Statistics and Its Interface Volume 3 (2010) 419–426

Singular spectrum analysis for image processing

Licesio J. Rodríguez-Aragón, and Anatoly Zhigljavsky*

A technique of image processing based on application of the Singular Spectrum Analysis (SSA) is discussed and illustrated on the problem of denoising the celebrated 'Lena' image corrupted with noise. Also, SSA-based distances between two images are introduced and suggested for a possible use in the face verification problem.

AMS 2000 subject classifications: Primary 37M10, 62M10; secondary 94A08.
Keywords and phrases: Singular spectrum analysis, Time series analysis, Distance between images, Face verification.

1. INTRODUCTION

A digital image is generally encoded as a matrix of grey level or colour values. In the case of grey level images, each element of the matrix is a grey level, in the case of colour images each element of the grid is a triplet of values for red, green and blue components. The two main sources of errors in image processing are categorized as blur and noise. Blur is intrinsic to image acquisition systems as digital images have a finite number of samples. Noise is another source of error; as indicated below there are different types of noise.

The search for efficient denoising methods is still a valid challenge in image processing [1]. In this paper we use the celebrated 'Lena' image to illustrate the performance of SSA for the problem of image denoising. We also introduce SSA-related distances between images and argue that these distances can be much more informative than the currently used L_2-based distances.

2. SSA FOR IMAGE PROCESSING

Assume that we have an image I of size $h\times w$ represented in the form of a matrix,

$$I = \|I_{i,j}\|_{\substack{i=1,\dots,h\\ j=1,\dots,w}} \tag{1}$$

where, for example, the values $I_{i,j}$ code the intensity of either a colour or a grey level, $0 \le I_{i,j} \le 255$.

We are going to study the extension of the Basic SSA procedure, as described in [12], for analyzing images. The first stage of the procedure is the transformation of the data (which is now the matrix (1) rather than a time series) into another matrix which is a version of the trajectory matrix in the Basic SSA. This matrix will also be called the trajectory matrix and denoted $\mathbf{X}$. The trajectory matrix $\mathbf{X}$ will be constructed so that it will contain all the information from the image and, moreover, it will retain the information about the neighbours of all pixels.

*Corresponding author.

2.1 Construction of the trajectory matrix

Similarly to the Basic SSA, we need to define the window which will be moved over the image. Unlike the one-dimensional case, our window has not only width, but height as well. Let the window have size $u \times v$, with $1 \le u \le h$, $1 \le v \le w$.

The window is then placed at all possible positions in the image. We shall refer to the top-left point in the window as 'the reference point of the window'; the range of all reference points is (i, j) with

$$1 \le i \le \tilde{u} = h - u + 1, \quad 1 \le j \le \tilde{v} = w - v + 1.$$

The window is moved from left to right and top to bottom. That is, the reference point is moved as follows:

$$(i,j) \to \begin{cases} (i, j+1) & \text{if } j \le w - v \\ (i+1, 1) & \text{if } j = w - v + 1 \text{ and } i \le h - u. \end{cases}$$

The window $W_{i,j}$, with reference point (i, j), covers a region of the image of size $u \times v$:

$$W_{i,j} = \|I_{i+k-1,j+l-1}\|_{\substack{k=1,\dots,u\\ l=1,\dots,v}} .$$

The elements of all windows W are then rearranged into column vectors using the *vec* operation; that is, by transposing rows one by one and writing them consecutively:

$$\overrightarrow{W} = \mathrm{vec}(W) = (W_1^T, W_2^T, \dots, W_u^T)^T,$$

where W_i is the ith row of the window W. In this way, all the windows $W_{i,j}$ are transformed into the vectors

$$\overrightarrow{W}_{i,j} = \mathrm{vec}(W_{i,j}) = (I_{i,j}, I_{i,j+1}, \dots, I_{i,j+v-1}, \\ I_{i+1,j}, \dots, I_{i+1,j+v-1}, \dots, I_{i+u-1,j+v-1})^T \in \mathbb{R}^{uv} .$$

Finally, the vectors $\overrightarrow{W}_{i,j}$ are arranged into the trajectory matrix $\mathbf{X}$ in the following way:

$$\mathbf{X} = (\overrightarrow{W}_{1,1}, \overrightarrow{W}_{1,2}, \dots, \overrightarrow{W}_{1,\tilde{v}}, \overrightarrow{W}_{2,1}, \dots, \overrightarrow{W}_{2,\tilde{v}}, \dots, \overrightarrow{W}_{\tilde{u},\tilde{v}}). \tag{2}$$

The size of the matrix $\mathbf{X}$ is $p \times q$ with $p = uv$, $q = \tilde{u}\tilde{v} = (h-u+1)(w-v+1)$. This procedure of formation of the trajectory matrix $\mathbf{X}$ is illustrated below for an image I of size $h \times w$ and a window of size 3×3: C

$$\begin{pmatrix} \ulcorner I_{1,1} & I_{1,2} & I_{1,3}\urcorner & I_{1,4} & \dots \\ I_{2,1} & I_{2,2} & I_{2,3} & I_{2,4} & \dots \\ \llcorner I_{3,1} & I_{3,2} & I_{3,3}\lrcorner & I_{3,4} & \dots \\ I_{4,1} & I_{4,2} & I_{4,3} & I_{4,4} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \to$$

$$\begin{pmatrix} I_{1,1} & \ulcorner I_{1,2} & I_{1,3} & I_{1,4}\urcorner & \dots \\ I_{2,1} & I_{2,2} & I_{2,3} & I_{2,4} & \dots \\ I_{3,1} & \llcorner I_{3,2} & I_{3,3} & I_{3,4}\lrcorner & \dots \\ I_{4,1} & I_{4,2} & I_{4,3} & I_{4,4} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \to \dots \to$$

$$\begin{pmatrix} \dots & I_{1,w-3} & \ulcorner I_{1,w-2} & I_{1,w-1} & I_{1,w}\urcorner \\ \dots & I_{2,w-3} & I_{2,w-2} & I_{2,w-1} & I_{2,w} \\ \dots & I_{3,w-3} & \llcorner I_{3,w-2} & I_{3,w-1} & I_{3,w}\lrcorner \\ \dots & I_{4,w-3} & I_{4,w-2} & I_{4,w-1} & I_{4,w} \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \to \dots \to$$

