B-Spline Signal Processing: Part II—Efficient Design and Applications

Michael Unser, *Member, IEEE*, Akram Aldroubi, and Murray Eden, *Life Fellow, IEEE*

***Abstract*—This paper describes a class of recursive filtering algorithms for the efficient implementation of B-spline interpolation and approximation techniques. In terms of simplicity of realization and reduction of computational complexity, these algorithms compare favorably with conventional matrix approaches. A filtering interpretation (low-pass filter followed by an exact polynomial spline interpolator) of smoothing spline and least squares approximation methods is proposed. These techniques are applied to the design of digital filters for cubic spline signal processing. An efficient implementation of a smoothing spline edge detector is proposed. It is also shown how to construct a cubic spline image pyramid that minimizes the loss of information in passage from one resolution level to the next. In terms of common measures of fidelity (e.g., visual quality, SNR), this data structure appears to be superior to the widely used Gaussian/Laplacian pyramid.**

I. Introduction

In a companion paper, we have shown that a variety of polynomial splines interpolation and approximation problems can be solved efficiently by shift invariant filtering [1]. Typical applications that could benefit from these techniques are signal interpolation [2], noise reduction [3], and data compression [4], [5].

Polynomial splines also provide a simple mechanism for switching between the discrete and continuous signal domains. Moreover, the simple piecewise polynomial form of the interpolating functions facilitates the transfer and implementation of a variety of mathematical methods available for the study of continuous functions (e.g., differentiation, integration, differential geometry) to the domain of the discrete. This design principle was illustrated in [1] by the derivation of digital filtering algorithms for signal differentiation and the evaluation of convolution integrals.

All B-spline processing techniques described in [1] are based on the well-known property that any polynomial spline $g^n(x)$ can be represented as a weighted sum of shifted B-spline basis functions ($\beta^n(x)$) and is therefore characterized by the discrete sequence of its B-spline coefficients [6], [7]. The primary step for processing a discrete signal $g(k)$ is therefore to determine the B-spline coefficients of its continuous representation $g^n(x)$. Three possible options have been considered for this purpose [1].

First, we may determine the B-spline coefficients that provide an exact interpolation, i.e., $g^n(k) = g(k)$. In effect, this constraint establishes a one-to-one correspondence between $\{g(k)\}$ and the sequence of its B-spline coefficients of order n: $\{y(k)\}$ [6], [7]. We call such a transformation a direct B-spline transform, as in [2]. In the particular case of equally spaced data points, the direct B-spline transform can be implemented by simple linear filtering:

$$y(k) = (b_1^n)^{-1} * g(k) \tag{1.1}$$

where $(b_1^n)^{-1}$ is the impulse response of the direct B-spline filter of order n.

The second option, which is to be preferred when there is a reason to believe that the data are corrupted by noise, is to evaluate the coefficients of a polynomial spline approximation subject to some smoothness constraints. This task is accomplished through the use of smoothing splines, which were introduced independently by Schoenberg [8] and Reinsh [3]. This technique can be viewed as a regularized version of the previous interpolation problem [9]. Here too, we have shown that for equally spaced data points, the B-spline coefficients of the smoothing spline of order $n = 2r - 1$ can be determined by linear filtering cf. [1, sec. IV-C]:

$$y_\lambda(k) = s_\lambda^n * g(k) \tag{1.2}$$

where s_λ^n is the impulse response of the smoothing spline filter of order n with scale (or regularization) parameter $\lambda \geq 0$. The magnitude of λ specifies the amount of smoothing. In particular, the choice $\lambda = 0$ corresponds to no smoothing at all, in which case (1.2) is equivalent to (1.1).

The third option is to select a least squares approximation constrained to an arbitrary (smaller) number of coefficients. This approach can be thought of as a data compression technique but it can be used for noise reduction as well. We have shown [1, sec. IV-D] that the least squares B-spline coefficients can be evaluated efficiently by simple filtering and down-sampling by a factor of m:

$$y_m(k) = s_m^n * [b_m^n * g]_{\downarrow m}(k) = [\mathring{b}_m^n * g]_{\downarrow m}(k) \tag{1.3}$$

where $\mathring{b}_m^n = [s_m^n]_{\uparrow m} * b_m^n$ is the optimal prefilter for a least squares spline approximation of order n with a reduction (or compression) factor m.

Manuscript received August 11, 1990; revised March 2, 1992.

M. Unser is with the Biomedical Engineering and Instrumentation Program, National Institutes of Health, Bethesda, MD 20892, on leave from INSERM U.2, Hôpital Henri-Mondor, F-94010, France.

A. Aldroubi and M. Eden are with the Biomedical Engineering and Instrumentation Program, National Institutes of Health, Bethesda, MD 20892.

IEEE Log Number 9205138.

1053-587X/93$03.00 © 1993 IEEE

The essence of B-spline processing is then to design discrete algorithms operating in the B-spline domain so as to perform mathematical operations or transformations on the underlying continuous signals $g^n(x)$; e.g., evaluation of derivatives and convolution integrals, the search for extrema or zero crossings, on the solution of differential equations. These are all computational tasks that are not well defined in any conventional discrete signal processing framework known to us. At the very end of the process, the result of these computations can be mapped back into the discrete signal domain by indirect B-spline transform of order p (where p may be different from n); a transformation that involves a convolution with a sampled B-spline kernel

$$\hat{g}(k) = b_1^p * y(k). \tag{1.4}$$

An extended form of this operation for signal reconstruction with an expansion factor m, of possible use for signal interpolation, is provided by [1, eq. (3.17)]. Separable extensions of these algorithms are not difficult to obtain for higher dimensional signals through the use of tensor product splines.

The purpose of this paper is to investigate the implementation of these techniques and to present some examples of applications. Section II describes a class of recursive algorithms for the efficient implementation of the B-spline transformations described by (1.1)–(1.3). It also compares this approach with a "carefully designed" matrix algorithm based on Gaussian elimination. In Section III, the global effect of both smoothing spline and least squares approximations is described in terms of equivalent low-pass filters acting on the input data sequence. Finally, in Section IV, we derive explicit filter formulas for cubic spline signal processing and present some image processing applications. In particular, we describe an efficient implementation of a cubic spline edge detection algorithm, which, somewhat unexpectedly, turns out to be equivalent to the Canny edge detector [10]. We also introduce a cubic spline image pyramid and show that this representation compares favorably with Burt's Gaussian/Laplacian pyramid [11], [12], a technique used in a variety of multiresolution image processing and computer vision algorithms [13].

The theoretical results in [1] were derived under the assumption of finite energy signals with an infinite duration (i.e., $g(k) \in l_2$). Since the signals (or images) encountered in practice are of finite duration: $\{g(k), k = 1, \cdots, K\}$, we have to impose a set of boundary conditions. For practical convenience and to avoid discontinuities, we have chosen to extend a signal on both sides by using its mirror image, a standard practice in image processing:

$$\begin{cases} g(-k) = g(k+1), & k = 0, \cdots, K-1 \\ g(k) = g(2K-k), & k = K, \cdots, 2K-1. \end{cases} \tag{1.5}$$

Such a signal is also embedded in an infinite sequence of period $2K - 1$ defined as

$$\forall k \in Z, \qquad g_p(k) = g(k \bmod (2K-1)).$$

This last property may be useful for implementing FFT-based algorithms, although this option is not considered further in this paper.

II. B-Spline Filter Implementation

In this section, we present a general procedure for implementing the B-spline filters derived in [1]. The present approach uses the fact that the basic structure of all these filters is very similar (symmetric all-pole IIR filters). Alternatively, these filters may also be implemented using an FIR approximation of their impulse response. A constrained least squares design technique that is well suited for this latter task is discussed in [14].

A generic symmetrical recursive filter may be decomposed into a cascade of elementary symmetrical exponential filters which themselves can be separated into two complementary causal and anticausal components. Such a decomposition can then be used to obtain simple implementation formulas for direct and least squares B-spline filters of any order. The first-order smoothing spline filter also fits well into this framework. Following a presentation of our main results, we compare the computational complexity of the present approach and a fast matrix implementation, with a special emphasis on the problem of cubic spline interpolation.

