

Efficiently DOWndating, Composing and Splitting Singular Value Decompositions Preserving the Mean Information

Javier Melenchón, Elisa Martínez

Communications and Signal Theory Department,
Enginyeria La Salle, Universitat Ramon Llull,
Pg. Bonanova, 8, 08002 Barcelona, SPAIN
jmelen@salle.url.edu, elisa@salle.url.edu

Abstract. Three methods for the efficient downdating, composition and splitting of low rank singular value decompositions are proposed. They are formulated in a closed form, considering the mean information and providing exact results. Although these methods are presented in the context of computer vision, they can be used in any field forgetting information, combining different eigenspaces in one or ignoring particular dimensions of the column space of the data. Application examples on face subspace learning and latent semantic analysis are given and performance results are provided.

1 Introduction

Process analysis can be carried out by means of the observation of its related data. In general, the more data obtained, the more detailed the analysis can be; however, redundancy is also increased. The latter effect is specially important when managing high dimensional data, e.g., video sequences, images, audio waveforms and document sets, which makes the analysis harder; nevertheless, this kind of data can usually be approximated by a subspace of low dimension. Working with these subspaces in the analysis process has two main advantages: *i*) the laws or rules of the system are described in a more intuitive way; *ii*) processing algorithms tend to be faster and require less memory space. Therefore, dimensionality reduction techniques [1] are welcome to remove all possible redundant information of high dimensional data. The reader is referred to [2], [3] and [4] for some examples involving data analysis in low dimensional spaces.

Given a data set, the well-known Karhunen-Loève expansion [5] can find its optimal orthogonal basis, which is the expression of its underlying subspace. This expansion has received different names in the literature, e.g., principal component analysis (PCA) [6], and can be computed with the singular value decomposition (SVD) [7], which is a powerful mathematical tool of linear algebra. Therefore, the SVD can be used to find the subspace of any data set, e.g., the low dimensional subspace for video and image sequences of natural scenes.

However, SVD computation is rather expensive, specially when using a batch algorithm [7], requiring the whole data matrix at once. For high dimensional data

such as video sequences, this last fact is a key point, as they can easily exhaust memory resources. Incremental schemes have recovered importance in the last decade, with the increasing interest in video and image sequence processing. Most efforts have been directed to incremental computation of SVD, but little attention has been given to other cases, as SVD downdating (or decremental SVD), composing little subspaces into a higher dimensional one or splitting an existing subspace into little ones.

1.1 Related Work

The first work introducing an incremental computation of SVD in the field of computer vision was [8]; although it is an efficient algorithm based on eigenvalue decomposition (EVD), it can only update one vectorized image per iteration (or column update), it does not account for the mean information and has some potential numerical instability; moreover, only the left singular vectors and singular values are obtained. Chandrasekaran et al. [9] proposed a more stable update algorithm based on the work of Gu et al. [10], where a direct SVD readjustment was given, also updating the right singular values; however it still cannot update more than one column per iteration and does not take into account the mean information. Fortunately, Hall et al. [11] included it in the incremental computation of SVD, also allowing for multiple columns update (or block update). This work is based on EVD, achieving a method to merge and split subspaces, which can be seen as updating and downdating the SVD; however, they only offer the left singular vectors and singular values. Later, these authors presented another approximation [12] based on direct SVD updating, obtaining the right singular vectors and achieving better numerical stability; nevertheless, they claimed that SVD downdate was impossible to achieve in closed form with their formulation. Brand [13] proposed a highly efficient and stable incremental SVD algorithm with block update and also pointed out a way of adapting the eigenspace defined by the SVD to nonstationary systems by means of decaying singular values; however, he did not take into account the mean information. The work of Skocaj et al. [14] is very similar to that of [11]; eigendecomposition is used, mean information update is taken into account and robust features are presented, however block update is not considered. Melenchon et al. [15] extended the work of [13] proposing a novel incremental algorithm through a reorthonormalization process, allowing block update with the stability and efficiency of [13] and mean update like in [12]; however, SVD downdating is not addressed in their work. Finally, Lim et al. [16] presented a new alternative based on R-bidiagonalization SVD (RSVD) [7]; mean update is taken into account, block update is offered and a forgetting factor is introduced, based on [17] and similarly to [13]. However, old information is progressively forgotten but never removed in [13] and [16]. The reader is referred to table 1 for a summary about the described works. Additional SVD early history can be read in [18].

