Applied Mathematics and Computation 374 (2020) 125046

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A comparison between the sampling Kantorovich algorithm for digital image processing with some interpolation and quasi-interpolation methods[☆]

Danilo Costarelli[1,*], Marco Seracini, Gianluca Vinti

Department of Mathematics and Computer Science, University of Perugia, 1, Via Vanvitelli, Perugia 06123, Italy

ARTICLE INFO

Article history:
Received 8 August 2019
Revised 8 November 2019
Accepted 5 January 2020

MSC:
65D18
65D05
65D07
41A05
41A25

Keywords:
Sampling Kantorovich
Interpolation
Quasi-interpolation
Image processing
PSNR
CPU time

ABSTRACT

In this paper we study the performance of the sampling Kantorovich (S–K) algorithm for image processing with other well-known interpolation and quasi-interpolation methods. The S-K algorithm has been implemented with three different families of kernels: central B-splines, Jackson type and Bochner–Riesz. The above method is compared, in term of PSNR (Peak Signal-to-Noise Ratio) and CPU time, with the bilinear and bicubic interpolation, the quasi FIR (Finite Impulse Response) and quasi IIR (Infinite Impulse Response) approximation. Experimental results show better performance of S-K algorithm than the considered other ones.

1. Introduction

The rescaling of an image is a widely studied problem in Digital Image Processing (D.I.P.). Typical methods developed to perform the above task are based on mathematical interpolation, see, e.g., [10,36]. For instance, bilinear and bicubic interpolation are among the most used interpolation methods for image rescaling, see e.g., [9,32].

The above methods are quite easy to implement and need of a small CPU time. On the other side, they provide not optimal results in terms of quality of the reconstruction, measured by the so-called PSNR (Peak Signal to Noise Ratio).

To overcome this limit, recently quasi-interpolation methods have been successfully used. From the theoretical point of view, the better performance of the latter approximation methods than the interpolation ones, has been proved providing estimates concerning the order of approximation, see e.g., [7]. For instance, quasi Finite Impulse Response (quasi

☆ *In memory of Prof. Domenico Candeloro, the Mentor, the Man and the Friend, with deep gratitude.*

* Corresponding author.

E-mail addresses: danilo.costarelli@unipg.it (D. Costarelli), marco.seracini@dmi.unipg.it (M. Seracini), gianluca.vinti@unipg.it (G. Vinti).

[1] According to the submission guideline of the journal, we certify that the general content of the manuscript, in whole or in part, is not submitted, accepted, or published elsewhere.

https://doi.org/10.1016/j.amc.2020.125046

FIR) and Infinite Impulse Response (quasi IIR) have been reviewed to face the rescaling problem. Numerical results confirm the theoretical ones, e.g., in case of non trivial multiple image rotation (see [13] again).

Concerning the quasi-interpolation methods for D.I.P., the so-called sampling Kantorovich (S-K) algorithm has been recently introduced (see [19]). The S–K algorithm is based on the theory of the sampling Kantorovich series S_w, $w > 0$, which are approximation operators particularly suitable for digital image reconstruction, in view of their mathematical expression, see e.g., [17,19].

More in detail, the advantage of the S-K operators resides in the use of the mean sample values calculated in some neighborhoods of $\frac{k}{w}$, differently from the usual employed approximation operators based on the pointwise sample values $f(\frac{k}{w})$. In fact, the S-K operators are totally independent from the pointwise behavior of the function being reconstructed and this makes them suitable in order to reconstruct not necessarily continuous signals such as images. The w parameter of the S-K operator determines the width of the neighborhood over which the mean values are computed and, at the same time, it is connected with the order of approximation. Bigger the value of w, better the quality of the reconstruction, independently from the image ratio and the image size. Once the signal has been reconstructed it is possible to chose whatever sampling frequency and zoom factor to resample its original version, achieving a qualitative improvement.

The implementation of the S-K algorithm needs the use of suitable kernels which, from the mathematical point of view, are discrete approximate identities in the sense described in [8], such as the central B-spline, the Jackson type kernels, and the Bochner–Riesz kernels [20,37]. In the case of the implementation of the S-K algorithm based upon the Jackson type kernels, some meaningful numerical results have been achieved in the engineering field, concerning the study of thermal bridges and the behavior of buildings under seismic actions, by means of thermographic images [4,5,12].

The main purpose of the present paper is to evaluate the performance of the S-K algorithm in image rescaling, in term of PSNR and CPU time, in comparison with the above mentioned interpolation and quasi-interpolation methods. Our goal is to obtain an objective evaluation of the performance of the S-K algorithm for different kernel types, studying their behavior when varying the parameter w of the operators and the order N of each kernels.

Now, we give a plan of the paper. In Section 2, we briefly recall the definition of the sampling Kantorovich series together with some basic aspects and the list of the used kernel functions. In Section 3, we give the definition of the PSNR, while in Section 4 the above mentioned interpolation and quasi-interpolation methods are explicitly considered. In Section 5, numerical results are provided, while the main conclusions of the paper are summarized in Section 6.

2. The sampling Kantorovich algorithm for digital image processing

Recently, many applications to various applied fields related to image processing have been studied thanks to the crucial contribution of the so-called sampling Kantorovich (S-K) algorithm, see e.g., [4,5,12]. In particular, the above algorithm revealed to be crucial and performing for the problems of image reconstruction, image enhancement and rescaling; its (optimized) implementation is based on a numerical version of the following formula:

$$(S_w f)(\underline{x}) \ := \ \sum_{\underline{k}\in\mathbb{Z}^n} \chi(w\underline{x}-\underline{k}) \left[w^n \int_{R_{\underline{k}}^w} f(\underline{u}) \ d\underline{u} \right], \qquad \underline{x}\in\mathbb{R}^n, \quad w>0, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a locally integrable function (signal/image) such that the above series is convergent for every $\underline{x}\in\mathbb{R}^n$, and

$$R_{\underline{k}}^w \ := \ \left[\frac{k_1}{w}, \frac{k_1+1}{w}\right] \times \left[\frac{k_2}{w}, \frac{k_2+1}{w}\right] \times \ldots \times \left[\frac{k_n}{w}, \frac{k_n+1}{w}\right],$$

are the sets in which we consider the mean values of the signal f.

The function $\chi : \mathbb{R}^n \to \mathbb{R}$ is called a *kernel* and it is chosen such that the following assumptions are satisfied:

$(\chi 1)$ χ belongs to $L^1(\mathbb{R}^n)$, and it is bounded in a ball containing the origin of $\mathbb{R}^n$;
$(\chi 2)$ for every $\underline{x}\in\mathbb{R}^n$, there holds:

$$\sum_{\underline{k}\in\mathbb{Z}^n} \chi(\underline{x}-\underline{k}) = 1;$$

$(\chi 3)$ for some $\beta > 0$, we assume that the discrete absolute moment of order β of χ is finite, i.e.,

$$m_\beta(\chi) \ := \ \sup_{\underline{u}\in\mathbb{R}} \sum_{\underline{k}\in\mathbb{Z}^n} |\chi(\underline{u}-\underline{k})| \cdot \|\underline{u}-\underline{k}\|_2^\beta \ < \ +\infty,$$

where $\| \cdot \|_2$ denotes the usual Euclidean norm of $\mathbb{R}^n$.

Assumptions $(\chi 1)$, $(\chi 2)$, and $(\chi 3)$ are typically satisfied by the discrete approximate identities, [8].

It is well-known that, S_w, $w > 0$, defined in (1), are called *sampling Kantorovich operators*, and they are approximation operators which are able to pointwise reconstruct continuous and bounded signals, and to uniformly reconstruct signals which are uniformly continuous and bounded, as $w \to +\infty$, [6,31,35]. Moreover, the operators S_w revealed to be suitable also to reconstruct not-necessarily continuous signals, e.g., in the L^p-sense, [17].

For further details about the optimized implementation of the S-K algorithm, see e.g., [5], where also a pseudo-code is available.

Now, we give a brief list of some well-known and important classes of kernels which satisfy the above assumptions $(\chi 1) - (\chi 3)$, and that can be used in order to implement (1).