A. Basic Decomposition Formulas

Let us consider the z transform of a generic symmetrical stable recursive filter of order $2N$:

$$H_{2N}(z) = \frac{c_0}{[z^N + z^{-N}] + \left(\sum_{k=1}^{N-1} a_k [z^k + z^{-k}]\right) + a_0} \tag{2.1}$$

where c_0 and $\{a_k, k = 0, \cdots, N-1\}$ are constant coefficients. A polynomial $P_{2N}(z)$ can be defined by multiplying the denominator of (2.1) by z^N. Clearly, since $H_{2N}(z) = H_{2N}(z^{-1})$, this polynomial satisfies the equation: $z^{-N}P_{2N}(z) = z^N P_{2N}(z^{-1})$. It follows that the zeros of $P_{2N}(z)$ occur in reciprocal pairs, i.e., $P_{2N}(z_i) = 0$ iff $P_{2N}(z_i^{-1}) = 0$. These roots, which are assumed to lie outside the unit circle, are denoted by $\{(z_i, z_i^{-1})$ with $|z_i| < 1, i = 1, \cdots, N\}$. Consequently, $H_{2N}(z)$ can be factored as

$$H_{2N}(z) = c_0 \prod_{i=1}^{N} H(z; z_i) \tag{2.2}$$

where $H(z; z_i)$ is defined as follows:

$$H(z; z_i) := \left(\frac{-z_i}{(1 - z_i z^{-1})(1 - z_i z)}\right)$$

$$= \frac{z_i}{(1 - z_i^2)}\left(\frac{1}{1 - z_i z^{-1}} + \frac{1}{1 - z_i z} - 1\right). \tag{2.3}$$

The global impulse response can then be determined explicitly by using a standard decomposition into simple partial fractions.

B. Implementation of the Basic Symmetrical Operator

Equation (2.2) suggests a straightforward filter implementation by a cascade of simple operators, which we now consider in greater detail. Obviously, this approach is only practical when the z_i's are all real. The smoothing cubic spline filter which does not satisfy this constraint is discussed separately in Section IV-B.

It is not difficult to show that $H(z; z_i)$ is the transfer function of a symmetrical exponential filter whose impulse response is

$$h(k; z_i) = \frac{z_i}{(1 - z_i^2)} z_i^{|k|}. \tag{2.4}$$

Starting from (2.3), we can derive the following recursive filter equations:

$$\begin{cases} y^+(k) = x(k) + z_i y^+(k-1), \\ \qquad (k = 2, \cdots, K) \\ y(K) = c_i(2y^+(K) - x(K)) \\ y(k) = z_i(y(k+1) - y^+(k)), \\ \qquad (k = K-1, \cdots, 1) \end{cases} \tag{2.5}$$

where $x(k)$ and $y(k)$ are the input and output signals, respectively, and where $c_i = -z_i/(1 - z_i^2)$ is a scaling constant. For boundary conditions specified by (1.5), the recursion begins with

$$y^+(1) = \sum_{k=1}^{k_0} z_i^{|k-1|} x(k) \tag{2.6}$$

where k_0 is chosen to ensure that $z_i^{|k_0|}$ is smaller than some prescribed level of precision. The second equation in (2.5) is borrowed from a sum decomposition (right-hand side of (2.3)) and is required to obtain a correct initialization of the backward recursion. The following arguments can be listed in favor of these particular boundary conditions:

- the boundary conditions are also satisfied by any filtered version of the signal provided that the impulse response of the filter is symmetrical; this property is essential for reversibility of the B-spline transform;
- they result in no visible border artifacts;
- they guarantee that the filtering of a constant signal induces a constant output.

The algorithm described by (2.5) requires no more than two additions and two multiplications per sample point. Since all operations are real, it is necessary to use a one-dimensional real array to store the input and output sequences with a precision sufficient to avoid propagation of errors by recursion. It is relatively straightforward to write a general subroutine that implements (2.1) from a succession of simple convolutions of the form (2.5). This approach is also applicable in higher dimensions through the successive use of the same one-dimensional filter along each dimension of the data. For digital images, there is in principle no need for floating point data storage other than the one-dimensional array required by the basic filtering module.

We note that for compatibility and to insure the reversibility of the transformations involved, the FIR operators used for signal reconstruction (indirect B-spline transform) should also use the same type of signal extrapolation (i.e., (1.5)).

C. Recursive Direct B-Spline Filters

We used the iterative equations [1, eqs. (3.3) and (3.4)] to determine the transfer functions of the direct B-spline filters for $n = 0$ to 7. These expressions together with their characteristic values c_0 and $\{z_i, i = 1, \cdots, [n/2]\}$ are given in Table I. The use of these parameters in the general filtering module described above allows an evaluation of the B-spline coefficients of order n with approximately $2[n/2]$ multiplications and $2[n/2]$ additions per sample point.

It is interesting to note that the roots of direct B-spline filters are simple and negative, as illustrated by the examples in Table I. This property was proved by Schoenberg [7, Lemma 8] in a different context.

D. First-Order Smoothing Spline Filter

The first-order smoothing spline filter is the simplest case of [1, eq. (4.13)] and its transfer function can be written as

$$S_\lambda^1(z) = \frac{1}{1 + \lambda(-z^{-1} + 2 - z)} = \frac{z_1/\lambda}{(1 - z_1 z^{-1})(1 - z_1 z)} = -H(z; z_1)/\lambda \tag{2.7}$$

where λ is a regularization parameter controlling the degree of smoothness of the first-order spline (piecewise linear) signal approximation. The smallest root z_1 ($0 \le z_1 \le 1$) of the characteristic polynomial is given by

$$z_1 = 1 + \frac{1}{2\lambda} - \frac{\sqrt{1 + 4\lambda}}{2\lambda}. \tag{2.8}$$

Clearly, this filter corresponds to a particular case of (2.2) with $c_0 = -1/\lambda$ and $N = 1$. Interestingly, it is identical to the first-order R-filter described by us in a different context [15]. A further discussion of the properties of this particular filter can be found in Section III-A of that earlier paper.

The case of higher order smoothing spline filters is somewhat more complicated because the roots are not necessarily simple and depend on λ. The special case of smoothing cubic splines, which is still tractable analytically, is considered in Section IV-C.

E. Least Squares Spline Filters

We have shown in [1] that least squares B-spline approximations with a decimation factor m can be deter-

TABLE I
TRANSFER FUNCTIONS AND POLES OF DIRECT B-SPLINE FILTERS FOR $n = 0$ TO 7

n	Direct B-Spline Filter: $B_1^n(z)^{-1}$	c_0	Poles: $\{\lvert z_i\rvert < 1, i = 1, \cdots, n\}$
0	1	1	—
1	1	1	—
2	$\dfrac{8}{z + 6 + z^{-1}}$	8	$z_1 = -3 + 2\sqrt{2} = -0.171573$
3	$\dfrac{6}{z + 4 + z^{-1}}$	6	$z_1 = -2 + \sqrt{3} = -0.267949$
4	$\dfrac{384}{z^2 + 76z + 230 + 76z^{-1} + z^{-2}}$	384	$z_1 = -0.361341$ $z_2 = -0.0137254$
5	$\dfrac{120}{z^2 + 26z + 66 + 26z^{-1} + z^{-2}}$	120	$z_1 = -0.430575$ $z_2 = -0.0430963$
6	$\dfrac{46080}{z^3 + 722z^2 + 10543z + 23548 + 10543z^{-1} + 722z^{-2} + z^{-3}}$	46080	$z_1 = -0.488295$ $z_2 = -0.0816793$ $z_3 = -0.00141415$
7	$\dfrac{5040}{z^3 + 120z^2 + 1191z + 2416 + 1191z^{-1} + 120z^{-2} + z^{-3}}$	5040	$z_1 = -0.53528$ $z_2 = -0.122555$ $z_3 = -0.00914869$

mined in three basic steps (cf. Fig. 3(a)). The practical aspects of this procedure are now considered in greater detail.

1) Prefiltering: The first step is a convolution with a discrete sampled B-spline basis function expanded by a factor m (e.g., b_m^n). This operation is equivalent to an indirect B-spline transform whose efficient implementation by the use of moving average filters of length m is discussed in [2]. For small expansion factors ($m = 2$ or 3), one may equally well implement this operation as a single convolution with a symmetrical FIR kernel b_m^n of length $(2m\lfloor n/2 \rfloor + m)$ which can be generated recursively by use of [1, eqs. (3.3) and (3.4)].

2) Decimation: The next step is a decimation by a factor of m that samples the signal at the position of the knots of the expanded B-spline basis functions. To insure the concordance between the approximation and reconstruction processes in the border regions, the knot points should be positioned so that the boundary conditions used for the initial and decimated sequences are fully compatible. For the mirror symmetry extrapolation of the signal specified by (1.5), this puts the constraint of having a knot at each extremity of the signal which is only possible for sequences of length $K = K'm + 1$ where K' is an integer number. The input signal should therefore initially be truncated or extrapolated in some manner to this length and the samples taken according to the sequence: $k = 1, m + 1, 2m + 1, \cdots, mk' + 1, \cdots, mK' + 1$. Alternatively, it is also possible to modify the initial conditions on the decimated sequence so that they are compatible with (1.5) at the finer resolution level.

