Least-Squares Image Resizing Using Finite Differences

Arrate Muñoz, *Student Member, IEEE*, Thierry Blu, *Member, IEEE*, and Michael Unser, *Fellow, IEEE*

***Abstract*—We present an optimal spline-based algorithm for the enlargement or reduction of digital images with arbitrary (noninteger) scaling factors. This projection-based approach can be realized thanks to a new finite difference method that allows the computation of inner products with analysis functions that are B-splines of any degree n. A noteworthy property of the algorithm is that the computational complexity per pixel does not depend on the scaling factor a. For a given choice of basis functions, the results of our method are consistently better than those of the standard interpolation procedure; the present scheme achieves a reduction of artifacts such as aliasing and blocking and a significant improvement of the signal-to-noise ratio. The method can be generalized to include other classes of piecewise polynomial functions, expressed as linear combinations of B-splines and their derivatives.**

***Index Terms*—Affine transform, boundary conditions, finite difference, interpolation, least-squares, oblique projection, scale, spline.**

I. Introduction

IMAGE resizing (magnification or reduction) is a common operation in image processing [1]. It is used whenever one wants to change the image resolution. For example, it is required on a routine basis in digital photography, multimedia, and electronic publishing [2], [3], for adapting the pixel size to the resolution of an output device (printer or monitor) [4], [5], and for generating preview images, or posting digital pictures on the Web.

Another important area of applications is medical imaging; typical instances are as follows.

- Reslicing for resolution normalization. This is to compensate for the fact that three-dimensiona (3-D) volumetric data (CT, SPECT or MRI) are often acquired in a nonisotropic fashion—the within-slice resolution is typically finer than the across-slice resolution [6].
- Image zooming. It is often used to focus on details for diagnostic purposes.
- Image pyramids for multi-scale processing. Many iterative image processing algorithms can be applied in a coarse-to-fine fashion. Working with smaller images reduces the computation time and also tends to improve robustness [7].

Many linear resizing techniques are available even though they have some limitations. The standard ones rely on interpolation [8]. The simplest methods are nearest-neighbor and bilinear interpolation, which correspond to fitting the image with a spline of degree 0 and 1, respectively. The piecewise constant model generates noticeable blocking artifacts, while the (bi-)linear one tends to lose details through image blurring. Better interpolation performance is achieved by switching to higher order models [8]; typical examples are Keys' short kernel convolution [9], or higher order spline interpolation which offers a better cost-performance ratio [10]–[12]. While interpolation works well for image magnification, it is not entirely suitable for image reduction because of potential aliasing problems. The standard remedy is to apply some kind of lowpass prefiltering prior to resampling. Although a complete suppression of aliasing is possible through the application of Shannon's ideal filter, this is not a widely used technique—it is computationally expensive and tends to introduce ringing artifacts (Gibbs oscillations).

The principal limitation of interpolation approaches is that they are not designed to minimize information loss. It therefore makes good sense to investigate the possibility of obtaining the best solution in the least-squares sense [13]. Indeed, the signal-to-noise ratio (SNR) is a standard figure of merit used in image processing. Even though it is widely used in the field, it has its limitation because it does not take into account the subjective aspects of visual perception [14]. However, it has the advantage of being easy to measure and amenable to optimization.

The least-squares solution is achieved by modifying the interpolation approach so that the resampling step gets replaced by the evaluation of inner products with the translates of a suitable analysis function $\tilde{\varphi}$. This computation is equivalent to applying a continuously-defined prefilter (antialias) to the interpolated function prior to resampling—the prefilter is not necessarily ideal but is chosen to be biorthogonal to the underlying interpolation kernel. Note that the method is conceptually similar to a wavelet decomposition [15]–[17], except that the scale factor is not restricted to be a power of two. While the basic principle of this projection method was introduced in [13], an exact least-squares implementation was only demonstrated for splines of degree 0 and 1. The practical limitation was the difficulty to perform an exact numerical implementation of the optimal prefilter for higher order splines. Lee *et al.* developed a higher order spline resizing algorithm by replacing the orthogonal projection of [13] by an oblique one [18]. They simplified

Manuscript received February 13, 2000; revised May 4, 2001. This work was supported by the Swiss National Science Foundation under Grant 2100-053540. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Brian L. Evans.

The authors are with Biomedical Imaging Group, DMT/IOA BM, Swiss Federal Institute of Technology Lausanne (EPFL), CH-1015 Lausanne EPFL, Switzerland (e-mail: arrate.munoz@epfl.ch; thierry.blu@epfl.ch; michael.unser@epfl.ch).

Publisher Item Identifier S 1057-7149(01)07465-6.

1057–7149/01$10.00 © 2001 IEEE

the procedure by replacing the optimal prefilter by a box function analysis (B-spline of degree 0) and made it more efficient by pre-computing the antiderivative of the function to be approximated.

Here, we present a generalization of this method that allows us to compute both oblique and orthogonal projections (least-squares approximations) for splines of any degree n. What makes the approach feasible in this more general setting is the new finite difference method presented in Section III; it allows an *exact* computation of the required for analysis functions that are B-splines of any degree n. The method works for both reduction and magnification of images with an arbitrary scaling factor and for any translation value. In Section V, we generalize the method to a whole class of piecewise polynomial functions (all linear combinations of B-splines), including some with optimal approximation properties. Another benefit of this work is that it leads us to the definition of a new operator formalism for splines and multi-rate signal processing (cf. Sections III and IV); thanks to these tools, we are able to simplify the derivation of the key formulas and we provide a presentation that is (hopefully) understandable and self-contained.

II. Philosophy of the Approach

When the image is represented using separable basis functions, the resizing problem can be solved optimally in a separable fashion. Consequently, the complexity is reduced from 2-D to 1-D.

In order to simplify the description of our algorithm, it is advantageous to use a continuous signal processing representation of operators defined in the continuous domain (scale change and shift). For that reason, we introduce the *digital-to-analog operator* which maps discrete sequences s_k to the distribution

$$s_k \mapsto s_\delta(x) = \sum_k s_k \delta(x-k)$$

where δ is the Dirac's delta distribution. Using this formulation, a digital filter g_k is represented by its equivalent continuous-space impulse response $g_\delta(x) = \sum_k g_k \delta(x-k)$.

The schematic continuous-space domain representation of the whole algorithm is given in Fig. 2. All boxes denote convolutions; g_δ, q_δ and h_δ are digital filters, while φ and φ_1 are true functions of the continuous variable x. The affine transformation $s(x/a+b)$ is represented via the combination of a shift [convolution with $\delta(x+b)$], and of a resizing $s(x) \rightarrow s(x/a)$ represented as —(%)—. We now describe the four main steps of the method.

A. Interpolation

The first step is to take the discrete input data s_k and to construct a continuous interpolating model $s(x) = \sum_k c_k \varphi(x-k)$, where the $\varphi(x-k)$'s are some specified basis functions. For this purpose, we take the samples s_k and convolve them with an appropriate prefilter g_k to get the coefficients $c_k = g_k * s_k$. The continuous-time function is obtained by convolving $c_\delta(x)$ with $\varphi(x)$. The prefilter $G(z) = \sum_k g_k z^{-k} = 1/(\sum_k \varphi_k z^{-k})$ is the convolution inverse of the sequence $\varphi(k)$. This particular choice ensures that the interpolation requirement is satisfied; i.e., $s(x)|_{x=k} = s_k$.

B. Affine Transformation (Conceptual Step)

We apply the affine transformation (scaling and shifting) to the function $s(x)$

$$f(x) = s\left(\frac{x}{a} + b\right).$$

The image is enlarged if $a > 1$ and shrunk if $a < 1$.

C. Projection-Based Signal Approximation

In the standard interpolation approach, the image gets resized by resampling $f(x)$ at the integers [19], as illustrated in Fig. 1. Here, we will consider an alternative approach in terms of approximation theory. Specifically, we find the best approximation $\tilde{f}(x) \in V_{\varphi_2}$ of $f(x)$ in some space $V_{\varphi_2} = \mathrm{span}_k\{\varphi_2(x-k)\}$ such that the L_2-approximation error $\epsilon_a(f) = \|f - \tilde{f}\|_{L_2}$ is minimized. From the Projection Theorem, we know that the least-squares solution to this problem is the orthogonal projection of $f(x)$ onto V_{φ_2} [20]

$$\tilde{f}(x) = Pf(x) = \sum_k c_2(k)\varphi_2(x-k) = c_{2,\delta} * \varphi_2(x)$$

with $c_2(k) = \langle f, \mathring{\varphi}_2(x-k)\rangle$, where $\mathring{\varphi}_2(x)$ is the dual of the analysis function $\varphi_2(x)$; in other words, $\mathring{\varphi}_2$ satisfies $\mathring{\varphi}_2 \in V_{\varphi_2}$ and $\langle \mathring{\varphi}_2(x), \varphi_2(x-k)\rangle = \delta_k$.

Rather than computing the inner product $\langle f, \mathring{\varphi}_2(x-k)\rangle$, we consider a slightly more general and also more flexible approach via the block diagram in Fig. 2. It corresponds to an *oblique* projection onto V_{φ_2} It uses an auxiliary analysis function $\varphi_1(x)$ which is essentially arbitrary.

First, we compute the inner products

$$c_1(k) = \langle f(x), \varphi_1(x-k)\rangle \tag{1}$$

which is equivalent to prefiltering $f(x)$ with $\varphi_1(-x)$ and sampling thereafter.

The cross-correlation sequence of $\varphi_1(x)$ and $\varphi_2(x)$ is given by $a_{12}(k) = \langle \varphi_1(x), \varphi_2(x-k)\rangle$. If $a_{12}(k) \neq \delta_k$, the projection of $f(x)$ onto V_{φ_2} perpendicular to V_{φ_1} requires an additional digital filtering correction q to satisfy the biorthogonality condition [21]. Thus, $c_2(k) = c_1(k) * q(k)$. The appropriate correction filter q is the convolution inverse of a_{12}: $q = a_{12}^{-1} \leftrightarrow 1/(\sum_k a_{12}(k) z^{-k})$. This is equivalent to using the analysis function $\tilde{\varphi}_2(x) = \sum_k q_k * \varphi_1(x-k) = q_\delta * \varphi_1(x)$, where $\varphi_2(x)$ and $\tilde{\varphi}_2(x)$ are biorthonormal. If $\varphi_1(x) \in V_{\varphi_2}$, then we get the orthogonal projection; otherwise, we have an oblique projection [21].

