2015/06/16 v0.5.1 IPOL article class

Published in Image Processing On Line on 2018–07–23.
Submitted on 2018–01–10, accepted on 2018–06–26.
ISSN 2105–1232 © 2018 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2018.221

Theory and Practice of Image B-Spline Interpolation

Thibaud Briand[1,2], Pascal Monasse[1]

[1] Université Paris-Est, LIGM (UMR CNRS 8049), ENPC, F-77455 Marne-la-Vallée, France
(thibaud.briand@enpc.fr, monasse@imagine.enpc.fr)
[2] Université Paris-Saclay, CMLA (UMR CNRS 8536), France

Communicated by Mauricio Delbracio *Demo edited by* Pascal Monasse and Thibaud Briand

Abstract

We explain how the B-spline interpolation of signals and, in particular, of images can be efficiently performed by linear filtering. Based on the seminal two-step method proposed by Unser et al. in 1991, we propose two slightly different prefiltering algorithms whose precisions are proven to be controlled thanks to a rigorous boundary handling. This paper contains all the information, theoretical and practical, required to perform efficiently B-spline interpolation for any order and any boundary extension. We describe precisely how to evaluate the kernel and to compute the B-spline interpolator parameters. We show experimentally that increasing the order improves the interpolation quality. As a fundamental application we also provide an implementation of homographic transformation of images using B-spline interpolation.

Source Code

The ANSI C99 implementation of the code that we provide is the one which has been peer reviewed and accepted by IPOL. The source code, the code documentation, and the online demo are accessible at the IPOL web part of this article[1]. Compilation and usage instructions are included in the `README.txt` file of the archive.

Keywords: interpolation; splines; linear filtering; boundary handling

1 Introduction

Interpolation consists in constructing new data points within the range of a discrete set of known data points. It is closely related to the concept of approximation [3], fitting [4], and extrapolation. In signal processing it is commonly expressed as the problem of recovering the underlying continuous signal from which the known data points are sampled. A continuous signal representation is handy when one wishes to implement numerically an operator that is initially defined in the continuous domain (e.g. edge detection, geometric transformation).

[1] https://doi.org/10.5201/ipol.2018.221

Under the assumption that the signal belongs to a given class of functions, the common principle of all interpolation schemes is to determine the parameters of the continuous image representation [6, 18]. A fundamental example is given by Shannon's sampling theory which states an equivalence between a band-limited function and its equidistant samples taken at a frequency that is superior or equal to the Nyquist rate [16]. According to the Shannon-Whittaker interpolation formula, a band-limited signal can be written as the convolution between a Dirac comb weighted by its samples and the cardinal sine (or sinc) function. However this result cannot be used in practice [20], notably because real-world signals have a bounded domain. In addition, the cardinal sine function has a decay in $1/x$ that is too slow for practical computations [17].

A non band-limited alternative approach that has been widely used since the 1990s is the spline representation. A spline of degree n is a continuous piece-wise polynomial function of degree n of a real variable with derivatives up to order $n-1$. This representation has the advantage of being equally justifiable on a theoretical and practical basis [19]. It can model the physical process of drawing a smooth curve and it is well adapted to signal processing thanks to its optimality properties. It is handy to consider the B-spline representation [15] where the continuous underlying signal is expressed as the convolution between the B-spline kernel, that is compactly supported, and the parameters of the representation, namely the B-spline coefficients. One of the strongest arguments in favor of the B-spline interpolation is that it approaches the Shannon-Whittaker interpolation as the order increases [2].

The determination of the B-spline coefficients is done in a prefiltering step which, in general, can be done by solving a band diagonal system of linear equations [8, 11]. For uniformly spaced data points, which is most of the time the case in signal processing, Unser et al. proposed in [21] an efficient and stable algorithm based on linear filtering that works for any order. More details, in particular regarding the determination of the interpolator parameters, were provided by the authors in later publications [22, 23].

As for Shannon's sampling theory, the spline representation is designed for infinite signals. Finite signals need to be extended in an arbitrary way in order to apply the interpolation scheme. This issue is commonly referred to as the boundary handling. The influence of the unknown exterior data decays with the distance to the boundary of the known data points [17] and is therefore often neglected. However in some applications this decay may be too slow or it can be relevant to consider a particular extension. Unser's algorithm is described with a symmetrical boundary handling which is a classical but restrictive choice. Additional boundary extensions are available in the implementation provided by [6]. The problem that arises in this prefiltering algorithm is that the boundary handling involves infinite sums that are approximated by truncation. Because it uses a recursive structure, the B-spline coefficients are computed *a priori* with an uncontrolled error.

In this article we provide all the information, theoretical and practical, required to perform B-spline interpolation for any order and any boundary extension. We propose two slightly different prefiltering algorithms based on Unser's algorithm but with additional computations that take into account the boundary extension. The computational errors are proven, theoretically and experimentally, to be controlled (up to dimension two) thanks to a correct boundary handling. The computational cost increases slowly with the desired precision (which can be set to the single precision in most of the applications). The first algorithm is general and works for any boundary extension while the second is applicable under specific assumptions. In addition, we describe precisely how to evaluate the B-spline kernel and to compute the B-spline interpolator parameters. We show experimentally that increasing the order improves the interpolation quality. As a fundamental application we also provide an implementation of homographic transformation of images using B-spline interpolation.

This article is accompanied by an online demo where the user can upload an image and apply a homographic transformation to it. The homography is chosen by selecting the transformation of the four corners of the image.

This article is organized as follows: Section 2 presents the B-spline interpolation theory of a discrete infinite unidimensional signal as a two-step method. Section 3 describes two prefiltering algorithms for a finite unidimensional signal whose computational errors are controlled thanks to a proper boundary handling. The extension to higher dimensions, with a particular focus to dimension two, is done in Section 4. Details concerning the provided numerical implementation are given in Section 5. Section 6 presents some experiments.

2 B-spline Interpolation of a Discrete Signal

In this section we present the B-spline interpolation of an *infinite* discrete unidimensional signal as a two-step interpolation method. We detail how the first step, i.e., the prefiltering step, can be decomposed into a cascade of exponential filters, themselves separated into two complementary causal and anti-causal components. We also explain how to evaluate by closed form formulas the B-spline values at arbitrary points.

2.1 B-spline Interpolation Theory

Let n be a non-negative integer. A spline of degree $n \geq 1$ is a continuous piece-wise polynomial function of degree n of a real variable with continuous derivatives up to order $n-1$. The junction abscissas between successive polynomials are called the *knots*.

Definition 1. *The normalized B-spline function of order n, noted $\beta^{(n)}$, is defined recursively by*

$$\beta^{(0)}(x) = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, & x = \pm\frac{1}{2} \\ 0, & \textit{otherwise} \end{cases} \qquad \textit{and for } n \geq 0, \quad \beta^{(n+1)} = \beta^{(n)} * \beta^{(0)}, \tag{1}$$

where the symbol $*$ denotes the convolution operator.

The normalized B-spline function of order n is even, compactly supported in $\mathrm{supp}(\beta^{(n)}) = [-\frac{n+1}{2}, \frac{n+1}{2}]$ and non-negative. Moreover, it has $n+2$ equally spaced knots $k \in \mathbb{Z} \cap \mathrm{supp}(\beta^{(n)})$ when n is odd and $n+2$ equally spaced knots $k \in \left(\mathbb{Z} + \frac{1}{2}\right) \cap \mathrm{supp}(\beta^{(n)})$ when n is even. An explicit formula for $\beta^{(n)}$ can be derived from its recursive definition [19] and will be used in Section 2.3. Figure 1 displays $\beta^{(n)}$ for $n = 0, \ldots, 3$.

The normalized B-spline functions are the basic atoms for constructing splines in the case of equally spaced knots. Let φ be a spline of degree n with equally spaced knots belonging to $\mathbb{Z}$ for n odd and to $\mathbb{Z} + \frac{1}{2}$ for n even. Then, as proved by Schoenberg [15], φ can be uniquely represented as the weighted sum of shifted normalized B-splines of order n, i.e.

$$\varphi(x) = \sum_{i \in \mathbb{Z}} c_i \beta^{(n)}(x - i), \tag{2}$$

where the weights $c = (c_i)_{i \in \mathbb{Z}} \in \mathbb{R}^{\mathbb{Z}}$ are called the B-spline coefficients. The spline φ is uniquely characterized by its B-spline coefficients, or equivalently by its samples $(\varphi(k))_{k \in \mathbb{Z}} \in \mathbb{R}^{\mathbb{Z}}$ at integer locations. Thus, the interpolation of a discrete signal $f \in \mathbb{R}^{\mathbb{Z}}$ by a spline of degree n, namely the B-spline interpolation of order n, can be defined as follows.