3) Recursive Filtering: The final step is to postfilter the decimated sequence with the least squares operator $S_m^n(z)$ (cf. [1, eq. (4.20)]); also a particular case of (2.1). To determine this filter's coefficients, we need to evaluate the autocorrelation function of the sequence b_m^n and downsample it by a factor m. We have performed these calculations for $m = 2, \cdots, 8$ and all B-spline functions up to order 3. Some of those results ($m = 2$ and 3), including all parameters for the general filtering module described in Section II-B, are given in Table II. In all cases, the number of characteristic roots, which equals the number of elementary exponential convolutions, is precisely equal to the order of the B-spline (n). An interesting property is suggested by the derivation in Appendix A: For m sufficiently large, the least squares spline filters s_m^n converge to a direct B-spline filter of order $2n + 1$:

$$\lim_{m \to +\infty} (m \cdot s_m^n(k)) = \left(b_1^{2n+1}\right)^{-1}(k). \qquad (2.9)$$

In fact, a comparison of the roots of the corresponding filters in Tables I and II shows that this tendency is already quite apparent for $m = 3$, especially for higher order splines.

These algorithms were implemented in Fortran on a low end workstation (standard 16 MHz Apple Macintosh IIcx) for biomedical image processing. An example is shown in Fig. 1 comparing the result of LS zero order and cubic approximations with a compression ratio 4 ($m = 2$) to the original image. The zero order approximation is obtained by taking averages over 2×2 neighborhoods (simple moving average). The B-spline coefficient image (Fig. 1(b)) is essentially a scaled down version of the original MRI scan (Fig. 1(a)). The reconstruction (Fig. 1(d)) has noticeable artifacts and has a signal-to-noise ratio (SNR) of 27.78 dB. The LS cubic spline coefficients provide a reduced image (Fig. 1(c)) that appears to be perceptually sharper than the original. The reconstruction (Fig. 1(e))

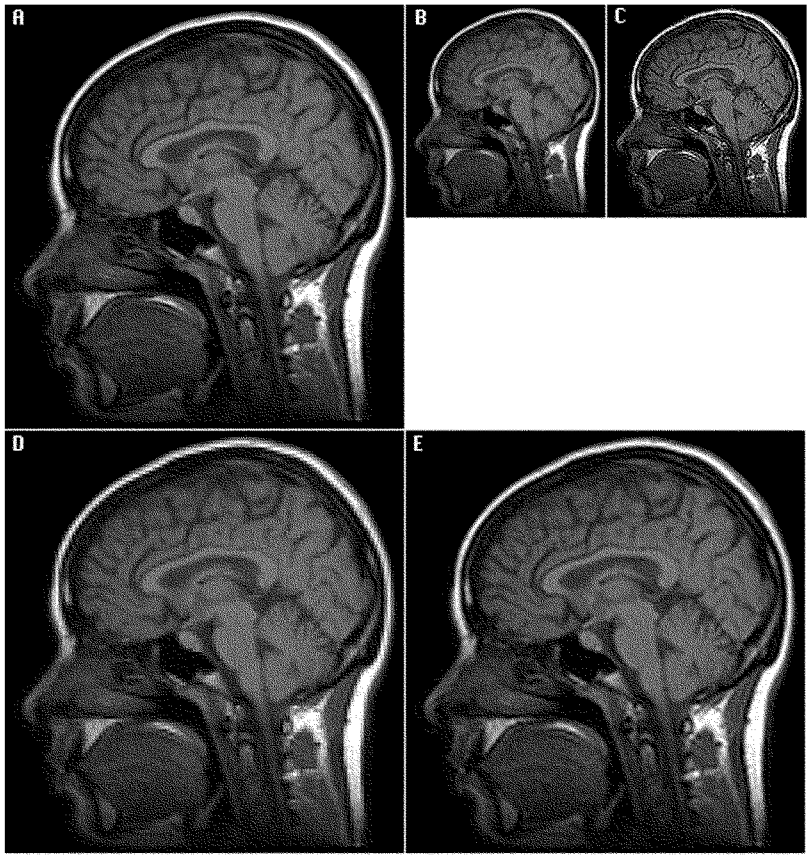

Fig. 1. Example of least squares spline image compression with a decimation factor 2. (a) Original 238 × 253 MRI scan, (b) least squares zero-order coefficients, (c) least squares cubic spline coefficients, (d) zero-order reconstruction, (e) cubic spline reconstruction.

TABLE II
TRANSFER FUNCTIONS AND POLES OF LEAST SQUARES B-SPLINE FILTERS FOR $n = 0$ TO 3 AND DECIMATION FACTORS $m = 2$ AND 3

Order n	Width m	Least Squares B-Spline Filter: $S_m^n(z)$	c_0	Poles: $\{\|z_i\| < 1, i = 1, \cdots, [n/2]\}$
0	2	1	1	—
1	2	$\frac{4}{z + 6 + z^{-1}}$	4	$z_1 = -0.171573$
2	2	$\frac{64}{z^2 + 28z + 70 + 28z^{-1} + z^{-2}}$	64	$z_1 = -0.446463$ $z_2 = -0.0395661$
3	2	$\frac{2304}{z^3 + 110z^2 + 1087z + 2212 + 1087z^{-1} + 110z^{-2} + z^{-3}}$	2304	$z_1 = -0.529604$ $z_2 = -0.122309$ $z_2 = -0.0100731$
0	3	1	1	—
1	3	$\frac{9}{4z + 19 + 4z^{-1}}$	$\frac{9}{4}$	$z_1 = -0.220789$
2	3	$\frac{5184}{131z^2 + 3364z + 8562 + 336z^{-1} + 131z^{-2}}$	$\frac{5184}{131}$	$z_1 = -0.427871$ $z_2 = -0.0437242$
3	3	$\frac{26244}{16z^3 + 1875z^2 + 18600z + 37750 + 18600z^{-1} + 1875z^{-2} + 16z^{-3}}$	$\frac{26244}{16}$	$z_1 = -0.534462$ $z_2 = -0.122347$ $z_3 = -0.00939172$

is of much better quality with SNR = 35.39 dB. The CPU times necessary for the evaluation of the results in Fig. 1 using a separable recursive implementation were 4 s (b), 25 s (c), 1 s (d), and 12 s (e).

A more detailed performance evaluation using a wider range of compression ratios is given in Table III in terms of the SNR in decibels for various B-spline approximations of order 0, 1, 2, and 3. Consistent with our expec-

TABLE III
COMPARISON OF VARIOUS COMPRESSION METHODS IN TERMS OF THEIR SIGNAL-TO-NOISE RATIO (DECIBELS) FOR THE "MRI" IMAGE IN FIG. 1(a). A DECIMATION BY (m, m) CORRESPONDS TO A COMPRESSION RATIO OF $R = 1/m^2$

Decimation	0 Order Spline	Bilinear Spline	Quadratic Spline	Cubic Spline	Quadratic Interpol.	Gaussian Pyramid	Spline Pyramid
(2, 2)	27.78	32.66	35.06	35.39	33.76	26.55	35.39
(3, 3)	24.38	27.85	28.84	29.08	27.41	—	—
(4, 4)	22.43	24.96	25.46	25.58	23.70	20.95	25.57
(5, 5)	21.20	23.22	23.61	23.73	21.50	—	—
(6, 6)	20.18	22.30	22.61	22.68	20.45	—	—
(7, 7)	19.68	21.22	21.52	21.62	19.33	—	—
(8, 8)	19.06	20.54	20.78	20.85	18.95	18.02	20.83

tations, the quality of the approximation diminishes with increasing reduction factors (m). What is more interesting is that for a given compression ratio the SNR's obtained for higher order splines are consistently superior. It appears, however, that the relative performance improvement decreases rapidly with n. The results of the quadratic spline compression method described by Toraichi *et al.* are also given in Table III [4]. It is not surprising that this technique, which uses a simple decimation (with no prefiltering) followed by an exact quadratic spline interpolation, produces lower SNR values. For comparison, we have also included the results obtained using Burt's Gaussian pyramid [11], [12]; a hierarchical image representation frequently used in multiresolution image processing and computer vision applications [13]. The somewhat inferior performance of the latter technique can be explained by reference to the expansion (or interpolation) mechanism described in [12]. It is suboptimal in that it does not minimize the approximation error, but it simplifies the implementation. An alternative cubic spline image pyramid that avoids this limitation is described in Section IV-C. Its results (cf. last column in Table III) are essentially equivalent to the ones obtained for the cubic spline least squares approximation.