First of all, we recall in Eqn 2 the definition of the one-dimensional central B-spline of order N (see e.g., [1,34]):

$$\beta^N(x) \;:=\; \frac{1}{(N-1)!}\sum_{i=0}^{N}(-1)^i\binom{N}{i}\left(\frac{N}{2}+x-i\right)_+^{N-1}, \qquad x\in\mathbb{R}. \tag{2}$$

The corresponding multivariate version of central B-spline of order N is given in Eqn 3:

$$\mathcal{B}_n^N(\underline{x}) \;:=\; \prod_{i=1}^{n}\beta^N(x_i), \qquad \underline{x}=(x_1,\ldots,x_n)\in\mathbb{R}^n. \tag{3}$$

Other important kernels are given by the so-called Jackson type kernels of order N *(see* Eqn 4*)*, defined in the univariate case by:

$$J_N(x) \;:=\; c_N\,\mathrm{sinc}^{2N}\left(\frac{x}{2N\pi\alpha}\right), \qquad x\in\mathbb{R}, \tag{4}$$

with $N\in\mathbb{N}$, $\alpha\ge 1$, and where c_N is a non-zero normalization coefficient, given by:

$$c_N \;:=\; \left[\int_{\mathbb{R}}\mathrm{sinc}^{2\,N}\left(\frac{u}{2\,N\pi\alpha}\right)du\right]^{-1}.$$

For the sake of completeness, we recall that the well-known *sinc*-function is defined as $\sin(\pi x)/\pi x$, if $x\neq 0$, and 1 if $x=0$, see e.g., [27–29]. As in case of the central B-splines, multivariate Jackson type kernels of order N *(see* Eqn 5*)* are defined by:

$$\mathcal{J}_N^n(\underline{x}) \;:=\; \prod_{i=1}^{n}J_N(x_i), \qquad \underline{x}=(x_1,\ldots,x_n)\in\mathbb{R}^n. \tag{5}$$

In particular, Jackson type kernels revealed to be very useful, e.g., for applications to the biomedical field, [11,18].

Finally, as a last important class of (radial) kernels we can mention the so called Bochner–Riesz kernels of order $N>0$ (see Eqn 6), defined as follows:

$$r_N(\underline{x}) \;:=\; \frac{2^N}{\sqrt{2\pi}}\,\Gamma(N+1)\,\|\underline{x}\|_2^{-N-1/2}J_{N+1/2}(\|\underline{x}\|_2), \qquad \underline{x}\in\mathbb{R}^n, \tag{6}$$

where J_λ is the Bessel function of order λ [14], and Γ is the usual Euler gamma function. For several examples of kernels, see, e.g., [2,3,15,16,20–24].

In Fig. 1, an example of the application of the S-K algorithm is shown, for various kernels. Here the "Starting image" has a dimension of 128×128 pixels, and has been obtained by reducing in size, the so-called "Target" image (256×256 pixels). For more details about the size reduction process, see Section 5. Now, if we rescale the starting image to the dimension of 256×256 pixels without using interpolation or quasi-interpolation algorithms, by means of a mere duplication of the pixels, we obtain the second image of the first column in Fig. 1 ("No interpolation" image). On the second column of Fig. 1, we have the reconstructed images (all of 256×256 pixels) obtained by the application of the S-K algorithm with various w and N (i.e., for kernels of various orders). More precisely, figure "A" has been obtained with the bi-dimensional central B-spline $\mathcal{B}_2^5$ with $w=5$. The figure "B" has been obtained with the bivariate Jackson type kernel $\mathcal{J}_{10}^2$ with $w=40$, and finally, figure "C" has been obtained with the bi-dimensional Bochner-Riesz kernel r_5 with $w=25$.

In what follows, we will evaluate the performance of the S-K algorithm in comparison with some well-known interpolation and quasi-interpolation algorithms for image processing, in term of the so-called PSNR and the CPU time.

3. The peak signal-to-noise ratio (PSNR)

The Peak Signal-to-Noise Ratio (PSNR) defined in Eqn 7 is a well known index in literature and it is often used to quantify the rate of similarity between two signals. It is expressed by the following formula:

$$PSNR \;=\; 10\cdot\log_{10}\left(\frac{(f_{max})^2}{MSE}\right), \tag{7}$$

where $f_{\max}$ is the maximum possible value of the signal, or function f (the full scale value), and MSE is the standard Mean Square Error, defined in the domain $D\subset\mathbb{R}^n$ of f, as follows:

$$MSE \;=\; \frac{\int_D |f(\underline{x})-f_r(\underline{x})|^2\,d\underline{x}}{\int_D d\underline{x}},$$

f being the original signal and f_r being the reconstructed version of the original signal f. Note that, usually, for real physical signals, $D\subset\mathbb{R}^n$ with $1\le n\le 4$. The PSNR is extensively used in image analysis and processing to evaluate, for example,

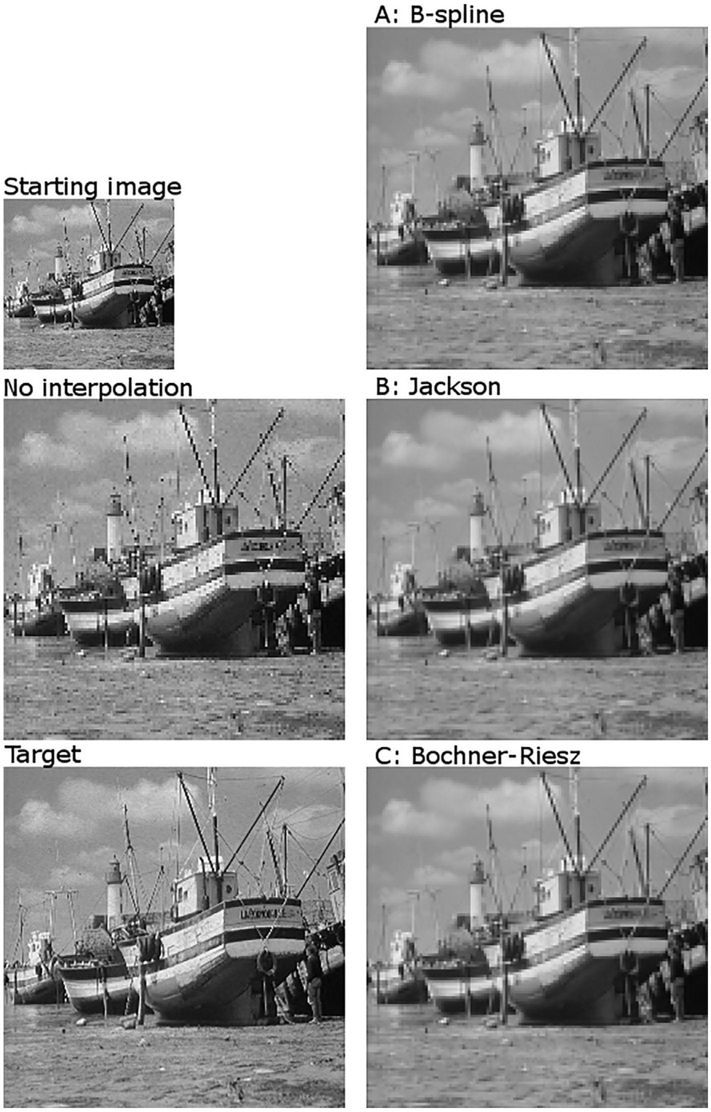

Fig. 1. On the second column, we have some reconstructions of the "Starting image" by the application of the S-K algorithm with various kernels.

the rate of similarity of two images after a watermarking process [30]. In the field of the image reconstruction, where the domain D is discrete, the 2-dimensional discrete version of the MSE is achieved replacing the integral by the summation symbol, as follows:

$$MSE_d \; = \; \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{|I(i,j) - I_r(i,j)|^2}{NM}, \tag{8}$$

where I is the original image, I_r is the reconstructed version of the original image I, N and M are the dimensions of the images.

In this paper, we use 8-bits gray levels images and in this case the maximum possible value is equal to 255. Hence:

$$PSNR \; = \; 10 \cdot \log_{10} \left(\frac{255^2}{MSE_d} \right).$$

To perform the measurement of the similarity between the original and the reconstructed images, we adopt the standard version of PSNR because it gives an objective, not observer-dependent, evaluation of the error after the reconstruction of the image, see e.g., [13].

To evaluate the PSNR with Matlab© we have used the native function psnr(). Before performing the calculation is appropriate to convert the image data, from the uint8 Matlab© specific data format, into a double. This is necessary because, if the psnr() function is applied to uint8 data it produces a zero difference between the original image and the reconstructed one every time the difference in (8) is less than zero: the latter could bring to erroneous numerical estimations.

4. Some interpolation and quasi-interpolation methods for digital image processing

The main purpose of this paper is to study the behavior of sampling Kantorovich operators in image reconstruction, i.e., the so-called S-K algorithm, in comparison with other well known methods in literature. For the aim of this study we have chosen, as reference for the state of the art, standard bilinear and bicubic methods other than quasi-Finite Impulse Response (quasi-FIR) and quasi-Infinite Impulse Response (quasi-IIR) filters as defined in [13,32]. As described above, sampling Kantorovich operators are quasi-interpolation operators. We expect that the quasi-interpolation methods give better results than the interpolation ones, as established in [7]. The choice of the reference algorithms is motivated by the fact that bilinear and bicubic, which are both interpolation methods, represent very performing algorithms in terms of time consuming and PSNR [26], respectively.