Table 1. Evolution about incremental computation of SVD in computer vision fields. Main features of past and proposed methods. Note the decaying singular values of [13], the robust features of [14], and the forgetfulness of [16].

Work	Year	Method	Update columns	Mean update	Down-date	Miscellanea
[8] Murakami et al.	1982	EVD	single	no	no	
[9] Chandrasekaran et al.	1997	SVD	single	no	no	
[11] Hall et al.	2000	EVD	multiple	yes	yes	
[12] Hall et al.	2002	SVD	multiple	yes	limited	
[13] Brand	2002	SVD	multiple	no	no	decaying
[14] Skocaj et al.	2003	EVD	single	yes	no	robust
[15] Melenchón et al.	2004	SVD	multiple	yes	no	
[16] Lim et al.	2005	RSVD	multiple	yes	no	forgetful
This paper	2006	SVD	multiple	yes	yes	$\left\{ \begin{array}{l} \text{splitted} \\ \text{composed} \end{array} \right.$

1.2 Contributions

Three novel methods for efficiently downdating the SVD (section 2.1), composing different SVD's (section 2.2) and splitting existing ones (section 2.3) are proposed in this work. They are presented in closed form, preserve the mean information and are based on the reorthonormalization process and mean extraction of [15]. Moreover, they become extremely efficient for low rank SVD's. Their application to video sequence data is shown in section 3. Additional experiments have been conducted with textual information in the Latent Semantic analysis (LSA) framework [3] to test the methods with sparse matrices. Concluding remarks and future work are provided in section 4.

2 SVD Computation

Let $\mathbf{D}_{m \times n}$ be a real matrix of full rank $r = \min(m, n)$, then its singular value decomposition can be expressed as a sum of r rank one matrices (matrix size will only be shown when necessary for clarity purposes):

$$\mathbf{D}_{m \times n} = \mathbf{U}'_{m \times m} \mathbf{\Sigma}'_{m \times n} (\mathbf{V}'_{n \times n})^T = \sum_{i=1}^r \sigma'_i \mathbf{u}'_i (\mathbf{v}'_i)^T . \quad (1)$$

where $\mathbf{U}' = [\mathbf{u}'_1 \dots \mathbf{u}'_m]$ and $\mathbf{V}' = [\mathbf{v}'_1 \dots \mathbf{v}'_n]$ are orthonormal matrices containing the eigenvectors of $\mathbf{D}\mathbf{D}^T$ and $\mathbf{D}^T\mathbf{D}$, respectively (a.k.a. right and left singular vectors of \mathbf{D}), and $\mathbf{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_r)$ is a diagonal matrix with the eigenvalues of both $\mathbf{D}\mathbf{D}^T$ and $\mathbf{D}^T\mathbf{D}$ (a.k.a. singular values of \mathbf{D}) in descending order. The SVD finds the best rank k approximation matrix of \mathbf{D} . Any other rank k matrix \mathbf{B} that is not the rank k SVD approximation will have greater error:

$$\underset{\text{rank}(\mathbf{B})=k}{\text{argmin}} \|\mathbf{D}_{m \times n} - \mathbf{B}_{m \times n}\|_2 = \mathbf{A}_{m \times n} = \mathbf{U}'_{m \times k} \mathbf{\Sigma}'_{k \times k} (\mathbf{V}'_{n \times k})^T . \quad (2)$$

where $\|\mathbf{D} - \mathbf{A}\|_2 = \sigma'_{k+1}$. The reader is referred to [7] for a proof of (2). The SVD of \mathbf{A} (2) is known as the truncated SVD of \mathbf{D} and also as its compact version when \mathbf{D} is a rank k matrix.