$$\begin{pmatrix} \dots & \dots & \dots & \dots & \dots \\ \dots & I_{h-3,w-3} & I_{h-3,w-2} & I_{h-3,w-1} & I_{h-3,w} \\ \dots & I_{h-2,w-3} & \ulcorner I_{h-2,w-2} & I_{h-2,w-1} & I_{h-2,w}\urcorner \\ \dots & I_{h-1,w-3} & I_{h-1,w-2} & I_{h-1,w-1} & I_{h-1,w} \\ \dots & I_{h,w-3} & \llcorner\ I_{h,w-2} & I_{h,w-1} & I_{h,w}\ \lrcorner \end{pmatrix}.$$

which gives

$$\mathbf{X} = \begin{pmatrix} I_{1,1} & I_{1,2} & \dots & I_{1,w-2} & \dots & I_{h-2,w-2} \\ I_{1,2} & I_{1,3} & \dots & I_{1,w-1} & \dots & I_{h-2,w-1} \\ I_{1,3} & I_{1,4} & \dots & I_{1,w} & \dots & I_{h-2,w} \\ I_{2,1} & I_{2,2} & \dots & I_{2,w-2} & \dots & I_{h-1,w-2} \\ I_{2,2} & I_{2,3} & \dots & I_{2,w-1} & \dots & I_{h-1,w-1} \\ I_{2,3} & I_{2,4} & \dots & I_{2,w} & \dots & I_{h-1,w} \\ I_{3,1} & I_{3,2} & \dots & I_{3,w-2} & \dots & I_{h,w-2} \\ I_{3,2} & I_{3,3} & \dots & I_{3,w-1} & \dots & I_{h,w-1} \\ I_{3,3} & I_{3,4} & \dots & I_{3,w} & \dots & I_{h,w} \end{pmatrix}$$

Note that the same element $I_{i,j}$ of the matrix (1) appears several times (up to p times) in the trajectory matrix $\mathbf{X}$ so that $\mathbf{X}$ has a particular structure; this structure is known as Hankel–block–Hankel [9]. This structure is difficult to describe analytically but easy algorithmically. Note also that there is a one-to-one correspondence between the Hankel–block–Hankel matrices of size $p \times q$ and the images of the form (1).

2.2 Formal description of the algorithm

Let us have an image of size $h \times w$ represented in the form of a matrix (1). We shall be constructing the trajectory matrix $\mathbf{X}$ as described above. Basic SSA operations will then be carried out with the matrix $\mathbf{X}$ as the trajectory matrix. In doing so we will assume that the parameters (u and v) of the window are chosen so that $p \le q$, and therefore the size $p \times p$ of the matrix $\mathbf{X}\mathbf{X}^T$ is smaller than or equal to the size $q \times q$ of the matrix $\mathbf{X}^T\mathbf{X}$.

Algorithm 1 (*Basic SSA for image processing*).

1. (*Computing the trajectory matrix*): define the window of size $u \times v$ (with $1 \le u \le h$, $1 \le v \le w$) and using the procedure described above and summarized in the formula (2), construct the trajectory matrix $\mathbf{X} = (x_{ij})_{i,j=1}^{p,q}$ where $p = uv$, $q = (h-u+1)(w-v+1)$, $p \le q$.

2. (*Constructing a matrix for applying SVD*): compute $\mathbf{X}\mathbf{X}^T$.

3. (*SVD of the matrix* $\mathbf{X}\mathbf{X}^T$): perform the SVD of $\mathbf{X}\mathbf{X}^T$; that is, represent $\mathbf{X}\mathbf{X}^T$ in the form $\mathbf{X}\mathbf{X}^T = \sum_{i=1}^{p} \lambda_i P_i P_i^T$, where $\lambda_1 \ge \dots \ge \lambda_p \ge 0$ are the eigenvalues of $\mathbf{X}\mathbf{X}^T$ and P_i are the normalized eigenvectors corresponding to λ_i ($i = 1, \dots, p$).

4. (*Selection of eigenvectors*): select a group of l ($1 \le l \le p$) eigenvectors $P_{i_1}, P_{i_2}, \dots, P_{i_l}$.

5. (*Reconstruction of the image*): compute the matrix $\tilde{\mathbf{X}} = \sum_{k=1}^{l} P_{i_k} P_{i_k}^T \mathbf{X}$ as an approximation to $\mathbf{X}$. Transition to an image of size $h \times w$ (represented as a matrix $\tilde{I}$ of size $h \times w$) can now be achieved by averaging over the corresponding entries of the matrix $\tilde{\mathbf{X}}$.

Note that this algorithm has been suggested in [3]. It was further discussed in [5–7] under the name 2D-SSA.

Similarly to the Basic SSA, we can suggest different modifications of Algorithm 1 such as SSA with single or double centering (see [4], Sect. 1.7). We have made a numerical comparative study of different versions of SSA for image processing. In most examples we considered, the difference between many versions of SSA was not significant; we hence do not consider these modifications in the present paper.

Note that Algorithm 1 does not present significant computational difficulties in comparison with the Basic SSA; chiefly the selection of the window size (u, v) determines the computational cost. Indeed, the most computationally difficult operation is the SVD of the matrix $\mathbf{X}\mathbf{X}^T$. The difficulty of this operation depends on the size of the matrix $\mathbf{X}\mathbf{X}^T$ which for the image processing SSA is equal to $p \times p$ with $p = uv$. Note that on a standard laptop, MATLAB routinely performs the SVD for a matrix of a size up to 1,000×1,000; see also [8] for a description of efficient techniques that could be used to increase the efficiency of the procedures of computation of the SVD for the matrices appearing in SSA.

2.3 SSA for analyzing colour images

If a colour image in M colours (e.g., $M = 3$) of size $h \times w$ is given, then we have M matrices $I^{(fi)}$ ($i = 1, \dots, M$) of size $h \times w$ rather than just one matrix I defined in (1). To perform an SSA reconstruction of the whole image, we can perform M parallel reconstructions of the image in i-th colour ($i = 1, \dots, M$) by applying M times Algorithm 1 to the matrices $I^{(i)}$ ($i = 1, \dots, M$). In doing this, we can naturally assume that the parameters of Algorithm 1 (which are the window size $u \times v$ and the indices of the eigenvectors chosen for the reconstruction) are fixed.