F. Comparison with Conventional Matrix Approaches

Usually, polynomial spline interpolation and approximation problems are solved using matrix formulations [16]. The main advantage of the B-spline representation is the simple structure of the underlying matrices which are banded Toeplitz. It follows that the corresponding system of equations can be solved very efficiently by careful Gaussian elimination or LU factorization [17]. To illustrate these techniques, we consider the case of cubic spline interpolation. The matrix formulation of this problem with boundary conditions specified by (1.5) is

$$\begin{bmatrix} u_1 & v_1 & 0 & \cdots & & \\ v_2 & u_2 & v_2 & \cdots & & \\ & & \cdots & & & \\ & & & \cdots & u_{K-1} & v_{K-1} \\ & & & \cdots & v_K & u_K \end{bmatrix} \cdot \begin{bmatrix} y(1) \\ y(2) \\ \cdots \\ y(K-1) \\ y(K) \end{bmatrix} = \begin{bmatrix} g(1) \\ g(2) \\ \cdots \\ g(K-1) \\ g(K) \end{bmatrix} \tag{2.10}$$

where $u_1 = u_2 = \ldots = u_K = \frac{4}{6}$, $v_2 = v_3 = \ldots = v_{K-1} = \frac{1}{6}$, and $v_1 = v_K = \frac{2}{6}$. This system of tridiagonal equations can be solved by using an adapted version of a fast algorithm described in [18, pp. 40–41]. The first step is an LU decomposition with a forward substitution:

$$\begin{aligned} & d_1 = u_1, \\ & \begin{cases} \alpha_i = v_i/d_{i-1} \\ d_i = u_i + \alpha_i v_{i-1}, \qquad (i = 2, \cdots, K). \\ y^+(i) = g(i) + \alpha_i g(i-1) \end{cases} \end{aligned} \tag{2.11}$$

The second step is a backsubstitution using the upper triangular factorization specified by the components of the vectors $\boldsymbol{d}$ and $\boldsymbol{v}$:

$$\begin{aligned} & y(K) = y^+(K)/d_K \\ & \begin{cases} \alpha_i = -v_i/d_i \\ y(i) = -y^+(i)/u_i + \alpha_i y(i+1), \end{cases} \\ & \qquad (i = K-1, \cdots, 1) \end{aligned} \tag{2.12}$$

This algorithm has a complexity of $8K$ flops ($5K$ multiplications + $3K$ adds) where K is the number of samples, which is about twice that of the filtering procedure described in Section II-C. Moreover, it requires additional intermediate storage for the modified diagonal elements of the upper triangular factorization. We note that this procedure is quite similar to the recursive filtering algorithm (2.5) described in Section II-B: the forward and backward substitutions correspond to the causal and anticausal recursive filtering equations, respectively. The only difference is the use of a nonconstant updating factor α_i in (2.11) and (2.12). During the forward substitution part of the algorithm, it is easy to verify that this quantity is

given by

$$\alpha_i = \frac{-1}{4 + \alpha_{i-1}}$$

with the initial condition $\alpha_1 = -\frac{1}{4}$. This nonlinear difference equation has a stable steady state $\alpha = -2 + \sqrt{3}$ that corresponds precisely to the value of the recursive filter coefficient for the direct cubic spline filter [2]. In fact, it can be verified that the convergence of α_i to this value is extremely fast (relative error less than 0.04% error for $i = 3$). The same type of analysis can be carried out for the backsubstitution part of the algorithm. These considerations clearly show that the filtering and matrix procedures are essentially equivalent and differ only in their way of handling the boundary conditions. The former is simpler and uses fewer operations.

For higher order spline interpolations and least squares approximations, it would appear that the advantage of the filtering procedure over the matrix approach is even greater, not to mention the difficulties that might be encountered in designing such matrix algorithms. For a B-spline interpolation of order n, the recursive filtering approach described in Section II-C uses $4[n/2]$ flops per sample. A comparable algorithm for banded matrices would require $2n[n/2]^2$ flops/sample for the LU factorization alone, and $8[n/2]$ flops/sample for the subsequent solution of the banded system of equations using forward and backward substitutions [17, pp. 150–151]. In this respect, we note that the matrix algorithm described above is superior in the sense that it performs the two tasks jointly and computes the LU factorization at no additional cost.

III. B-Spline Approximation and Low-Pass Filtering

In the first part of this study, we have offered an interpretation of B-spline interpolation as the convolution of a sampled sequence with a continuous interpolation function referred to as the cardinal spline. This interpolation function resembles a sinc function and its frequency response approximates an ideal low-pass filter (cf. [1, sec. III-D]).

A similar filtering interpretation can be made for signal approximation using either smoothing or least squares splines. The key idea is to decompose these operations into a purely discrete filtering procedure implementing the approximation part of the process, and a subsequent polynomial spline interpolation that produces an exact interpolation of the intermediate discrete values. It will be shown here that the first part of this process is a special form of low-pass filtering.

A. Smoothing Spline Approximation

In the case of smoothing spline approximation, the discrete part of the process is completed when the B-spline coefficients are mapped back into the signal domain. This requires an additional indirect B-spline transform. Thus, based on [1, eq. (4.13)], the transfer function of the equivalent discrete filter is

$$H_\lambda^n(z) = B_1^n(z) S_\lambda^n(z) = \frac{B_1^n(z)}{B_1^n(z) + \lambda(-z + 2 - z^{-1})^{(n+1)/2}} \quad (3.1)$$

where $B_1^n(z)$ is given by [1, eq. (3.9)]. The frequency response is obtained by replacing z by $e^{j2\pi f}$. Clearly, we have that $H_\lambda^n(z)|_{z=1} = 1$ as $B_1^n(z)|_{z=1} = 1$ and $(-z + 2 + 1/z)|_{z=1} = 0$; it follows that the gain at zero frequency is precisely one. In fact, $H_\lambda^n(z)$ always behaves as a low-pass filter and the strength of this filtering is modulated by the parameter λ. As an example, we show the graph of the frequency response of a cubic B-spline filter for various values of λ (Fig. 2).

B. Least Square Spline Approximation

The case of the least square B-spline approximation is slightly more complicated because we require a decimation by a factor m during signal approximation followed by an expansion by the same factor for signal reconstruction. The mechanism by which a filtering/interpolation decomposition can be obtained is illustrated in Fig. 3. The first step is to apply an indirect B-spline transformation to the B-spline coefficients to map these back into the decimated discrete signal domain. This operation must be compensated globally by an inverse direct B-spline filter (cf. Fig. 3(b)). The last manipulation is to move both discrete filters on the other side of the decimation and expansion modules. This, in turn, requires an up-sampling of their impulse response by a factor of m (cf. Fig. 3(c)).

The components prior to decimation form the procedure's discrete filtering part characterized by the transfer function

$$\mathring{H}_m^n(z) = \mathring{B}_m^n(z) B_1^n(z^m) = \frac{B_m^n(z) B_1^n(z^m)}{\frac{1}{m}\sum_{k=0}^{m-1} B_m^n(ze^{2\pi k/m})^2}. \quad (3.2)$$

The components on the left produce an interpolation with an expansion factor m (see for instance [2]). Such a discrete B-spline interpolator is characterized by the transfer function

$$H_m^n(z) = \frac{B_m^n(z)}{B_1^n(z^m)}. \quad (3.3)$$

The notation $\mathring{H}_m^n(z)$ in (3.2) is consistent with the fact that $\mathring{H}_m^n(z)$ is defined as a pseudoinverse of $H_m^n(z)$. The graphs of the frequency responses of corresponding filters for $n = 1, 3, 5$ and a decimation factor $m = 2$ are shown in Fig. 4. It is apparent from those displays that the prefilter $\mathring{h}_m^n$ acts like a low-pass filter with a cutoff frequency at $1/m$. This operator is therefore quite similar to the anti-aliasing low-pass filters used in conventional sampling theory [19]. We also note that the prefilter approaches an ideal low-pass filter for increasing values of n. This result is consistent with the property that a cardinal spline inter-

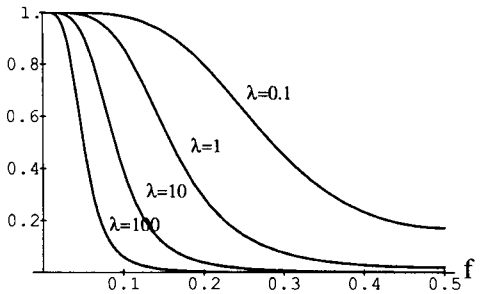

Fig. 2. Frequency response of a smoothing cubic spline filter for various values of λ.

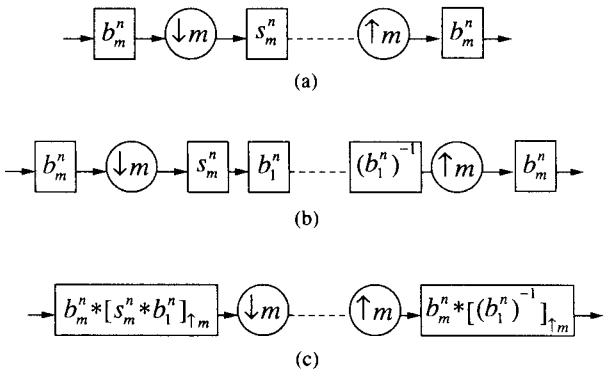

Fig. 3. Three equivalent implementations for the least squares polynomial spline approximation of a signal.