On the other side, FIR and IIR which are both quasi-interpolation methods, appear to be more performing in the PSNR sense in comparison with interpolation algorithms (see [13] again).

It is well-known that, most common quasi-interpolation methods need the use of boundary conditions [13]. One of the advantages of using S-K algorithm is that it can work without specifying any particular boundary conditions: we assume that the pixels outside the image have the constant value equal to zero, in fact they do not provide additional informations and we do not resort to any speculative methods to assign them suitable values.

All the methods used in this paper for the evaluation of the quality of digital image reconstruction by S-K algorithm can be expressed as a double convolution:

$$f(\underline{x}) \;=\; \sum_{\underline{k}\in\mathbb{Z}^2} c_{\underline{k}}\,\varphi(\underline{x}-\underline{k}), \quad (c_{\underline{k}}) = (f_{\underline{k}}) * (p_{\underline{k}}), \quad \underline{x}\in\mathbb{R}^2, \tag{9}$$

where the coefficients $(c_{\underline{k}})$ are obtained by a discrete filtering $(p_{\underline{k}})$ of $(f_{\underline{k}})$, where $(f_{\underline{k}})$ is a discrete version of the original image f, and $\varphi(\underline{x})$ is a given kernel, see e.g., [25,33].

Bilinear method consists of a linear interpolation for functions of two variables: this interpolation method has been implemented in Matlab© using (9) with $\varphi=\beta^1$, being β^n a generic central B-spline of order n, and $p_{\underline{k}}=1$, for every $\underline{k}$.

Bicubic method consists on the implementation of (9) with $\varphi=\beta^3$ and $p_{\underline{k}}=1$, for every $\underline{k}$.

Quasi-FIR method has been implemented by (9) with $\varphi=\beta^1$ and where the coefficients $(c_{\underline{k}})$ are computed by the matrix convolution between the original image $(f_{\underline{k}})$ and the filtering matrix:

$$A=\begin{bmatrix} -\dfrac{1}{144} & -\dfrac{7}{72} & -\dfrac{1}{144}\\ -\dfrac{7}{72} & -\dfrac{49}{36} & -\dfrac{7}{72}\\ -\dfrac{1}{144} & -\dfrac{7}{72} & -\dfrac{1}{144}\end{bmatrix}.$$

The matrix A is generated using the following transfer function $H(z)$ (i.e., the z-transform of impulse response of the filter, [13]):

$$H(z) = -\frac{1}{12}z^{-1} + \frac{7}{6} - \frac{1}{12}z.$$

For the sake of completeness, we recall that $H(z)=\sum_{k\in\mathbb{Z}} h_k z^{-k}$ is the z-transform of any digital filter (h_k).

Quasi-IIR method has been implemented according to (9) by using both $\varphi=\beta^1$ and $\varphi=\beta^3$ and where the coefficients $(c_{\underline{k}})$ are computed by a product between the original image $(f_{\underline{k}})$ and a suitable filtering matrix A_I.

The matrix A_I is generated using the transfer function $H(z)$ expressed by:

$$H(z) \;=\; Y(z)\,/X(z),$$

where $X(z)$ and $Y(z)$ are respectively the z-transform of the input and the output of the filter, with:

$$Y(z) = \left(I - \frac{A}{m}\right)^{-1} - \frac{X(z)}{m}.$$

Here, m is a suitable coefficient determined by $H(z)$.

In case of $\varphi=\beta^1$, the transfer function (in the z-transform domain) is the following:

$$H(z) \;=\; \frac{1}{12}z^{-1} + \frac{5}{6} + \frac{1}{12}z$$

giving $m = \frac{25}{36}$, and the matrix A is a Toeplitz matrix with Toeplitz blocks, of the form:

$$A = \begin{bmatrix} A_1 & A_2 & 0 & 0 & \dots & \dots & 0 \\ A_2 & A_1 & A_2 & 0 & \dots & \dots & 0 \\ 0 & A_2 & A_1 & A_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 & A_2 & A_1 \end{bmatrix}$$

with A_1 and A_2 Toeplitz matrices defined as follows:

$$A_1 = -\begin{bmatrix} 0 & \frac{5}{72} & 0 & 0 & \dots & 0 \\ \frac{5}{72} & 0 & \frac{5}{72} & \dots & \dots & 0 \\ 0 & \frac{5}{72} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & \dots & \dots & \frac{5}{72} \\ 0 & \dots & \dots & 0 & \frac{5}{72} & 0 \end{bmatrix},$$

$$A_2 = -\begin{bmatrix} \frac{5}{72} & \frac{1}{144} & 0 & 0 & \dots & \dots & 0 \\ \frac{1}{144} & \frac{5}{72} & \frac{1}{144} & \dots & \dots & \dots & 0 \\ 0 & \frac{1}{144} & \frac{5}{72} & \frac{1}{144} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & \dots & \dots & \dots & \frac{1}{144} \\ 0 & \dots & \dots & \dots & 0 & \frac{1}{144} & \frac{5}{72} \end{bmatrix}.$$

In case of $\varphi = \beta^3$, the transfer function $H(z)$ (in the z-transform domain) is the following:

$$H(z) = -\frac{1}{720}z^{-2} + \frac{31}{180}z^{-1} + \frac{79}{120} + \frac{31}{180}z - \frac{1}{720}z^2,$$

giving $m = c^2$, and A is a Toeplitz matrix with Toeplitz blocks, of the form:

$$A = \begin{bmatrix} A_1 & A_2 & A_3 & 0 & \dots & \dots & 0 \\ A_2 & A_1 & A_2 & A_3 & \dots & \dots & 0 \\ A_3 & A_2 & A_1 & A_2 & A_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & .A_3 & A_2 & A_1 & A_2 \\ 0 & \dots & \dots & \dots & A_3 & A_2 & A_1 \end{bmatrix},$$

with A_1, A_2 and A_3 Toeplitz matrices defined as follows:

$$A_1 = -\begin{bmatrix} 0 & bc & ac & 0 & \dots & 0 \\ bc & 0 & bc & ac & \dots & 0 \\ ac & bc & 0 & \dots & \dots & bc \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & ac & bc & 0 \end{bmatrix},$$

$$A_2 = -\begin{bmatrix} bc & b^2 & ab & 0 & \dots & \dots & 0 \\ b^2 & bc & b^2 & bc & \dots & \dots & 0 \\ ab & b^2 & bc & b^2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & ab & b^2 & bc \end{bmatrix},$$

Table 1
Pseudo-code of the S-K algorithm.

Objective: Magnification of the starting image I Inputs: I image, $L \times M$ pixel resolution, w, kernel type specific parameter N, zoom factor $R = 2$
Main steps: • Calculation of the initially $w \times w$ scaled image I_w; • convolution between the chosen kernel and I_w; • resampling of I_w according with the zoom factor R.
Output: Greyscale magnified image I_{SK}

$$A_3 = -\begin{bmatrix} ac & ab & a^2 & 0 & \dots & \dots & 0 \\ ab & ac & ab & a^2 & \dots & \dots & 0 \\ a^2 & ab & ac & ab & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & a^2 & ab & ac \end{bmatrix},$$

and $a = \frac{1}{720}, b = \frac{31}{180}, c = \frac{79}{120}$.

5. Numerical examples

In this paper we focus our attention on the capability of sampling Kantorovich operators, based upon various kernels, to reconstruct images by the S-K algorithm, as described in Section 2. In particular, we consider the problem of rescaling. In Fig. 2 the flowchart of the above method is shown and the corresponding pseudo code can be found in Table 1.

For our software simulation we use a standard set of 5 square images (file names: ‘lena’, ‘baboon’, ‘cameraman’, ‘boat’, ‘barbara’),.png (Portable Network Graphix) file format, with dimensions varying from 16 × 16 pixel to 128 × 128 pixels, doubled in size at each step (16, 32, 64, 128), for a total number of 20 images collected by dimension. Note that, in general the S-K algorithm is not specific for square images, but it can be applied in order to reconstruct and enhance images with any resolution.

To generate the above sets of images, for each file name we start from a 256 × 256 pixels sized image and halving its size at each step, achieving five 128 × 128 pixels images, five 64 × 64 pixels images and so on.