However, we typically would get a better result if we perform a multivariate SSA procedure. To do this, we need to

Figure 1. 'Lena' image.

compute the trajectory matrix $\mathbf{X}$ according to

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}^{(1)} \\ \cdots \\ \mathbf{X}^{(M)} \end{pmatrix}, \tag{3}$$

where the individual trajectory matrices $\mathbf{X}^{(i)}$ are computed by (2). The SSA reconstruction of the trajectory matrix $\mathbf{X}$ given in the form (3) will then correspond to a simultaneous reconstruction of the image. We expect to observe an advantage of performing the simultaneous reconstruction of M colour images over M parallel reconstructions, if there is a dependence between these M images.

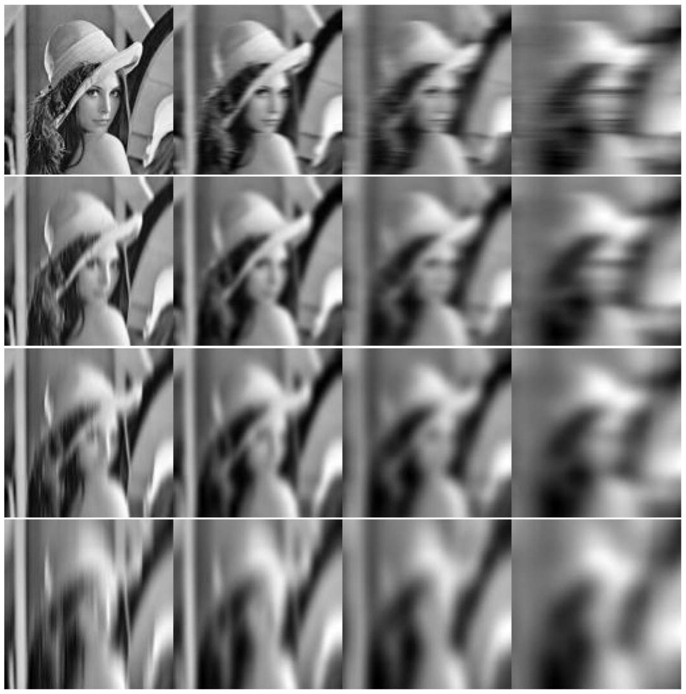

Figure 2. SSA reconstruction of the 'Lena' image using only the first eigenvector for different window sizes $u \times v$: $u = 1, 5, 10, 20$ (from top to bottom), $v = 1, 5, 10, 20$ (from left to right).

3. APPLICATION TO THE 'LENA' IMAGE

3.1 'Lena' image with no noise

Assume that we are given the standard 'Lena' image of size 128×128 pixels in a grey scale (coded with numbers from 0 to 255), see Fig. 1. Assume that the image is given in the form of a matrix I as in (1), with $h = w = 128$.

In this section, we apply Algorithm 1 for reconstructing this 'Lena' image in the case when there is no noise. Each reconstruction (that is, application of Algorithm 1) gives us a matrix $\tilde{I}$ of size 128×128. Typically, the reconstructed images $\tilde{I} = ||\tilde{I}_{ij}||_{i,j=1}^{128}$ have values with a range different from $[0, 255]$. For the purpose of the graphical representation of these reconstructed images, they are always rescaled so that $\min_{i,j} \tilde{I}_{ij} \to 0$ and $\max_{i,j} \tilde{I}_{ij} \to 255$; these images are then painted using the standard grayscale colourmap.

The 'Lena' image has been processed with different window sizes $u \times v$. Fig. 2 shows the results of the SSA reconstruction of the image using only the first eigenvector for $u, v \in \{1, 5, 10, 20\}$. The top row of figures corresponds to the value $u = 1$; the second, third and fourth rows correspond to the values $u = 5, 10, 20$ respectively. Similarly, the columns from left to right correspond to the values $1, 5, 10, 20$ for v.

Assume that we choose the first $l \leq p = uv$ eigenvectors for the reconstruction. If we use all $p = uv$ eigenvectors (that is, if $l = p$), then the image is reconstructed perfectly; that is why the reconstructed image in the top left corner of Fig. 2 coincides with the original image. For fixed l and smaller values of p, the reconstruction is better; for larger p the reconstructions with small l are smoothed versions of the image. If $u \neq v$ then this smoothing is non-symmetric; for example, if $u > v$ then the reconstructed images in Fig. 2 are smoother in the vertical direction than in the horizontal one.

To see the influence of the individual components, in Fig. 3 we plot the reconstruction of the image using one eigenvector only; here we use the window size 10×10 (that is, $u = v = 10$). Specifically, at Step 4 of Algorithm 1 we have $l = 1$, $P_{i_1} = k$, $k = 1, \ldots, 12$; that is, only the k-th eigenvector has been used for the reconstruction in the k-th image ($k = 1, \ldots, 12$). Fig. 3 shows that each of the first twelve eigenvectors reflects some features of the original image.

Fig. 4 illustrates the relative importance of the components of the SVD decomposition in Step 3 of Algorithm 1; this importance is expressed in terms of the magnitude of the respective eigenvalues. The eigenvalues from 1 to 10 are plotted in the first graph and the eigenvalues from 2 to 10 are plotted in the second. As the first eigenvalue is almost 30 times larger than the second eigenvalue, we present the second plot where the first eigenvalue is omitted.