Definition 2 (B-spline interpolation)**.** *The B-spline interpolate of order n of a discrete signal $f \in \mathbb{R}^{\mathbb{Z}}$ is the spline $\varphi^{(n)}$ of degree n defined for $x \in \mathbb{R}$ by*

$$\varphi^{(n)}(x) = \sum_{i \in \mathbb{Z}} c_i \beta^{(n)}(x - i), \tag{3}$$

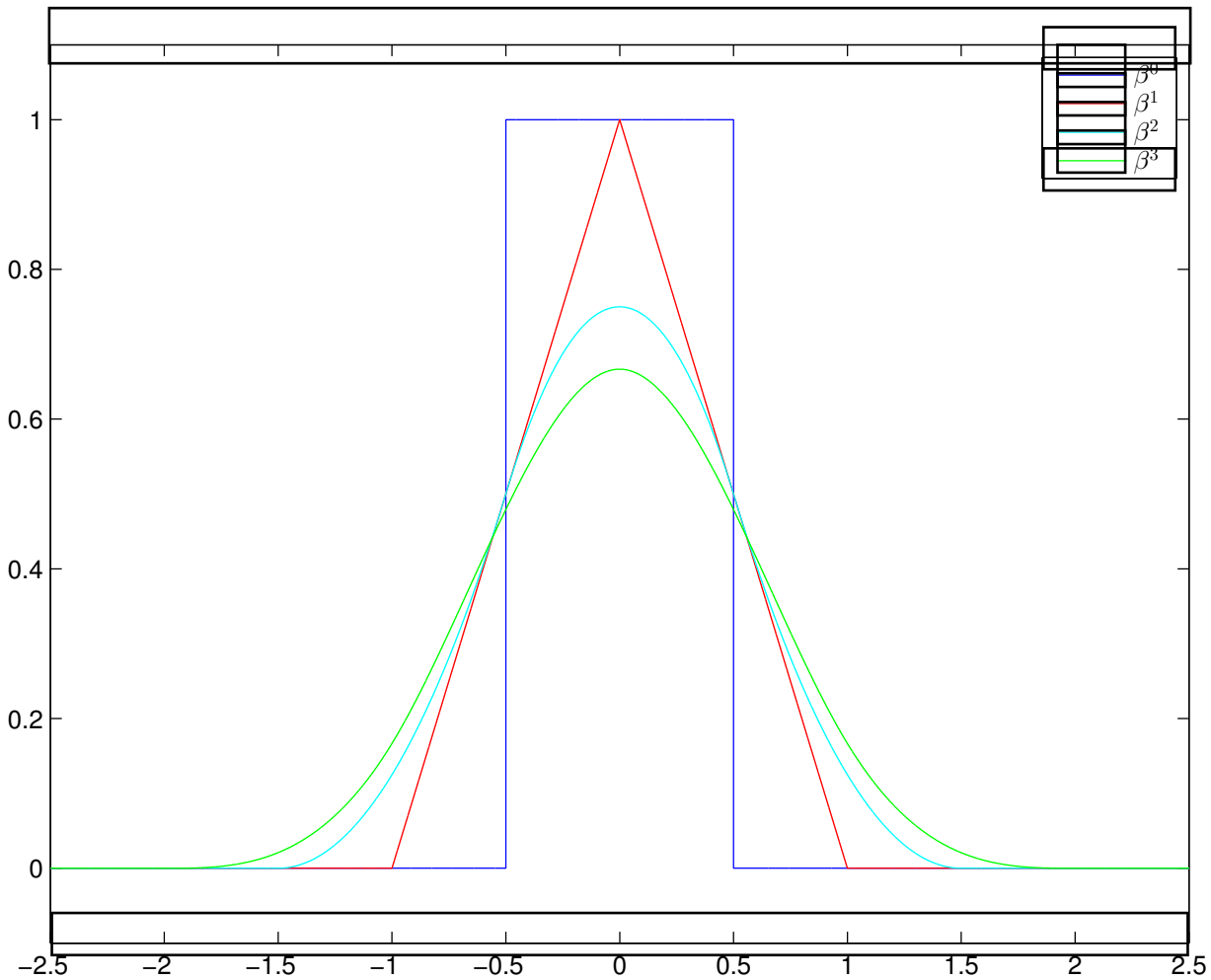


Figure 1: Normalized B-spline functions $\beta^{(n)}$ for $0 \leq n \leq 3$.

where the B-spline coefficients $c = (c_i)_{i\in\mathbb{Z}}$ are uniquely characterized by the interpolating condition

$$\varphi^{(n)}(k) = f_k, \quad \forall k \in \mathbb{Z}. \tag{4}$$

Given a signal f and a real x, computing the right hand side of (3) requires two evaluations: the signal $c = (c_i)_{i\in\mathbb{Z}}$ and $\beta^{(n)}(x-i)$. This explains the two steps involved in the computation:

- Step 1 (prefiltering or direct B-spline transform) provides a B-spline representation of the signal. The computation of the B-spline coefficients c is done in the prefiltering step detailed in Section 2.2. Except in the simplest cases, $n = 0$ or $n = 1$, in which case $\beta^{(n)}(k-i) = \delta_i(k)$ and so $c_i = f_i$, the determination of the coefficients c_i from f so as to satisfy (4) is not straightforward.
- Step 2 (indirect B-spline transform [21]) reconstructs the signal values from the B-spline representation. Given the Dirac comb of B-spline coefficients $\mathbf{c} = \sum_{i\in\mathbb{Z}} c_i\delta_i$ the value of $\varphi^{(n)}(x)$ in (3) is computed at any location $x \in \mathbb{R}$ as a convolution of $\mathbf{c}$ with the finite signal $\beta^{(n)}(x-.)$, whose computation is explained in Section 2.3.

The B-spline interpolation can be expressed also as a direct interpolation

$$\varphi^{(n)}(x) = \sum_{i\in\mathbb{Z}} f_i\, \eta^{(n)}(x-i) \quad (x \in \mathbb{R}), \tag{5}$$

where $\eta^{(n)}$ is called the cardinal spline function of order n [19, 22]. In [2] it is proven that the cardinal spline Fourier transform approaches the ideal filter (i.e., the Fourier transform of the cardinal sine)

when n goes to infinity. This result makes the link between the Shannon's sampling theory and the B-spline interpolation. For $n = 0$ and $n = 1$, we have $c_i = f_i$ and $\eta^{(n)} = \beta^{(n)}$ so that the direct and indirect methods coincide. These interpolations correspond respectively to the nearest neighbor and linear interpolation methods [6]. For $n \geq 2$, $\eta^{(n)}$ is no longer compactly supported so that the two-step representation becomes more efficient. See [18] for more information about two-step interpolation methods.

2.2 Prefiltering Step

The prefiltering step (or direct B-spline transform) consists in computing the B-spline coefficients c introduced in (3).

2.2.1 Expression as a Discrete Convolution

The interpolating condition (4) can be written as

$$\varphi^{(n)}(k) = \sum_{i \in \mathbb{Z}} c_i \beta^{(n)}(k - i) = f_k, \quad \forall k \in \mathbb{Z}. \tag{6}$$

Let us define $b^{(n)} \in \mathbb{R}^{\mathbb{Z}}$ by $b_i^{(n)} = \beta^{(n)}(i)$ for $i \in \mathbb{Z}$. Then, we recognize in (6) the convolution equation

$$c * b^{(n)} = f. \tag{7}$$

Solving for c in the latest equation is efficiently done thanks to transfer functions:

Definition 3 (Z-transform (or transfer function) [9])**.** *The Z-transform of a discrete signal $y \in \mathbb{R}^{\mathbb{Z}}$ is the formal power series $\mathcal{Z}[y]$ defined by*

$$\mathcal{Z}[y](z) = \sum_{i \in \mathbb{Z}} y_i z^{-i}. \tag{8}$$

The set of complex points for which the Z-transform summation converges is called the region of convergence (ROC). In particular, if the ROC contains the unit circle then the Z-transform can be inverted and characterizes the signal.

The advantage of introducing this transform is its property of transforming a convolution into a simple multiplication, so that inverting a convolution amounts to a division in the Z-domain:

Proposition 1 (Convolution property [9])**.** *Let $v \in \mathbb{R}^{\mathbb{Z}}$ and $w \in \mathbb{R}^{\mathbb{Z}}$ be two discrete signals. Then,*

$$\mathcal{Z}[v * w] = \mathcal{Z}[v]\mathcal{Z}[w], \tag{9}$$

on the intersection of their regions of convergence.