When computing the orthogonal projection, we obtain a resized image with minimum loss of information in the least-squares sense. If instead, we choose an oblique projection, the approximation is only slightly suboptimal, depending on the angle between $V_{\varphi_1} = \mathrm{span}_k\{\varphi_1(x-k)\}$ and $V_{\varphi_2} = \mathrm{span}_k\{\varphi_2(x-k)\}$ [21]. Moreover, the rate of convergence depends on the approximation order properties of the

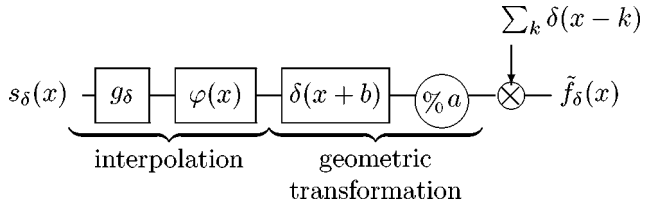

Fig. 1. General scheme for the standard interpolation approach.

synthesis function alone; the analysis function has essentially no influence on the asymptotic approximation error [22].

D. Resampling of the Projection at the Integers

Finally, we have to resample the projection at the integers ($\tilde{f}(l) = \tilde{f}(x)|_{x=l}$) to get the output of the system $\tilde{f}_\delta(x) = \sum_l \tilde{f}(l)\delta(x-l)$. This is achieved by postfiltering with $h_k = \varphi_2(k)$, the sampled version of the synthesis function.

If we compare the block diagram in Figs. 1 and 2, we see that the standard interpolation approach corresponds to the simplified situation where $\varphi_1(x) = \delta(x)$ and $q_k * h_k = \delta_k$. We also see that the main difficulty with our new approach is the computation of the inner products $\langle f(x), \varphi_1(x-k)\rangle$ involving continuously-defined functions that are specified on different grids.

III. Splines and Related Notions

Before describing our spline resizing algorithm, we introduce a new operator formalism for splines which will facilitate the derivation of our method. In this section we assume that all signals $f(x)$ and discrete sequences s_k are compactly supported.

A. Continuous Differential Operators

The conventional derivative operator is

$$Df(x) = \frac{df(x)}{dx}.$$

The unique inverse of D is the antiderivative operator

$$D^{-1}f(x) = \int_{-\infty}^{x} f(\tau)\,d\tau$$

i.e., $DD^{-1}f(x) = D^{-1}Df(x) = f(x)$.

The operator D^{-1} also corresponds to the convolution of $f(x)$ with the unit step function. The $(n+1)$-fold convolution of the step function yields the one-sided power function $x_+^n/n!$ where

$$x_+^n = \begin{cases} x^n, & \text{if } x \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

In particular, the unit step function is x_+^0. We may also write the $(n+1)$-fold integral operator as

$$D^{-(n+1)}f(x) = \frac{x_+^n}{n!} * f(x).$$

Note that the composition rule $D^{-(n_1+1)}D^{-(n_2+1)} = D^{-(n_1+n_2+2)}$ corresponds to the following composition property of one-sided power functions

$$\frac{x_+^{n_1}}{n_1!} * \frac{x_+^{n_2}}{n_2!} = \frac{x_+^{n_1+n_2+1}}{(n_1+n_2+1)!}. \tag{2}$$

B. Discrete Differential Operators

We define the backward finite difference operator as

$$\Delta = \delta(x) - \delta(x-1).$$

This is also a discrete convolution operator (digital filter) whose z-transform is

$$\Delta(z) = 1 - z^{-1}.$$

When working with ℓ_1-sequences, we can consider the inverse operator Δ^{-1} defined uniquely as

$$\Delta^{-1} = \sum_{n\geq 0} \delta(x-n)$$

whose z-transform is

$$\Delta^{-1}(z) = (1-z^{-1})^{-1}.$$

It can be defined as the running sum filter

$$(\Delta^{-1} * s)_k = \sum_{n\leq k} s_n$$

we thus have $\Delta^{-1} * \Delta * s = \Delta * \Delta^{-1} * s = s$.

In our implementation, we will extend the application of Δ^{-1} to periodic sequences with zero average. In general, Δ^{-1} does not preserve the zero-mean property of the input. In Section IV-C, we will however show how to overcome this difficulty. We will also show that Δ^{-1} exchanges the standard symmetric and antisymmetric boundary conditions. Note that $y_k = \Delta^{-1} * s_k$ can be implemented very efficiently using the recursive equation

$$y_k = y_{k-1} + s_k. \tag{3}$$

C. B-Splines

The purest form of a polynomial spline of degree n is the one-sided power function x_+^n which has a unique singularity of order n at the origin. While a polynomial spline can always be written as a sum of shifted one-sided power functions, it is more convenient to work with B-splines as basis functions [12]; these are obtained through the following finite difference process:

$$\beta^n(x) = \Delta^{n+1} * \frac{x_+^n}{n!} * \delta\left(x + \frac{n+1}{2}\right). \tag{4}$$

Note that the shift by $(n+1)/2$ recenters the finite difference operation so that the result is a centered B-spline. Since a convolution with $x_+^n/n!$ is equivalent to the $(n+1)$-fold integral $D^{-(n+1)}$, we can rewrite the B-spline as

$$\beta^n(x) = \Delta^{n+1} * D^{-(n+1)} * \delta\left(x + \frac{n+1}{2}\right). \tag{5}$$

The centered B-spline of degree n has the following remarkable properties:

- positivity: $\beta^n(x) \geq 0$;
- compact support: $[-(n+1)/2, (n+1)/2)$;
- symmetry: $\beta(x) = \beta(-x)$;
- partition of unity: $\sum_k \beta^n(x-k) = 1$.

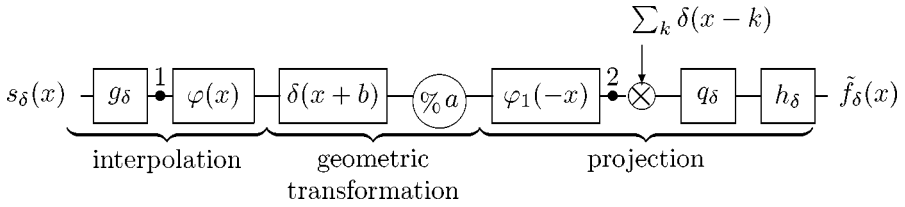

Fig. 2. General scheme for the proposed projection method.

The B-spline relations that are especially relevant for this paper are

- n_1th derivative

$$D^{n_1}\beta^n(x) = \Delta^{n_1} * \beta^{n-n_1}\left(x+\frac{n_1}{2}\right) \tag{6}$$

- n_1-fold integral

$$D^{-n_1}\beta^n(x) = \Delta^{-n_1} * \beta^{n+n_1}\left(x-\frac{n_1}{2}\right) \tag{7}$$

with $n_1 \le n$ in both cases. These can all be derived rather easily using the previously defined operator formalism as shown in the Appendix.

D. B-Spline Inner Products

All the steps in the method described in Section II are rather standard and can be performed using digital filtering, except for the computation of the inner product for an arbitrary resizing factor a (not assumed to be a power of two as in the case of wavelets). The basis for our method is the following formula for computing B-spline inner products and follows from the definition of the B-spline (4)

$$\begin{aligned}\langle f(x), \beta^{n_1}(x-y)\rangle &= f * \beta^{n_1}(y)\\ &= \Delta^{n_1+1} * \left[D^{-(n_1+1)} f\left(y+\frac{n_1+1}{2}\right)\right].\end{aligned} \tag{8}$$

Thus, we can compute the inner products rather simply by applying finite differences to the (n_1+1)-fold integral of f. What makes an exact computation possible and tractable analytically is the fact that the (n_1+1)-fold integral of a spline is a spline with a corresponding increase of the degree. Specifically, if $s(x) = \sum_k c_k \beta^n(x-k)$ where the sum is finite

$$\begin{aligned}D^{-(n_1+1)}s(x) = \sum_k &\left(\Delta^{-(n_1+1)} * c\right)_k \beta^{n+n_1+1}\\ &\cdot\left(x-k-\frac{n_1+1}{2}\right).\end{aligned}$$

IV. Spline Resizing Algorithm

Our reason for using B-splines—or some close relatives—is that these are functions for which we know how to compute the required inner products. They also have excellent approximation properties [23], [24]. Moreover, they have the shortest support for a given approximation order, which means that the computational complexity is minimized.

A. Derivation of the Algorithm

Thus, we choose our basis functions to be B-splines. In that case, our algorithm has the following parameters: $\varphi(x) = \beta^n(x)$, $\varphi_1(x) = \beta^{n_1}(x)$ and $\varphi_2(x) = \beta^{n_2}(x)$. This implies that $a_{12}(k) = q^{-1}(k) = \beta^{n_1+n_2+1}(x)|_{x=k}$ and $\phi_2(x) = q_\delta * \beta^{n_2}(x)$.

In the sequel, we will derive our final form of the resizing algorithm graphically by using the exchange rules for the one-sided power functions and for the shift given in Figs. 4 and 5, together with the convolution rule for one-sided power functions. The proof of the exchange rule for the one-sided power functions is as follows.

Proof: We can write the expression on the right handside in Fig. 4 as

$$\int f\left(\frac{\tau}{a}\right)\frac{1}{a^{n+1}}(x-\tau)_+^n\,d\tau.$$

We then make the change of variable $u = \tau/a$

$$\int f(u)\frac{1}{a^{n+1}}(x-au)_+^n a\,du = \int f(u)\left(\frac{x}{a}-u\right)_+^n du$$

which corresponds to the expression on the left handside. ■

The proof for the exchange rule of a shift by b is trivial.