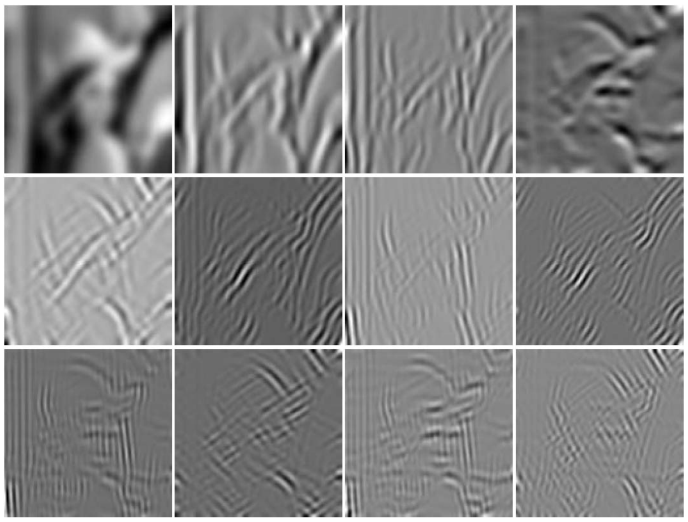

Figure 3. *SSA reconstruction of the 'Lena' image using the k-th eigenvector only $(k = 1, \ldots, 12)$; from left to right and from top to bottom.*

3.2 Denoising the 'Lena' image

In this section, we demonstrate the results of application of Algorithm 1 for denoising the 'Lena' image corrupted with different kinds of noise.

The uncorrupted 'Lena' image ('Lena') is shown as the left image of Fig. 5, 'Lena' image corrupted with salt and pepper noise of different intensity (images 'L1', 'L2', 'L3') is shown in the first column of Fig. 6; Fig. 7 and Fig. 8 show the 'Lena' image corrupted with Gaussian white noise of different intensity (images 'L4' and 'L5') and distorted 'Lena' images ('L6' and 'L7'). We also show the SSA-based reconstruction of these images using the first l eigenvectors $(1 \leq l \leq 10)$ for the window size 10×10. To do the reconstruction, we used the first eigenvector only (the second column in Figs. 5, 6, 7 and 8), the first three eigenvectors (the third column), and the first ten eigenvectors (the fourth column).

To measure the distance between the original image and the reconstructions of the corrupted images, let us assume that we have two images $I^{(1)} = \|I^{(1)}_{i,j}\|$, $I^{(2)} = \|I^{(2)}_{i,j}\|$ of the same size $h \times w$. To measure the dissimilarity between the two images we shall use the normalized squared Frobenius distance

$$(4) \qquad d(I^{(1)}, I^{(2)}) = \left[\frac{1}{hw} \sum_{i,j=1}^{h,w} \left| I^{(1)}_{i,j} - I^{(2)}_{i,j} \right|^2 \right],$$

which is the distance used most commonly. For more sophisticated distances between images, see Sec. 4 below.

In order to measure the denoising efficiency of the SSA decomposition, the distances between the original 'Lena' image, Fig. 1, and the reconstructions of the corrupted images, Figs. 5, 6, 7 and 8, have been computed. In Figs. 9, 10 and

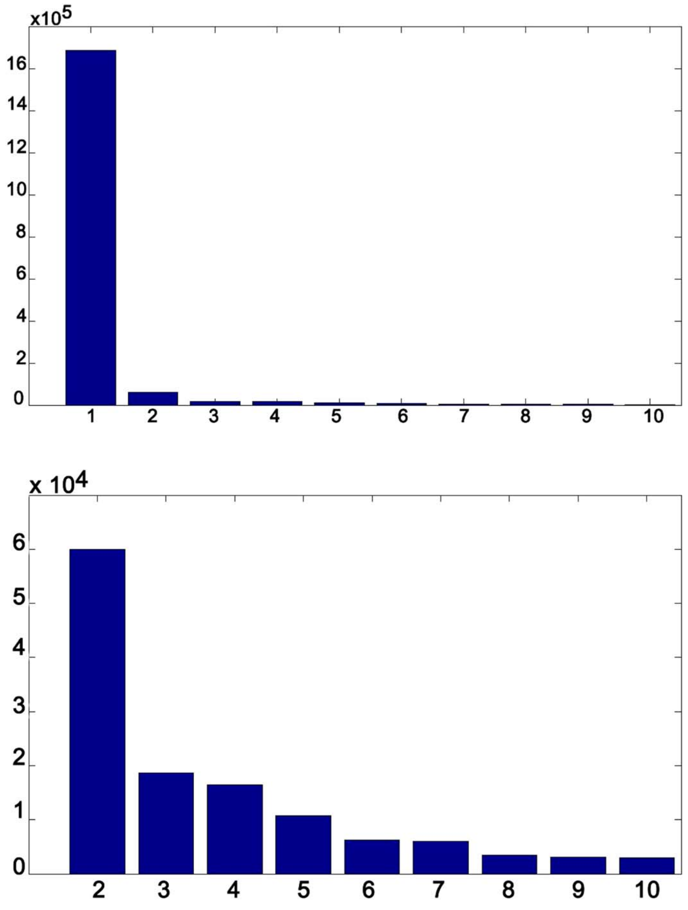

Figure 4. *Top: The first ten eigenvalues in the SSA decomposition of the 'Lena' image with a window of size $u = v = 10$. Bottom: the eigenvalues 2–10.*

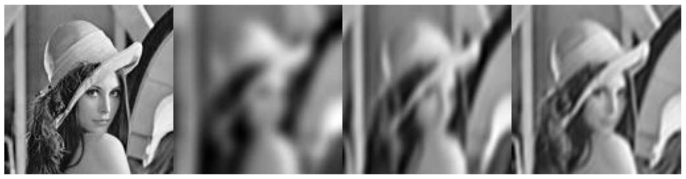

Figure 5. *Reconstruction of the 'Lena' image with the window size $u = v = 10$ for a different number of eigenvectors (all, the first only, the first three, the first ten).*

11 we plot the distances between the original 'Lena' image and its reconstructions.

In Figs. 9, 10 and 11 the circles show the distances between the original 'Lena' image and its reconstructions using the first l eigenvectors $(l = 1, 2, \ldots, 10)$. The crosses in Figs. 9, 10 and 11 represent the normalized squared Frobenius distances between the original 'Lena' image and the reconstructions of the corresponding noisy image. The horizontal line represents the normalized squared Frobenius distance between the original 'Lena' image and the original corrupted image.