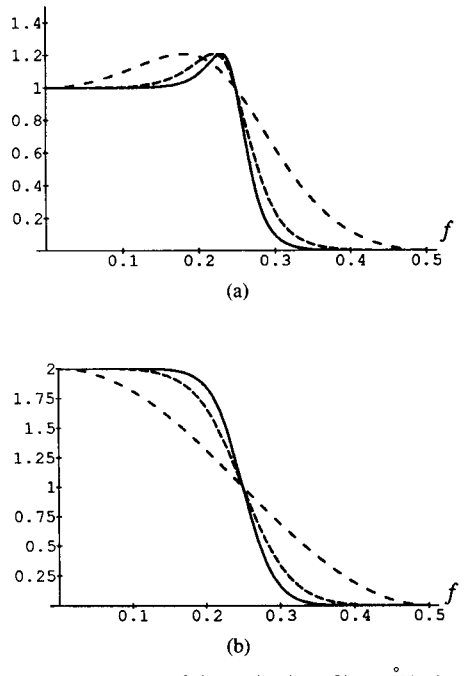

Fig. 4. Frequency responses of the optimal prefilter ($\mathring{H}_2^n(f)$) and its corresponding B-spline interpolator ($H_2^n(f)$) with an expansion factor of two for $n = 1, 3$, and 5. Legend: $n = 1$: - - - - (linear), $n = 3$: · · · · (cubic), $n = 5$: ———— (quintic).

polator approaches an ideal sinc interpolator as n tends to infinity [20].

The least squares spline approximation of a signal $g(k)$ may also be viewed as the projection of this signal into S_m^n: the subspace of polynomial splines of order n with a knot spacing m. This property also implies that the error $e_m(k) = g(k) - g_m^n(k)$ is orthogonal to $g_m^n(k)$, the expanded representation of this approximation.

IV. Cubic Spline Image Processing

In this section, we will concentrate on the design and the study of the properties of the operators for cubic spline signal processing because higher order splines will rarely be required in practice. We will also present some image processing applications of these techniques.

We recall that the cubic B-spline coefficients that provide an exact signal interpolation are obtained by convolution with the direct cubic spline filter (direct cubic spline transform) given by

$$(b_1^3)^{-1}(k) \overset{z}{\leftrightarrow} B_1^3(z)^{-1} = \frac{6}{z + 4 + z^{-1}}. \qquad (4.1)$$

The operation is fully reversible (indirect cubic spline transform) by convolution with a sampled cubic spline kernel:

$$b_1^3(k) \overset{z}{\leftrightarrow} B_1^3(z) = \tfrac{1}{6}(z + 4 + z^{-1}). \qquad (4.2)$$

For image processing applications, these filters are applied successively along the rows and columns (see, for example, [2]). In the case of noisy data, an approximate cubic spline representation can be obtained through the use of the smoothing cubic spline filters described in Section IV-B.

A. Gradient and Laplacian Operators

Once an exact or approximate B-spline image representation has been obtained, the evaluation of first-order and second-order derivatives along the vertical or horizontal direction involve simple convolutions with the basic one-dimensional difference operators $|1 \ -1|$ and $|1 \ -2 \ 1|$, respectively, (cf. [1, Sec. IV-A]). If we are using a cubic spline representation, the first-order derivative will be a quadratic spline while the second-order derivative will be a piecewise linear function.

To map those results back into the image domain, we must perform an indirect B-spline transform. In the case of a single derivative, this requires an additional convolution with $c_1^2(k)$ (cf. [1, Table I]). It is worth noting that by combining the difference and indirect transformation, we get the equivalent antisymmetrical operator: $|\frac{1}{2} \ \frac{1}{2}| * |1 \ -1| = |\frac{1}{2} \ 0 \ -\frac{1}{2}|$, which is frequently used as the discrete approximation of a differentiator. In the case of the second-order derivative, the indirect mapping is the identity transform since the first order B-spline coefficients correspond to the pixel values themselves.

To evaluate image gradients or Laplacians which are frequently used for edge detection, we apply these operators in the two principal directions. This is illustrated by the general block diagram in Fig. 5. The convolution mask for the vertical and horizontal gradient components are derived from the tensor product of the symmetric differ-

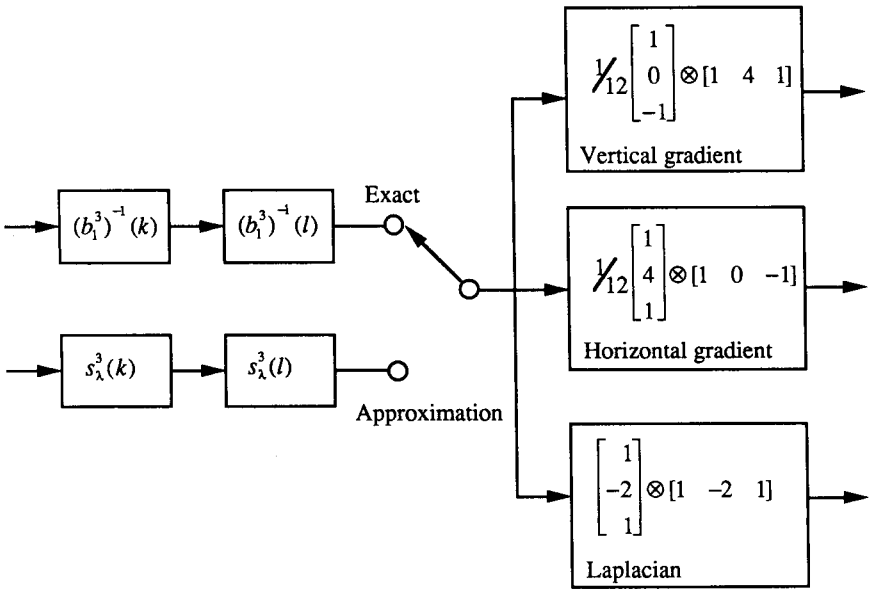

Fig. 5. Two-dimensional cubic spline gradient and Laplacian operators.

ence kernel and the sampled cubic B-spline (4.2) that performs the indirect cubic spline transformation. The operators used in the last part of this system are very similar to those used conventionally in image processing for edge detection: The Laplacian mask is identical to the most frequently used discrete Laplacian operator and the vertical and horizontal gradient masks are similar to the Sobel operators [21], [22]. The essential difference, however, is the use of a prefilter to determine the cubic B-spline coefficients. As mentioned previously, one may choose between an exact representation (direct B-spline transform) which is to be preferred for noise free data, or an approximate representation (smoothing splines) which is more robust for noisy images. The use of a least squares approximation techniques for a multiresolution feature extraction is also conceivable, in a way that is analogous to the scale-space approach described by Witkin [23].

B. Smoothing Cubic Spline Filters

The smoothing cubic spline filter is obtained by setting $r = 2$ in (4.13) in [1] yielding

$$S_\lambda^3(z) = \frac{6}{z + 4 + z^{-1} + 6\lambda(z^{-2} - 4z^{-1} + 6 - 4z + z^2)}. \tag{4.3}$$

This transfer function can be factorized into a product of complementary causal and anticausal responses

$$S_\lambda^3(z) = S^+(z)S^+(1/z)$$

with

$$S^+(z) = \frac{1 - 2\rho\cos(\omega) + \rho^2}{1 - 2\rho\cos(\omega)z^{-1} + \rho^2 z^{-2}}. \tag{4.4}$$

The quantities ρ and ω are the magnitude and argument of the two smallest complex conjugate roots ($z_1 = \rho e^{j\omega}$ and $z_2 = \rho e^{-j\omega}$) of the characteristic polynomial in the denominator of (4.3). The explicit determination of these roots is rather complicated but can be carried out analytically by making a change of variable $u = (z + z^{-1})$ and then solving a system of nested quadratic equations. After a series of lengthy calculations, we find that

$$\rho = \left(\frac{24\lambda - 1 - \sqrt{\xi}}{24\lambda}\right)\left(\frac{48\lambda + 24\lambda\sqrt{3 + 144\lambda}}{\xi}\right)^{1/2} \tag{4.5}$$

$$\tan(\omega) = \left(\frac{144\lambda - 1}{\xi}\right)^{1/2} \tag{4.6}$$

where

$$\xi = 1 - 96\lambda + 24\lambda\sqrt{3 + 144\lambda}.$$

We have checked the validity of these formulas numerically for particular values of λ. The regularization parameter λ is directly related to ρ and ω by

$$\lambda = \frac{\rho^2}{(1 - 2\rho\cos(\omega) + \rho^2)^2}. \tag{4.7}$$

It can also be shown that the impulse response of this filter is

$$s_\lambda^3(k) = c_0\rho^{|k|}\left[(\cos(\omega|k|) + \gamma\sin(\omega|k|))\right] \tag{4.8}$$

where

$$\gamma = \frac{1 - \rho^2}{1 + \rho^2}\cdot\frac{1}{\tan(\omega)}$$

and where the normalizing term c_0 is given by

$$c_0 = \frac{1 + \rho^2}{1 - \rho^2}\cdot\frac{1 - 2\rho\cos(\omega) + \rho^2}{1 + 2\rho\cos(\omega) + \rho^2}.$$

