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & B_{k-2} & M_{k-1} & B_{k-1}^T \\ & & & B_{k-1} & M_k \end{bmatrix} \tag{15}$$

and columns of $Q_k = (X_1, \ldots, X_k)$ are orthonormal. By comparing the left and right hand sides of $Q_k A = Q_k T_k$, we have

$$AX_r = X_{r-1} B_{r-1}^T + X_r M_r + X_{r+1} B_r,\ r = 1, \ldots, k-1, \tag{16}$$

where B_0 is defined to be 0. From the orthogonality of Q, we have that

$$M_r = X_r^T A X_r,\ r = 1, \ldots, k. \tag{17}$$

Another version of Block Lanczos algorithm is Block Lanczos with Warm Start (BLWS) (see [17]) where, at each iteration matrix X_r is initialized with the subspace obtained in the previous iteration.

In our experiments we propose as a warm start for X_1 a fixed number of singular values of matrix A computed apriori as in [10]. This implementation offers us a good starting point and ensure us of satisfactory results.

Block Lanczos Algorithm:

Input: $m \times m$ symmetric matrix A, $m \times d$ orthogonal matrix X_1, k.
1. Initialization: $M_1 = X_1^T A X_1;\ B_0 = 0.$
2. for r = 1 : k - 1 do
$R_r = AX_r - X_r M_r - X_{r-1} B_{r-1}^T;$
$(X_{r+1}, B_r) = qr(R_r);$ *(The QR decomposition)*
$M_{r+1} = X_{r+1}^T A X_{r+1};$
end for
Output: $Q_k = (X_1, \ldots, X_k)$ *and* T_k *as in (15).*

3. EXPERIMENTS

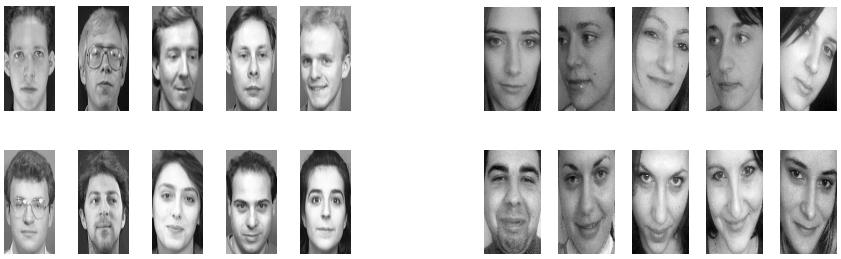
In our experiments we use three datasets.

1. The ORL dataset (also known as AT&T dataset) consists of pictures expressing different facial expressions.

2. The CTOVF face dataset (generated by ourselves) consists of pictures expressing different facial expressions and the head position of the subjects is not always straight.

3. A larger dataset composed of the two datasets mentioned above.

	ORL	**CTOVF**	**ORL+CTOVF**
# subjects	40	11	51
# images	400	110	510
resolution	92×112	92×112	92×112



ORL dataset CTOVF dataset

Fig. 1.: Samples of the used datasets

The experiments were performed from two different viewpoints: first, we take each dataset (ORL and CTOVF) as a different entity in comparing the recognition rate obtained with each of them, latter, we consider a larger dataset, composed of the other two mentioned above.

We have divided the dataset into two nonoverlapping subsets: training subset and testing subset. All three datasets have 10 images for every person. We have conducted two types of experiments: for the first type we put 8 (80%) out of all 10 images in the training set and 2 (20%) images in the testing set and for the second type we put 9 (90%) images in the training set and one image (10%) in the testing set. For all these types of splitting we used different levels of truncation k.

We use two performance indicators: recognition rate and average query time. Recognition rate is the ratio between the number of correct results given by the algo-

rithm (true positive) and the total number of tests performed with that algorithm. The average query time is the average of all query times when searching to identify a person. The preprocessing time is the dataset preparation stage, when we construct the smaller dimension subspace and project the dataset onto it. This stage is performed only once, before starting the search for a person. Since it is performed only once, in what follows, we will not take into account this performance indicator.

	COD		PCA		Lanczos		Block Lanczos	
	RR	AQT	RR	AQT	RR	AQT	RR	AQT
k=40	95%	0.0027	70%	0.0046	95%	0.0025	95%	0.0019
k=60	95%	0.0030	71.25%	0.0041	95%	0.0031	96.25%	0.0025
k=80	95%	0.0034	72.25%	0.0043	96.25%	0.0035	96.25%	0.0029

Table 1: ORL 80%+20%

where RR=Recognition Rate, AQT=Average Query Time (in seconds), and k=level of truncation.

In Table 1 are presented the results for the ORL dataset, 80% of images for training and 20% of the images for testing (80%+20%). In this case the highest recognition rate is 96.25% obtained for Lanczos and Block Lanczos algorithms.