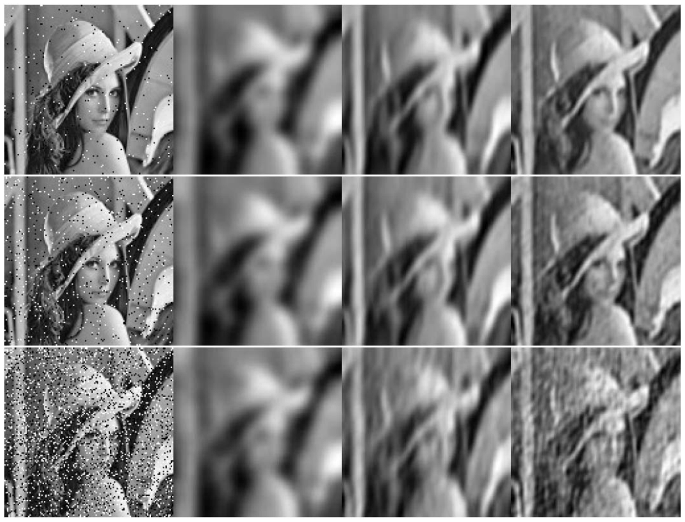

Figure 6. Denoising the 'Lena' image corrupted with salt and pepper noise of different intensity, 'L1', 'L2' and 'L3'.

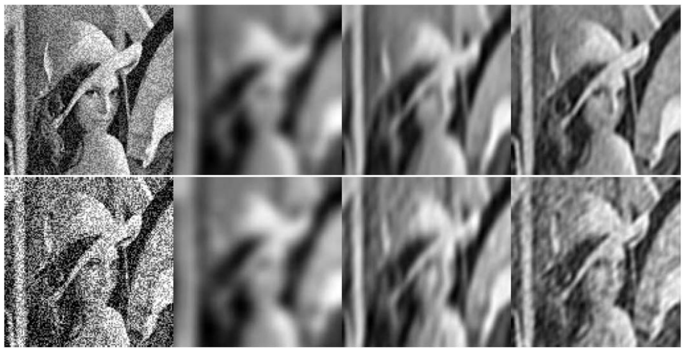

Figure 7. Denoising the 'Lena' image corrupted with Gaussian white noise of different intensity, 'L4' and 'L5'.

The numerical values of the squared normalized Frobenius distances plotted in Figs. 9, 10 and 11, are given in Table 1. The first row gives the distances between the original 'Lena' image and all test (noisy) images. In the following rows we give the distances between the 'Lena' image and the respective reconstructed images. The reconstructions have been done using the first l eigenvectors ($l = 1, 2, \ldots, 10$) and the size of the window is 10×10.

From the Figs. 9, 10 and 11 and Table 1, we can observe that the original 'Lena' image is much closer to some of the reconstructed images than to the noisy image taken for the analysis. This can be considered as an evidence of good performance of SSA as a tool for image denoising.

In Fig. 12 the normalized squared Frobenius distance from the original 'Lena' image to the reconstructions using the first l eigenvectors of images 'L1' and 'L2', is plotted, where we use all possible values for l: $l = 1, 2, \ldots, 100 = uv$. The reference distances between the original 'Lena' image

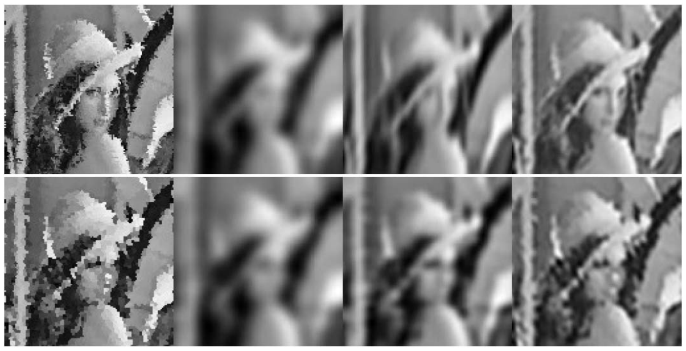

Figure 8. Denoising the distorted 'Lena' images, 'L6' and 'L7'.

Table 1. Distances between the original 'Lena' image and the reconstructions of noisy images. In the first column the distance between the original 'Lena' image and their reconstructions is given as reference. In the first row distance between the original 'Lena' image and each of the 7 test images is presented

l	Lena	L1	L2	L3	L4	L5	L6	L7
0	0	350	1034	2778	798	2927	671	1046
1	1149	1162	1192	1292	1179	1253	1194	1228
2	676	684	712	813	701	786	741	799
3	533	543	573	684	561	659	615	740
4	442	455	488	603	473	582	537	653
5	367	382	419	541	403	527	473	620
6	313	329	375	499	351	496	435	602
7	265	286	328	473	311	467	399	619
8	237	259	305	464	286	460	391	620
9	212	237	288	462	267	460	379	620
10	189	218	274	460	251	460	364	630

and its reconstructions reduce to zero as the number of eigenvalues approaches uv. For noisy images, the distances initially decrease and afterwards, as the reconstructions get closer to the original noisy image, approach the reference line. This reference line represents the normalized squared Frobenius distance between the original 'Lena' image and the related noisy image.

4. SSA-BASED DISTANCES BETWEEN IMAGES

4.1 Defining the SSA-based distances

Assume that we have two images $I^{(1)} = \|I^{(1)}_{i,j}\|$, $I^{(2)} = \|I^{(2)}_{i,j}\|$ of the same size $h \times w$. To measure the dissimilarity between the two images it is customary to use the L_2-based distance (4).

In this section we introduce a family of distances based on the trajectory matrices of the images and their SVD expansions. Our experience shows that these SSA-based distances