The SVD is often used when computing the PCA of a dataset \mathbf{D} approximated by \mathbf{A} , which needs the extraction of the mean information; moreover, it is needed in statistical methods used in classification problems, like mahalanobis distance. Let the mean be $\bar{\mathbf{a}}_{m \times 1} = m^{-1} \mathbf{A}_{m \times n} \cdot \mathbf{1}_{n \times 1}$, it is often required to obtain a mean centered dataset $\hat{\mathbf{A}} = \mathbf{A} - \bar{\mathbf{a}} \cdot \mathbf{1}$ with the following truncated SVD:

$$\mathbf{A}_{m \times n} = \bar{\mathbf{A}}_{m \times n} + \bar{\mathbf{a}}_{m \times 1} \cdot \mathbf{1}_{1 \times n} = \mathbf{U}_{m \times k} \boldsymbol{\Sigma}_{k \times k} (\mathbf{V}_{n \times k})^T + \bar{\mathbf{a}}_{m \times 1} \cdot \mathbf{1}_{1 \times n} \quad (3)$$

2.1 Decremental SVD

In this section, the problem of downdating the SVD preserving the mean information is addressed. Dropping columns of any data matrix can be considered as a radical forgetting action. Given (3), if p columns are removed from \mathbf{A} , a new matrix $\mathbf{A}_{m \times l}^d$ is obtained, where $l = n - p$. The SVD of $\mathbf{A}_{m \times l}^d$ can be updated efficiently with (4), without recomputing it from scratch.

$$\begin{aligned} \mathbf{A} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T + \bar{\mathbf{a}} \cdot \mathbf{1} &\Rightarrow \mathbf{A}^d = \mathbf{U} \boldsymbol{\Sigma} \tilde{\mathbf{V}}^T + \bar{\mathbf{a}} \cdot \mathbf{1} = \mathbf{U} \boldsymbol{\Sigma} \hat{\mathbf{V}}^T + \mathbf{U} \boldsymbol{\Sigma} \bar{\mathbf{v}}^T \cdot \mathbf{1} + \bar{\mathbf{a}} \cdot \mathbf{1} = \\ &= \mathbf{U} \boldsymbol{\Sigma} \mathbf{R}^T \mathbf{Q}^T + (\Delta \bar{\mathbf{a}}^d + \bar{\mathbf{a}}) \cdot \mathbf{1} = \mathbf{U}^d \boldsymbol{\Sigma}^d \mathbf{V}_t^T \mathbf{Q}^T + \bar{\mathbf{a}}^d \cdot \mathbf{1} = \\ &= \mathbf{U}_{m \times k}^d \boldsymbol{\Sigma}_{k \times k}^d (\mathbf{V}_{l \times k}^d)^T + \bar{\mathbf{a}}_{m \times 1}^d \cdot \mathbf{1}_{1 \times l} = \mathbf{A}_{m \times l}^d \end{aligned} \quad (4)$$

here, matrix $\tilde{\mathbf{V}}^T$ is not orthonormal and contains the columns of \mathbf{V}^T corresponding to the non-dropped columns of \mathbf{A} . It is centered around its mean row $\bar{\mathbf{v}} = l^{-1} (\mathbf{1}_{1 \times l} \cdot \tilde{\mathbf{V}}_{l \times k})$, obtaining matrix $\hat{\mathbf{V}}$. The mean update $\Delta \bar{\mathbf{a}}^d$ is computed as $\mathbf{U} \boldsymbol{\Sigma} \bar{\mathbf{v}}^T$. The updated mean $\bar{\mathbf{a}}^d$ is obtained as $\bar{\mathbf{a}} + \Delta \bar{\mathbf{a}}^d$. Finally, the expression $\mathbf{U} \boldsymbol{\Sigma} \hat{\mathbf{V}}^T$ is reorthonormalized as $\mathbf{U}^d \boldsymbol{\Sigma}^d (\mathbf{V}^d)^T$ with the QR decomposition of $\hat{\mathbf{V}} = \mathbf{Q} \mathbf{R}$, the SVD of $\mathbf{U} \boldsymbol{\Sigma} \mathbf{R}^T = \mathbf{U}^d \boldsymbol{\Sigma}^d \mathbf{V}_t^T$ (with lower cost than that of $\mathbf{U} \boldsymbol{\Sigma} \hat{\mathbf{V}}^T$, since $k < n$) and the identity $\mathbf{V}^d = \mathbf{Q} \mathbf{V}_t$.