The size reduction process proceeds by a mean of the original image: the gray-level mean is calculated shifting a 2 × 2 cell and associating for each step a new pixel in the reduced size image. In this way we can have an “original” reference for each image and we use it to compare with the post-processing results (for example on the size reduction process, see Fig. 3). Each image of size $M \times M$ is then doubled in size using the S-K algorithm [4] and the result is compared with the reference image of size $2M \times 2M$, using the PSRN defined in Section 3. For the implementation of the S-K algorithm, three different families of kernels have been used for the reconstruction process: the central B-spline, the Jackson type kernels and the Bochner–Riesz kernels, varying w and the order N of each kernel from $w = 5$ to $w = 50$ (with step size 5) and from $N = 1$ to $N = 10$ (with step size 1). The S-K algorithm with central B-spline, Jackson type and Bochner–Riesz kernel shows results in general depending on N and w.

For the central B-spline kernels, the PSNR exhibits its maximum (that expresses the best achieved performance) for $N \in \{4, 5\}$ and $w = 5$. This trend reproduces for all considered images, independently from the size. For instance, if we plot the trend of the PSNR in function of N, for some values of w (see Fig. 4), we obtain in general a concave function with some small oscillations, due to numerical computational errors, as w and N increases. For this reason, we are going to consider, as reference, the results with $w \geq 15$; here $w = 15$ represents a lower bound for the stability of the approximation error, in the sense that the error is almost constant for every N.

Moreover for $w \geq 15$ the PSNR shows no significant improvements when varying w. The choice of lower values for w and N determines a lower execution time and this behavior is common to each kernel, as we will see in the final part of this section.

The results of the application of the S-K algorithm with both Jackson type and Bochner–Reisz kernels exhibit a saturation of the PSNR when the order N increases (see Fig. 5).

Here, with the word saturation we intend that the PSNR has a not meaningful variation.

The Jackson type kernels as well as the Bochner–Riesz type kernels, exhibit an improvement of the PSNR as w increases.

To obtain a lower bound for w and N for which the S-K algorithm saturates, we introduce the *a posteriori* gain speed (see Eqn 10), defined as:

$$V_{gain} := \frac{G_{i_{max}}}{|\Delta_t|}, \tag{10}$$

where $G_{i_{\max}}$ is the maximum gain, in terms of PSNR, when varying N, between two subsequent values of w (among those considered), Δ_t is the mean difference of CPU time between two subsequent values of w.

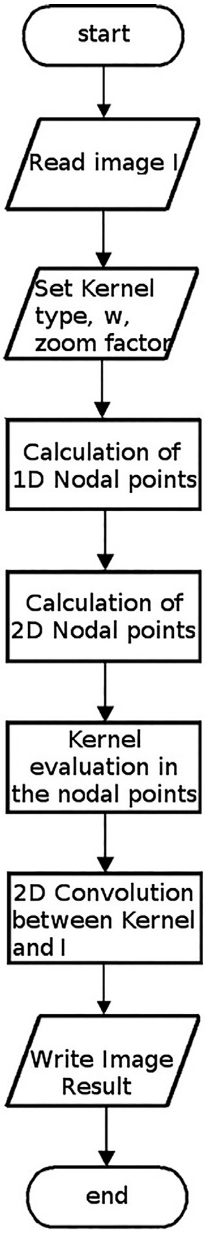

Fig. 2. Flowchart of the S-K algorithm.

The index V_{gain} is positive if and only if $G_{i_{\max}} > 0$, according to the fact that we can accept the increase of the execution time only when the achieved results for a certain w are better than the ones from the previous considered w values.

From the results of Table 2 we can observe the evolution of V_{gain}, as w increases. We mark out that, in case of Jackson type kernels since order $w = 20$, V_{gain} has a value significantly high. For instance, passing from $w = 5$ to $w = 10$, we have an increase of PSNR which is almost 114 times bigger than the increase of the CPU time, passing from $w = 10$ to $w = 15$, we have an increase of PSNR which is almost 16 times bigger than the increase of the CPU time, and so on. Note that, in case of $w = 25$ and $w = 30$ we have a double of the CPU time with respect to the improvement of the PSNR, then it appears disadvantageous to apply the S-K algorithm with such value of w. Finally, we also observe that some values of V_{gain} are negative but approximatively near to zero, and this is due to numerical errors. Analogous considerations can be done for the numerical results of Table 2 related to the case of Bochner–Riesz kernels.

The reference methods (bilinear, bicubic, quasi FIR, quasi IIR with β^1 and β^3) show their best results in terms of PSNR as the image size generally increases. Results are shown in Tables 3–7.

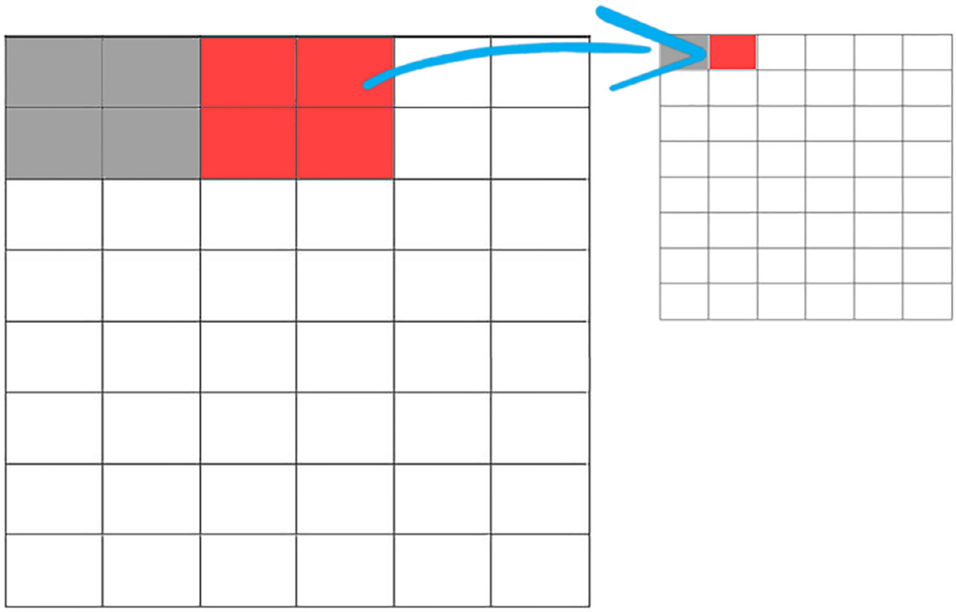

Fig. 3. The size reduction process associates to a 2 × 2 pixels area (on the left) a single pixel (on the right) having as value the mean value of the 2 × 2 cell.

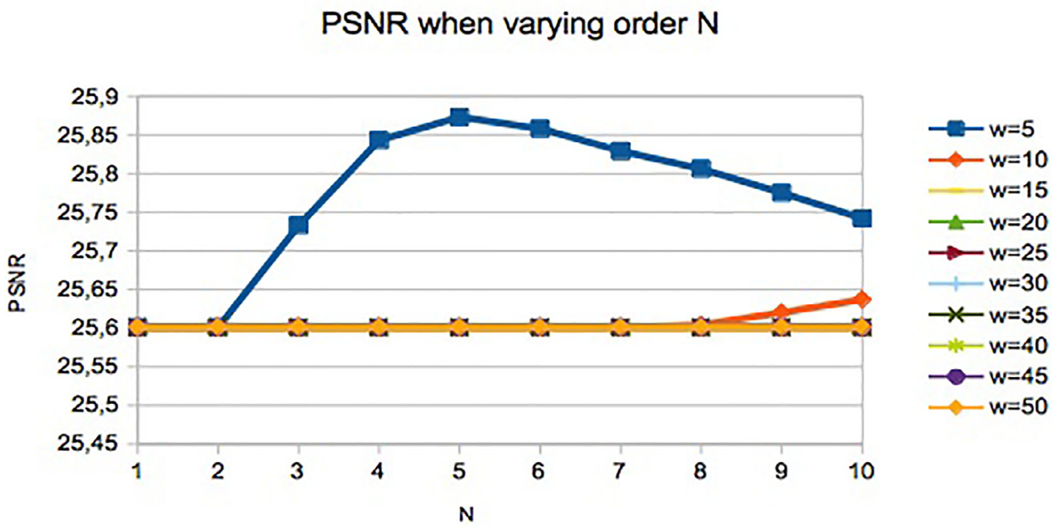

Fig. 4. Central B-spline trend for various N. Best results are for $w = 5$ and $N = 5$. The graph refers to the reconstruction of 'lena.png', starting with size 32 × 32 pixels, and reconstructed with size 64 × 64 pixels. The trend in the graph is qualitatively the same for all the considered images. For $w \geq 15$ the trend of the PSNR is the same of the orange one.