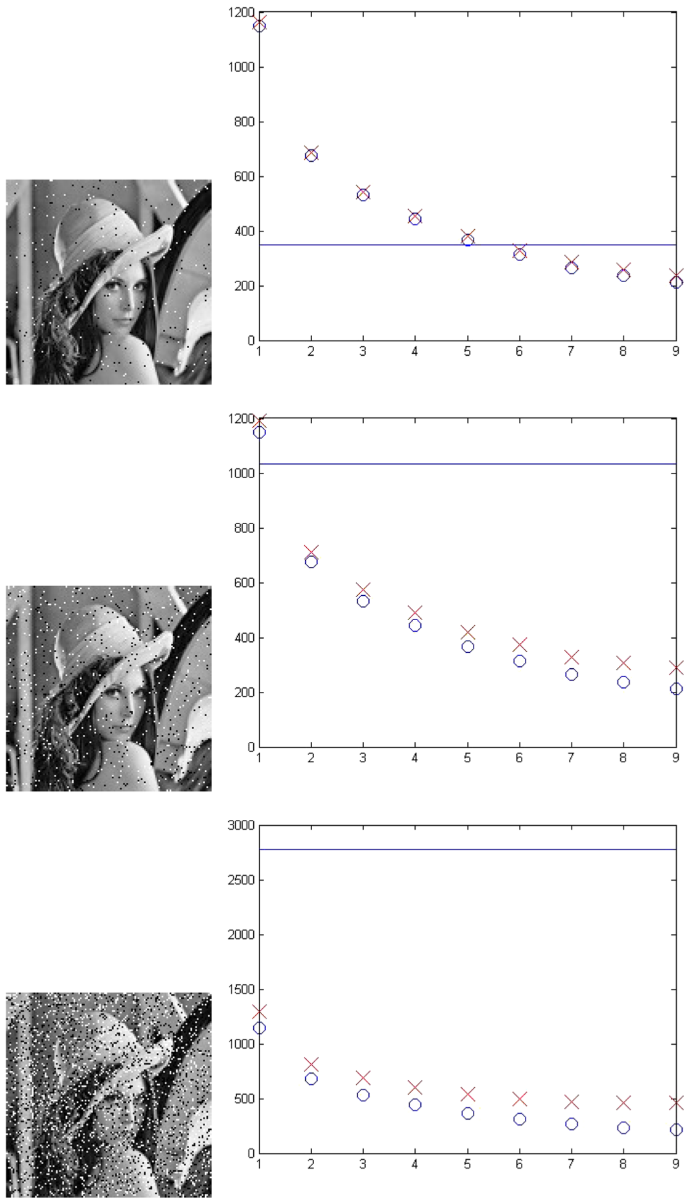

Figure 9. The evolution of the squared Frobenius distance for the reconstruction of 'Lena' image corrupted with salt and pepper noise of different intensity (images L1,L2,L3).

reflect the dissimilarity between two images much more naturally than the distance (4).

Let us fix the size $u \times v$ of the moving window and let $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$ be two trajectory matrices of size $p \times q$ associated with $I^{(1)}$ and $I^{(2)}$. First, we normalize the trajectory matrices as follows:

$$Y_1 = \mathbf{X}^{(1)} / \sqrt{\operatorname{tr}(\mathbf{X}^{(1)}(\mathbf{X}^{(1)})^T)}\,,$$

$$Y_2 = \mathbf{X}^{(2)} / \sqrt{\operatorname{tr}(\mathbf{X}^{(2)}(\mathbf{X}^{(2)})^T)}$$

Then we make the SVD of the matrices $Y_1 Y_1^T$ and $Y_2 Y_2^T$.

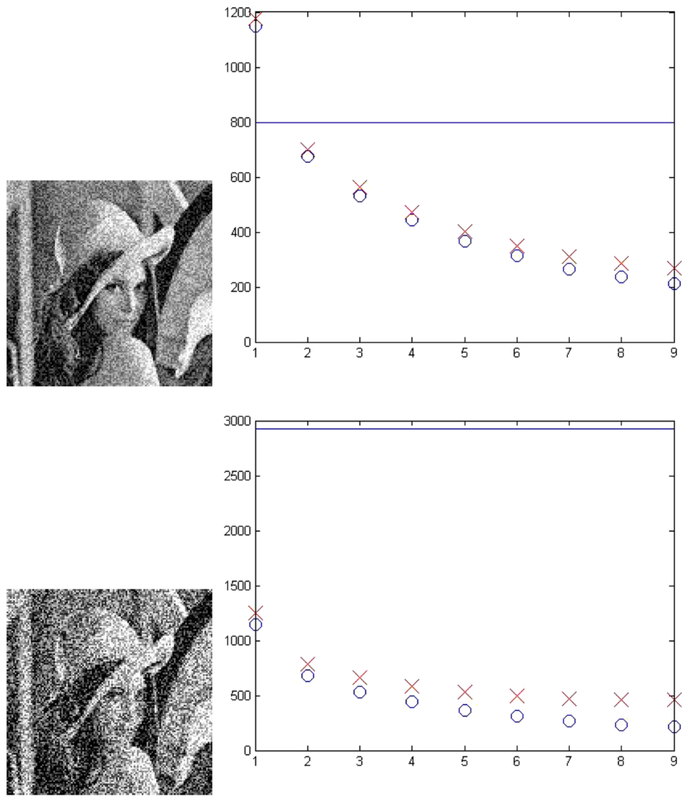

Figure 10. The evolution of the squared Frobenius distance for the reconstruction of 'Lena' image corrupted with Gaussian noise of different intensity (images L4, L5).

Let $\lambda_1 \geq \cdots \geq \lambda_p$ and $\mu_1 \geq \cdots \geq \mu_p$ be the corresponding eigenvalues. As the matrices $Y_1 Y_1^T$ and $Y_2 Y_2^T$ are non-negative definite and $\operatorname{tr}(Y_1 Y_1^T) = \operatorname{tr}(Y_2 Y_2^T) = 1$, we have $\lambda_i \geq 0$, $\mu_i \geq 0$ for all i and $\sum_{i=1}^{p} \lambda_i = \sum_{i=1}^{p} \mu_i = 1$.

Let us now analyze the two images $I^{(1)}$ and $I^{(2)}$ simultaneously. To do this, we create a joint (normalized) trajectory matrix $Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}$ so that

$$(5) \qquad YY^T = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} \begin{pmatrix} Y_1 & Y_2 \end{pmatrix} = \begin{pmatrix} Y_1 Y_1^T & Y_1 Y_2^T \\ Y_2 Y_1^T & Y_2 Y_2^T \end{pmatrix}.$$

Denote the eigenvalues of the $(2p \times 2p)$-matrix YY^T by $\nu_1 \geq \cdots \geq \nu_{2p} \geq 0$. Since $\operatorname{tr}(YY^T) = \operatorname{tr}(Y_1 Y_1^T) + \operatorname{tr}(Y_2 Y_2^T) = 2$, we have $\sum_{i=1}^{2p} \nu_i = 2$. If it is needed, from the SVD of the matrix YY^T we can perform the reconstruction of both images, $I^{(1)}$ and $I^{(2)}$, using Algorithm 1.