We now proceed by successive modifications of the block diagram in Fig. 2. We have extracted the operators between the marks 1 and 2 for simplicity. The final result is shown in Fig. 3(e).

In Fig. 3(a), $\varphi(x)$ and $\varphi_1(-x)$ are substituted by their explicit expression using (4). Using the rules in Figs. 4 and 5, the boxes are reorganized in such a way that all one-sided power functions and shifts are moved to the left side of the scale change $-\!\!\!

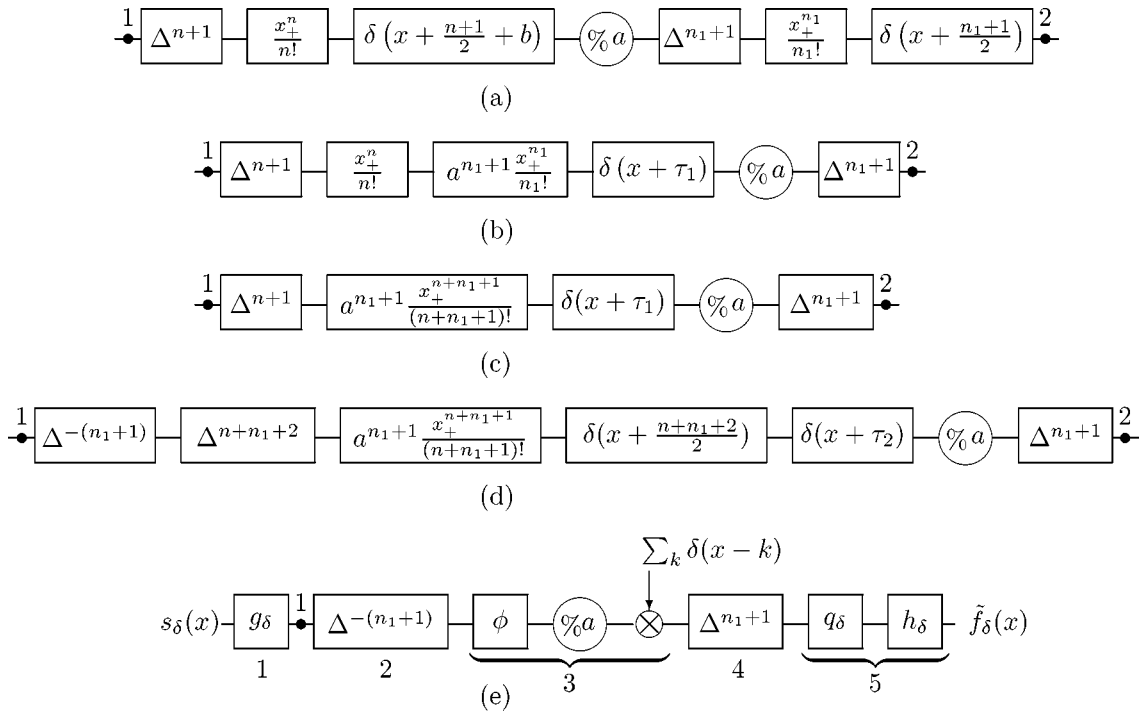

Fig. 3. Diagram that shows the full process to get our algorithm. (a) Substitution of $\beta^n(x)$ and $\beta^{n_1}(-x)$ by their explicit time expression. (b) Scheme obtained using the exchange rule for $x_+^n/n!$ and $\delta(x+b)$, with $\tau_1 = ((n+1)/2) + ((n_1+1)/2a) + b$. (c) Application of convolution rule of $x_+^n/n!$. (d) Application of $\Delta^{n_1+1} * \Delta^{-(n_1+1)} = I$, $\Delta^{n_1+1} * \Delta^{n+1} = \Delta^{n+n_1+2}$ and $\tau_2 = ((n_1+1)/2)(1/a-1) + b$. (e) Equivalent form of Fig. 1 with $\phi(x) = a^{n_1+1}\beta^{n+n_1+1}(x+\tau_2)$, $g = (b^n)^{-1}$, $q = a^{-1} = (b^{n_1+n_2+1})^{-1}$ and $h = b^{n_2}$. The numbers below the diagram indicate the main steps in the implementation.

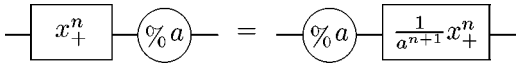

Fig. 4. Exchange rule for x_+^n.

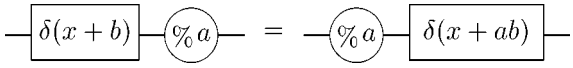

Fig. 5. Exchange rule for a shift by b.

from s_k (input samples). The filter is implemented recursively using a cascade of simple causal and anticausal operators as described in [25].

2) (n_1+1)-running sums corresponding to the operator $\Delta^{-(n_1+1)}$; these are computed recursively as well by iterating (3).
3) Geometric transformation and resampling using a spline interpolation model of degree $(n+n_1+1)$ [basis function $\phi(x)$].
4) (n_1+1)-centered finite differences, corresponding to the operator Δ^{n_1+1}.
5) Digital postfiltering with the sampled synthesis function $\phi_2(x)|_{x=k} = q_k * h_k$, where q is an IIR filter implemented using the recursive routines developed in [25].

The algorithm is now almost fully described. The only remaining issue is the extension to periodic signals and the consistent handling of boundary conditions. The main difficulty comes from step 2 which involves running sum filters which, in principle, are neither symmetric nor antisymmetric.

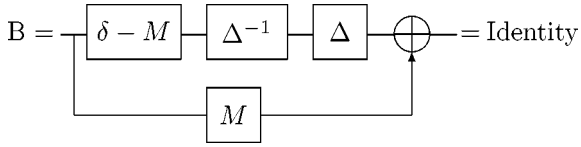

Fig. 6. Identity diagram for symmetric boundary input signals.

C. Boundary Conditions and Discrete Differential Operators

1) Signal Extensions: We will consider periodization and the two types of boundary conditions shown in Figs. 9 and 10.

• *Periodization:* It is easy to verify that the general projection-based resizing scheme shown in Fig. 2 has a meaning not only for *finite support* signals, but also for *periodic* signals. However, the implementation proposed in Fig. 3(e) is undefined for such inputs because of step 2, which attempts to compute an infinite sum of periodic data. A way to overcome this difficulty is to restrict ourselves to periodic signals that have a *zero mean*, i.e., such that $\sum_k s(x-k) = 0$, and to express the P-periodic signal $s(x)$ as $\lim_{m\to\infty} s_{[-mP,\,(m+1)P]}(x)$ where $s_{[-mP,\,(m+1)P]}(x)$ is the restriction of $s(x)$ to the support $[-mP,\,(m+1)P]$. The compactly supported signal $s_{[-mP,\,(m+1)P]}(x)$ can also be expressed by a convolution

$$s_{[-mP,\,(m+1)P]}(x) = s_{[0,\,P]}(x) * \underbrace{\sum_{|k|\le m} \delta(x-kP)}_{\Pi_m(x)}.$$

Then, since Δ^{-1} commutes with the periodization operator $\Pi_m(x)$ and, because $s_{[0,\,P]}(x)$ has zero mean, $\Delta^{-1} * s_{[0,\,P]}$ is also finitely supported within $[0,\,P]$ (more precisely: within $[0,\,P-1]$). This implies that Δ^{-1} keeps its meaning as m

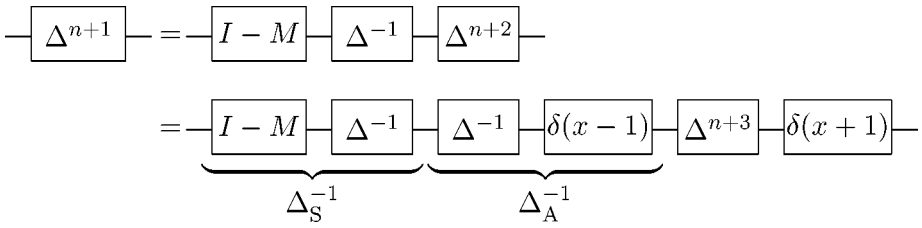

Fig. 7. Result from applying the identity given in Fig. 6 to a finite differences operator.

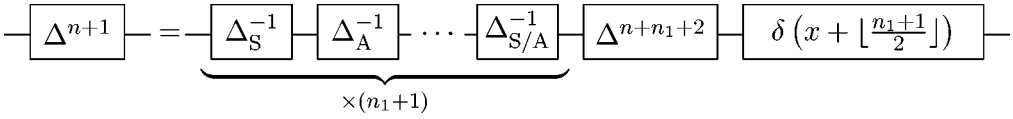

Fig. 8. Equivalence derived from Fig. 7.

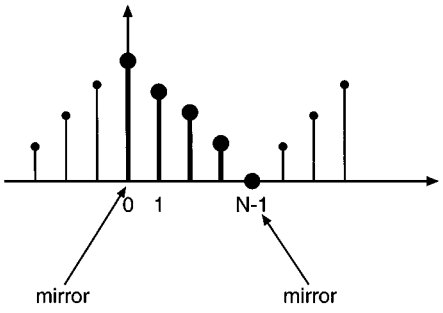

Fig. 9. Signal extended using symmetric boundary conditions.

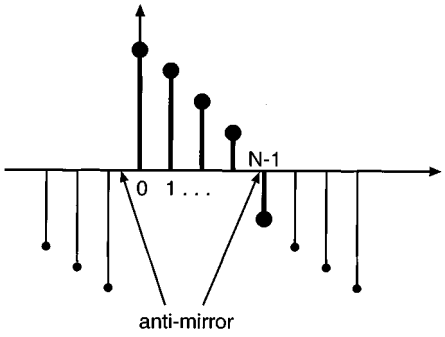

Fig. 10. Signal extended using antisymmetric boundary conditions.

tends to infinity. If $s(x) = \sum_k s_k \delta(x-k)$, where $s_k = s_{k+P}$ (periodicity) and $\sum_{k=0}^{P-1} s_k = 0$ (zero-mean requirement), then we have

$$(\Delta^{-1} * s)_k = \sum_{l=0}^{k \bmod P} s_l.$$

Since we have to apply the operator Δ^{-1} repeatedly, a process that does not preserve the zero-mean property, we will indicate in the next subsection how to enforce this property on any periodic input.