The finite discrete convolution kernel $b^{(n)}$ is entirely characterized by its Z-transform $B^{(n)} = \mathcal{Z}[b^{(n)}]$ (whose ROC is $\mathbb{C} \setminus \{0\}$ since $\beta^{(n)}$ is compactly supported). As $B^{(n)}$ has no zeros on the unit circle [2], the inverse operator $(b^{(n)})^{-1}$ exists and is uniquely defined by its Z-transform, noted $(B^{(n)})^{-1}$, which verifies (formally)

$$(B^{(n)})^{-1} = \mathcal{Z}[(b^{(n)})^{-1}] = \frac{1}{\mathcal{Z}[b^{(n)}]} = \frac{1}{B^{(n)}}, \tag{10}$$

and whose ROC contains the full unit circle. Finally, the prefiltering step boils down to the discrete convolution

$$c = (b^{(n)})^{-1} * f. \tag{11}$$

2.2.2 Decomposition into Elementary Filters

The filter $(b^{(n)})^{-1}$ can be decomposed into elementary causal and anti-causal filters. Using the fact that $\beta^{(n)}$ is even and supported in $\left[-\frac{n+1}{2}, \frac{n+1}{2}\right]$, denoting $\tilde{n} = \lfloor \frac{n}{2} \rfloor$, we can write

$$B^{(n)}(z) = b_0^{(n)} + \sum_{i=1}^{\tilde{n}} b_i^{(n)}(z^i + z^{-i}). \tag{12}$$

Schoenberg proved that $B^{(n)}$ has only negative (simple) zeros [14, lemma 8]. By the symmetry $B^{(n)}(z) = B^{(n)}(z^{-1})$ for $z \neq 0$, these zeros can be grouped in reciprocal pairs (α, α^{-1}). Denoting by $R^{(n)}$ the set of zeros of $B^{(n)}$ yields

$$R^{(n)} = \bigcup_{i=1}^{\tilde{n}} \{z_i, z_i^{-1}\}, \quad \text{with } -1 < z_1 < \cdots < z_{\tilde{n}} < 0. \tag{13}$$

The z_i's are called the poles of the B-spline interpolation. Their practical computation is dealt with in Appendix A. As $z^{\tilde{n}} B^{(n)}(z)$ is a polynomial whose roots are the elements of $R^{(n)}$, $B^{(n)}$ can be rewritten for $z \neq 0$ as

$$B^{(n)}(z) = b_{\tilde{n}}^{(n)} z^{-\tilde{n}} \prod_{i=1}^{\tilde{n}} (z - z_i)(z - z_i^{-1}), \tag{14}$$

which gives for $z \notin R^{(n)} \cup \{0\}$,

$$(B^{(n)})^{-1}(z) = \gamma^{(n)} \prod_{i=1}^{\tilde{n}} H(z; z_i), \tag{15}$$

where

$$\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}, \tag{16}$$

and

$$H(z; z_i) = \frac{-z_i}{(1 - z_i z^{-1})(1 - z_i z)}. \tag{17}$$

Let $-1 < \alpha < 0$, α playing the role of one z_i. Denote by $k^{(\alpha)} \in \mathbb{R}^{\mathbb{Z}}$ the causal filter and by $l^{(\alpha)} \in \mathbb{R}^{\mathbb{Z}}$ the anti-causal filter defined for $i \in \mathbb{Z}$ by

$$k_i^{(\alpha)} = \begin{cases} 0 & i < 0 \\ \alpha^i & i \geq 0 \end{cases} \quad \text{and} \quad l_i^{(\alpha)} = \begin{cases} 0 & i > 0 \\ \alpha^{-i} & i \leq 0. \end{cases} \tag{18}$$

In terms of Z-transform we have for $|z| > |\alpha|$,

$$\mathcal{Z}[k^{(\alpha)}](z) = \sum_{i=0}^{\infty} \alpha^i z^{-i} = \frac{1}{1 - \alpha z^{-1}}, \tag{19}$$

and for $|z| < |\alpha^{-1}|$,

$$\mathcal{Z}[l^{(\alpha)}](z) = \sum_{i=-\infty}^{0} \alpha^{-i} z^{-i} = \frac{1}{1 - \alpha z}. \tag{20}$$

Set $h^{(\alpha)} = -\alpha l^{(\alpha)} * k^{(\alpha)}$. The filter $h^{(\alpha)}$ is called exponential filter because it is shown, using for instance (41), that for $j \in \mathbb{Z}$,

$$h_j^{(\alpha)} = \frac{\alpha}{\alpha^2 - 1} \alpha^{|j|}. \tag{21}$$

Define the domain $D_\alpha = \{z \in \mathbb{C}, |\alpha| < |z| < |\alpha^{-1}|\}$. Applying Proposition 1 to $h^{(\alpha)}$ and combining (19), (20) and (17), we get the following equality, valid on D_α

$$\mathcal{Z}[h^{(\alpha)}] = -\alpha \mathcal{Z}[k^{(\alpha)}]\mathcal{Z}[l^{(\alpha)}] = H(.;\alpha). \tag{22}$$

Since $z_1 < \cdots < z_{\tilde{n}} < 0$ we have $D_{z_1} \subset \cdots \subset D_{z_{\tilde{n}}}$. Thus with Proposition 1 and (22) we get for $z \in D_{z_1}$,

$$\mathcal{Z}\left[\gamma^{(n)} h^{(z_{\tilde{n}})} * \cdots * h^{(z_1)}\right](z) = \gamma^{(n)} \prod_{i=1}^{\tilde{n}} H(z; z_i) = (B^{(n)})^{-1}(z). \tag{23}$$

Since D_{z_1} contains the unit circle, this yields

$$(b^{(n)})^{-1} = \gamma^{(n)} h^{(z_{\tilde{n}})} * \cdots * h^{(z_1)}, \tag{24}$$

and provides a new expression for c,

$$c = \gamma^{(n)} h^{(z_{\tilde{n}})} * \cdots * h^{(z_1)} * f. \tag{25}$$

To simplify, define recursively the signal $c^{(i)} \in \mathbb{R}^{\mathbb{Z}}$ for $i \in \{0, \ldots, \tilde{n}\}$ by

$$c^{(0)} = f \quad \text{and for } i \geq 1, \quad c^{(i)} = h^{(z_i)} * c^{(i-1)}. \tag{26}$$

We have $c = \gamma^{(n)} c^{(\tilde{n})}$. Thus the computation of the prefiltering step can be decomposed[2] into $\tilde{n}$ successive filtering steps with exponential filters that can themselves be separated into two complementary causal and anti-causal components. The corresponding algorithm for prefiltering an infinite discrete signal is presented in Algorithm 1. This is a theoretical algorithm that cannot be used in practice because it requires an infinite input signal. Turning this algorithm into a practical one, that is, applicable to a finite signal, is the subject of Section 3.

Algorithm 1: Theoretical prefiltering of an infinite signal

Input : A discrete infinite signal $f = (f_i)_{i\in\mathbb{Z}}$ and the B-spline interpolation order n
Output: The B-spline coefficients $c = (c_i)_{i\in\mathbb{Z}}$ of f

1 Compute the poles $(z_i, z_i^{-1})_{1\leq i\leq \tilde{n}}$ and the normalization coefficient $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ (Appendix A)
2 Define $c^{(0)} = f$
3 **for** $i = 1$ **to** $\tilde{n}$ **do**
4 | Compute $c^{(i)} = h^{(z_i)} * c^{(i-1)}$ where $h^{(z_i)}$ is given by (21)
5 **end**
6 Normalization: $c = \gamma^{(n)} c^{(\tilde{n})}$

2.3 Normalized B-spline Function Evaluation

The evaluation of the normalized B-spline function $\beta^{(n)}$ at any location $x \in \mathbb{R}$ is necessary in order to perform the indirect B-spline transform (see Algorithm 6 and Algorithm 7). As $\beta^{(n)}$ is continuous, even and compactly supported in $[-\frac{n+1}{2}, \frac{n+1}{2}]$, it is sufficient to evaluate $\beta^{(n)}(x)$ for $0 \leq x < \frac{n+1}{2}$. Moreover, $\beta^{(n)}$ is a piece-wise polynomial function so this evaluation can be efficiently performed after the precomputation of the polynomial coefficients between each pair of successive knots.

[2] In general the same decomposition principle is applicable to linear interpolation methods whose kernel is symmetrical and compactly supported [18].

We use the explicit expression of $\beta^{(n)}$, justified in Appendix B: for $n \geq 1$ and $x \in \mathbb{R}$,

$$\beta^{(n)}(x) = \frac{1}{n!} \sum_{i=0}^{n+1} (-1)^i \binom{n+1}{i} \left(x - i + \frac{n+1}{2} \right)_+^n, \tag{27}$$

where for all $y \in \mathbb{R}$, $y_+ = \max(y, 0)$ denotes the positive part of y. Let $0 \leq x < \frac{n+1}{2}$. Starting from (27) and by symmetry we get for $n \geq 1$,

$$\beta^{(n)}(x) = \beta^{(n)}(-x) = \frac{1}{n!} \sum_{i=0}^{n+1} (-1)^i \binom{n+1}{i} \left(\frac{n+1}{2} - i - x \right)_+^n, \tag{28}$$

To get rid of the + subscript in this equation, we observe that

$$\frac{n+1}{2} - i - x > 0 \Leftrightarrow x - \frac{n+1}{2} + i < 0 \Leftrightarrow \left\lfloor x - \frac{n+1}{2} + i \right\rfloor \leq -1 \Leftrightarrow i \leq k,$$

with

$$k = \left\lceil \frac{n+1}{2} - x \right\rceil - 1.$$

Therefore (28) can be rewritten as the restricted sum

$$\beta^{(n)}(x) = \frac{1}{n!} \sum_{i=0}^{k} (-1)^i \binom{n+1}{i} \left(\frac{n+1}{2} - i - x \right)^n. \tag{29}$$

We then expand the powers to write $\beta^{(n)}(x)$ as a sum of monomials. We observe that $0 \leq k \leq \tilde{n}$.