It is evident from the results shown in Table 8 that the S-K algorithm produces better results than the reference methods, even with a small value of w (e.g., $w = 5$). The central B-spline kernels give the best results in terms of PSNR: this behavior stands with low values of w. When w increases the central B-spline kernels show a fast saturation compared to Bochner–Riesz and Jackson type ones (see Tab. 9 again, and Fig. 6). As w increases Jackson type kernels express the best performance.

It is possible to compare the S-K algorithm using the above kernels, for different values of w and N. In the Tables 9 and 11 the mean values of PSNR and CPU time have been computed for all $1 \leq N \leq 10$, $N \in \mathbb{N}$, and for all the reconstructed images, when varying w. The Fig. 6 shows qualitatively the PSNR trends for different kernels, while Fig. 8 shows the CPU time for different kernels. In terms of PSNR, central B-spline kernels give better results for low values of w and its PSNR is inversely proportional to w, until $w = 15$. For $15 \leq w < 25$, the Bochner-Riesz kernels show better performance than all the other ones. For $w \geq 25$, the best results are given by the Jackson type kernels (see Fig. 7).

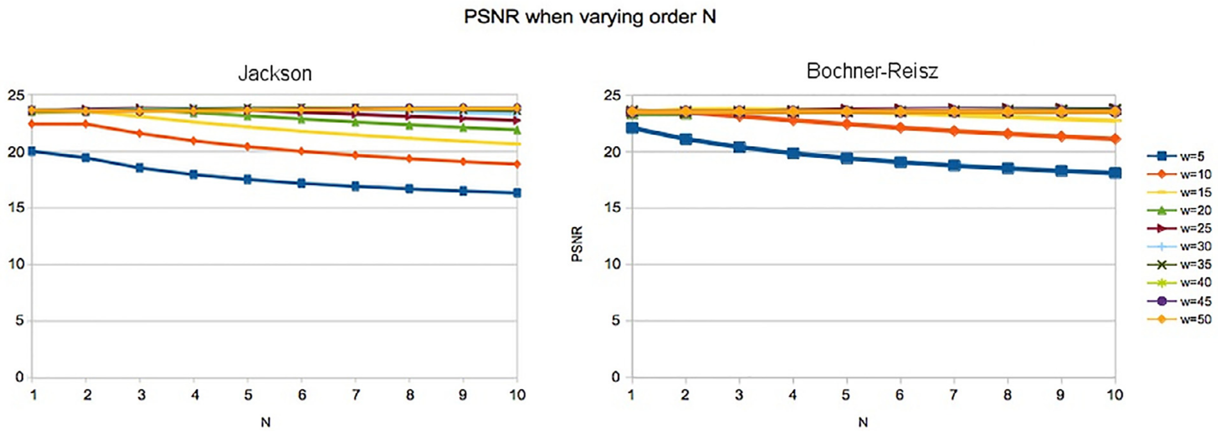

Fig. 5. Jackson type kernel results' trend (on the left) and Bochner-Reisz type kernel results' trend (on the right) for various N. Best results are for $w = 5$ and $N = 5$. The graph refers to the reconstruction of 'lena.png', starting size 32 × 32 pixels, reconstructed size 64 × 64 pixels. The trend in the graph is qualitatively the same for all the considered images.

Table 2
Incremental time for the reconstructions performed by Jackson type and Bochner–Riesz kernels, on square sized images of 16 × 16 pixels. The showed values are the mean of the results for the entire set of images (Lena, Boat, etc).

Jackson			
PSNR	w	Time (s)	V_{gain} (PSNR/ms)
17.209	5	0.044	–
19.773	10	0.058	114.531
21.059	15	0.093	16.669
21.711	20	0.134	6.166
22.082	25	0.251	1.141
22.242	30	0.352	0.546
22.293	35	0.489	0.124
22.289	40	0.587	-0.013
22.262	45	0.692	-0.087
22.231	50	0.847	-0.070
Bochner-Riesz			
PSNR	w	Time (s)	V_{gain} (PSNR/ms)
18.992	5	0.060	–
21.244	10	0.073	77.898
22.018	15	0.121	5.901
21.938	20	0.403	-0.104
22.246	25	0.356	-2.233
22.209	30	0.494	-0.093
22.164	35	0.687	-0.080
19.709	40	1.302	-1.220
22.149	45	1.067	-3.620
22.126	50	1.452	-0.021

For what concerns the S-K algorithm CPU time, implemented as in [5], it depends on the size of the original image being reconstructed, on the used kernel χ and on w.

All the code has been written and executed in Matlab©, version 8.4.0.150421 (R2014b) on a pc running Microsoft Windows©10 Home Version 10.0.

The S-K algorithm performs significantly faster with respect to the quasi-FIR and quasi-IIR (see Table 10 again). In particular, the CPU time of the quasi-IIR depends on the complexity of the algorithm used to invert the matrix $(I - A)$; it is well known that the time for this calculation increases with the size of the matrix A that is proportional to the size of the image to reconstruct (as happens, e.g., in Cholesky decomposition and other well-known methods). In terms of CPU time the best performance of the S-K algorithm are achieved in case of central B-spline kernels, that result to be almost constant when varying w, while in case of both Jackson type and Bochner-Riesz kernels, the CPU time increases with respect to w (see Fig. 8). In Tables 12, 13, 14 a numerical simulation by an image with a larger value of starting size (the starting size varies from 16 × 16 to 256 × 256) with respect to the previous ones has been considered, in order to show the variation of the CPU time. In fact, this experiment allows to evaluate the computational efficiency of the S-K algorithm.

Table 3
Numerical results obtained by using bilinear interpolation for different image sizes for each file of the dataset. At the bottom of each size, the mean PSNR, the mean execution time, and the standard deviation are computed.

Original size	Reconstructed size	PSNR	Time (s)	Filename
16	32	15.483	0.019	baboon
16	32	14.582	0.019	barbara
16	32	15.774	0.019	boat
16	32	15.036	0.019	cameraman
16	32	16.006	0.023	lena
Mean		15.376	0.020	
Std. Dev.		0.573	0.002	
32	64	17.673	0.050	baboon
32	64	16.684	0.050	barbara
32	64	17.372	0.050	boat
32	64	16.957	0.051	cameraman
32	64	18.383	0.051	lena
Mean		17.414	0.050	
Std. Dev.		0.661	0.001	
64	128	19.426	0.173	baboon
64	128	19.208	0.171	barbara
64	128	19.341	0.170	boat
64	128	18.702	0.172	cameraman
64	128	20.922	0.173	lena
Mean		19.520	0.172	
Std. Dev.		0.833	0.001	
128	256	19.942	0.812	baboon
128	256	21.296	0.772	barbara
128	256	20.934	0.818	boat
128	256	20.704	0.840	cameraman
128	256	23.074	0.652	lena
Mean		21.190	0.779	
Std. Dev.		1.164	0.075	

Table 4
Numerical results obtained by using bicubic interpolation for different image sizes for each file of the dataset. At the bottom of each size, the mean PSNR, the mean execution time, and the standard deviation are computed.

Original size	Reconstructed size	PSNR	Time (s)	Filename
16	32	16.614	0.018	baboon
16	32	16.321	0.019	barbara
16	32	17.390	0.020	boat
16	32	16.617	0.020	cameraman
16	32	17.584	0.114	lena
Mean		16.905	0.038	
Std. Dev.		0.549	0.042	
32	64	18.870	0.050	baboon
32	64	18.153	0.049	barbara
32	64	18.929	0.049	boat
32	64	18.361	0.049	cameraman
32	64	19.887	0.049	lena
Mean		18.840	0.049	
Std. Dev.		0.672	0.000	
64	128	20.565	0.162	baboon
64	128	20.569	0.163	barbara
64	128	20.683	0.164	boat
64	128	20,112	0.162	cameraman
64	128	22.339	0.166	lena
Mean		20.854	0.163	
Std. Dev.		0.859	0.002	
128	256	21.000	0.650	baboon
128	256	22.465	0.619	barbara
128	256	22.278	0.665	boat
128	256	22.105	0.666	cameraman
128	256	24.486	0.622	lena
Mean		22.467	0.644	
Std. Dev.		1.264	0.023	

Table 5
Numerical results obtained by using FIR quasi-interpolation for different image sizes for each file of the dataset. At the bottom of each size, the mean PSNR, the mean execution time, and the standard deviation are computed.