• *Symmetry:* To minimize boundary artifacts, we extend our signal $\{s_k\}_{k=0,\ldots,N-1}$ using symmetric mirror boundary conditions defined as $s_{-k} = s_k$, and $s_{N-1-k} = s_{N-1+k}$, for $k = 0, 1, \ldots, N-1$. This process is repeated on the newly extended signal, $\{s_k\}_{k=-N+1,\ldots,2N-3}$ and so further. As can readily be verified, this is equivalent to requiring that $s(-x) = s(x)$ and that $s(x)$ is $(2N-2)$-periodic. In other words, it is sufficient to specify what happens around the origin; the symmetry on the other end is propagated automatically through the periodization process.

• *Antisymmetry:* Another complementary technique interesting to us because it is satisfied by signals that appear naturally in the method consists in extending the signal using antisymmetric mirror boundary conditions. It is defined as $s_k = -s_{-k-1}$, and $s_{N-1+k} = -s_{N-2-k}$, for $k = 0, 1, \ldots, N-1$, repeated on the further extensions of the signal. As can readily be verified, this is equivalent to requiring that $s(-x) = -s(x-1)$ and that $s(x)$ is $(2N-2)$-periodic. Note however that, unlike the symmetric extension, this one cannot be applied to arbitrary signals, as it requires that $s_{N-1} = -s_{N-2}$. Actually, an antisymmetric signal is always zero mean. Once again, it is sufficient to specify the antisymmetry around the origin; the antisymmetry on the other end is propagated automatically through the periodization process.

2) Propagation of the Boundary Conditions: Any shift-invariant operator preserves the $(2N-2)$ periodicity, but not necessarily the symmetry. We therefore need to investigate how Δ and Δ^{-1} propagate symmetric and antisymmetric boundary conditions. We need also to correct for the fact that the considered periodic signals are not necessarily zero mean.

The finite differences operator Δ inverses symmetry. Specifically, it transforms antisymmetric into symmetric boundary conditions and symmetric into *shifted* antisymmetric boundary conditions. The following theorem claims that Δ^{-1} has a similar behavior.

Theorem 1: The operator Δ^{-1} transforms symmetries according to

• *Symmetric Input:* If $s(x) = s(-x)$ and s has a zero mean, then $u = \Delta^{-1} * s$ satisfies $u(-x) = -u(x-1)$. Thus, if s satisfies symmetric boundary conditions, then $\Delta^{-1} * s$ satisfies antisymmetric boundary conditions.

• *Antisymmetric Input:* If $s(-x) = -s(x-1)$ and s has a zero mean, then $u = \Delta^{-1} * s$ satisfies $u(-x) = u(x-2)$. Thus, if s satisfies antisymmetric boundary conditions, then $\delta(x-1) * \Delta^{-1} * s$ satisfies symmetric boundary conditions.

Proof: It is sufficient to prove the property for a finitely supported signal $s(x)$ that satisfies the zero-mean property

$\sum_k s(x-k)=0$, because of the definition of Δ^{-1} for periodic signals.

- *Symmetric Input:* We have

$$\begin{aligned} u(-x) &= \sum_{k\geq 0} s(-x-k) && \text{definition of } \Delta^{-1} \\ &= \sum_{k\geq 0} s(x+k) && \text{symmetry property} \\ &= -\sum_{k\leq -1} s(x+k) && \text{zero-mean property} \\ &= -u(x-1). \end{aligned}$$

- *Antisymmetric Input:* We have

$$\begin{aligned} u(-x) &= \sum_{k\geq 0} s(-x-k) && \text{definition of } \Delta^{-1} \\ &= \sum_{k\geq 0} -s(x+k-1) && \text{antisymmetry property} \\ &= \sum_{k\leq -1} s(x+k-1) && \text{zero-mean property} \\ &= u(x-2). \end{aligned}$$

■

We have defined our inverse finite differences operator for finitely supported signals, or zero mean periodic signals. We will now show how to deal with non zero mean P-periodic signals in our algorithm. Let us define the moving average filter

$$M(x)=\frac{1}{P}\sum_{k=0}^{P-1}\delta(x-k).$$

We can then consider the identity block diagram in Fig. 6, which holds for symmetric boundary input signals. The key idea is that the output of $\boxed{\delta - M}$ is a signal of zero mean, while that of $\boxed{M}$ is a constant, whenever the input is P-periodic.

Since the finite differences operator kills constant signals, we can write the equivalence shown in Fig. 7 for an input with symmetric boundary conditions. Furthermore, in order to implement the boundary conditions as specified in Theorem 1, it is necessary to define a "symmetric" version of Δ^{-1}, Δ_{S}^{-1}, for symmetric inputs and an "antisymmetric" version, Δ_{A}^{-1}, for antisymmetric inputs. Note that Δ_{A}^{-1} is simply Δ^{-1} (delayed by one sample) because an antisymmetric input is of zero mean. That is why, the filter $\boxed{\delta - M}$ disappear at this stage. This is iterated (n_1+1)-times to yield the equivalence in Fig. 8. Note that the alternation between symmetric and antisymmetric boundary conditions adds a delay of $\lfloor (n_1+1)/2 \rfloor$.

Thus, in practice, we will modify the block diagram in Fig. 3(e) to use an alternation of Δ_{S}^{-1} and Δ_{A}^{-1} instead of $\Delta^{-(n_1+1)}$, adding the appropriate delay. This ensures that the boundary conditions are correctly propagated throughout. This modification is necessary for the behavior of the algorithm to be fully consistent; in particular, this ensures that for n odd and for an integer scaling (including $a=1$) the method is fully reversible with no boundary artifacts.

V. Generalization of the Method

We now show that the projection method can also be implemented exactly for a more general class of piecewise polynomial functions.

A. Linear Combinations of Shifted B-Splines

We consider the case where the basis functions are linear combinations of shifted splines

$$\psi(x)=\sum_i \alpha_i \beta^n(x+h_i) \longleftrightarrow \hat{\psi}(\omega)=F(\omega)\hat{\beta}^n(\omega) \qquad (9)$$

with $F(\omega)=\sum_i \alpha_i e^{j\omega h_i}$. These functions are piecewise polynomial. However, they are not necessarily splines and the knots are not necessarily uniformly spaced. Then, the functions $\varphi(x)$, $\varphi_1(x)$ and $\varphi_2(x)$ used in our algorithm depicted in Fig. 2, are defined by $(\alpha_i, h_i)=(\alpha_{0,i}, h_{0,i})$, $(\alpha_{1,i}, h_{1,i})$ and $(\alpha_{2,i}, h_{2,i})$, respectively; moreover, we have $a_{12}(k)=q_k^{-1}=\sum_{i,j}\alpha_{1,i}\alpha_{2,j}\beta^{n_1+n_2+1}(k+h_{1,i}-h_{2,j})$.

If we follow the same process as in Section IV-A, we end up with a diagram similar to Fig. 3(e) with a kernel that is now

$$\phi(x)=a^{n_1+1}\sum_{i,j}\alpha_{0,i}\alpha_{1,j}\beta^{n+n_1+1}(x+\tau_{i,j})$$

where $\tau_{i,j}=h_{0,i}-(h_{1,j}/a)+b+((n_1+1)/2)(1/a-1)$.

B. Linear Combination of B-Spline Derivatives

A generating function $\varphi(x)$ is said to be of order L if the approximation error at step a decays like a^L as a tends to zero. Specifically, from approximation theory [22], we have

$$\|f-f_a\|_{L^2}=C_\varphi^-\|f^L\|_{L^2}a^L+o(a^L)$$

where

f_a	approximation of f at step a;
C_φ^-	some constant that depends on φ only;
$\|f^L\|_{L^2}$	norm of the Lth derivative of f.

In the case of cubic B-splines, $C_\varphi^-=(240\sqrt{21})^{-1}$.

The necessary and sufficient condition for achieving this rate of decay is the reproduction of polynomials of degree $n=L-1$: $\{1, x, x^2, \ldots, x^{L-1}\}\in \mathrm{span}\{\varphi(x-k)\}_{k\in\mathbb{Z}}$ (Strang–Fix conditions) [26]. The quality of the approximation of $f(x)$ depends strongly on the order L of the interpolator and not so much on N_φ (the size of the support). Nevertheless, N_φ determines the computational cost.

We showed in [27], [28] that the functions that minimize the support N_φ for a given order L are linear combinations of B-spline derivatives

$$\varphi(x)=\sum_{k=0}^{L-1}\gamma_k\frac{d^k}{dx^k}\beta^{L-k}(x) \qquad (10)$$

which specifies the maximum order minimal support (MOMS) class of functions.

In particular, the B-splines of degree n are the smoothest functions for a given order of approximation ($L=n+1$).

We get a simple expression for $\varphi(x)$ in terms of one-sided power functions by using the relation between splines of different degrees

$$\varphi(x) = \sum_{i=0}^{n} \gamma_i D^i \beta^n(x) = \sum_{i=0}^{n} \gamma_i \Delta^i * \beta^{n-i}\left(x + \frac{i}{2}\right)$$
$$= \Delta^{n+1} * \sum_{i=0}^{n} \gamma_i \frac{x_+^{n-i}}{(n-i)!} * \delta\left(x + \frac{n+1}{2}\right).$$

We now select $\varphi(x)$, $\varphi_1(x)$ and $\varphi_2(x)$ to be linear combinations of B-spline derivatives of degree n, n_1 and n_2, respectively, with coefficients $\gamma_{0,i}$, $\gamma_{1,i}$ and $\gamma_{2,i}$. With this particular choice, we get

$$a_{12}(k) = \langle \varphi_1(x), \varphi_2(x-k) \rangle$$
$$= \sum_{j=0}^{n_1} \gamma_{1,j} \sum_{i=0}^{n_2} \gamma_{2,i} \Delta^{i+j} * \beta^{n_1+n_2-i-j+1}\left(k + \frac{i+j}{2}\right)$$

and the final scheme is the same as Fig. 3(e) with

$$\phi(x) = \sum_{i=0}^{n} \gamma_{0,i} \sum_{j=0}^{n_1} \gamma_{1,j} a^{n_1-j+1} \Delta^{i+j} * \beta^{n+n_1-i-j+1}(x + \tau_{i,j})$$

and $\tau_{i,j} = ((n_1+1)/2)(1/a - 1) + (i+j)/2 + b$.