Polynomial expression. Define $y = \frac{n+1}{2} - x - k$, so that $0 < y \leq 1$. Using the relation $\frac{n+1}{2} - i - x = y + (k - i)$ and the binomial expansion, (29) becomes

$$\begin{aligned} \beta^{(n)}(x) &= \frac{1}{n!} \left((-1)^k \binom{n+1}{k} y^n + \sum_{i=0}^{k-1} (-1)^i \binom{n+1}{i} \sum_{j=0}^{n} \binom{n}{j} y^j (k-i)^{n-j} \right) && (30) \\ &= \frac{1}{n!} \left((-1)^k \binom{n+1}{k} y^n + \sum_{j=0}^{n} \left(\binom{n}{j} \sum_{i=0}^{k-1} (-1)^i \binom{n+1}{i} (k-i)^{n-j} \right) y^j \right). && (31) \end{aligned}$$

For $0 \leq j \leq n$ define the polynomial coefficients

$$C_{k,j}^{(n)} = \begin{cases} \binom{n}{j} \sum_{i=0}^{k-1} (-1)^i \binom{n+1}{i} (k-i)^{n-j} & 0 \leq j \leq n-1 \\ \sum_{i=0}^{k} (-1)^i \binom{n+1}{i} & j = n \end{cases}, \tag{32}$$

so that

$$\beta^{(n)}(x) = \frac{1}{n!} \sum_{j=0}^{n} C_{k,j}^{(n)} y^j. \tag{33}$$

Polynomial expression near 0. Assume $k = \tilde{n}$, i.e., $0 \leq x < \frac{n+1}{2} - \tilde{n}$. This upper bound is 0.5 for even n and 1 for odd n. Using the binomial expansion, we have

$$\begin{aligned} \beta^{(n)}(x) &= \frac{1}{n!} \sum_{i=0}^{\tilde{n}} (-1)^i \binom{n+1}{i} \sum_{j=0}^{n} (-1)^j \binom{n}{j} x^j \left(\frac{n+1}{2} - i \right)^{n-j} && (34) \\ &= \frac{1}{n!} \sum_{j=0}^{n} \left((-1)^j \binom{n}{j} \sum_{i=0}^{\tilde{n}} (-1)^i \binom{n+1}{i} \left(\frac{n+1}{2} - i \right)^{n-j} \right) x^j. && (35) \end{aligned}$$

For $0 \leq j \leq n$, define the polynomial coefficients

$$D_{\tilde{n},j}^{(n)} = (-1)^j \binom{n}{j} \sum_{i=0}^{\tilde{n}} (-1)^i \binom{n+1}{i} \left(\frac{n+1}{2} - i\right)^{n-j}$$
$$= \frac{1}{2^{n-j}} \binom{n}{j} \sum_{i=0}^{\tilde{n}} (-1)^{i+j} \binom{n+1}{i} (n+1-2i)^{n-j}, \tag{36}$$

so that

$$\beta^{(n)}(x) = \frac{1}{n!} \sum_{j=0}^{n} D_{\tilde{n},j}^{(n)} x^j. \tag{37}$$

The function $\beta^{(n)}$ is even and $(n-1)$-times differentiable, so that $\frac{d^j \beta^{(n)}}{dx^j}(0) = 0$ for any odd j such that $0 \leq j \leq n-1$. In other words, $D_{\tilde{n},j} = 0$ for $0 \leq j \leq n-1$, j odd. Applying (37) results in only $n+1-\tilde{n}$ terms (i.e., $\tilde{n}+1$ if n is even and $\tilde{n}+2$ if n is odd) in the sum instead of $n+1$ with (33).

In practice, the polynomial coefficients $(C_{l,j}^{(n)})_{0 \leq l \leq \tilde{n}-1, 0 \leq j \leq n}$ and $(D_{\tilde{n},j}^{(n)})_{0 \leq j \leq n}$ are precomputed[3] before the indirect B-spline transform, as detailed in Algorithm 2. Then, the evaluation of $\beta^{(n)}(x)$ is done by Horner's method [5].

Algorithm 2: Coefficients of the piecewise polynomial expression of $\beta^{(n)}$

Input : Order n of the B-spline
Output: The coefficients $C_{k,j}^{(n)}$ and $D_{\tilde{n},j}^{(n)}$ for $j = 0 \dots n$, $k = 0 \dots \tilde{n}-1$

1 Compute $\binom{n}{j}$ for $j = 0 \dots n$ using recursive formulas $\binom{n}{0} = 1$, $\binom{n}{j} = \frac{n-j+1}{j}\binom{n}{j-1}$ for $j = 1 \dots \tilde{n}$ and $\binom{n}{j} = \binom{n}{n-j}$ for $j > \tilde{n}$.
2 Compute $\binom{n+1}{i} = \frac{n+1}{i}\binom{n}{i-1}$ for $i = 1 \dots \tilde{n}$
3 Compute matrix P with $P_{ij} = j^i$ for $i = 0 \dots n$, $j = 1 \dots n+1$ using $P_{0,\cdot} = 1$ and $P_{i,j} = jP_{i-1,j}$ for $i \geq 1$
4 Compute $C_{k,j}^{(n)}$ for $j = 0 \dots n$, $k = 0 \dots \tilde{n}-1$ applying (32)
5 Compute $D_{\tilde{n},j}^{(n)}$ applying (36).

3 B-spline Interpolation of a Finite Signal

In practice the signal $\underline{f}$ to be interpolated is finite and discrete, i.e., $\underline{f} = (\underline{f}_i)_{0 \leq i \leq K-1}$ for a given positive integer K. There exists an infinite number of coefficients $(c_i)_{i \in \mathbb{Z}}$ satisfying the interpolating condition (6) for $0 \leq k \leq K-1$. To insure uniqueness, an arbitrary extension of $\underline{f}$ outside $\{0, \dots, K-1\}$ is necessary. B-spline interpolation theory can then be applied to the extended signal $f \in \mathbb{R}^{\mathbb{Z}}$. To simplify the notations in the following, no distinction will be made between the signal $\underline{f}$ and its extension f when there is no ambiguity.

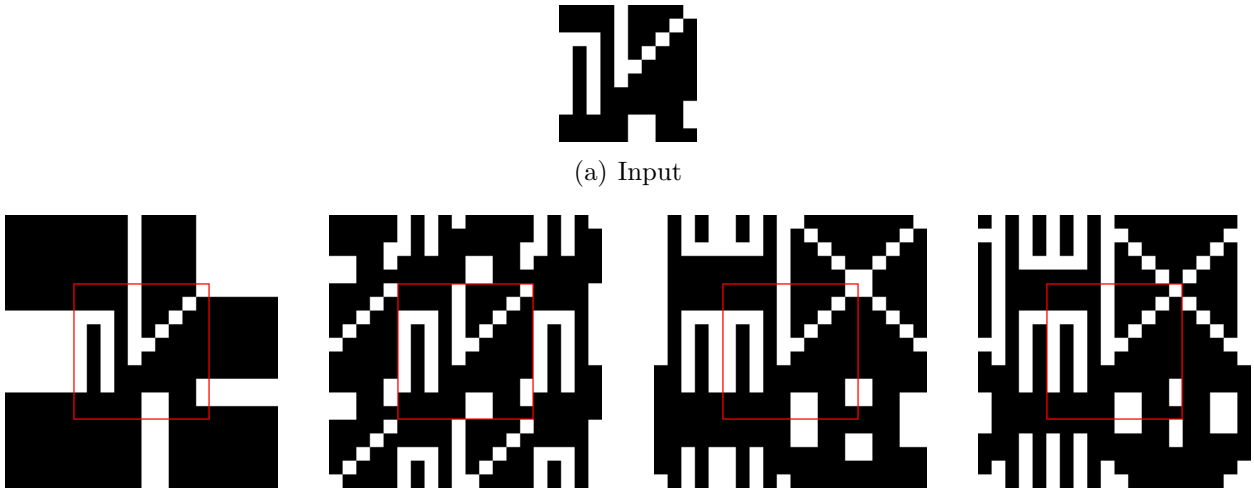
Extension on a finite domain. Let $x \in [0, K-1]$. To compute the interpolated value $\varphi^{(n)}(x)$, the indirect B-spline transform in (3) only requires the c_i's for $i \in \left(x + \left]-\frac{n+1}{2}, \frac{n+1}{2}\right[\right) \cap \mathbb{Z} \subset \{-\tilde{n}, \dots, K-1+\tilde{n}\}$. In addition, even though the value f_k for $k \in \mathbb{Z}$ contributes to every B-spline coefficient c_i, this contribution vanishes when $|k-i|$ tends to infinity. As presented in the following, to compute $\varphi^{(n)}(x)$

[3] Note that [12] provides a recursive algorithm, that is not used in our implementation, for computing these coefficients.

with a relative precision ε it is sufficient to extend the signal to a finite domain $\{-L^{(n,\varepsilon)}, \dots, K-1+L^{(n,\varepsilon)}\}$ where $L^{(n,\varepsilon)}$ is a positive integer that only depends on the B-spline order n and the desired precision ε. The precision is relative to the values of the signal. A relative precision of ε means that the error committed is less than $\varepsilon \sup_{k\in\mathbb{Z}} |f_k| = \varepsilon \|f\|_\infty$. In practice the images are large enough so that $L^{(n,\varepsilon)} < K$ and it is possible to express the extension as a boundary condition[4] around 0 and $K-1$. The most classical boundary condition choices are summarized in Table 1 and represented in Figure 2.