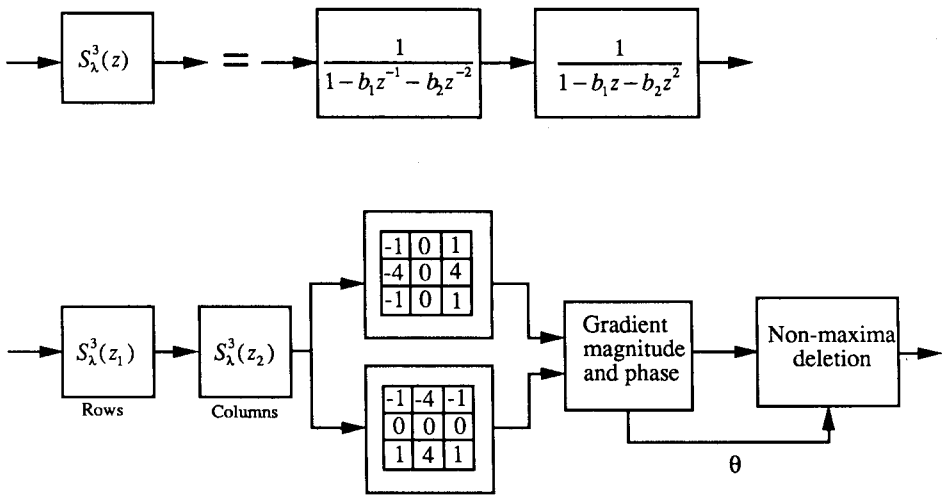

Fig. 6. Block diagram of cubic spline edge detector.

We have observed empirically that this filter resembles a normalized (zero mean and unit sum) Gaussian filter with variance

$$\sigma_w^2 \cong \sqrt{2\lambda}. \tag{4.9}$$

By taking advantage of the product decomposition (4.3) and using the same technique as in Section II, the cubic spline filter can be implemented recursively with as few as four operations per sample value. A two-dimensional version of this algorithm is outlined in the upper part of Fig. 6 with the following values for the filter coefficients: $b_1 = \rho \cos(\omega)$ and $b_2 = -\rho^2$. It should be mentioned that this smoothing cubic spline filter is very similar to the second order R-filter $W_2(z)$ derived by us in a different context using regularization theory [15]. The main difference is the presence of $B_1^3(z)$ in the denominator of (4.3) which leads to somewhat different expressions for the parameters ρ and ω.

The concept of a smoothing spline edge detector was initially proposed by Poggio *et al.* to deal with the ill-posed nature of differentiation in the presence of noise [24], [25]. These authors showed that the corresponding regularization filter is very similar to a Gaussian. A cubic spline edge detector using a similar concept and the computational techniques developed in this section is represented schematically in Fig. 6. The underlying principle, which is common to many recent edge detection schemes [26], [27], is to search for the maxima of the first-order derivative in the direction of the gradient or, equivalently, to detect the zero crossings of the second directional derivative. In our system, this task is accomplished in four steps. First, the image is filtered with a smoothing spline filter which provides a representation in terms of cubic B-spline coefficients. The parameter λ is adjusted by taking into account the noise thought to be present in the image and the minimum size of the objects or details that are of interest to the observer. Second, the sampled values of the image gradient are evaluated by convolution with the appropriate horizontal and vertical templates (cf. Fig. 5). Third, these values are converted into polar coordinates. Finally, all points that are not local maxima along a line pointing in the direction of the gradient are set to zero (nonmaxima suppression).

Interestingly enough, it can be shown (cf. Appendix B) that the block diagram in Fig. 6 is very similar to the Canny/Deriche edge detector (CDED) [10], [15], [28]. This latter operator is known to be optimal with respect to a particular criterion that takes into account both the efficiency of detection in the presence of noise and the reliability in localization [10]. The design of the edge detector that we are proposing here uses fewer operations than the fast Deriche implementation of CDED which is also recursive. We note, however, that these algorithms are not rigorously equivalent and that the design considerations are different. From a practical point of view, however, these differences have a relatively minor effect and the output of these edge detectors are essentially equivalent.

An example of edge detection for a high resolution electron micrograph of herpes simplex type II [29] is shown in Fig. 7. Typically, such micrographs (Fig. 7(a)) are quite noisy due to the use of low electron dose techniques intended to preserve the integrity of the specimen insofar as possible. As one might expect, the cubic spline edge detection algorithm (Fig. 7(d)) is superior to a simple Roberts edge detector (Fig. 7(b)) [30].

C. The Cubic Spline Pyramid

Another interesting application is the generation of a multiresolution pyramid in a way that is very similar to the techniques described in [11]–[13], [31]. A pyramid representation of a signal g is a sequence of fine-to-coarse approximations: $g_{(0)}, g_{(1)}, \cdots, g_{(N)}$, with the number of samples reduced by a factor of two in each of the principal directions from one level to the next. In the present case, we generate such a hierarchical data structure by computing a sequence of nested cubic spline least squares approximations with a decimation factor of two. The cubic

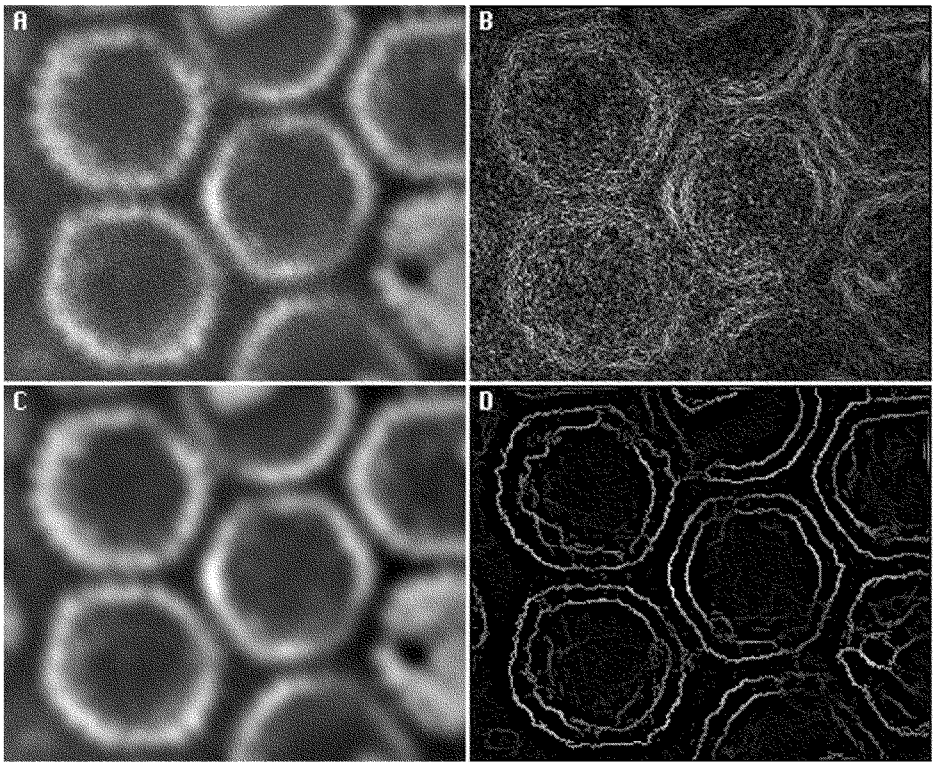

Fig. 7. Example of edge detection. (a) 226 × 184 region of interest of a digitized electron micrograph of herpes simplex virus type 2 (negatively stained), (b) output of a standard Sobel operator, (c) result of cubic spline smoothing with $\lambda = 8$ ($\sigma_w = 2$), (d) gradient magnitude after nonmaxima deflection algorithm.

spline filters relevant to this context (cf. [1, sec. IV-D]) are

$$b_2^3(k) \overset{z}{\leftrightarrow} B_2^3(z) = \frac{32 + 23[z + z^{-1}] + 8[z^2 + z^{-2}] + [z^3 + z^{-3}]}{48} \tag{4.10}$$

$$s_2^3(k) \overset{z}{\leftrightarrow} S_2^3(z) = \frac{48^2}{2212 + 1087[z + z^{-1}] + 110[z^2 + z^{-2}] + [z^3 + z^{-3}]}. \tag{4.11}$$

To insure that the different levels of the pyramid provide a lower resolution copy of the image that is visually as close as possible to the original one, we use a signal representation in terms of sampled values (cf. Section III-B), as opposed to a B-spline representation that tends to overemphasize higher frequencies (cf. Fig. 1(c)) (these two representations are equivalent and are related to each other through the B-spline transform).