Interestingly, the generalized scheme has the same computational cost as the B-spline algorithm. The basis functions are polynomials of the same degree as the corresponding splines, they have the same support and the recursive prefilters have the same degree.

Note that this particular setting also constitutes a limit case of the previous one. Specifically, we can approximate the derivative operator using finite differences and make the sampling step h tend to 0

$$\varphi(x) = \sum_{i=0}^{n} \gamma_i D^i \beta^n(x) = \lim_{h \to 0} \sum_{i=0}^{n} \gamma_i \frac{1}{h^i} \Delta_h^i * \beta^n(x)$$

$$\hat{\varphi}(\omega) = \left(\sum_{i=0}^{n} \gamma_i (j\omega)^i\right) \hat{\beta}^n(w) \simeq \sum_{i=0}^{n} \gamma_i \left(\frac{-1+e^{j\omega h}}{h}\right)^i \hat{\beta}^n(\omega)$$
$$= \sum_{k=0}^{n} \alpha_k e^{j\omega h_k} \hat{\beta}^n(\omega) = F(\omega) \hat{\beta}^n(\omega)$$

with $\alpha_k = \sum_{i \geq k} \binom{k}{i} \gamma_i (1/h^i)(-1)^{i-k}$ and $h_k = kh$.

Among the functions that minimize the support for a given order L, one interesting case are the O-MOMS where "O" stands for optimal. They are the functions belonging to the MOMS family (10) that minimize the approximation error constant C_φ^-. They can be determined recursively as indicated in [27] and [28]. The expression for $n = 3$ is

$$\varphi_4(x) = \beta^3(x) + \frac{1}{42} \frac{d^2}{dx^2} \beta^3(x).$$

If L is odd, $\varphi_L(x)$ is discontinuous; if L is even, $\varphi_L(x)$ is continuous but its derivative is no. So, they are at most continuous. The value of the constant for $\varphi_4(x)$ is $C_{\varphi_4}^- = 5040^{-1}$, so that we may expect the following asymptotic improvement over the cubic spline case: $(C_{\varphi_4}/C_{\beta^4}) = 1/\sqrt{21}$.

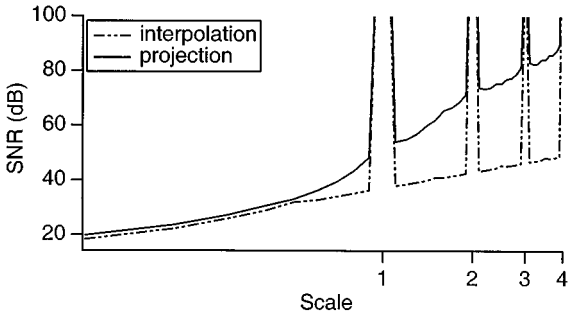

Fig. 11. Least-squares versus standard method for linear splines.

VI. Experimental Results

We used a series of back and forth experiments to evaluate and compare the various resizing algorithms. A test image—the MR scan (Fig. 17)—is scaled by a factor of a and then reset to its initial size using the reverse transformation (scaling by a factor of a^{-1}) with the same algorithm. The loss of information is measured by the relative mean square difference between the approximation and the initial digital image, expressed in decibels (dB). The experiment is repeated for many scaling factors and the peak signal-to-noise ratio (SNR) is represented as a function of the scale in a logarithmic plot. Scale factors smaller than 1 correspond to image reduction, while scale factors larger than 1 represent enlargement. Obviously, most information is lost in the reduction step, not in the enlargement one. Note, however, that magnification is not fully reversible unless the zooming factor is an integer.

A. Least-Squares versus Interpolation

Our first goal was to compare the performance of our projection algorithm with the more standard interpolation method that fits the image with a spline of the same degree and then resamples it at the required rate. The detailed results for $n = 1$ (linear splines) are given in Fig. 11. It is clear from this plot that the least-squares method outperforms the standard one (bilinear interpolation), even though the underlying model is the same in both cases. The visual improvement can be substantial, as illustrated in Figs. 19 and 20. We observe that the small-scale details are much better preserved with our optimal approach (see Figs. 19 and 20) and the contrast is enhanced because of the reduction of aliasing. Fig. 18 illustrates the reduction of blocking artifacts of the projection method with respect to the standard one.

Interestingly, the projection method also provides some improvement for image magnification. For $a > 1$, the gain is of the order of 20 dB. The distance between the two curves when $a > 1$ reflects the differences between the asymptotic orthogonal projection constant (which is small) and the interpolation one (which is larger) [29]. We also note that the error curve exhibits peaks at the integers, which simply reflects the fact that the signal is preserved exactly for integer zooming factors. In this particular case, the interpolation and projection methods are equivalent because the corresponding spline spaces are nested

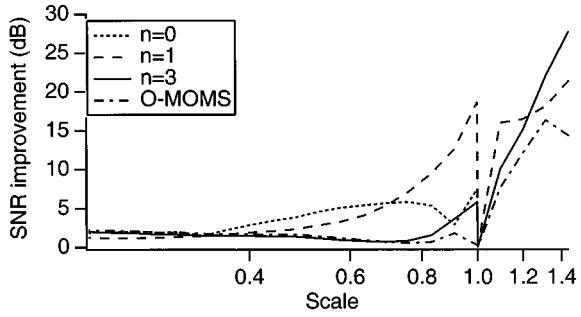

Fig. 12. Least-squares versus standard method for linear splines.

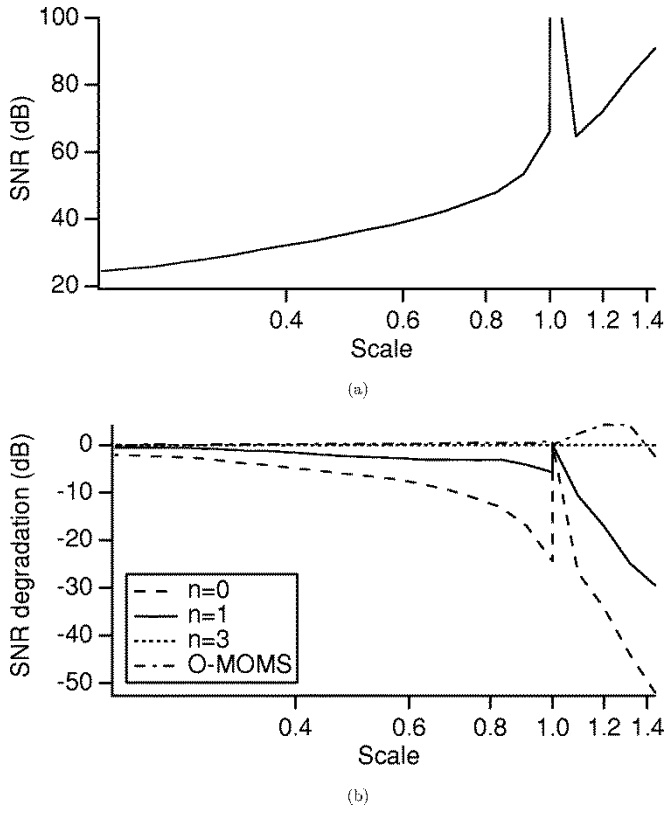

Fig. 13. SNR measures for the least-squares projection method: (a) cubic splines and (b) comparison of cubic splines with $n = 1, 0$ degree splines and O-MOMS.

(which implies that the projection error is zero). This is a property that holds for all B-splines of odd degrees, but not for the O-MOMS; for splines of even degrees, it is only true for odd magnification factors.

The superiority of the least-squares method is also apparent for the other interpolation models as shown in Fig. 12. This graph displays the relative SNR improvement of least-squares versus interpolation for splines of degree 0, 1, and 3, as well as the cubic O-MOMS. For small scaling factors ($a < 0.4$), the improvement is typically better than 2 dB, irrespective of the model used. The fundamental reason for the lesser performance of standard interpolation is aliasing. The effect is more pronounced for large reduction factors or when the image contains a lot of high frequency information.

Another visual example is provided in Fig. 16. Here, we observe a substantial improvement in the perceptual quality of the

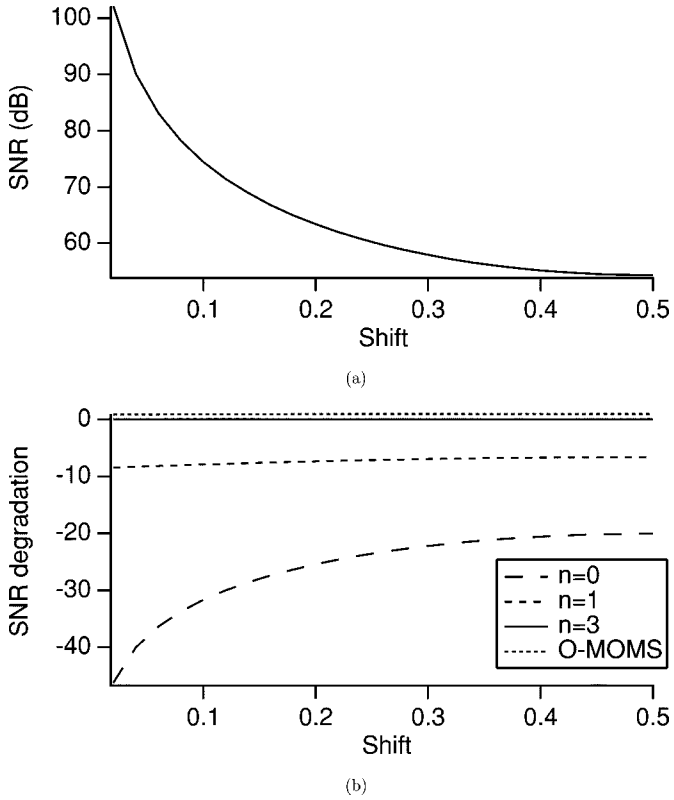

Fig. 14. SNR measures for the shift variations using the projection method. (a) Cubic splines and (b) comparison in the performance of different splines with the cubic ones.

projection method over the standard one in rescaling text. The interpolation model used was cubic.