Extension	Signal $abcde$
Constant	$aaa\|abcde\|eee$
Half-symmetric	$cba\|abcde\|edc$
Whole-symmetric	$dcb\|abcde\|dcb$
Periodic	$cde\|abcde\|abc$

Table 1: Classical boundary extensions of the signal $abcde$ by $L = 3$ values.



(a) Input

(b) Constant (c) Periodic (d) Half-symmetric (e) Whole-symmetric

Figure 2: Classical extensions by $L = 5$ values of a binary image of size 10×10. The boundary of the original image is colored in red.

Prefiltering computation using the extension. As presented in Algorithm 1, for computing the B-spline coefficients using the prefiltering decomposition given in (25), only the first exponential filter $h^{(z_1)}$ is applied directly to $f = c^{(0)}$. Therefore, for $i \geq 1$ the intermediate filtered signals $c^{(i)}$ are known *a priori* only where they are computed. Considering this, the two following approaches are proposed in order to perform the prefiltering.

- **Approach 1** (See Section 3.2.1). The intermediate filtered signals $c^{(i)}$ are computed in a larger domain than $\{-\tilde{n}, \dots, K-1+\tilde{n}\}$. This works with any extension.
- **Approach 2** (See Section 3.2.2). The extension, expressed as a boundary condition, is chosen so that it is transmitted after the application of each exponential filter $h^{(z_i)}$. The intermediate filtered signals $c^{(i)}$ (and the B-spline coefficients c) verify the same boundary condition and only need to be computed in $\{0, \dots, K-1\}$.

[4]Otherwise the extension is obtained by iterating the boundary condition.

In the rest of this section, we first detail the general method for computing an exponential filter application on a finite domain. Then we propose two algorithms, corresponding to both above-mentioned approaches, for computing the B-spline coefficients of a finite signal with a given precision. Finally we present the simple algorithm for performing the indirect B-spline transform, i.e., for evaluating the interpolated values.

3.1 Application of the Exponential Filters

Let $s \in \mathbb{R}^{\mathbb{Z}}$ be an infinite discrete signal. Let $-1 < \alpha < 0$ and $L_{\text{ini}} < L_{\text{end}}$. The application of the exponential filter $h^{(\alpha)} = -\alpha l^{(\alpha)} * k^{(\alpha)}$ to the signal s is computed in the domain of interest $\{L_{\text{ini}}, \ldots, L_{\text{end}}\}$ as follows.

Causal filtering. To simplify the notation we set $s^{(\alpha)} = k^{(\alpha)} * s$ so that $h^{(\alpha)} * s = -\alpha l^{(\alpha)} * s^{(\alpha)}$. Given the initialization

$$s^{(\alpha)}_{L_{\text{ini}}} = \left(k^{(\alpha)} * s\right)_{L_{\text{ini}}} = \sum_{i=0}^{+\infty} \alpha^i s_{L_{\text{ini}}-i}, \tag{38}$$

the application of the causal filter $k^{(\alpha)}$ to s can be computed recursively from $i = L_{\text{ini}}+1$ to $i = L_{\text{end}}$ according to the recursion formula

$$s^{(\alpha)}_i = s_i + \alpha s^{(\alpha)}_{i-1}. \tag{39}$$

Anti-causal filtering initialization. With a simple partial fraction decomposition we can rewrite $H(z;\alpha)$, the Z-transform of $h^{(\alpha)}$ introduced in (17), as

$$H(z;\alpha) = \frac{\alpha}{\alpha^2 - 1}\left(\frac{1}{1-\alpha z^{-1}} + \frac{1}{1-\alpha z} - 1\right). \tag{40}$$

Thus $h^{(\alpha)}$ can also be written as

$$h^{(\alpha)} = \frac{\alpha}{\alpha^2 - 1}\left(k^{(\alpha)} + l^{(\alpha)} - \delta_0\right), \tag{41}$$

and we have

$$h^{(\alpha)} * s = \frac{\alpha}{\alpha^2 - 1}\left(k^{(\alpha)} * s + l^{(\alpha)} * s - s\right). \tag{42}$$

This last formula provides[5] an expression for the initialization of the (renormalized) anti-causal filtering,

$$\left(h^{(\alpha)} * s\right)_{L_{\text{end}}} = \frac{\alpha}{\alpha^2 - 1}\left(s^{(\alpha)}_{L_{\text{end}}} + \left(l^{(\alpha)} * s\right)_{L_{\text{end}}} - s_{L_{\text{end}}}\right), \tag{43}$$

where

$$\left(l^{(\alpha)} * s\right)_{L_{\text{end}}} = \sum_{i=0}^{+\infty} \alpha^i s_{L_{\text{end}}+i}. \tag{44}$$

Anti-causal filtering. The renormalized anti-causal filtering is computed recursively from $i = L_{\text{end}} - 1$ to $i = L_{\text{ini}}$ according to the following formula,

$$\begin{aligned}
\left(h^{(\alpha)} * s\right)_i &= \left(-\alpha l^{(\alpha)} * s^{(\alpha)}\right)_i && (45)\\
&= -\alpha\left(s^{(\alpha)}_i + \alpha\left(l^{(\alpha)} * s^{(\alpha)}\right)_{i+1}\right) && (46)\\
&= \alpha\left(\left(h^{(\alpha)} * s\right)_{i+1} - s^{(\alpha)}_i\right). && (47)
\end{aligned}$$

[5] It also provides another way of applying the exponential filter that is not considered because of its higher complexity.

Approximation of the initialization values. The two infinite sums in (38) and (44) cannot be computed numerically. Let N be a non-negative integer. The initialization values are approximated by truncating the sums at index N so that

$$\left(k^{(\alpha)} * s\right)_{L_{\text{ini}}} \simeq \sum_{i=0}^{N} \alpha^i s_{L_{\text{ini}}-i}, \tag{48}$$

and

$$\left(l^{(\alpha)} * s\right)_{L_{\text{end}}} \simeq \sum_{i=0}^{N} \alpha^i s_{L_{\text{end}}+i}. \tag{49}$$

Algorithm. The general method for computing the application of the exponential filter $h^{(\alpha)}$ to a discrete signal in a finite domain is summarized in Algorithm 3. Note that it is sufficient to know the signal in the domain $\{L_{\text{ini}} - N, \dots, L_{\text{end}} + N\}$. It consists of $6(N+1)+4(L_{\text{end}} - L_{\text{ini}})$ operations.

Algorithm 3: Application of the exponential filter $h^{(\alpha)}$ to a discrete signal

Input : A pole $-1 < \alpha < 0$, a range of indices $L_{\text{ini}} < L_{\text{end}}$, a truncation index N and a discrete signal s (whose values are known in $\{L_{\text{ini}} - N, \dots, L_{\text{end}} + N\}$)

Output: The filtered signal $h^{(\alpha)} * s$ at indices $\{L_{\text{ini}}, \dots, L_{\text{end}}\}$

```
Compute s^(α)_{L_ini} using (48)
for i = L_ini + 1 to L_end do
    Compute s^(α)_i using (39)
end
Compute (l^(α) * s)_{L_end} using (49)
Compute (h^(α) * s)_{L_end} using (43)
for i = L_end − 1 to L_ini do
    Compute (h^(α) * s)_i using (47)
end
```

3.2 Prefiltering of a Finite Signal

Define $(\mu_j)_{1\le j\le \tilde{n}}$ by[6]

$$\begin{cases} \mu_1 = 0 \\ \mu_k = \left(1 + \frac{1}{\log|z_k| \sum_{i=1}^{k-1} \frac{1}{\log|z_i|}}\right)^{-1}, \quad 2 \le k \le \tilde{n}. \end{cases} \tag{50}$$

Let $\varepsilon > 0$. Define for $1 \le i \le \tilde{n}$,

$$N^{(i,\varepsilon)} = \left\lfloor \frac{\log\left(\varepsilon \rho^{(n)} (1 - z_i)(1 - \mu_i) \prod_{j=i+1}^{\tilde{n}} \mu_j\right)}{\log|z_i|} \right\rfloor + 1, \tag{51}$$

where

$$\rho^{(n)} = \left(\prod_{j=1}^{\tilde{n}} \frac{1 + z_j}{1 - z_j}\right)^2. \tag{52}$$

[6]The definition of $(\mu_j)_{1\le j\le \tilde{n}}$ is justified in Appendix C.3.

We propose two algorithms for computing the B-spline coefficients of a finite signal with precision ε. In both cases they are computed by successively applying the exponential filters to the intermediate filtered signals using Algorithm 3 with the truncation indices $(N^{(i,\varepsilon)})_{1\leq i\leq \tilde{n}}$. The difference between both algorithms lies in the computation domains. The choice for the truncation indices guarantees a precision ε for $n \leq 16$, as stated in the next theorem.