We have shown in Section III-B that the standard decimation technique, which uses a prefilter followed by a down-sampler (see, for instance, [32]), could also be used for computing least squares B-spline approximations. The optimal prefilter that is required for this purpose is represented on the left side of Fig. 3(c) and is characterized by (3.2). Based on these results, the successive levels of the cubic spline pyramid are computed iteratively as

$$\begin{cases} g_{(0)}(k) = g(k) \\ g_{(i+1)}(k) = [\mathring{h}_2^3 * g_{(i)}]_{\downarrow 2}(k), \\ \qquad\qquad (i = 1, \cdots, N-1) \end{cases} \tag{4.12}$$

where the prefilter is given by

$$\mathring{h}_2^3(k) = [s_2^3 * b_1^3]_{\uparrow 2} * b_2^3(k). \tag{4.13}$$

The second equation in (4.12) defines the basic REDUCE operation. It can be verified that the transfer function of the optimal prefilter is

$$\mathring{H}_2^3(z) = \frac{8(4 + [z^2 + z^{-2}])(32 + 23[z + z^{-1}] + 8[z^2 + z^{-2}] + [z^3 + z^{-3}])}{2212 + 1087[z^2 + z^{-2}] + 110[z^4 + z^{-4}] + [z^6 + z^{-6}]}. \tag{4.14}$$

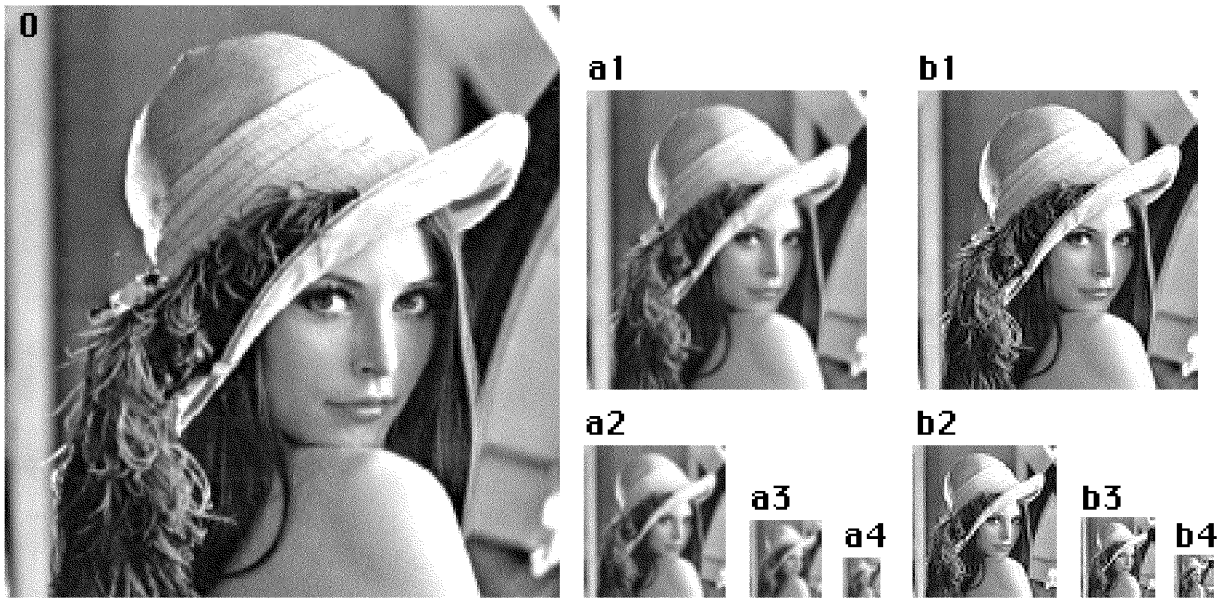

Fig. 8. Comparison of image pyramids. (0): Initial test image (level 0), (a1–b4): levels 1 to 4 of Burt's Gaussian pyramid with $a = 3/8$, (b1–b4): levels 1 to 4 of the least squares cubic spline pyramid.

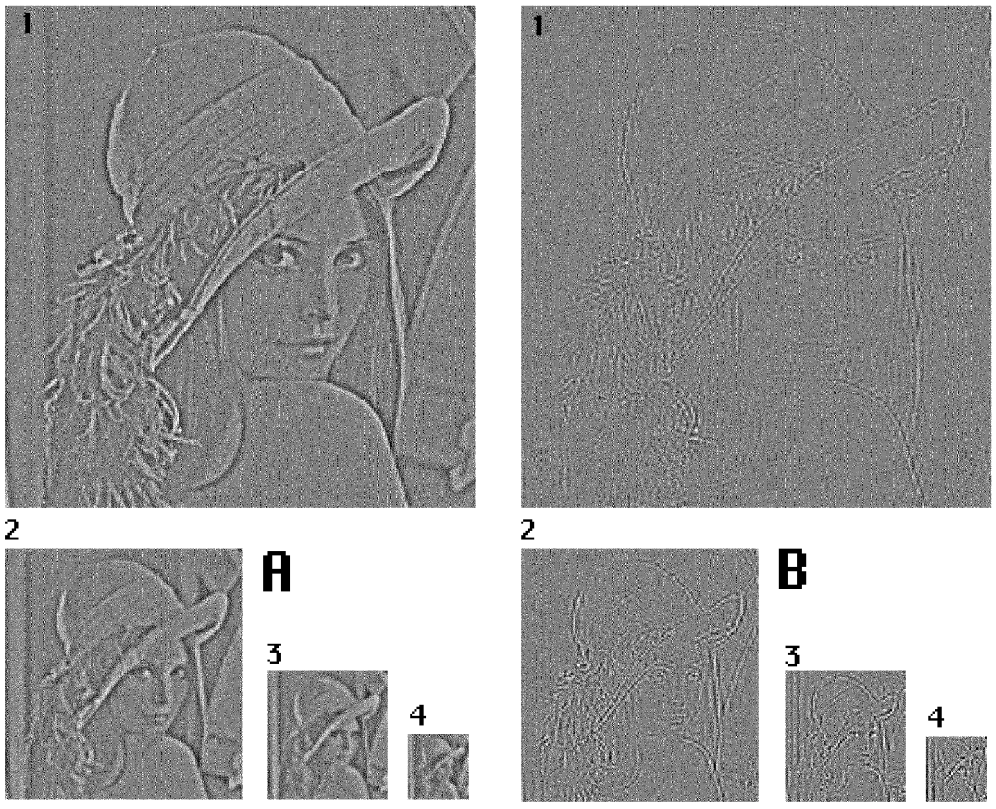

Fig. 9. Comparison of Laplacian pyramids for the image in Fig. 9-(0). (A1–4): four bottom levels the basic Laplacian pyramid with $a = 3/8$, (B1–4): four bottom levels of the cubic spline difference pyramid.

Based on inspection of its frequency response (Fig. 4(a)), it appears that this operator is essentially a low-pass filter with a slight emphasis of the higher frequencies in the bandpass region. The cubic spline pyramid of a standard test image is shown in Fig. 8. For comparison, we have also included a representation of Burt's Gaussian pyramid with $a = \frac{3}{8}$. We note that the sharpness of the least squares cubic spline pyramid (8b) is preserved at all resolution levels while the corresponding images in the Gaussian pyramid (8a) seem increasingly blurred in comparison.

Cubic splines also provide us with a complementary interpolation mechanism from any coarser level of the pyramid onto a finer sampling grid. This process is accomplished efficiently by defining an EXPAND function that

performs a cubic spline interpolation with an expansion factor 2. The operation is described as

$$\hat{g}_{(i-1)}(k) = h_2^3 * [g_{(i)}]_{\uparrow 2}(k) \tag{4.15}$$

where h_2^3 is the impulse response of a cubic spline interpolator with a zooming factor 2 and is given by

$$h_2^3(k) = [(b_1^3)^{-1}]_{\uparrow 2} * b_2^3(k). \tag{4.16}$$

The transfer function of this filter is obtained by substituting (4.2) and (4.10) in (3.3):

$$H_2^3(z) = \frac{32 + 23[z + z^{-1}] + 8[z^2 + z^{-2}] + [z^3 + z^{-3}]}{8(z^2 + 4 + z^{-2})}. \tag{4.17}$$

We note that $\hat{g}_{(i-1)}(k)$ is precisely the least squares cubic spline approximation of $g_{(i-1)}(k)$.

Burt and Crowley have introduced the concept of a Laplacian pyramid that displays the information lost during decimation in a Gaussian pyramid [12], [33]. In the present context, an analogous representation can be obtained by taking the difference between the signal at a given resolution level and its least squares cubic spline approximation reconstructed from the samples at the next coarser level:

$$\Delta g_{(i-1)}(k) = g_{(i-1)}(k) - \hat{g}_{(i-1)}(k). \tag{4.18}$$

Fig. 9 provides a comparison between Burt's Laplacian pyramid (LP) and the cubic spline difference pyramid evaluated according to (4.18). Identical intensity scaling factors were applied to all images to facilitate the comparison. For the initial LP, the amount of information lost at each level is quite significant; the initial subject is still readily recognizable. In the case of the cubic spline pyramid, the energy of the difference is reduced drastically but very high frequency artifacts are propagated in this representation, albeit with very little energy.

Based on those results, it appears that the present technique could be used to improve the performance of the coding scheme described in [12]. As multiresolution techniques are increasingly common in image processing, there may be many other potential applications including image segmentation [11], [34], edge detection [35], feature extraction, and a variety of multigrid algorithms for computer vision [36] that may be developed from the concepts we have presented.