2.2 Composed SVD

Given matrices \mathbf{A} , \mathbf{B} and their compact SVD's (5), (6), that of $\mathbf{C} = [\mathbf{A}^T \mathbf{B}^T]^T$ (7) can be obtained efficiently from them (8), if $m = q + p$ and $k = r + s$.

$$\mathbf{A}_{q \times n} = \mathbf{U}_{q \times r}^A \boldsymbol{\Sigma}_{r \times r}^A (\mathbf{V}_{n \times r}^A)^T + \bar{\mathbf{a}}_{q \times 1}^A \cdot \mathbf{1}_{1 \times n} \quad (5)$$

$$\mathbf{B}_{p \times n} = \mathbf{U}_{p \times s}^B \boldsymbol{\Sigma}_{s \times s}^B (\mathbf{V}_{n \times s}^B)^T + \bar{\mathbf{a}}_{p \times 1}^B \cdot \mathbf{1}_{1 \times n} \quad (6)$$

$$\mathbf{C}_{m \times n} = \mathbf{U}_{m \times k}^C \boldsymbol{\Sigma}_{k \times k}^C (\mathbf{V}_{n \times k}^C)^T + \bar{\mathbf{a}}_{m \times 1}^C \cdot \mathbf{1}_{1 \times n} \quad (7)$$

$$\begin{aligned} \mathbf{C} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} &= \begin{bmatrix} \mathbf{U}^A \boldsymbol{\Sigma}^A (\mathbf{V}^A)^T \\ \mathbf{U}^B \boldsymbol{\Sigma}^B (\mathbf{V}^B)^T \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{a}}^A \\ \bar{\mathbf{a}}^B \end{bmatrix} = \begin{bmatrix} \mathbf{U}^A \boldsymbol{\Sigma}^A \mathbf{R}_A^T \\ \mathbf{U}^B \boldsymbol{\Sigma}^B \mathbf{R}_B^T \end{bmatrix} \mathbf{Q}_t^T + \begin{bmatrix} \bar{\mathbf{a}}^A \\ \bar{\mathbf{a}}^B \end{bmatrix} = \\ &= \mathbf{U}^C \boldsymbol{\Sigma}^C \mathbf{V}_t^T \mathbf{Q}_t^T + \bar{\mathbf{a}}^C = \mathbf{U}^C \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T + \bar{\mathbf{a}}^C \end{aligned} \quad (8)$$

where the QR decomposition of $[\mathbf{V}^A \ \mathbf{V}^B]$ is $\mathbf{Q}_t [\mathbf{R}_A \ \mathbf{R}_B]$ and the SVD is done to matrix $[(\mathbf{U}^A \boldsymbol{\Sigma}^A \mathbf{R}_A^T)^T (\mathbf{U}^B \boldsymbol{\Sigma}^B \mathbf{R}_B^T)^T]^T$, obtaining $\mathbf{U}^C \boldsymbol{\Sigma}^C \mathbf{V}_t^T$ (with lower cost than computing the SVD of $[\mathbf{A}^T \ \mathbf{B}^T]^T$), as $k < n$); finally, matrix \mathbf{V}^C is computed as $\mathbf{Q} \mathbf{V}_t$ and $\bar{\mathbf{a}}^C$ is obtained as the vertical concatenation of $\bar{\mathbf{a}}^A$ and $\bar{\mathbf{a}}^B$. Any SVD's of matrices with the same number of columns can be composed.

2.3 Splitted SVD

This case is the opposite of that in section 2.2: given a SVD (7), the desire is to obtain two (or more) SVD's, (5) and (6), splitting the subspace definition of the starting one (in this case $k = r = s$). It can be achieved as follows, using the reorthonormalization process proposed in section 2.1:

$$\begin{aligned} \mathbf{C} &= \begin{bmatrix} \mathbf{U}_s^A \\ \mathbf{U}_s^B \end{bmatrix} \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T + \begin{bmatrix} \bar{\mathbf{a}}^A \\ \bar{\mathbf{a}}^B \end{bmatrix} = \begin{bmatrix} \mathbf{Q}^A \mathbf{R}^A \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T \\ \mathbf{Q}^B \mathbf{R}^B \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{a}}^A \\ \bar{\mathbf{a}}^B \end{bmatrix} = \\ &= \begin{bmatrix} \mathbf{Q}^A \mathbf{U}_t^A \boldsymbol{\Sigma}^A (\mathbf{V}^A)^T \\ \mathbf{Q}^B \mathbf{U}_t^B \boldsymbol{\Sigma}^B (\mathbf{V}^B)^T \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{a}}^A \\ \bar{\mathbf{a}}^B \end{bmatrix} = \begin{bmatrix} \mathbf{U}^A \boldsymbol{\Sigma}^A (\mathbf{V}^A)^T + \bar{\mathbf{a}}^A \\ \mathbf{U}^B \boldsymbol{\Sigma}^B (\mathbf{V}^B)^T + \bar{\mathbf{a}}^B \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}. \quad (9) \end{aligned}$$

where $(\mathbf{U}^C)^T = [(\mathbf{U}_s^A)^T (\mathbf{U}_s^B)^T]^T$ and $(\bar{\mathbf{a}}^C)^T = [(\bar{\mathbf{a}}^A)^T (\bar{\mathbf{a}}^B)^T]^T$; QR decomposition is done to $\mathbf{U}_s^A = \mathbf{Q}^A \mathbf{R}^A$ and $\mathbf{U}_s^B = \mathbf{Q}^B \mathbf{R}^B$; the SVD is computed to $\mathbf{R}^A \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T = \mathbf{U}_t^A \boldsymbol{\Sigma}^A (\mathbf{V}^A)^T$ and $\mathbf{R}^B \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T = \mathbf{U}_t^B \boldsymbol{\Sigma}^B (\mathbf{V}^B)^T$ with lower cost than those of $\mathbf{U}_s^A \boldsymbol{\Sigma}^A (\mathbf{V}^A)^T$ and $\mathbf{U}_s^B \boldsymbol{\Sigma}^B (\mathbf{V}^B)^T$, since $k < m$; finally, $\mathbf{U}^A = \mathbf{Q}^A \mathbf{U}_t^A$ and $\mathbf{U}^B = \mathbf{Q}^B \mathbf{U}_t^B$. Rows of matrix \mathbf{C} can be reordered with some permutation matrix \mathbf{P} , obtaining \mathbf{C}_r ; therefore, \mathbf{U}^C and $\bar{\mathbf{a}}^C$ can also be reordered by the same \mathbf{P} , further obtaining \mathbf{U}_r^C and $\bar{\mathbf{a}}_r^C$ in (10). Consequently, rows of matrix \mathbf{C} can be splitted into any desired groups: first, row grouping can be achieved with (10); second, (9) can be applied to the reordered SVD.

$$\mathbf{C}_r = \mathbf{P} \mathbf{C} = \mathbf{P} \mathbf{U}^C \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T + \mathbf{P} \bar{\mathbf{a}}^C = \mathbf{U}_r^C \boldsymbol{\Sigma}^C (\mathbf{V}^C)^T + \bar{\mathbf{a}}_r^C. \quad (10)$$

3 Results

The proposed algorithms of decremental, composed and splitted SVD have been applied to computer vision and LSA fields. Datasets and tests are presented in sections 3.1 and 3.2. A 3GHz processor with 2GB of RAM has been used.

3.1 Test Data Used

A video data sequence with a size of 320×240 and 482 frames (at 25 fps) has been recorded for the testing of the performance of given algorithms, similarly to [14–16]. This sequence shows a human face while it is speaking and making different gestures. Using a tracking algorithm like in [15] and [16], the face in each frame can be aligned w.r.t. the first one, so pixel value variations are due only to appearance changes. Collecting all face pixels into columns (one per frame), a

matrix \mathbf{F} can be obtained; if the face pixels are classified into R regions, then R matrices \mathbf{F}_r can be built, one per facial region. Here, the forehead, both eyes and mouth have been considered. Moreover, a big textual sparse matrix \mathbf{T} [19] related to LSA has been also considered. The columns represent its 600 documents and the rows offer its 12018 possible words. Efficient SVD algorithms can be applied to this kind of matrices [20]; however, they are outperformed by the proposed algorithms (see 3.2).