Original size	Reconstructed size	PSNR	Time (s)	Filename
16	32	16.319	0.019	baboon
16	32	15.926	0.021	barbara
16	32	17.267	0.020	boat
16	32	16.471	0.023	cameramen
16	32	17.477	0.241	lena
Mean		16.692	0.065	
Std. Dev.		0.656	0.099	
32	64	18.530	0.048	baboon
32	64	17.879	0.049	barbara
32	64	18.698	0.048	boat
32	64	18.228	0.048	cameramen
32	64	19.739	0.054	lena
Mean		18.615	0.049	
Std. Dev.		0.702	0.003	
64	128	20.256	0.170	baboon
64	128	20.354	0.160	barbara
64	128	20.492	0.161	boat
64	128	19.908	0.163	cameramen
64	128	22.178	0.165	lena
Mean		20.638	0.164	
Std. Dev.		0.888	0.004	
128	256	20.709	0.750	baboon
128	256	22.260	0.612	barbara
128	256	22.043	0.662	boat
128	256	21.868	0.769	cameramen
128	256	24.266	0.610	lena
Mean		22.229	0.681	
Std. Dev.		1.287	0.075	

Table 6
Numerical results obtained by using IIR quasi-interpolation with β^1 for different image sizes for each file of the dataset. At the bottom of each size, the mean PSNR, the mean execution time, and the standard deviation are computed.

Original size	Reconstructed size	PSNR	Time (s)	Filename
16	32	13.615	0.029	baboon
16	32	15.185	0.029	barbara
16	32	14.363	0.028	boat
16	32	14.990	0.026	cameraman
16	32	16.580	0.032	lena
Mean		14.947	0.029	
Std. Dev.		1.100	0.002	
32	64	14.232	0.189	baboon
32	64	17.164	0.161	barbara
32	64	16.836	0.189	boat
32	64	15.755	0.205	cameraman
32	64	18.510	0.163	lena
Mean		16.499	0.181	
Std. Dev.		1.604	0.019	
64	128	14.555	5.437	baboon
64	128	18.511	5.329	barbara
64	128	19.840	5.464	boat
64	128	16.945	5.521	cameraman
64	128	19.600	4.856	lena
Mean		17.890	5.321	
Std. Dev.		2.187	0.269	
128	256	15.629	228.440	baboon
128	256	20.060	226.970	barbara
128	256	17.404	223.610	boat
128	256	17.483	228.850	cameraman
128	256	20.428	233.970	lena
Mean		18.201	228.368	
Std. Dev.		2.011	3.749	

Table 7
Numerical results obtained by using IIR quasi-interpolation with β^3 for different image sizes for each file of the dataset. At the bottom of each size, the mean PSNR, the mean execution time, and the standard deviation are computed.

Original size	Reconstructed size	PSNR	Time (s)	Filename
16	32	14.603	0.029	baboon
16	32	13.816	0.031	barbara
16	32	15.760	0.030	boat
16	32	13.823	0.028	cameraman
16	32	15.601	0.033	lena
Mean		14.721	0.030	
Std. Dev.		0.935	0.002	
32	64	14.328	0.170	baboon
32	64	13.884	0.174	barbara
32	64	15.635	0.186	boat
32	64	14.185	0.169	cameraman
32	64	14.907	0.169	lena
Mean		14.588	0.174	
Std. Dev.		0.693	0.007	
64	128	17.303	5.647	baboon
64	128	15.227	5.446	barbara
64	128	17.435	5.404	boat
64	128	13.883	5.405	cameraman
64	128	14.859	4.931	lena
Mean		15.741	5.367	
Std. Dev.		1.566	0.263	
128	256	15.694	228.850	baboon
128	256	15.913	220.950	barbara
128	256	17.658	215.050	boat
128	256	15.419	229.590	cameraman
128	256	14.670	222.110	lena
Mean		15.871	223.310	
Std. Dev.		1.104	6.028	

Table 8
The mean values of the PSNR computed on all the images of the dataset, for the considered methods. The last three columns of the table on the bottom refer to the kernels used for the implementation of the S-K algorithm, with $w = 5$. In particular, the mean PSNR is computed considering the above kernels for all the orders $1 \leq N \leq 10$. From the results of these tables, it is evident that S-K algorithm gives the best performance, in terms of PSNR, compared to other methods. In particular, B-spline kernels gives the highest (best) values of PSNR.

Starting size	Bilinear	Bicubic	quasi FIR	quasi IIR β^1
16	15.376	16.905	16.692	14.947
32	17.414	18.840	18.615	16.499
64	19.520	20.854	20.638	17.890
128	21.190	22.467	22.229	18.201
Starting size	quasi IIR β^3	B-splines	Bochner–Riesz	Jackson
16	14.721	22.096	18.993	17.209
32	14.588	23.743	21.047	19.242
64	15.741	25.569	22.545	21.204
128	15.871	26.815	25.07	24.137

Table 9
The mean values of the PSNR computed on all the images of the dataset with their relative dimension, processed by the S-K algorithm, based upon the above kernels. Also here, the mean PSNR is computed considering the above kernels for all the orders $1 \leq N \leq 10$.

w	B-spline	Bochner–Riesz	Jackson
5	24.5555	21.589	20.0779
15	24.2577	24.397	23.81
25	24.4577	24.412	24.733
35	24.4577	24.286	24.802
50	24.4577	24.085	24.645

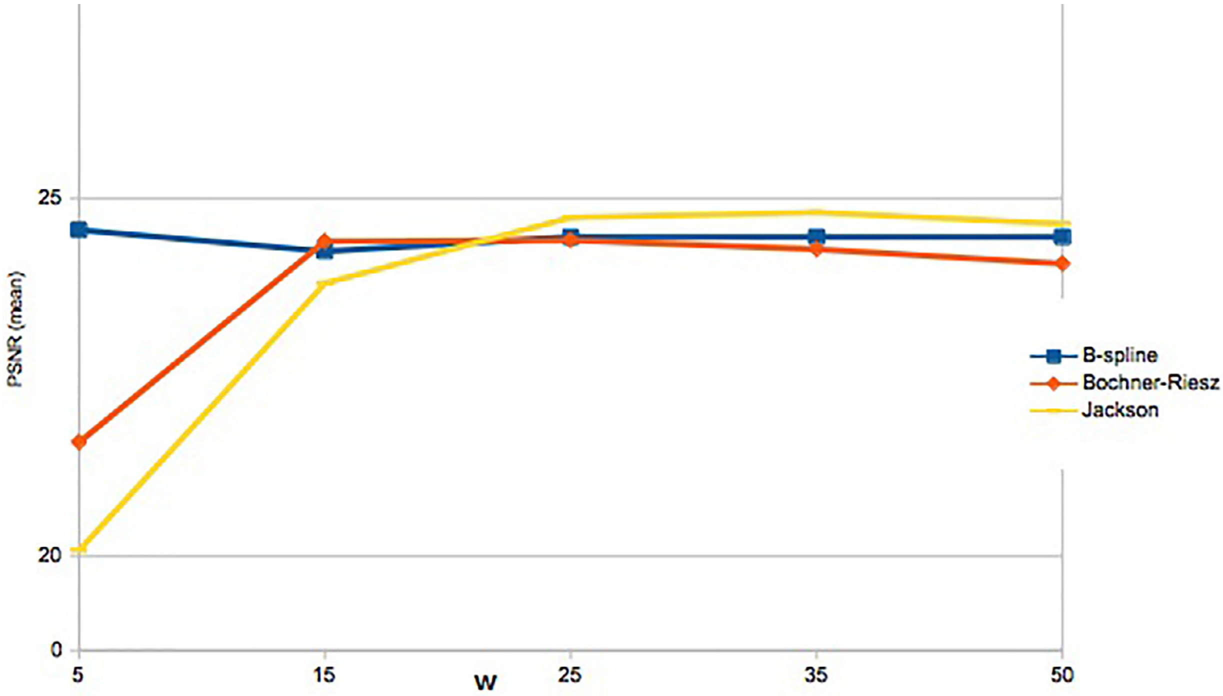

Fig. 6. Trend of PSNR after the reconstruction of the images by the S-K algorithm. The saturation process occurs as w increases.

Table 10
The mean values of CPU time (expressed in seconds) computed on all the images of the dataset, for the considered methods. The last three columns in the lower part of the table refer to the kernels used for the implementation of the S-K algorithm, with $w = 5$. In particular, the mean CPU time is computed considering the above kernels for all the orders $1 \leq N \leq 10$. From the results of these tables, it is evident that bilinear and bicubic give the best performance. In particular, B-spline kernels show the best performance.