Consider two extreme cases:
(a) $Y_1 = Y_2$ (proportional images), and
(b) $Y_1 Y_2^T = 0$ (orthogonal images).

In the case (a) we have $\mu_i = \lambda_i$ $(i = 1, \ldots, p)$, $YY^T = \begin{pmatrix} Y_1 Y_1^T & Y_1 Y_1^T \\ Y_1 Y_1^T & Y_1 Y_1^T \end{pmatrix}$ and therefore $\nu_i = 2\lambda_i$ for $i = 1, \ldots, p$ and $\nu_i = 0$ for $i = p+1, \ldots, 2p$.

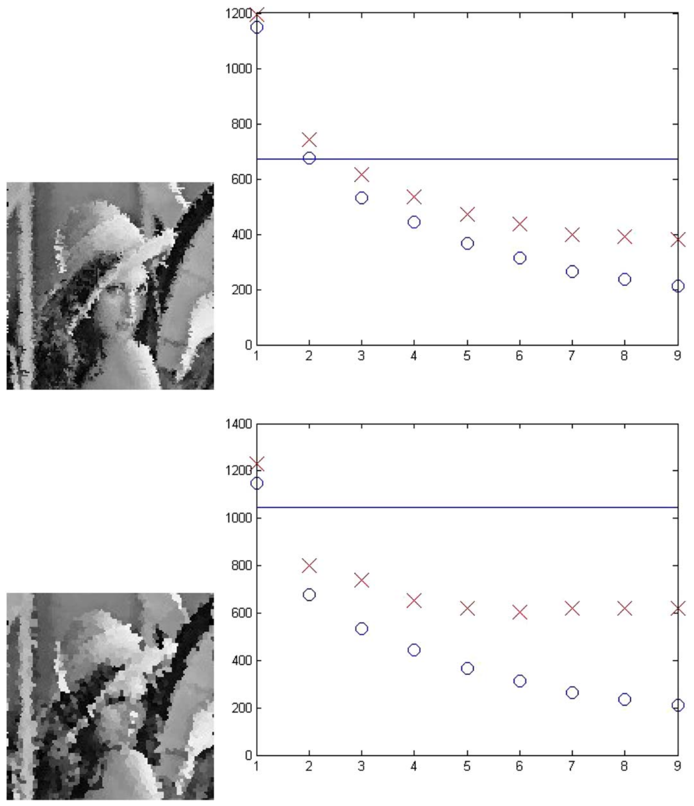

Figure 11. The evolution of the squared Frobenius distance for the reconstruction of distorted 'Lena' images (images L6, L7).

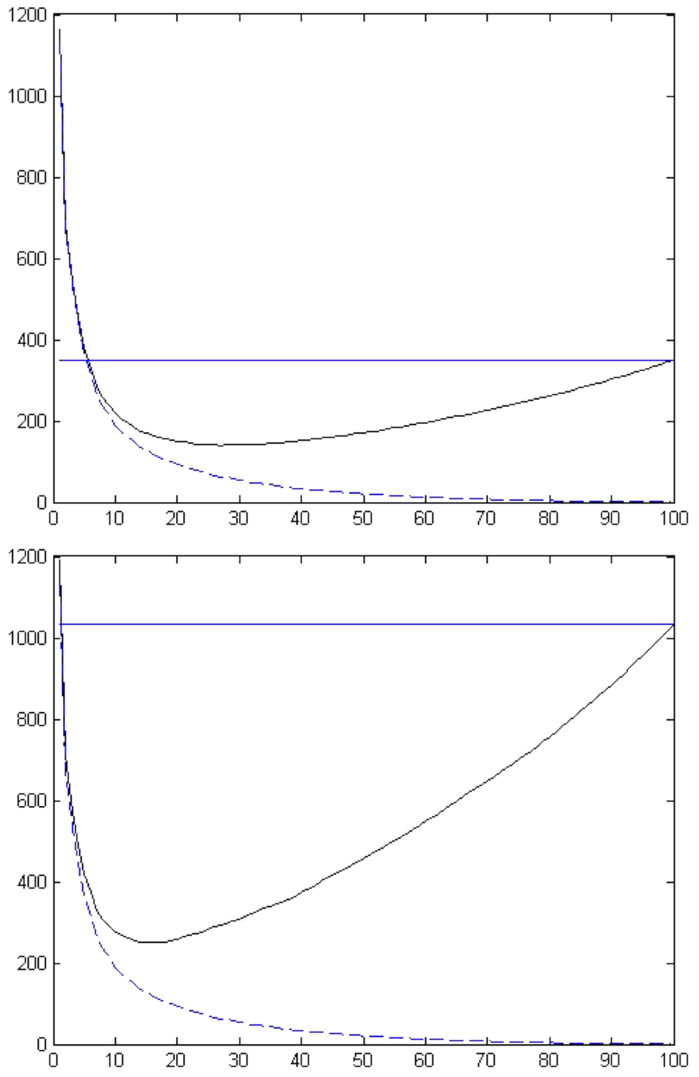

Figure 12. The evolution of the normalized squared Frobenius distance for the reconstruction of distorted 'Lena' images computed for $l = 1, 2, \ldots, 100 = uv$. Top: Distances for 'L1'. Bottom: Distances for 'L2'. Dashed line is the distance between the original 'Lena' and its reconstructions, to be used as reference.

In the case (b), the set of eigenvalues $\{\nu_i\}_{i=1}^{2p}$ is a union of the sets $\{\lambda_i\}_{i=1}^{p}$ and $\{\mu_i\}_{i=1}^{p}$.

A very useful result of [11] implies that for any matrix of the form (5) (where the matrices Y_1 and Y_2 are arbitrary) and any positive integer k we have

$$\sum_{j=1}^{k} \lambda_j + \sum_{j=1}^{k} \mu_j \geq \sum_{j=1}^{k} \nu_j \tag{6}$$

(here we assume that $\lambda_j = 0$ if j is larger than the rank of the matrix Y_1; similarly for the eigenvalues μ_j and ν_j).