B. Comparison of Basis Functions

Now that we have established the superiority of the projection method, it is interesting to compare the various basis functions. In particular, we are interested in evaluating the effect of the order parameter L. For this comparison, we use as our reference the least-squares method with cubic splines, which corresponds to the error graph in Fig. 13(a). The relative performance comparison of the various models is shown in Fig. 13(b). As expected, the SNR improves as the order of the spline increases. For small reduction factors ($a < 0.5$), cubic splines perform 1.0 dB better than linear splines, and 2.5 dB better than the piecewise constant model $(n = 0)$. For large scale factors, this difference gets magnified. If we now compare the O-MOMS and cubic B-splines, which have the same support $(W = 4)$ and the same order $(L = 4)$, we find that the former offer slightly better performance across all scales ($+0.15$ dB at small scales), which confirms their optimality.

We also compared the methods when the image is only shifted forward by a factor b and backward by the same factor without resizing. Fig. 14 shows the results. We observe that the O-MOMS give the best value in terms of SNR, 1 dB over cubic splines, while the linear splines (resp., piecewise constant) are 10 dB (resp., 25 dB) below the cubic ones. Thus, it appears that higher order correlates with improved shift-invariance, in accordance with the theoretical findings in [23].

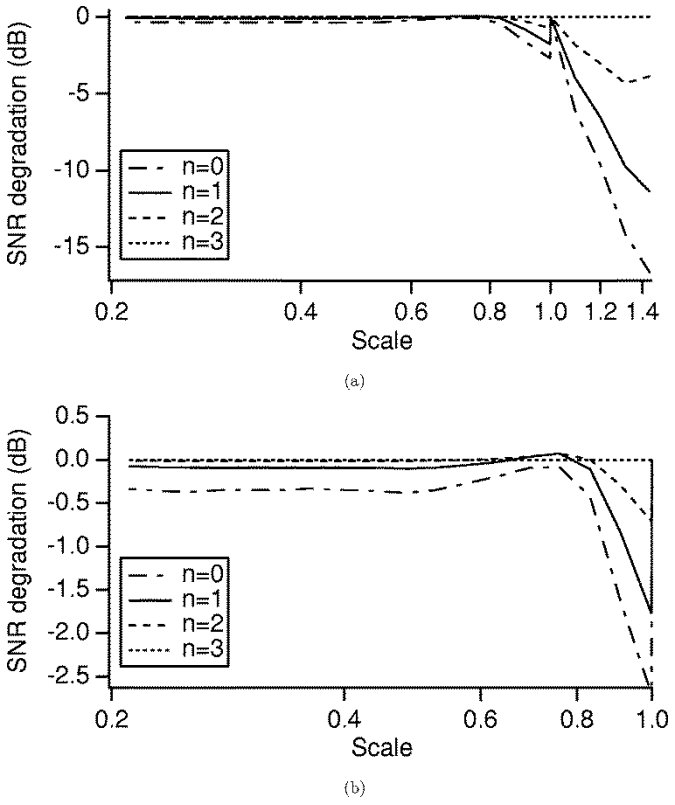

Fig. 15. Loss in performance by using oblique projection instead of least-squares. (a) Full scale range [0.2 : 1.4] and (b) reduced scale range [0.2 : 1.0] to magnify the difference at low scale factors.

Fig. 16. Resizing method using cubic splines applied to text with a scale factor 0.33; top—original text; bottom—reduced image: (left) using standard method and (right) using orthogonal projection.

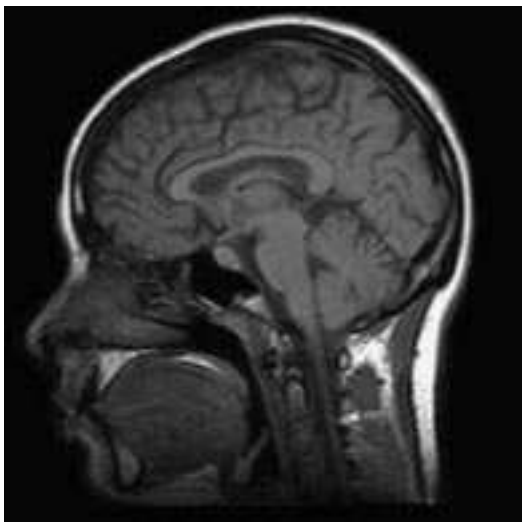

Fig. 17. Original magnetic resonance (MR) image.

C. Oblique versus Orthogonal Projection

When we pick the analysis degree n_1 different from n, our method implements an oblique projection instead of an orthogonal one. In Fig. 15, we see that such an oblique projection only brings a slight degradation of 0.4 dB when $n_1 = 0$ and 0.15 dB when $n_1 = 1$ compared to the orthogonal scheme, with the advantage of a lesser computational complexity [$O(n_1+n+1)$ instead of $O(2n+1)$]. These results are consistent with the theory developed in [22].

VII. Discussion

A. Relation to Previous Work

One of the main contributions of the paper is the development of a method based on finite differences for computing B-spline inner products. The key idea is that the integral of a spline is another spline of higher degree. We were able to formalize the approach by defining an inverse finite differences operator (running sum filter) and to use this tool to our advantage for computing multiple B-spline integrals. We note that the idea of precomputing an integral to facilitate the evaluation of inner products was introduced in [18]. However, the approach was restricted to the case of a box function which corresponds to the choice of $n_1 = 0$ in the present method.

The principle of least-squares image resizing was first proposed by one of us in [13]. The initial formulation of this algorithm did not use integrals, but rather an intermediate kernel function defined as the convolution of two B-splines of different width. In the original paper, the exact form of this kernel was only worked out for $n \leq 1$. More recently, we were able to obtain an explicit kernel formula [30].

While this new formula makes the original algorithm also applicable to splines of degree higher than 1, it is much less efficient computationally than the approach that we are proposing here. The essential difference is that the present approach has a complexity per computed output point that is independent of the scaling factor. In the original method, the complexity was proportional to the size of the convolution kernel $(n_1 + 1) + a^{-1}(n_2 + 1)$ and to the number of operations required to evaluate the spline kernel. In other words, the original method had a strong penalty for large reduction factors.

When the reduction factor is an integer, there exist alternative filtering/decimation techniques which are equivalent to the present algorithm (least-squares spline approximation) [31]; these are also very efficient computationally, but they require the design of a separate prefilter for each scale factor a.

When the scale parameter is a power of 2, the method is equivalent to a wavelet decomposition [15] because splines satisfy a two-scale difference equation.

B. Computational Issues

We can easily trade computational speed against image quality. The most important choice is the underlying signal model (the spline degree n) which determines the approximation properties of the solution. The second parameter, n_1, can be selected to obtain the optimal least-squares solution $(n_1 = n)$, or a slightly suboptimal one which corresponds

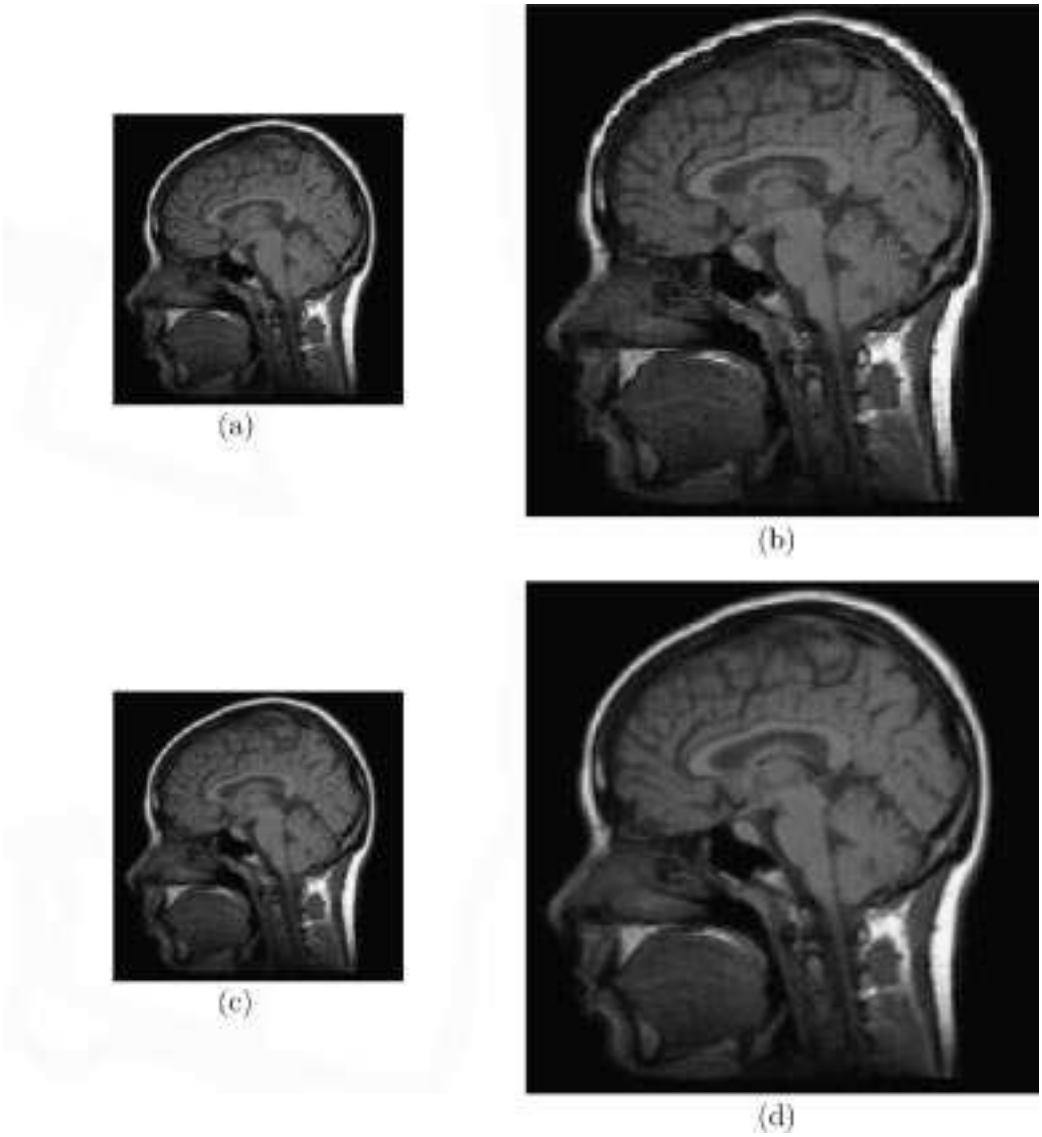

Fig. 18. Example of image reduction by a factor $a = \sqrt{\pi}$ using splines of degree 0. Notice that the projection method reduces blocking artifacts: (a) Reduced image using standard method; (b) enlarged version of the image (a) (SNR = 25.8 dB); (c) reduced image using orthogonal projection; and (d) enlarged version of the image (c) (SNR = 30.7 dB).

to oblique projection $(-1 \leq n_1 < n)$. For the limiting case $n_1 = -1$, we recover the traditional interpolation approach provided that we define the B-spline of degree -1 as the Dirac delta distribution (ideal sampler). The larger $n_1 \leq n$, the better the quality but at the expensive of more computations.