	COD		PCA		Lanczos		Block Lanczos	
	RR	AQT	RR	AQT	RR	AQT	RR	AQT
k=40	95%	0.0026	72.5%	0.0043	95%	0.0027	95%	0.0021
k=60	92.5%	0.0035	72.5%	0.0045	95%	0.0033	95%	0.0027
k=80	92.5%	0.0037	72.5%	0.0048	95%	0.0036	95%	0.0031

Table 2: ORL 90%+10%

In Table 2 are presented the results for the ORL dataset, 90% of images for training and 10% of the images for testing (90%+10%). In this case the highest recognition rate is 95% obtained for COD, Lanczos, and Block Lanczos algorithms.

In Table 3 are presented the results for the CTOVF dataset, 80% of images for training and 20% of the images for testing (80%+20%). In this case the highest recognition rate is 90.91% obtained for COD, Lanczos, and Block Lanczos algorithms.

In Table 4 are presented the results for the CTOVF dataset, 90% of images for training and 10% of the images for testing (90%+10%). In this case the highest recognition rate is 90.91% obtained for Lanczos algorithm for $k = 40$, in the rest

	COD		PCA		Lanczos		Block Lanczos	
	RR	AQT	RR	AQT	RR	AQT	RR	AQT
k=40	86.36%	0.0016	86.36%	0.0012	86.36%	0.0013	90.91%	0.0008
k=60	90.91%	0.0019	86.36%	0.0012	90.91%	0.0019	90.91%	0.0013
k=80	90.91%	0.0023	86.36%	0.0015	90.91%	0.0026	90.91%	0.0018

Table 3: CTOVF 80%+20%

	COD		PCA		Lanczos		Block Lanczos	
	RR	AQT	RR	AQT	RR	AQT	RR	AQT
k=40	81.82%	0.0015	72.73%	0.0015	90.91%	0.0018	81.82%	0.0009
k=60	81.82%	0.0020	72.73%	0.0015	81.82%	0.0022	81.82%	0.0014
k=80	81.82%	0.0024	72.73%	0.0017	81.82%	0.0027	81.82%	0.0019

Table 4: CTOVF 90%+10%

of the combinations the recognition rate is 81.82% obtained for COD, Lanczos, and Block Lanczos algorithms.

	COD		PCA		Lanczos		Block Lanczos	
	RR	AQT	RR	AQT	RR	AQT	RR	AQT
k=40	93.14%	0.003	71.57%	0.0051	93.14%	0.0032	92.16%	0.0023
k=60	92.16%	0.0035	71.57%	0.0050	93.14%	0.0035	94.12%	0.0029
k=80	91.18%	0.0042	71.57%	0.0054	94.12%	0.0039	95.1%	0.0037

Table 5: ORL+CTOVF 80%+20%

In Table 5 are presented the results for the ORL+CTOVF dataset, 80% of images for training and 20% of the images for testing (80%+20%). In this case the highest recognition rate is 95.1% obtained for the Block Lanczos algorithm.

	COD		PCA		Lanczos		Block Lanczos	
	RR	AQT	RR	AQT	RR	AQT	RR	AQT
k=40	92.16%	0.0032	78.43%	0.0055	94.12%	0.0032	94.12%	0.0026
k=60	94.12%	0.0038	78.43%	0.0057	94.12%	0.0038	92.16%	0.0032
k=80	90.2%	0.0044	78.43%	0.0061	94.12%	0.0046	92.16%	0.0039

Table 6: ORL+CTOVF 90%+10%

In Table 6 are presented the results for the ORL+CTOVF dataset, 90% of images for training and 10% of the images for testing (90%+10%). In this case the highest recognition rate is 94.12% obtained for for COD, Lanczos, and Block Lanczos algorithms.

The results for the recognition rate from Table 7 are computed as a weighted average $(80 * RR_{ORL} + 22 * RR_{CTOVF})/102$, where RR_{ORL} is the recognition rate for ORL dataset and RR_{CTOVF} is the recognition rate for CTOVF dataset, for the splitting 80% of images for training and 20% of the images for testing (80%+20%). We wanted to check if the fact that we considered the two datasets as a single one, influenced in any way the results in terms of recognition rate. From Tables 5 and 7 we can conclude that the recognition rate obtained for the larger dataset is almost the same as the weighted average for the recognition rates for the ORL and CTOVF datasets.

ORL+CTOVF 80%+20%	COD	PCA	Lanczos	Block Lanczos
k=40	93.13%	73.52%	93.13%	94.11%
k=60	94.11%	74.50%	94.11%	95.09%
k=80	94.11%	75.29%	95.09%	95.09%

Table 7: ORL+CTOVF 80%+20%

The results for the recognition rate from Table 8 are computed as a weighted average $(40 * RR_{ORL} + 11 * RR_{CTOVF})/51$, where RR_{ORL} is the recognition rate for ORL dataset and RR_{CTOVF} is the recognition rate for CTOVF dataset, for the splitting 90% of images for training and 10% of the images for testing (90%+10%). From Tables 6 and 8 we can conclude that the recognition rate obtained for the larger dataset is almost the same as the weighted average for the recognition rates for the ORL and CTOVF datasets.