Define

$$F_1(t) = \sum_{j=1}^{[t]} \lambda_j \ \text{ and } \ F_2(t) = \sum_{j=1}^{[t]} \mu_j$$

so that $F_1(t)$ and $F_2(t)$ can be interpreted as c.d.f.'s defined on the integers $\{1, \ldots, p\}$. Similarly, we define $F(t) = \frac{1}{2}\sum_{j=1}^{[t]} \nu_j$ so that the c.d.f. $F(t)$ defines a discrete probabilistic measure on $\{1, \ldots, 2p\}$.

In our notation, the inequality (6) means that $F_1(t) + F_2(t) - 2F(t) \geq 0$ for all $t \geq 0$.

Therefore, the rationale for defining the distances between two images $I^{(1)}$ and $I^{(2)}$ is as follows: if the difference $G(t) = F_1(t) + F_2(t) - 2F(t)$ is small, then the images I_1 and I_2 are similar; if, on the contrary, the difference $G(t)$ is large then the images are different. Indeed, in the case (a) $G(t) = 0$ for all t and in the case (b) $G(t) > 0$ for all t, $1 \leq t < rank(Y_1) + rank(Y_2)$.

For large t, the behaviour of the c.d.f.'s $F_1(t)$, $F_2(t)$ and $F(t)$ resembles the c.d.f. of eigenvalues of random matrices and therefore the dissimilarity between images better manifests itself at the values of $G(t)$ for relatively small t (the range of values of t should depend on the size of the image and the size of the window). Therefore, a natural definition of the distance is

$$d_1(I^{(1)}, I^{(2)}) = \int_0^k G(t)dt = \sum_{j=1}^{k}(\lambda_j + \mu_j - \nu_j) \tag{7}$$

where k is small compared to p. Another natural distance is

$$d_\infty(I^{(1)}, I^{(2)}) = \max_{0 \le t < \infty} G(t) = \max_{j=1,\dots,p} (\lambda_j + \mu_j - \nu_j)\,. \tag{8}$$

We have made extensive numerical experiments with the SSA-based distances (7) and (8); these experiments have shown that these distances reflect the similarity (and dissimilarity) between images much finer than the standard distance (4) and the distances between the subspaces created by the SVD decompositions of the trajectory matrices of the two images taken separately.

4.2 Application to face verification

The technique described above can be applied to solving the face verification problem which is the problem of making the decision of whether a given image belongs to the subject that claims a certain identity. Such problems are usually simulated with databases of images such that different sets of images have been taken for each individual.

Usually feature extraction methods, including multivariate dimension reduction methods, are used to extract discriminating patterns. Afterwards classifiers are trained and tested through an exhaustive cross-validation of the whole database. Examples of these strategies can be found for example in [10] for the FRAV2D database of colour images and in [2] for the FRAV3D database of 3-dimensional laser scans.

The advantages of applying SSA to the verification problems is the possibility of applying the distances between images, such as (7), (8), to directly compare reference images and possible candidates, instead of comparing features through a learning algorithm. First results of application of the SSA-based distances to face verification problems are very encouraging: the success rate for the SSA-based face verification techniques is comparable to the best results obtained by the application of the support vector machines (SVM). An advantage of SSA over SVM is the fact that all the manipulations with data in SSA are transparent and controllable; this is not true in the case of SVM. On the other hand, the use of the SSA-based distances require higher computational costs for large-scale problems than that for the SVM. The authors intend to accomplish a comprehensive comparative study of SSA and SVM techniques in face recognition problems and provide a thorough discussion of advantages and disadvantages of both approaches in a separate publication.

Received 31 March 2010

REFERENCES

[1] Buades A., Coll B., and Morel J.M. (2005). A review of image denoising algorithms, with a new one, *Multiscale Model. Simul.*, **2**, 490–530. MR2162865

[2] Conde C., Serrano A., Rodríguez-Aragón L.J., and Cabello E. (2006). An automatic 2D, 2.5D and 3D score-based fusion face verification system. In Proceedings of the IEEE Computer Architecture for Machine Perception and Sensing, 207–212.

[3] Danilov, D., and Zhigljavsky, A.A., eds (1997) *Principal Components of Time Series: "Caterpillar" Method,* St. Petersburg University (in Russian).

[4] Golyandina N.E., Nekrutkin V.V., and Zhigljavsky, A. (2001). *Analysis of Structure of Time Series: SSA and Related Techniques*, CRC / Chapman & Hall, Boca Raton. MR1823012

[5] Golyandina N.E., Usevich K.D., and Florinsky I.V. (2007). Filtering of digital terrain models by two-dimensional singular spectrum analysis. *International Journal of Ecology and Development*, **8**, 81–94.

[6] Golyandina N.E., and Usevich K.D. (2009). An algebraic view on finite rank in 2D-SSA. In Proceedings of the 6th St. Petersburg Workshop on Simulation. Vol 1, 308–313.

[7] Golyandina N.E., and Usevich K.D. (2010). 2D-extension of singular spectrum analysis: algorithm and elements of theory. In: Matrix Methods: *Theory, Algorithms, Applications* World Scientific, 449–473.

[8] Korobeynikov A. (2010) Computation- and space-efficient implementation of SSA. *Statistics and Its Interface*, **3**, 357–368.

[9] Markovsky I., Willems J.C., Van Huffel S., and De Moor, B. (2006). *Exact and approximate modeling of linear systems: A behavioral approach,* Mathematical Modeling and Computation, SIAM. MR2207544

[10] Rodríguez-Aragón L.J., Conde C., Serrano A., and Cabello E.(2006). Comparing and combining spatial dimension reduction methods in face verification. *Lecture Notes in Computer Science*, **4224**, 645–653.

[11] Thompson, R.C., and Therianos, S. (1972). The eigenvalues of complementary principal submatrices of a positive definite matrix, *Canadian J. Math.*, **24**, 658–667. MR0294375

[12] Zhigljavsky, A. (2010) Singular Spectrum Analysis for time series: Introduction to this special issue. *Statistics and Its Interface*, **3**, 255–258.

Licesio J. Rodríguez-Aragón
Department of Mathematics, University of Castilla-La Mancha
13071 - Ciudad Real, Spain
E-mail address: L.RodriguezAragon@uclm.es

Anatoly Zhigljavsky
School of Mathematics, Cardiff University
Cardiff CF2 4YH, UK
E-mail address: ZhigljavskyAA@cf.ac.uk