The expensive part of the algorithm is the resampling with the kernel ϕ [step 3 in Section IV-B], which is equivalent to a spline interpolation of degree $(n+n_1+1)$. The cost of the rest of the procedure is negligible in comparison: it involves digital filtering only—either short kernel FIR or fast recursive IIR. Thus, we can consider that the total cost per computed output point is proportional to $(n + n_1 + 1)$ times the number of operations required to evaluate ϕ [B-spline of degree $(n + n_1 + 1)$].

One practical limitation of the present approach is the potential propagation of roundoff errors during the multiple integration process. This requires working with high precision arithmetic. Our implementation uses the double type in C and can handle values up to $n_1 = 4$ with typical image of size $512*512$.

C. Extensions of the Method

The property that the integral of a spline is another spline of higher degree is also valid on nonuniform grids. Therefore, it is also possible to extend the method for the conversion of nonuniform splines to uniform ones using the same least-squares principle [32].

In principle, our algorithm can also be extended to higher dimensions and to nonseparable geometric affine operators. One may catch the intuition of this extension by stressing the key feature of our setting: the function $\varphi_1(x)$ which appears in the general projection-based scheme of Fig. 2 is built using shifted versions of functions—the one-sided power functions x_+^n—that are easily exchanged through the geometric transformation—the scaling operator.

The idea is thus to choose a function ρ that can easily be exchanged through the geometric transformation, and to require that φ_1 belongs to the space generated by the uniform shifts of ρ; in our algorithm, $\rho(x) = x_+^n$.

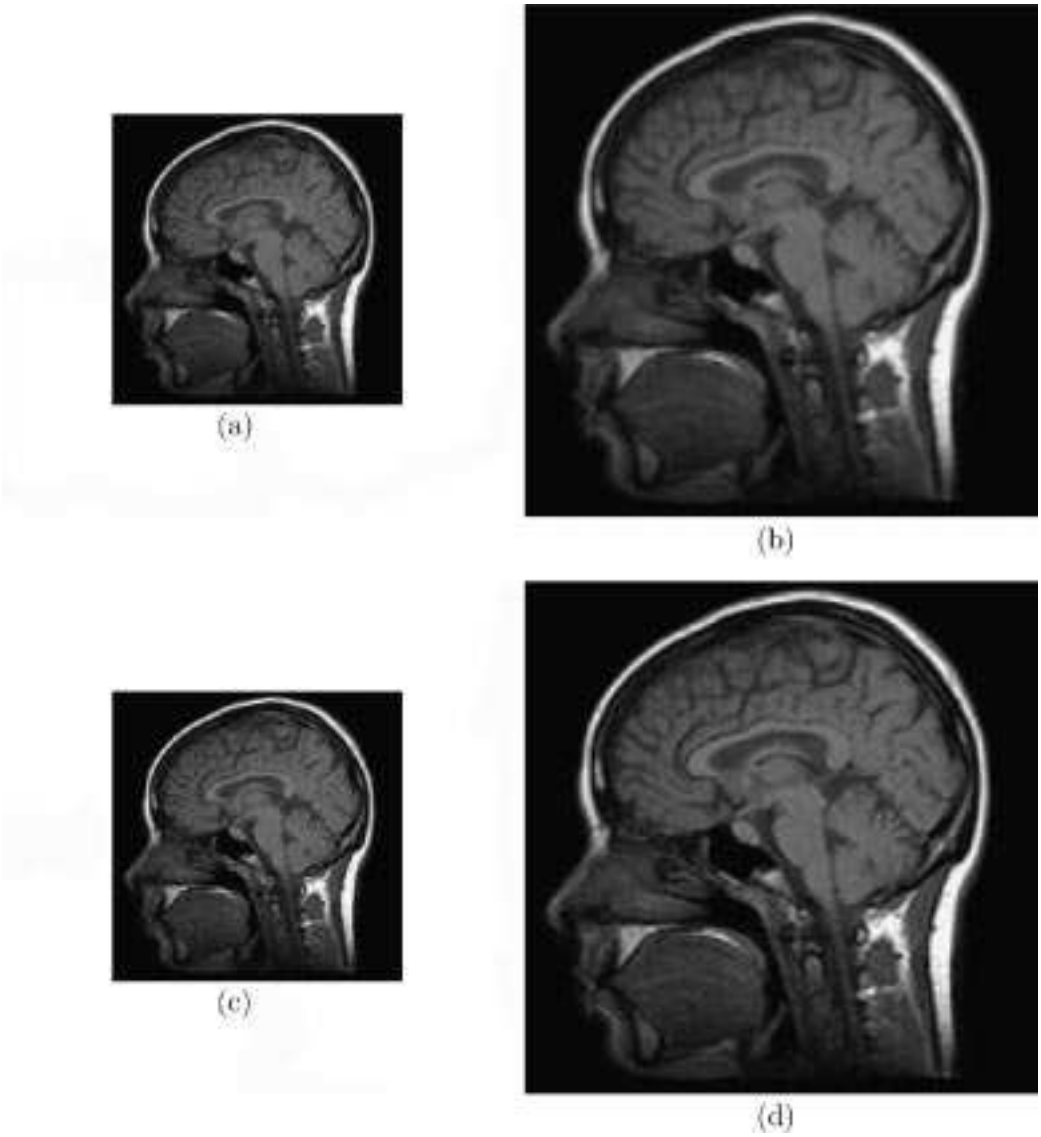

Fig. 19. Same experiment as Fig. 18, but with linear splines. Notice the aliasing reduction from (b) to (d) (less contrast of the features). See also the difference images in Fig. 20: (a) Reduced image using standard method; (b) enlarged version of the image (a) (SNR = 31.9 dB); (c) reduced image using orthogonal projection; and (d) enlarged version of the image (c) (SNR = 35 dB).

For instance, if we wanted to implement rotations and scalings of an N-dimensional digital signal, we could define $\rho(\mathrm{x}) = ||\mathrm{x}||^n$, that is, a radial basis function. This radial basis function can be localized using a digital filter Δ_ρ; that is to say, $\Delta_\rho * \rho$ defines a function that has some appropriate decay as $||\mathrm{x}|| \to \infty$. In our algorithm, this localization filter is simply the finite differences operator Δ^{n+1}, which transforms x_+^n into a B-spline of degree n. We would finally need to compute the convolution of the localized radial basis function with φ to get the function ϕ shown in Fig. 3(e).

VIII. Conclusions

In this paper, we have generalized Lee *et al.*'s method for image resizing using both oblique and orthogonal projections. We have demonstrated that the new method outperforms the standard interpolation techniques. It is especially advantageous for image reduction because of the built-in antialiasing mechanism. An attractive property of the present implementation is that the complexity per output point does not depend on the scaling factor. Our resizing algorithm works for arbitrary scaling factors (image magnification or reduction). We believe that it should be useful in applications where image quality is a key concern.

The formulation of the resizing problem that has been presented is rather general. By varying some key parameters, we switch between optimal least-squares solution, oblique projection and interpolation. We have also described algorithmic solutions for basis functions other than B-splines, the most notable example being the O-MOMS.

A demonstration of our method is available on the web at http://bigwww.epfl.ch/demo/resize.

Appendix

To show the power of our operator formalism, we derive the two key formulas for differential calculus.

n_1th derivative: $D^{n_1}\beta^n(x) = \Delta^{n_1} * \beta^{n-n_1}(x + n_1/2)$.

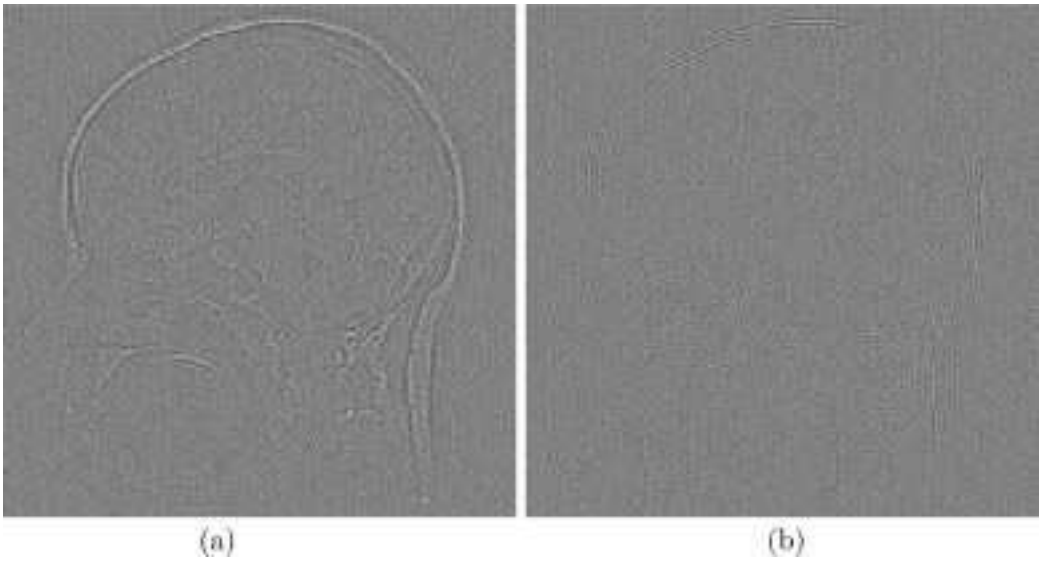

Fig. 20. Difference between the original and the enlarged version of the reduced image obtained in Fig. 19 with the linear spline resizing methods. (a) Standard method and (b) orthogonal projection.