ORL+CTOVF 80%+20%	COD	PCA	Lanczos	Block Lanczos
k=40	92.15%	72.54%	94.11%	92.15%
k=60	90.19%	72.54%	92.15%	92.15%
k=80	90.19%	72.54%	92.15%	92.15%

Table 8: ORL+CTOVF 90%+10%

4. CONCLUSIONS

In this paper we compare four algorithms: Eigenfaces, COD, Lanczos and Block Lanczos using two performance indicators: recognition rate and average query time. The results for the three datasets are different because they are also based on the specificity of the dataset, not only on the algorithm. All considered algorithms in this paper are real-time recognition methods (i.e. they give the answer in few milliseconds).

Our proposed algorithm COD (see [9]) has a higher recognition rate than the PCA algorithm and in most cases has about the same recognition rate as the Lanczos and Block Lanczos algorithms. It also has a smaller average query time than the Lanczos method in most cases.

Our intialization for Block Lanczos Warm Start algorithm gives the same recognition rate as the Lanczos method but smaller average query time. Other different initializations can be considered (see [1], [17], [10]).

The recognition rate obtained for the larger dataset (ORL and CTOVF as a single dataset) can be obtained as a weighted average of the recognition rate obtained for the ORL dataset and the one obtained for the CTOVF dataset.

References

[1] Cai J.-F., Candes E.J., Shen Z., *A Singular Value Thresholding Algorithm for Matrix Completion*, SIAM Journal on Optimization, 20 (4) (2010), 1956-1982.

[2] Chen J., Saad Y., *Lanczos Vectors versus Singular Vectors for Effective Dimension Reduction*, IEEE Transactions on Knowledge and Data Engineering, 21(8) (2009), 1091-1103.

[3] Dax A., *Orthogonalization Via Deflation: A Minimum Norm Approach for Low-Rank Approximations of a Matrix*, Journal of Computational and Applied Mathematics, **234(11)**, (2010), 3091-3103.

[4] Dax A., *A minimum norm approach for low-rank approximations of a matrix*, SIAM Journal on Matrix Analysis and Applications, **30(1)**, (2008), 236-260.

[5] Duda, R. O., Hart, P. E. and Stork, D. G., *Pattern Classification*, second ed., Wiley-Interscience, New York, 2001.

[6] Glineur F., Lu L., Van Dooren P., Wang X., *Extended Lanczos Bidiagonalization for Dimension Reduction in Information Retrieval*, 8th International Conference on Natural Computation (ICNC'12), Maui, Hawaii, 2012.

[7] Golub G., Van Loan C., *Matrix Computation, 3rd Edition*, The John Hopkins University Press, Baltimore 1996.

[8] Grecu (Liţă) L., Pelican E., *Comparative Study of NN, PCA and COD algorithms for Pattern Recognition*, Nineth Workshop on Mathematical Modelling of Environmental and Life Sciences Problems, November 1-4, 2012, Constanta, Romania.

[9] Grecu (Liţă) L., Pelican E., *Customized Orthogonalization via Deflation Algorithm with Applications in Face Recognition*, submitted.

[10] Larsen R.M., *Lanczos bidiagonalization with partial reorthogonalization,* Department of Computer Science, Aarhus University, Technical report, DAIMI PB-357, code available at http://soi.stanford.edu/r̃munk/PROPACK/, 1998.

[11] Murty M. N. and Susheela Devi V., *Pattern Recognition. An Algorithmic Approach*, Springer, First Edition, 2011.

[12] Pelican E., Grecu L., *Comparison Between Some Matrix Methods with Applications in Pattern Recognition*, Applied Linear Algebra Conference, May 24-28, 2010, Novi Sad.

[13] Pelican E., Grecu L., *Low-Rank Matrix Methods in Pattern Recognition*, Balkan Conference on Operational Research (BALCOR), (2009), ISBN: 973-86979-9-9.

[14] Pelican E., Grecu L., *Solving the Pattern Recognition Problem with some Low-Rank Approximation Based Algorithms*, XIème Colloque Franco-Roumain de Mathématiques Appliquées, 2012.

[15] Turk M., Pentland A., *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience, **1**, (1991), 71-86.

[16] Turk M., Pentland A., *Face Recognition using Eigenfaces* , Computer Vision and Pattern Recognition Proceedings CVPR '91, (1991), 586-591.

[17] Wei S., Lin Z., *Accelerating Iterations Involving Eigenvalue or Singular Value Decomposition by Block Lanczos with Warm Start*, Microsoft Technical Report MSR-TR-2010-162.