3.2 Experiments

The computational cost of the three proposed methods is $O(k^2(k+m+n))$, while the memory requirements are $O(km+kn)$. Direct batch computation algorithms for the decremental, composed and splitted schemes have been considered for comparison purposes and involve recomputing original matrices and obtaining the desired SVD's directly. They have the same computational and memory costs, $O(m^2n+n^2m)$ and $O(mn)$, which are higher than the proposed ones when $n \gg k$ and $m \gg k$. In all the experiments, the results offered by both approaches were practically the same (spectral norms of difference matrices are less than 10^{-13}). Figure 1 shows the executed experiments.

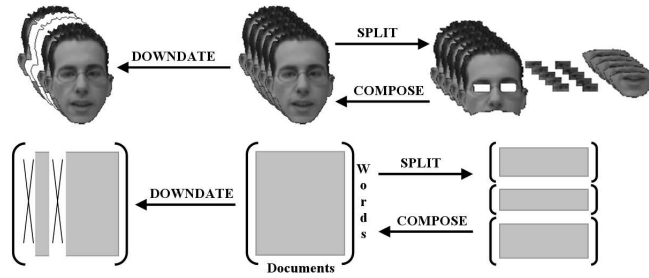


Fig. 1. Description of the experiments on facial appearance (up) and textual data (bottom), both represented by a matrix; the former has faces in its columns and pixels in its rows and the latter has documents in its columns and words in its rows. From left to right: the original data without some faces or documents left; the whole grouped data; the splitted data into 4 facial regions or 3 word sets. Every case has a SVD or set of SVD's; the left decompositions can be provided by downdating the central ones with the decremental SVD; the right ones can be achieved dividing the central decompositions with the splitted SVD, which can also be obtained with the composed SVD from the right ones.

Starting from the truncated SVD of the four matrices \mathbf{F}_r of ranks 17, 10, 9 and 14, corresponding to forehead, both eyes and mouth, respectively, the intention is to compose them in order to obtain the truncated SVD of rank $50 = 17 + 10 + 9 + 14$ of the whole matrix \mathbf{F} . Using the composed SVD algorithm (sect. 2.2) both time and memory requirements are reduced w.r.t. the batch approaches. If this matrix was to be divided into the original four facial regions \mathbf{F}_r ,

the splitted SVD of section 2.3 obtains the original truncated SVD’s faster and with less memory requirements than the batch process; note that each truncated SVD keeps its original rank. Next, considering the SVD of \mathbf{F} , the information of the first 52 frames and the ones from 266 to 325 are removed; the decremental SVD stated in section 2.1 can be executed, obtaining the result far sooner than with the batch scheme and with less memory resources (see table 2).

Regarding the textual data matrix \mathbf{T} , a truncated SVD of rank 40 is initially computed. First, the intention is to forget the last 200 documents (as if they had never been observed); with the explained decremental SVD, the time and memory resources spent are of smaller orders of magnitude than those of the batch method. Second, the original truncated SVD of \mathbf{T} is divided into three textual LSA, grouping the first 6437 words, the next 1814 ones and the rest; with the proposed splitted SVD, the result is obtained in less time and with less memory requirements than with the batch process. Joining the resulting SVD’s can be done with the composed SVD, which, like before, takes less time and memory than the batch method (table 2).

Table 2. Computational, memory resources and relative errors of the proposed algorithms and their batch counterpart. The latter are obtained with the norm $\|\cdot\|_2$.

Operation	Source data	Time		Memory		Error	
		Proposed	Batch	Proposed	Batch	Proposed	Batch
Decremental	Images	1’53	30’14	24’43	201’94	$< 10^{-14}$	$< 10^{-13}$
Decremental	Text	0’41	13’08	8’00	79’84	$< 10^{-14}$	$< 10^{-13}$
Composed	Images	1’48	46’09	39’86	245’27	$< 10^{-14}$	$< 10^{-13}$
Composed	Text	2’95	27’19	38’98	131’62	$< 10^{-14}$	$< 10^{-14}$
Splitted	Images	1’42	79’00	28’55	141’27	$< 10^{-14}$	$< 10^{-14}$
Splitted	Text	0’44	38’33	9’67	64’66	$< 10^{-13}$	$< 10^{-14}$