Starting size	Bilinear	Bicubic	quasi FIR	quasi IIR β^1
16	0.020	0.038	0.065	0.029
32	0.050	0.049	0.049	0.181
64	0.172	0.163	0.164	5.321
128	0.779	0.644	0.681	228.368

Starting size	quasi IIR β^3	B-spline	Bochner–Riesz	Jackson
16	0.030	0.035	0.039	0.044
32	0.174	0.083	0.206	0.172
64	5.367	0.236	0.523	1.254
128	223.310	0.844	4.565	3.447

Table 11
The mean values of the CPU time (expressed in seconds) on all the images of the dataset with their relative dimension, processed by the S-K algorithm, based upon the above kernels. Also here, the mean CPU time is computed considering the above kernels for all the orders $1 \leq N \leq 10$.

w	B-spline	Bochner-Riesz	Jackson
5	0.2997	6.236	7.204
15	0.4581	38.272	34.326
25	0.6965	132.379	98.632
35	0.9944	241.444	182.441
50	1.6	205.64	351.601

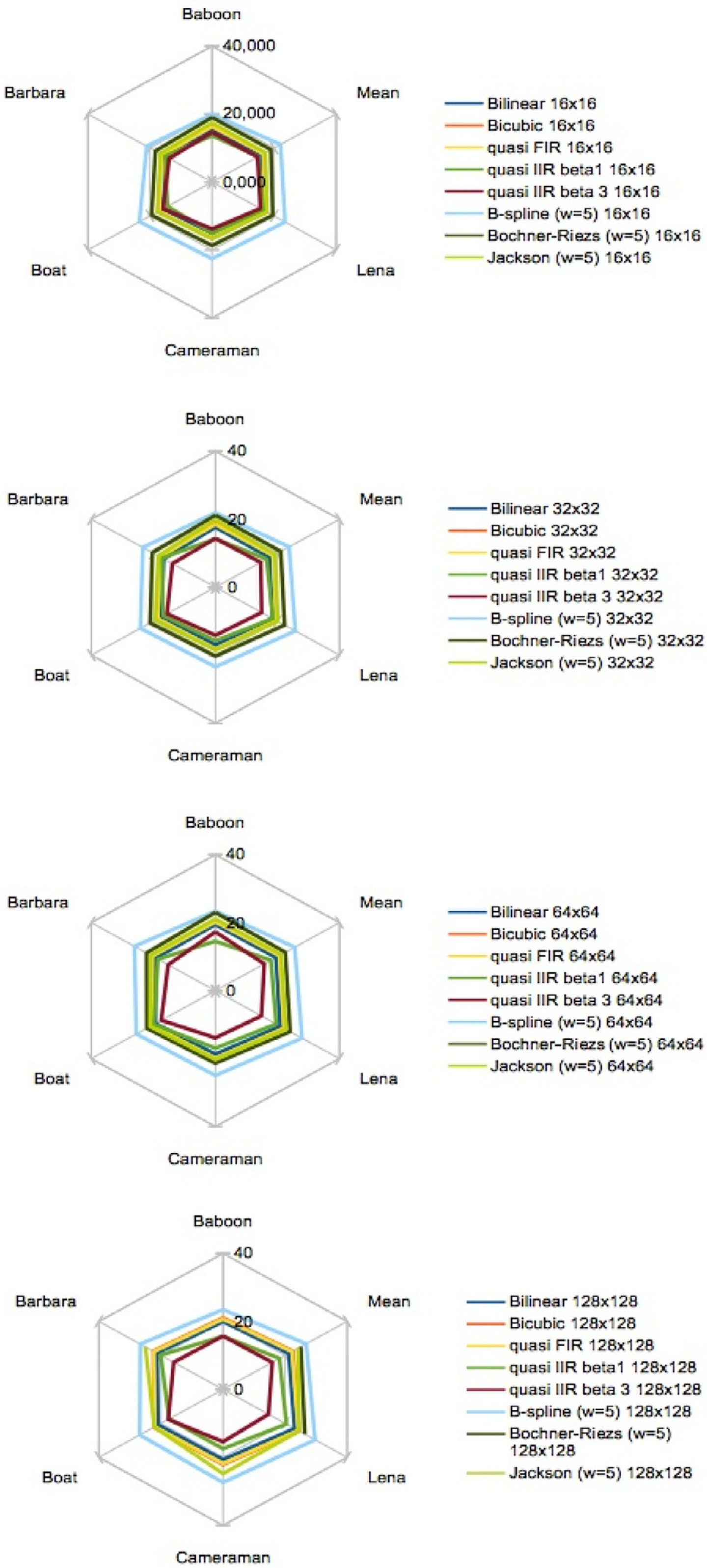

Fig. 7. Graphical representation of the numerical results listed in Table 8.

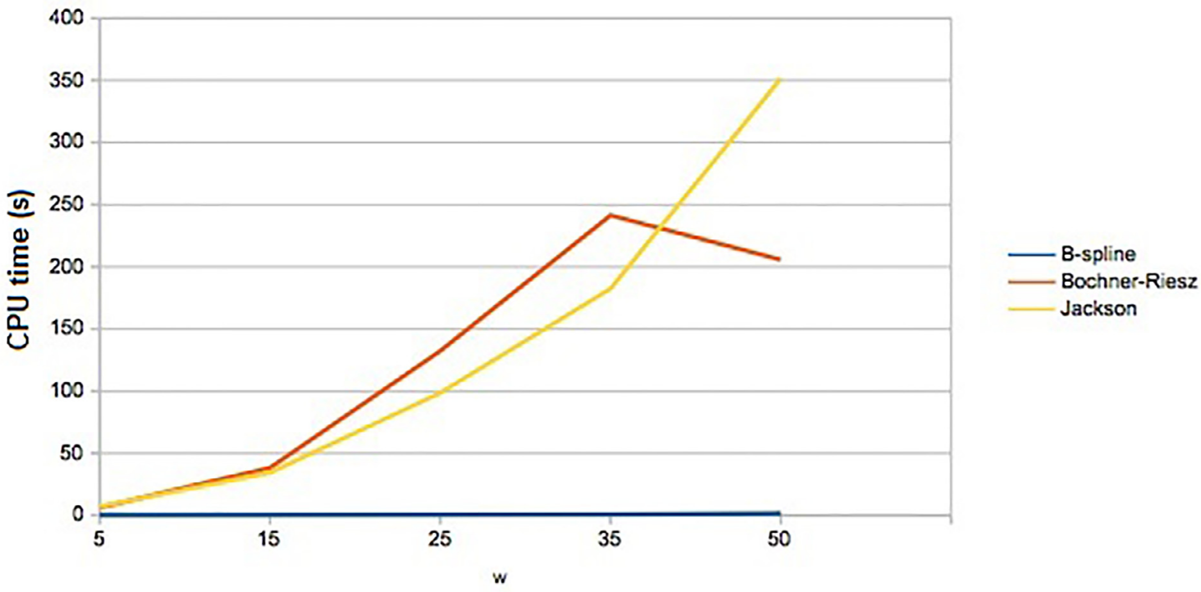

Fig. 8. Trend of the CPU time after the reconstruction of the images by the S-K algorithm.

Table 12
The CPU time (expressed in seconds) computed by varying the starting size of the images and processed by the S-K algorithm based upon the Jackson type kernel, with $N = 10$, $w = 25$, and zoom factor $R = 2$.

Jackson (N=10, W=25, R=2)	
Size	Time (s)
16×16	0.052
32×32	0.173
64×64	0.664
128×128	2.856
256×256	11.757

Table 13
The CPU time (expressed in seconds) computed by varying the starting size of the images and processed by the S-K algorithm based upon the Bochner–Riesz type kernel, with $N = 10$, $w = 25$, and zoom factor $R = 2$.

Bochner–Riesz (N=10, W=25, R=2)	
Size	Time (s)
16×16	0.081
32×32	0.214
64×64	0.838
128×128	3.571
256×256	16.004

Table 14
The CPU time (expressed in seconds) computed by varying the starting size of the images and processed by the S-K algorithm based upon the B-spline type kernel, with $N = 5$, $w = 25$, and zoom factor $R = 2$.

B-spline (N=5, W=25, R=2)	
Size	Time (s)
16×16	0.126
32×32	0.261
64×64	0.599
128×128	1.779
256×256	6.235

6. Final remarks and conclusions

In this paper we have compared the S-K algorithm with other meaningful well-known methods for image processing. Experimental results have shown better performance of S-K algorithm in terms of PSNR and CPU time than the considered other ones. Moreover, we have tested the S-K algorithm with three different families of kernels (central B-splines, Jackson type and Bochner–Riesz kernels) for different values of N and w. In general, we obtained that for values of $w \leq 15$, central B-splines provide the best results; for $15 < w < 25$, the Bochner–Riesz kernels seems to be the most performing, while if $w \geq 25$, the Jackson type kernels are the best ones. These results suggest how to proceed in the choice of the kernel and w before the application of S-K algorithm in concrete cases, such those studied in [4,5,18].