Theorem 1. *Assume $n \leq 16$. Let $\varepsilon > 0$ and f be a finite signal of length at most 4 (arbitrarily extended to $\mathbb{Z}$). The computation of the B-spline coefficients of f using Algorithm 4 or Algorithm 5 with the truncation indices $(N^{(i,\varepsilon)})_{1\leq i\leq \tilde{n}}$ has a precision of ε, i.e., the error committed is less than $\varepsilon\|f\|_\infty$.*

Proof. See Appendix C.1. □

Notice that $N^{(i,\varepsilon)} = O(\log \varepsilon)$, which shows that the truncation indices remain moderate even for high precision specification ε. However, a problematic factor is $\log|z_i|$ in the denominator, which increases the indices when z_i is close to -1. This is not unexpected, since a root with modulus close to 1 involves a slowly decaying exponential filter.

3.2.1 Approach 1: Extended Domain

The first approach for computing the prefiltering coefficients with precision ε consists in computing the intermediate filtered signals $c^{(i)}$ in a larger domain than $\{-\tilde{n}, \ldots, K-1+\tilde{n}\}$. For $0 \leq j \leq \tilde{n}$ define

$$L_j^{(n,\varepsilon)} = \tilde{n} + \sum_{i=j+1}^{\tilde{n}} N^{(i,\varepsilon)}, \tag{53}$$

which can be computed recursively using

$$\begin{cases} L_{\tilde{n}}^{(n,\varepsilon)} & = \tilde{n}, \\ L_j^{(n,\varepsilon)} & = L_{j+1}^{(n,\varepsilon)} + N^{(j+1,\varepsilon)}, \quad j = \tilde{n}-1 \text{ to } 0. \end{cases} \tag{54}$$

The input signal $f = c^{(0)}$ is first extended to $\{-L_0^{(n,\varepsilon)}, \ldots, K-1+L_0^{(n,\varepsilon)}\}$. Then, for $i = 1$ to $\tilde{n}$, $c^{(i)}$ is computed in $\{-L_i^{(n,\varepsilon)}, \ldots, K-1+L_i^{(n,\varepsilon)}\}$ from the values of $c^{(i-1)}$ in $\{-L_{i-1}^{(n,\varepsilon)}, \ldots, K-1+L_{i-1}^{(n,\varepsilon)}\}$ using Algorithm 3 with truncation index $N^{(i,\varepsilon)}$. The prefiltering algorithm on a larger domain is presented in Algorithm 4.

3.2.2 Approach 2: Transmitted Boundary Condition

The second approach for computing the prefiltering coefficients with precision ε consists in extending the input signal with a boundary condition that is transmitted after the application of the exponential filters. Assume that the $c^{(i)}$ for $0 \leq i \leq \tilde{n}$ share the same boundary condition. Then, $c^{(i)}$ can be computed at any index from the values of $c^{(i-1)}$ in $\{0, \ldots, K-1\}$ using Algorithm 3 (with truncation index $N^{(i,\varepsilon)}$). Indeed, we notice that the initialization values in (48) and (49) only depend on the values of $c^{(i-1)}$ in $\{0, \ldots, K-1\}$. The prefiltering algorithm with a transmitted boundary condition is described in Algorithm 5.

Note that the initialization values in (38) and (44) can be expressed as weighted sums of the $c_k^{(i-1)}$ for $k \in \{0, \ldots, K-1\}$, where the weights depend on k, the poles z_i and the boundary condition. For specific boundary conditions the weights, and therefore the initialization values, can be exactly computed. However these explicit expressions are not used in practice because they involve sums of K terms, i.e., more computation.

Algorithm 4: Prefiltering algorithm on a larger domain

Input : A finite discrete signal f of length K, a boundary extension, a B-spline interpolation order $n \leq 16$ and a precision ε

Output: The B-spline coefficients c of order n at indices $\{-\tilde{n}, \dots, K-1+\tilde{n}\}$ with precision ε

Precomputations

Compute the poles $(z_i, z_i^{-1})_{1 \leq i \leq \tilde{n}}$ and the normalization coefficient $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ (Appendix A).

for $1 \leq i \leq \tilde{n}$ **do**

  Compute the truncation index $N^{(i,\varepsilon)}$ using (51)

end

Define $L_{\tilde{n}}^{(n,\varepsilon)} = \tilde{n}$

for $j = \tilde{n} - 1$ **to** 0 **do**

  Compute $L_j^{(n,\varepsilon)} = L_{j+1}^{(n,\varepsilon)} + N^{(j+1,\varepsilon)}$

end

Prefiltering the signal:

Set $c_k^{(0)} = f_k$ for $k \in \{-L_0^{(n,\varepsilon)}, \dots, K-1+L_0^{(n,\varepsilon)}\}$ using the boundary extension

for $i = 1$ **to** $\tilde{n}$ **do**

  Compute $c_k^{(i)} = \left(h^{(z_i)} * c^{(i-1)}\right)_k$ for $k \in \{-L_i^{(n,\varepsilon)}, \dots, K-1+L_i^{(n,\varepsilon)}\}$ using Algorithm 3 with truncation index $N^{(i,\varepsilon)}$

end

Renormalize $c = \gamma^{(n)} c^{(\tilde{n})}$

Algorithm 5: Prefiltering algorithm with a transmitted boundary condition

Input : A finite discrete signal f of length K, a boundary condition that is transmitted, a B-spline interpolation order $n \leq 16$ and a precision ε

Output: The B-spline coefficients c of order n at indices $\{0, \dots, K-1\}$ with precision ε

Precomputations:

Compute the poles $(z_i, z_i^{-1})_{1 \leq i \leq \tilde{n}}$ and the normalization coefficient $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ (Appendix A).

for $1 \leq i \leq \tilde{n}$ **do**

  Compute the truncation index $N^{(i,\varepsilon)}$ using (51)

end

Prefiltering of the signal:

Set $c_k^{(0)} = f_k$ for $k \in \{0, \dots, K-1\}$

for $i = 1$ **to** $\tilde{n}$ **do**

  Compute $c_k^{(i-1)}$ for $k \in \{-N^{(i,\varepsilon)}, \dots, -1\} \cup \{K, \dots, K-1+N^{(i,\varepsilon)}\}$ using the boundary condition

  Compute $c_k^{(i)} = \left(h^{(z_i)} * c^{(i-1)}\right)_k$ for $k \in \{0, \dots, K-1\}$ using Algorithm 3 with truncation index $N^{(i,\varepsilon)}$

end

Renormalize $c = \gamma^{(n)} c^{(\tilde{n})}$

Particular cases. Among the four classical boundary extensions presented in Table 1, the periodic, half-symmetric and whole-symmetric boundary conditions are transmitted after the application of an exponential filter. For the periodic extension the filtered signal by any filter always remains periodic. For the half-symmetric and whole-symmetric extensions it is a consequence of the symmetry of the exponential filters. However, this property is not satisfied for the constant extension, so that

Algorithm 5 is not applicable in this case.

For the three boundary conditions that are transmitted, the anti-causal initialization value in (43) can be computed without using (49) as follows. Let s be a finite signal of length K and $-1 < \alpha < 0$. We recall that for the two symmetrical extensions the signal is extended to $\mathbb{Z}$ by periodization.

- **Periodic extension.** As s is a K-periodic signal then $s^{(\alpha)} = k^{(\alpha)} * s$ is also K-periodic. Noting N the truncation index, we can write

$$\begin{aligned}
\left(l^{(\alpha)} * s^{(\alpha)}\right)_{K-1} &\simeq \sum_{i=0}^{N} s^{(\alpha)}_{K-1+i}\alpha^i && (55)\\
&= s^{(\alpha)}_{K-1} + \alpha \sum_{i=0}^{N-1} s^{(\alpha)}_{K+i}\alpha^i && (56)\\
&= s^{(\alpha)}_{K-1} + \alpha \sum_{i=0}^{N-1} s^{(\alpha)}_{i}\alpha^i. && (57)
\end{aligned}$$

It yields,

$$\left(h^{(\alpha)} * s\right)_{K-1} \simeq -\alpha \left(s^{(\alpha)}_{K-1} + \alpha \sum_{i=0}^{N-1} s^{(\alpha)}_{i}\alpha^i \right). \quad (58)$$

- **Half-symmetric extension.** For $i \geq 0$ we have $s_{K+i} = s_{K-1-i}$ so that we can write

$$\begin{aligned}
\left(l^{(\alpha)} * s\right)_{K-1} &= s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K+i}\alpha^i && (59)\\
&= s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K-1-i}\alpha^i && (60)\\
&= s_{K-1} + \alpha s^{(\alpha)}_{K-1}. && (61)
\end{aligned}$$

It yields with (43),

$$\left(h^{(\alpha)} * s\right)_{K-1} = \frac{\alpha}{\alpha - 1} s^{(\alpha)}_{K-1} = \frac{\alpha}{\alpha - 1} \left(k^{(\alpha)} * s\right)_{K-1}. \quad (62)$$

- **Whole-symmetric extension.** For $i \geq 0$ we have $s_{K+i} = s_{K-2-i}$ so that we can write

$$\begin{aligned}
\left(l^{(\alpha)} * s\right)_{K-1} &= s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K+i}\alpha^i && (63)\\
&= s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K-2-i}\alpha^i && (64)\\
&= s_{K-1} + \alpha s^{(\alpha)}_{K-2}. && (65)
\end{aligned}$$

It yields with (43),

$$\left(h^{(\alpha)} * s\right)_{K-1} = \frac{\alpha}{\alpha^2 - 1} \left(s^{(\alpha)}_{K-1} + \alpha s^{(\alpha)}_{K-2} \right) = \frac{\alpha}{\alpha^2 - 1} \left(\left(k^{(\alpha)} * s\right)_{K-1} + \alpha \left(k^{(\alpha)} * s\right)_{K-2} \right). \quad (66)$$

In practice, when one of these three boundary conditions is used in Algorithm 5, a slightly different version of Algorithm 3 is called in Line 9. The boundary extension is added to the input list and the computation of $\left(h^{(\alpha)} * s\right)_{K-1}$ (see Line 5 and Line 6) is done using (58), (62) or (66) according to the extension case. Note that the anti-causal initialization value admits an exact expression for the two symmetric extensions. As presented in Appendix D exact expressions also hold for the causal initializations of the three extensions and for the anti-causal initialization with periodic extension. However they are not considered because they involve more computations.