V. Conclusion

The main objective of this series of papers has been to derive digital filtering techniques for solving the classical problems of B-spline interpolation and approximation. Efficient recursive algorithms have been exhibited that provide practical alternatives to the more standard matrix approaches commonly used in this context. The filtering approach is also interesting conceptually for the new interpretations it suggests. For instance, we have seen that a polynomial spline interpolator acts like a low-pass filter and that its impulse response is very similar to a "sinc" function. In fact, our recent theoretical results suggest that this analogy is more profound and that the classical interpolation method for bandlimited signals correspond to a polynomial spline interpolation with $n \to +\infty$ [20]. Likewise, the method of least squares B-spline approximation can be viewed as an extension of conventional sampling theory for bandlimited signals [37], [38].

Dealing with continuous signal (or image) representations may also suggest new processing algorithms, or, at least, provide a sounder theoretical underpinning for some earlier approaches. For instance, we have shown that it is relatively easy to compute quantities such as derivatives, gradients, or Laplacians by simple convolution with finite difference operators in the B-spline domain. In addition, we have found the filters for the evaluation of the cubic spline gradient and Laplacian to be very similar to certain well known image processing operators (Sobel and Laplacian). Similarly, it can be verified that the use of quadratic or bilinear splines yields a gradient estimate similar to the one used in the Roberts edge detector [30]. The use of smoothing spline or least squares approximations allows a straightforward extension of these techniques to the treatment of noisy data. As an example, we have described a smoothing cubic spline edge detection algorithm, which we have shown to be equivalent to the optimal Canny edge detector [10]. An interesting feature of the present algorithm is that its computational cost is low and is independent on the size of the smoothing window: Only 10 multiplications and 10 additions per pixel are required to compute both components of the gradient.

Finally, we have shown that the method of least squares spline approximation is well suited to the generation of scale-space or multiresolution signal representations. This concept has been illustrated with the design of a cubic spline pyramid which stands as an interesting alternative to the widely used Gaussian/Laplacian pyramid. Such least squares techniques could be useful in a variety of multi-resolution image processing algorithms.

Appendix A
Convergence of the Least Squares Filters

We recall that the least squares B-spline filter s_m^n [1, Sec. IV-D] is defined as

$$s_m^n(k) = \left([b_m^n * b_m^n]_{\downarrow m}\right)^{-1}(k). \tag{A-1}$$

To study the limiting form of this operator as m goes to infinity, we consider the limit:

$$\lim_{m \to +\infty} \left(\frac{1}{m} \sum_{l=-\infty}^{+\infty} b_m^n(l) b_m^n(k-l) \right).$$

Using the definition of the discrete B-spline kernels and defining $x = k/m$ where k is fixed, this expression is writ-

ten as

$$\lim_{m\to+\infty}\left(\frac{1}{m}\sum_{l=-\infty}^{+\infty}\beta^n(l/m)\beta^n(x-l/m)\right).$$

The term inside the bracket can be interpreted as a Riemann sum with step size: $\Delta y = 1/m$. As m goes to infinity, or equivalently, as Δy tends to zero, this sum converges to the integral:

$$\int_{-\infty}^{+\infty}\beta^n(y)\beta^n(x-y)\,dy = \beta^{2n+1}(x) \qquad \text{(A-2)}$$

which is evaluated based on the convolution property of B-spline functions [1, eq. (2.9)]. Accordingly, we have that

$$\lim_{m\to+\infty}\left(\frac{1}{m}\sum_{l=-\infty}^{+\infty}b_m^n(l)b_m^n(k-l)\right) = \lim_{m\to+\infty}\beta^{2n+1}(k/m) = \lim_{m\to+\infty}b_m^{2n+1}(k) \qquad \text{(A-3)}$$

from which, we conclude

$$\lim_{m\to+\infty}\frac{1}{m}[b_m^n * b_m^n]_{\downarrow m}(k) = \lim_{m\to+\infty}[b_m^{2n+1}]_{\downarrow m}(k) = b_1^{2n+1}(k). \qquad \text{(A-4)}$$

Appendix B
Link with the Canny/Deriche Operator

According to Deriche, the transfer function of the 1D edge detector satisfying Canny's condition of optimality with some appropriate boundary conditions is given by [28]

$$F(z) = \frac{az^{-1}}{1 + b_1z^{-1} + b_2z^{-2}} - \frac{az}{1 + b_1z + b_2z^2} \qquad \text{(B-1)}$$

with

$$a = -ce^{-\alpha}\sin(\omega)$$
$$b_1 = -2e^{-\alpha}\cos(\omega)$$
$$b_2 = e^{-2\alpha} \qquad \text{(B-2)}$$

where α and ω are the filter parameters and where c is a scaling constant. We choose to rewrite this expression as

$$F(z) = \frac{a(1-b_2)(z^{-1}-z)}{(1 + b_1z^{-1} + b_2z^{-2})(1 + b_1z + b_2z^2)}. \qquad \text{(B-3)}$$

If we define $c_1 = a(1-b_2)$ and $\rho = e^{-\alpha}$, we see that

$$F(z) = S_\lambda^3(z)(z^{-1}-z) \qquad \text{(B-4)}$$

which clearly indicates that this filter corresponds to the discrete differentiation of the smoothing cubic spline filter (Section IV-B). If we apply a smoothing spline low-pass filter, as defined by (3.1), to reduce noise in the orthogonal direction, we obtain the two-dimensional filters for the evaluation of the horizontal and vertical gradient components:

$$F_x(z_1, z_2) = S_\lambda^3(z_1)S_\lambda^3(z_2)(z_1^{-1} - z_1)B_1^3(z_2)$$
$$F_y(z_1, z_2) = S_\lambda^3(z_1)S_\lambda^3(z_2)B_1^3(z_1)(z_2^{-1} - z_2). \qquad \text{(B-5)}$$

It is clear from (B-5) that the smoothing needs to be performed only once and that these operations are equivalent to the ones performed by the block diagram in Fig. 6. In the Deriche algorithm as well as in the approach described in [15], the smoothing filter used in the orthogonal direction does not include the term $B_1^3(z)$.

References

[1] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing: Part I—Theory," *IEEE Trans. Signal Processing*, this issue, pp. 821-833.

[2] M. Unser, A. Aldroubi, and M. Eden, "Fast B-spline transforms for continuous image representation and interpolation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 13, pp. 277-285, Mar. 1991.

[3] C. H. Reinsh, "Smoothing by spline functions," *Numer. Math.*, vol. 10, pp. 177-183, 1967.

[4] K. Toraichi, S. Yang, M. Kamada, and R. Mori, "Two-dimensional spline interpolation for image reconstruction," *Patt. Recog.*, vol. 21, pp. 275-284, 1988.

[5] M. Unser, "Recursive filters for fast B-spline interpolation and compression of digital images," *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 1233, pp. 337-347.

[6] I. J. Schoenberg, "Contribution to the problem of approximation of equidistant data by analytic functions," *Quart. Appl. Math.*, vol. 4, pp. 45-99, 112-141, 1946.

[7] I. J. Schoenberg, "Cardinal interpolation and spline functions," *J. Approximation Theory*, vol. 2, pp. 167-206, 1969.

[8] I. J. Schoenberg, "Spline functions and the problem of graduation," *Proc. Nat. Acad. Sci.*, vol. 52, pp. 947-950, 1964.

[9] A. N. Tikhonov and V. Y. Arsenin, *Solution of Ill-Posed Problems*. Washington, DC: Winston and Sons, 1977.

[10] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-8, pp. 679-697, 1986.

[11] P. J. Burt, "Fast algorithms for estimating local image properties," *Comput. Graph. Image Processing*, vol. 21, pp. 368-382, 1983.

[12] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact code," *IEEE Trans. Commun.*, vol. COM-31, pp. 337-345, Apr. 1983.

[13] A. Rosenfeld, *Multiresolution Image Processing*. New York: Springer-Verlag, 1984.

[14] M. Unser and M. Eden, "Optimal FIR approximations of inverse filters and perfect reconstruction filter banks," *Signal Processing*, to be published.

[15] M. Unser, A. Aldroubi, and M. Eden, "Recursive regularization filters: Design, properties, and applications," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 13, pp. 272-277, Mar. 1991.

[16] C. de Boor, *A Practical Guide to Splines*. New York: Springer-Verlag, 1978.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 1989.

[18] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*. Cambridge, MA: Cambridge University Press, 1986.

[19] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[20] A. Aldroubi, M. Unser, and M. Eden, "Cardinal spline filters: Stability and convergence to the ideal sinc interpolator," *Signal Processing*, vol. 28, no. 2, pp. 127-138, Aug. 1992.

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

[22] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[23] A. P. Witkin, "Scale-space filtering," in *Proc. 4th Int. Joint Conf. Artificial Intell.*, 1983, pp. 1019-1022.

[24] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314-319, 1985.