The experimental trends achieved for each used kernel show the typical saturation behavior of the approximation processes.

The numerical results confirm that the proposed algorithm is suitable for image processing, in particular in image reconstruction.

Acknowledgments

The authors are members of the Gruppo Nazionale per l'Analisi Matematica, la Probabilitá e le loro Applicazioni (GNAMPA) of the Istituto Nazionale di Alta Matematica (INdAM).

The authors are partially supported by the "Department of Mathematics and Computer Science" of the University of Perugia (Italy). Moreover, the first author of the paper has been partially supported within the 2019 GNAMPA-INdAM Project "Metodi di analisi reale per l'approssimazione attraverso operatori discreti e applicazioni", while the third author within the projects: (1) Ricerca di Base 2017 dell'Università degli Studi di Perugia - "Metodi di teoria degli operatori e di Analisi Reale per problemi di approssimazione ed applicazioni", (2) Ricerca di Base 2018 dell'Università degli Studi di Perugia - "Metodi di Teoria dell'Approssimazione, Analisi Reale, Analisi Nonlineare e loro Applicazioni", (3) "Metodi e processi innovativi per lo sviluppo di una banca di immagini mediche per fini diagnostici" funded by the Fondazione Cassa di Risparmio di Perugia.

This research has been accomplished within RITA (Research ITalian network on Approximation).

References

[1] G. Allasia, R. Cavoretto, A. De Rossi, A class of spline functions for landmark-based image registration, Math. Methods Appl. Sci. 35 (2012) 923–934.
[2] L. Angeloni, D. Costarelli, G. Vinti, A characterization of the convergence in variation for the generalized sampling series, Annales Academiae Scientiarum Fennicae Mathematica 43 (2018) 755–767.
[3] L. Angeloni, D. Costarelli, G. Vinti, A characterization of the absolute continuity in terms of convergence in variation for the sampling Kantorovich operators, Mediterranean J. Math. 16 (2) (2019), doi:10.1007/s00009-019-1315-0.
[4] F. Asdrubali, G. Baldinelli, F. Bianchi, D. Costarelli, L. Evangelisti, A. Rotili, M. Seracini, G. Vinti, A model for the improvement of thermal bridges quantitative assessment by infrared thermography, Appl. Energy 211 (2018) 854–864.
[5] F. Asdrubali, G. Baldinelli, F. Bianchi, D. Costarelli, A. Rotili, M. Seracini, G. Vinti, Detection of thermal bridges from thermographic images by means of image processing approximation algorithms, Appl. Math. Comput. 317 (2018) 160–171.
[6] C. Bardaro, P.L. Butzer, R.L. Stens, G. Vinti, Kantorovich-type generalized sampling series in the setting of Orlicz spaces, Sampl. Theory Signal Image Process. 6 (1) (2007) 29–52.
[7] T. Blu, M. Unser, Quantitative Fourier analysis of approximation techniques: part i - interpolators and projectors, IEEE Trans. Signal Process. 47 (10) (1999).
[8] P.L. Butzer, R.J. Nessel, Fourier Analysis and Approximation I, Academic Press, New York-London, 1971.
[9] R.E. Carlson, F.N. Fritsch, An algorithm for monotone piecewise bicubic interpolation, SIAM J. Numer. Anal. 26 (1) (1985) 230–238.
[10] M.J. Chen, C.H. Huang, W.L. Lee, A fast edge-oriented algorithm for image interpolation, Image Vis. Comput. 23 (9) (2005) 791–798.
[11] E. Cieri, D. Costarelli, B. Fiorucci, G. Isernia, M. Seracini, G. Simonte, G. Vinti, Computed tomography post-processing for abdominal aortic aneurysm lumen recognition in unenhanced exams, Ann. Vasc. Surg. 60 (2019) 407–414.
[12] F. Cluni, D. Costarelli, A.M. Minotti, G. Vinti, Enhancement of thermographic images as tool for structural analysis in earthquake engineering, NDT E Int. 70 (2015) 60–72.
[13] L. Condat, T. Blu, M. Unser, Beyond interpolation: optimal reconstruction by quasi interpolation, IEEE Trans. Image Proc. 16 (2007) 1195–1206.
[14] D. Constales, H. De Bie, P. Lian, A new construction of the Clifford-Fourier kernel, J. Fourier Anal. Appl. 23 (2) (2017) 462–483.
[15] L. Coroianu, D. Costarelli, S.G. Gal, G. Vinti, The max-product generalized sampling operators: convergence and quantitative estimates, Appl. Math. Comput. 355 (2019) 173–183.
[16] L. Coroianu, D. Costarelli, S.G. Gal, G. Vinti, The max-product sampling Kantorovich operators with generalized kernels, Analysis and Applications, 2019, doi:10.1142/S0219530519500155.
[17] D. Costarelli, A.M. Minotti, G. Vinti, Approximation of discontinuous signals by sampling Kantorovich series, J. Math. Anal. Appl. 450 (2) (2017) 1083–1103.
[18] D. Costarelli, M. Seracini, G. Vinti, A segmentation procedure of the aorta artery from CT images without contrast medium, Math. Methods Appl. Sci. 43 (2020) 114–133, doi:10.1002/1099-1476.
[19] D. Costarelli, G. Vinti, Approximation by multivariate generalized sampling Kantorovich operators in the setting of Orlicz spaces, Bollettino U.M.I. 9 (IV) (2011) 445–468.
[20] D. Costarelli, G. Vinti, Degree of approximation for nonlinear multivariate sampling Kantorovich operators on some functions spaces, Numer. Funct. Anal. Optim. 36 (8) (2015) 964–990.
[21] D. Costarelli, G. Vinti, Approximation by max-product neural network operators of Kantorovich type, Results Math. 69 (3) (2016) 505–519.
[22] D. Costarelli, G. Vinti, Max-product neural network and quasi-interpolation operators activated by sigmoidal functions, J. Approx. Theory 209 (2016) 1–22.
[23] D. Costarelli, G. Vinti, An inverse result of approximation by sampling Kantorovich series, Proc. Edinb. Math. Soc. 62 (1) (2019) 265–280.
[24] D. Costarelli, G. Vinti, Inverse results of approximation and the saturation order for the sampling Kantorovich series, J. Approx. Theory 242 (2019) 64–82.
[25] G. Fix, G. Strang, Fourier analysis of the finite element method in Ritz–Galerkin theory, Stud. Appl. Math. 48 (1969) 268–273.
[26] R.G. Keys, Cubic convolution interpolation for digital image processing, IEEE Trans. Acoust. Speech Signal Process. 29 (6) (1981).

[27] Y.S. Kolomoitsev, A. Krivoshein, M.A. Skopina, Differential and falsified sampling expansions, J. Fourier Anal. Appl. (2017), doi:10.1007/s00041-017-9559-1.
[28] Y.S. Kolomoitsev, M.A. Skopina, Approximation by multivariate Kantorovich–Kotelnikov operators, J. Math. Anal. Appl. 456 (1) (2017) 195–213.
[29] A. Krivoshein, M.A. Skopina, Multivariate sampling-type approximation, Anal. Appl. 15 (4) (2017) 521–542.
[30] S.M. Mousavi, A. Naghsh, A.S.R. Abu-Bakar, Watermarking techniques used in medical images: a survey, J. Digit. Imaging 27 (2014) 714–729.
[31] O. Orlova, G. Tamberg, On approximation properties of generalized Kantorovich-type sampling operators, J. Approx. Theory 201 (2016) 73–86.
[32] J. A. Parker, R.V. Kenyon, D.E. Troxel, Comparison of interpolating methods for image resampling, IEEE Trans. Med. Imaging 2 (1) (1983).
[33] M.A. Skopina, Band-limited scaling and wavelet expansions, Appl. Comput. Harmon Anal. 36 (1) (2014) 143–157.
[34] M.A. Unser, Ten Good Reasons for Using Spline Wavelets, in: Optical Science, Engineering and Instrumentation'97, in: International Society for Optics and Photonics, 1997, pp. 422–431.
[35] G. Vinti, L. Zampogni, A general approximation approach for the simultaneous treatment of integral and discrete operators, Adv. Nonlinear Stud. 18 (4) (2018) 705–724.
[36] L. Zhang, W. Wu, An edge-guided image interpolation algorithm via directional filtering and data fusion, IEEE Trans. Image Proc. 15 (8) (2006) 2226–2238.
[37] D. Costarelli, G. Vinti, Saturation by the Fourier transform method for the sampling Kantorovich series based on bandlimited kernels, Anal. Math. Phys. 9 (2019) 2263–2280.