3.3 Indirect B-spline Transform: Computation of the Interpolated Value

The indirect B-spline transform reconstructs the signal values from the B-spline representation. Given the B-spline coefficients c in $\{-\tilde{n}, \dots, K-1+\tilde{n}\}$, the interpolated value $\varphi^{(n)}(x)$ can be computed with precision ε, using (3), as the convolution of $\sum_{i=-\tilde{n}}^{K-1+\tilde{n}} c_i \delta_i$ with the compactly supported function $\beta^{(n)}$. The computation of the indirect B-spline transform at location $x \in [0, K-1]$ is presented in Algorithm 6. Assume the B-spline coefficients are computed with precision ε using either Algorithm 4 or Algorithm 5 then $\varphi^{(n)}(x)$ is also computed with precision ε because $\sum_{k\in\mathbb{Z}} \beta^{(n)}(x-k) \leq 1$. We recall that in Algorithm 5 the B-spline coefficients are known in $\{-\tilde{n}, \dots, -1\} \cup \{K, \dots, K-1+\tilde{n}\}$ thanks to the boundary condition.

Algorithm 6: Indirect B-spline transform

Input : A finite discrete signal f of length K, a B-spline interpolation order n, the corresponding B-spline coefficients c in $\{-\tilde{n}, \dots, K-1+\tilde{n}\}$ and $x \in [0, K-1]$

Output: The interpolated value $\varphi^{(n)}(x)$

1 $x_0 \leftarrow \lceil x - (n+1)/2 \rceil$
2 Initialize $\varphi^{(n)}(x) \leftarrow 0$
3 **for** $k = 0$ **to** $\max(n, 1)$ **do**
4 | Update $\varphi^{(n)}(x) \leftarrow \varphi^{(n)}(x) + c_k \beta^{(n)}(x - (x_0 + k))$
5 **end**

The computation of (3) at a point x involves only a finite sum. Noting $r = (n+1)/2$ the radius of the support of $\beta^{(n)}$, we have for $n \geq 1$

$$\begin{aligned} \beta(x-k) > 0 &\Leftrightarrow -r < x - k < r \\ &\Rightarrow \lceil x - r \rceil \leq k \leq \lfloor x + r \rfloor. \end{aligned}$$

In general, we have $\lfloor x+r \rfloor - \lceil x-r \rceil + 1 = 2r = n+1$ except when $x \pm r$ is an integer. In the latter case, we have only $2r-2$ terms since $\beta^{(n)}(x-.)$ vanishes at the bounds $k = x \pm r$. For $n = 0$, $\beta^{(n)}$ is special because it does not vanish at the bounds of its support, and in that case up to 2 terms are involved. In any case, we have at most $\max(1, n) + 1$ terms.

4 Extension to Higher Dimensions

The one-dimensional B-spline interpolation theory (Section 2) and practical algorithms (Section 3) are easily extended to higher dimensions by using tensor-product basis functions. The multi-dimensional prefiltering is performed by applying the unidimensional prefiltering successively along each dimension. The indirect B-spline transform is efficiently computed as the convolution with a separable and compactly supported function.

4.1 Definitions

Note d the dimension. The Normalized B-spline function of order n is defined in dimension d as follows.

Definition 4. *The Normalized B-spline function of order n and dimension d, noted $\beta^{(n,d)}$, is defined for $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ by*

$$\beta^{(n,d)}(x) = \prod_{j=1}^{d} \beta^{(n)}(x_j). \tag{67}$$

Then, the B-spline interpolate of order n of a discrete signal $f \in \mathbb{R}^{\mathbb{Z}^d}$ can be naturally defined as follows.

Definition 5. *The B-spline interpolate of order n of a discrete signal $f \in \mathbb{R}^{\mathbb{Z}^d}$ is the function $\varphi^{(n)} : \mathbb{R}^d \mapsto \mathbb{R}$ defined for $x \in \mathbb{R}^d$ by*

$$\varphi^{(n)}(x) = \sum_{i \in \mathbb{Z}^d} c_i \beta^{(n,d)}(x - i), \tag{68}$$

where the B-spline coefficients $c = (c_i)_{i \in \mathbb{Z}^d}$ are uniquely defined by the interpolation condition

$$\varphi^{(n)}(k) = f_k, \quad \forall k \in \mathbb{Z}^d. \tag{69}$$

4.2 Prefiltering Decomposition

The B-spline interpolation in dimension d is also a two-step interpolation method. The prefiltering step is decomposed as follows. Define $b^{(n,d)} = \prod_{j=1}^{d} b^{(n,d,j)}$ where for $1 \leq j \leq d$ and $k = (k_1, \ldots, k_d) \in \mathbb{Z}^d$, $b_k^{(n,d,j)} = b_{k_j}$. Then, the interpolating condition is rewritten as

$$f = c * b^{(n,d)}. \tag{70}$$

Using the separability of $b^{(n,d)}$ and Z-transform based arguments as in Section 2.2.2, the prefiltering can be expressed as the filtering

$$c = \left(b^{(n,d)}\right)^{-1} * f, \tag{71}$$

where for $k = (k_1, \ldots, k_d) \in \mathbb{Z}^d$

$$\left(\left(b^{(n,d)}\right)^{-1}\right)_k = \prod_{j=1}^{d} \left(\left(b^{(n)}\right)^{-1}\right)_{k_j}. \tag{72}$$

The prefiltering filter $\left(b^{(n,d)}\right)^{-1}$ being separable, the B-spline coefficients c can be computed by filtering f successively along each dimension by $\left(b^{(n)}\right)^{-1}$. In other words, the multi-dimensional prefiltering is decomposed in successive unidimensional prefilterings along each dimension.

4.3 Algorithms in 2D

A particular and interesting case is given by $d = 2$ where the finite discrete signals to be interpolated are images. The B-spline coefficients are obtained by applying successively the unidimensional prefiltering on the columns and on the rows.

More precisely, let $g \in \mathbb{R}^{\mathbb{Z}^2}$ and $(i,j) \in \mathbb{Z}^2$. We denote $C^j(g) \in \mathbb{R}^{\mathbb{Z}}$ the j-th column of g so that $C^j(g)_i = g_{i,j}$. Similarly, we denote $R^i(g) \in \mathbb{R}^{\mathbb{Z}}$ the i-th row of g so that $R^i(g)_j = g_{i,j}$. Define $c_{\text{col}}(f) \in \mathbb{R}^{\mathbb{Z}^2}$, the unidimensional prefiltering of the columns of f, by their columns

$$C^j(c_{\text{col}}(f)) = C^j(f) * (b^{(n)})^{-1}. \tag{73}$$

Then, the B-spline coefficients c are given by the unidimensional prefiltering of the lines of c_{col} i.e.

$$R^i(c) = R^i(c_{\text{col}}(f)) * (b^{(n)})^{-1}. \tag{74}$$

In practice the images are finite and an arbitrary extension has to be chosen. Let f be an image of size $K \times L$. In order to compute interpolated values in $[0, K-1] \times [0, L-1]$ the B-spline coefficients c of f have to be computed in $\{-\tilde{n}, \dots, K-1+\tilde{n}\} \times \{-\tilde{n}, \dots, L-1+\tilde{n}\}$. According to (73) and (74) it is done by applying Algorithm 4 or Algorithm 5 successively on the columns and on the rows.

Theorem 2. *Assume $n \leq 16$. Let $\varepsilon > 0$ and f be a finite image of size at most 4 along each dimension (arbitrarily extended to $\mathbb{Z}^2$). Denote $\varepsilon' = \frac{\varepsilon \rho^{(n)}}{2}$. The computation of the B-spline coefficients of f by applying Algorithm 4 or Algorithm 5 successively on the columns and on the rows with truncation indices $(N^{(i,\varepsilon')})_{1 \leq i \leq \tilde{n}}$ (as in Algorithm 8) have a precision of ε.*

Proof. See Appendix C.2 □

Theorem 2 provides a control of the error that is committed during the two-dimensional prefiltering. Note that to insure a precision ε using Algorithm 4, $c_{\text{col}}(f)$ has to be computed in $\{-\tilde{n}, \dots, K-1+\tilde{n}\} \times \{-L_0^{(n,\varepsilon')}, \dots, L-1+L_0^{(n,\varepsilon')}\}$ i.e. for the columns indexed by $j \in \{-L_0^{(n,\varepsilon')}, \dots, L-1+L_0^{(n,\varepsilon')}\}$. In Figure 3 is displayed, for the four classical boundary conditions, the extended image used during the prefiltering (using Algorithm 4) for a precision of $\varepsilon = 10^{-8}$ i.e. 8 digits and $n = 11$. The image is extended of $L_0^{(n,\varepsilon')} = 125$ pixels from its boundary.