Proof:

$$D^{n_1}\beta^n(x) = D^{n_1} * \Delta^{n+1} * D^{-(n+1)} * \delta\left(x + \frac{n+1}{2}\right)$$

using (4)

$$= \Delta^{n_1} * \Delta^{n+1-n_1} * D^{-(n+1)+n_1} * \delta\left(x + \frac{n+1-n_1}{2}\right) * \delta\left(x + \frac{n_1}{2}\right)$$

conmutativity of "$*$" and $\Delta^{n_1} * \Delta^{-n_1} = I$

$$= \Delta^{n_1} * \beta^{n-n_1}\left(x + \frac{n_1}{2}\right) \qquad \text{using (7).}$$

■

n_1-fold integral: $D^{-n_1}\beta^n(x) = \Delta^{-n_1} * \beta^{n+n_1}(x - (n_1/2))$.

Proof:

$$D^{-(n_1)}\beta^n(x) = D^{-(n_1)} * \Delta^{n+1} * D^{-(n+1)} * \delta\left(x + \frac{n+1}{2}\right)$$

$$= \Delta^{-n_1} * \Delta^{n+n_1+1} * D^{-(n+n_1+1)} * \delta\left(x + \frac{n+1+n_1}{2}\right) * \delta\left(x - \frac{n_1}{2}\right)$$

$$= \Delta^{-n_1} * \beta^{n+n_1}\left(x - \frac{n_1}{2}\right).$$

■

ACKNOWLEDGMENT

The authors wish to thank P. Thévenaz, of the Swiss Federal Institute of Technology, for his careful proofreading of this paper. The authors thank also the reviewers for their insightful comments.

REFERENCES

[1] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978.

[2] V. Di Lecce, G. Dimauro, A. Guerriero, S. Impedoro, G. Pirlo, and A. Salzo, "Electronic document image resizing," in *Proc. Int. Conf. Document Analysis Recognition*, Los Alamitos, CA, Sept. 20–22, 1999.

[3] H. F. Schantz, "Optical imaging and (OCR) recognition technology (recognology)," *Remit. Doc. Process. Today*, vol. 14, no. 7, pp. 11–15, 1992.

[4] C. A. Glasbey and G. W. Horgan, *Image Analysis for the Biological Sciences*. West Sussex, U.K.: Wiley, 1995.

[5] N. Hekotetou and A. N. Venetsanopoulos, "Color image interpolation for high resolution acquisition and display devices," *IEEE Trans. Consum. Electron.*, vol. 41, no. 4, pp. 1118–1126, 1995.

[6] R. Bajcsy and S. Kovačič, "Multiresolution elastic matching," *Comput. Graph. Image Process.*, vol. 46, pp. 1–21, 1989.

[7] J. Kybic, P. Thévenaz, A. Nirkko, and M. Unser, "Unwarping of unidirectionally distorted epi images," *IEEE Trans. Med. Imag.*, vol. 19, pp. 80–93, Feb. 2000.

[8] P. Thévenaz, T. Blu, and M. Unser, *Image Interpolation and Resampling*. New York: Academic, 2000.

[9] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 29, pp. 1153–1160, 1981.

[10] T. Blu, P. Thévenaz, and M. Unser, "Generalized interpolation: Higher quality at no additional cost," in *Proc. IEEE Int. Conf. Image Processing*, Kobe, Japan, Oct. 1999, pp. 25–28.

[11] E. H. W. Meijering, W. J. Niessen, and M. A. Viergever, "Piecewise polynomial kernels for image interpolation: A generalization of cubic convolution," in *Proc. IEEE Int. Conf. Image Processing*, Kobe, Japan, Oct. 25–28, 1999.

[12] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Trans. Signal Processing*, vol. 16, pp. 22–38, Nov. 1999.

[13] M. Unser, A. Aldroubi, and M. Eden, "Enlargement or reduction of digital images with minimum loss of information," *IEEE Trans. Image Processing*, vol. 4, pp. 247–258, Mar. 1995.

[14] N. Damera-Vankata, T. D. Kite, W. S. Geisler, B. L. Evans, and A. C. Bovik, "Image quality assessment based on a degradation model," *IEEE Trans. Image Processing*, vol. 9, pp. 636–650, Apr. 2000.

[15] S. G. Mallat, "A theory of multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.

[16] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge, 1996.

[17] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[18] C. Lee, M. Eden, and M. Unser, "High-quality image resizing using oblique projection operators," *IEEE Trans. Image Processing*, vol. 7, pp. 679–692, May 1998.

[19] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 508–517, 1978.

[20] A. Aldroubi and M. Unser, "Sampling procedures in function spaces and asymptotic equivalence with Shannon's sampling theory," *Numer. Funct. Anal. Optim.*, vol. 15, no. 1/2, pp. 1–21, 1994.
[21] M. Unser and A. Aldroubi, "A general sampling theory for nonideal acquisition devices," *IEEE Trans. Signal Processing*, vol. 42, pp. 2915–2925, Nov. 1994.
[22] M. Unser, "Approximation power of biorthogonal wavelet expansions," *IEEE Trans. Signal Processing*, vol. 44, pp. 519–527, Mar. 1996.
[23] T. Blu and M. Unser, "Quantitative fourier analysis of approximation techniques: Part I—Interpolators and projectors," *IEEE Trans. Signal Processing*, vol. 47, pp. 2783–2795, Oct. 1999.
[24] ——, "Quantitative fourier analysis of approximation techniques: Part II—Wavelets," *IEEE Trans. Signal Processing*, vol. 47, pp. 2796–2806, Oct. 1999.
[25] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing: Part I—Theory and Part II—Efficient design and applications," *IEEE Trans. Signal Processing*, vol. 41, pp. 834–848, Feb. 1993.
[26] G. Strang and G. Fix, "A Fourier analysis of the finite-element variational method," in *Constructive Aspect of Functional Analysis*. Rome, Italy: Edizini Cremonese, 1971, pp. 796–830.
[27] T. Blu, P. Thévenaz, and M. Unser, "Minimum-support interpolators with optimum approximation properties," in *Proc. IEEE Int. Conf. Image Processing*, vol. 3, Chicago, IL, Oct. 4–7, 1998, pp. 242–245.
[28] ——, "MOMS: Maximal order minimal support interpolation," *IEEE Trans. Image Processing*, vol. 10, pp. 1069–1080, July 2001.
[29] M. Unser and I. Daubechies, "On the approximation power of convolution-based least squares vesus interpolation," *IEEE Trans. Signal Processing*, vol. 45, pp. 1697–1711, July 1997.
[30] S. Horbelt, A. Muñoz, T. Blu, and M. Unser, "Spline kernels for continuous-space image processing," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Istanbul, Turkey, June 5–9, 2000.
[31] M. Unser, A. Aldroubi, and M. Eden, "A family of polynomial spline wavelet transforms," *IEEE Trans. Signal Processing*, vol. 30, pp. 141–162, Feb. 1993.
[32] A. Muñoz, T. Blu, and M. Unser, "Non-uniform to uniform grid conversion using least-squares splines," in *Proc. IEEE Eur. Conf. Signal Processing*, Tampere, Finland, Sept. 4–8, 2000.

Arrate Muñoz was born in Pamplona, Spain, in 1973. She did her studies in telecommunications engineering in the Public University of Navarra (UPNA), Navarra, Spain, from which she graduated in July 1997. She is currently pursuing the Ph.D. in the Biomedical Imaging Group, EPFL, Lausanne, Switzerland.

Her main research topics are geometrical transformations of images, nonuniform splines, and non-Euclidean norms applied to multiresolution.

Thierry Blu (M'96) was born in Orléans, France, in 1964. He graduated from the École Polytechnique, France, in 1986 and from Télécom Paris (ENST), France, in 1988. In 1996, he received the Ph.D. in electrical engineering from ENST for a study on iterated rational filter banks applied to wide band audio coding.

He is currently with the Biomedical Imaging Group at the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, on leave from France Télécom CNET (National Center for Telecommunications Studies), Issy-les-Moulineaux, France. His research interests include (multi-)wavelets, multiresolution analysis, multirate filter banks, approximation and sampling theory, and psychoacoustics.

Michael Unser (M'89–SM'94–F'99) received the M.S. (summa cum laude) and Ph.D. degrees in electrical engineering in 1981 and 1984, respectively, from the Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland.

From 1985 to 1997, he was with the Biomedical Engineering and Instrumentation Program, National Institutes of Health, Bethesda, MD. He is now Professor and Head of the Biomedical Imaging Group, EPFL. His main research area is biomedical image processing. He has a strong interest in sampling theories, multiresolution algorithms, wavelets, and the use of splines for image processing. He is the author of 90 published journal papers in these areas. He is on the editorial boards of *Signal Processing*. He serves as regular chair for the SPIE Conference on Wavelets, held annually since 1993.

Dr. Unser is an Associate Editor for the IEEE TRANSACTIONS ON MEDICAL IMAGING. He has been on the editorial boards of several other journals including the IEEE TRANSACTIONS ON IMAGE PROCESSING (1992–1995) and IEEE SIGNAL PROCESSING LETTERS (1994–1998). He received the 1995 Best Paper Award and the 2000 Magazine Award from the IEEE Signal Processing Society. In January 1999, he was elected Fellow of the IEEE "for contributions to the theory and practice of splines in signal processing."