4 Concluding Remarks and Future Work

In this work, three novel methods for SVD computation have been presented. One for downdating the SVD, another one for joining or composing SVD’s, increasing the dimension of the final column space, and a last one for dividing or splitting a SVD, obtaining smaller SVD’s. They are very efficient for truncated SVD’s and for compact ones of low rank matrices, with sublinear computational and memory costs w.r.t. the amount of data (and cubic and linear w.r.t. the rank, which is of smaller orders of magnitude than the size). Moreover, they are solved in closed form and do not provide additional error terms. Some application examples of face subspace learning and LSA have been provided. Future work involving the application of these methods to face segmentation, tracking and synthesis will be carried out.

Acknowledgments. Thanks to G. Cobo and X. Sevillano for the LSA data.

References

1. Kirby, M.: Geometric Data Analysis. An empirical Approach to Dimensionality Reduction and the Study of Patterns. John Wiley and Sons, Inc., New York. (2001)
2. Sirovich, L., Kirby, M.: A low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, Vol. 4. (1987) 529–524
3. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, Vol. 41, No. 6. (1990) 391–407
4. Turk, M., Pentland, A.: Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, Vol. 3, No. 1. (1991) 71–86
5. Loève, M.: Probability Theory. Van Nostrand, Princeton, New Jersey. (1955)
6. Jolliffe, I.T.: Principal Component Analysis. Springer-Verlag, New York. (1986)
7. Golub, G.H., Van Loan, C.F.: Matrix Computations, Johns Hopkins University Press (1983)
8. Murakami, H., Kumar, V.: Efficient calculation of primary images from a set of images. *Trans. on PAMI*, Vol. 4, No. 5. (1982) 511–515
9. Chandrasekaran, S., Manjunath, B., Wang, Y., Winkeler, J., Zhang, H.: An Eigenspace Update Algorithm for Image Analysis. *Graphical Models and Image Processing*, Vol. 59, No. 5. (1997) 321–332
10. Gu, M., Eisenstat, S.T.: A stable and fast algorithm for updating the singular value decomposition. Tech. report YALEU/DCS/RR-966. Department of Computer Science, Yale University, New Haven (1993)
11. Hall, P.M., Marshall, A.D., Martin, R.R.: Merging and Splitting Eigenspace Models. *Trans. on PAMI*, Vol. 22, No. 9. (2000) 1042–1049
12. Hall, P.M., Marshall, A.D. and Martin, R.R.: Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition. *Image and Vision Computing*, Vol. 20, No. 13-14. (2002) 1009–1016
13. Brand, M.: Incremental Singular Value Decomposition of Uncertain Data with Missing Values. *Proc. of ECCV'02*, Vol. 1. (2002) 707–ff
14. Skocaj, D., Leonardis, A.: Weighted and robust incremental method for subspace learning *Proc. of ICCV'03*, Vol. 2. (2003) 1494–1501
15. Melenchón, J., Meler, L., Iriondo, I.: On-the-fly Training. *Articulated Motion and Deformable Objects, LNCS*, Vol. 3179. 2nd best paper award (2004) 146–153
16. Lim, J., Ross, D., Lin, R.S., Yang, M.H.: Incremental Learning for Visual Tracking. *Advances in Neural Information Processing Systems*, Vol. 18. (2005) 793–800
17. Levy, A., Lindenbaum, M.: Sequential Karhunen-Loeve basis extraction and its application to images. *Trans. on Image Processing*, Vol. 9, No. 8. (2000) 1371–1374
18. Stewart, G.W.: On the early history of the singular value decomposition. *Society for Industrial and Applied Mathematics Review*, Vol. 35. (1993) 551–566
19. Cobo, G., Sevillano, X., Alías, F., Socoró, J.C.: Técnicas de representación de textos para clasificación no supervisada de documentos. *Procesamiento del Lenguaje Natural*, Vol. 37. (2006) 329–336
20. Lehoucq, R.B., Sorensen, D.C., Yang, C.: ARPACK Users' Guide. Society for Industrial and Applied Mathematics. (1998)