The two-dimensional indirect B-spline transform is efficiently performed, as described in Algorithm 7, as a convolution with the separable and compactly supported function $\beta^{(n,2)}$. Finally, the two-dimensional B-spline interpolation of an image is presented in Algorithm 8.

5 Numerical Implementation Details

In this section we explain how computations are performed in practice. For any order n, algorithms for computing the poles along with the normalization constant and evaluating the B-spline function are detailed.

5.1 Provided Implementation

In the provided implementation of Algorithm 8, the 2D B-spline interpolation can be performed for $0 \leq n \leq 16$. The order limitation is due to numerical errors in the computation of the poles and the kernel evaluation. We replace $\beta^{(n)}$ by $n!\beta^{(n)}$ and $\gamma^{(n)}$ by

$$\gamma'^{(n)} = \frac{\gamma^{(n)}}{n!} = \begin{cases} 2^n & n \text{ even} \\ 1 & n \text{ odd.} \end{cases}, \tag{75}$$

This prevents numerical errors that could occur when n is large because of the useless renormalization by $n!$. The following entities are precomputed:

- the poles $(z_i)_{1 \leq i \leq \tilde{n}}$ as described in Apprendix A,



(a) Constant (b) Periodic



(c) Half-symmetric (d) Whole-symmetric

Figure 3: Extended image used during the prefiltering using Algorithm 4 for $\varepsilon = 10^{-8}$ i.e. 8 digits and $n = 11$. The image is extended of $L_0^{(n,\varepsilon')} = 125$ pixels from its boundary.

- the normalized B-spline coefficients $(C_{l,j})_{0 \leq l \leq \tilde{n}, 0 \leq j \leq n}$ as described in Section 2.3,
- the normalization constant $\gamma'^{(n)}$ defined in (75),
- the truncation indices $(N^{(i,\varepsilon')})_{1 \leq i \leq \tilde{n}}$ defined in (51) with $\varepsilon' = \frac{\rho^{(n)}\varepsilon}{2}$.

The first three items are tabulated for $n \leq 11$. For information purposes, $B^{(n)}$, $\gamma^{(n)}$, $\gamma'^{(n)}$ and the corresponding poles are displayed in Table 2 for $2 \leq n \leq 7$. Note that the computations are performed in double-precision floating-point format to prevent from round-off error. Multi-channel images, and in particular color images, are handled by applying the prefiltering algorithm on each channel independently, which gives the multi-channel B-spline coefficients. Then, the interpolated values are computed by applying the indirect B-spline transform to the multi-channel B-spline coefficients.

Algorithm 7: Two-dimensional indirect B-spline transform

Input : An image f of size $K \times L$, a B-spline interpolation order n, the corresponding B-spline coefficients c in $\{-\tilde{n}, \dots, K-1+\tilde{n}\} \times \{-\tilde{n}, \dots, L-1+\tilde{n}\}$ and a location $(x, y) \in [0, K-1] \times [0, L-1]$

Output: The interpolated value $\varphi^{(n)}(x, y)$

$x0 \leftarrow \lceil x - (n+1)/2 \rceil$
for $k = 0$ **to** $\max(1, n)$ **do**
  Tabulate xBuf$[k] \leftarrow \beta^{(n)}(x - (x0 + k))$
end
Initialize $\varphi^{(n)}(x, y) \leftarrow 0$
for $l = 0$ **to** $\max(1, n)$ **do**
  $y0 \leftarrow \lceil y - (n+1)/2 \rceil$
  Initialize $s \leftarrow 0$
  for $k = 0$ **to** $\max(1, n)$ **do**
    Update $s \leftarrow s + c_{x0+k, y0+l}$ xBuf$[k]$
  end
  Update $\varphi^{(n)}(x, y) \leftarrow \varphi^{(n)}(x, y) + s\, \beta^{(n)}(y - (y0 + l))$
end

Algorithm 8: Two-dimensional B-spline interpolation

Input : An image f of size $K \times L$, an extension method, a B-spline interpolation order $n \leq 16$, a precision ε and a list of pixel locations $(x_j, y_j)_{1 \leq j \leq J} \in ([0, K-1] \times [0, L-1])^J$

Output: The interpolated values $(\varphi^{(n)}(x_j, y_j))_{j \in J}$ (with precision ε)

Compute $\varepsilon' = \frac{\rho^{(n)} e}{2}$.
Compute with precision ε' the unidimensional prefiltering of the columns of f, noted c_{col}, using Algorithm 4 or Algorithm 5
Compute with precision ε' the unidimensional prefiltering of the rows of c_{col}, noted c, using Algorithm 4 or Algorithm 5
for $j = 1$ **to** J **do**
  Compute, from the B-spline coefficient c, the interpolated value $\varphi^{(n)}(x_j, y_j)$ using Algorithm 7
end

Homographic transformation. In the field of computer vision homographies [7, 13] are widely used to relate images of a scene assimilable to planar surfaces (or when the camera motion is a rotation around the optical center). As a fundamental application of Algorithm 8 we provide an implementation of homographic (or projective) transformation of images. Given a 2D homography h and an image f of size $K \times L$, the homographic transformation of f by h is the image f_h of size $K' \times L'$ verifying

$$\forall (i, j) \in \{0, \dots, K'-1\} \times \{0, \dots, L'-1\}, \quad (f_h)_{i,j} = f\left(h^{-1}(i, j)\right). \tag{76}$$

It is done by applying Algorithm 8 at locations $(h^{-1}(i, j))_{(i,j) \in \{0,\dots,K'-1\} \times \{0,\dots,L'-1\}}$. In the provided implementation the output image has the same size as the input, i.e., $K' = K$ and $L' = L$. If the location $h^{-1}(i, j)$ does not belong to $[0, K-1] \times [0, L-1]$ the value $(f_h)_{i,j}$ is arbitrarily set to 0 to avoid extrapolation. The implementation inputs are:

- an image f,

n	$B^{(n)}$	poles
2	$(z^{-1}+6+z)/8$	-0.1715728752538099
3	$(z^{-1}+4+z)/6$	-0.26794919243112281
4	$(z^{-2}+76z^{-1}+230z+76z+1)/384$	-0.36134122590021989
		-0.013725429297339109
5	$(z^{-2}+26z^{-1}+66z+26z+1)/120$	-0.4305753470999743
		-0.043096288203264443
6	$(z^{-3}+722z^{-2}+10543z^{-1}+23548+10543z+722z^2+z^3)/46080$	-0.48829458930303893
		-0.081679271076238694
		-0.0014141518083257976
7	$(z^{-3}+120z^{-2}+1191z^{-1}+2416+1191z+120z^2+z^3)/5040$	-0.53528043079643672
		-0.12255461519232777
		-0.0091486948096082266

Table 2: $B^{(n)}$ and the (approximate values of the) poles for $2 \leq n \leq 7$.

- a homography h,
- a B-spline order $n \in \{0, \ldots, 16\}$,
- one of the four classical extensions of Table 1,
- a desired precision ε,
- a choice between the two proposed prefiltering algorithms (larger or exact domain).

5.2 Online Demo

This article is accompanied by an online demo where the user can upload an image and apply to it an homographic transformation. The choice of the homography is made by selecting the images of the four corners of the image. In Figure 4 we display an example of the online demo use which corresponds to the homographic transformation of the 512×512 *Lena* image by the homography h defined by

$$\begin{cases} h(0,0) & = (25,13) \\ h(0,511) & = (11,500) \\ h(511,0) & = (480,12) \\ h(511,511) & = (468,482). \end{cases} \tag{77}$$

We recall that the pixels outside $[0,511]^2$ are arbitrarily set to 0.

6 Experiments

In this section we make several experiments using Algorithm 8. The input image used is a standard test image, *Lena*, a gray-level image of size 512×512.

6.1 Computational Cost

The computational cost of the B-spline interpolation depends on the order n, the size of the input signal (and its dimension), the desired precision ε and the number of interpolated values. The



Figure 4: Example of the online demo use. The *Lena* image is transformed by the homography h defined in (77). We use order 11, the half-symmetric boundary condition, $\varepsilon = 10^{-6}$ and the exact domain.

total length of the extension has a dependency with respect to n and ε that is not straightforward so that it is difficult to express the complexity of the prefiltering step in general. It is given by $2L_0^{(n,\varepsilon)} = 2\left(\tilde{n} + \sum_{i=1}^{\tilde{n}} N^{(i,\varepsilon)}\right)$ in dimension one and $2L_0^{(n,\varepsilon')}$ in dimension two. We display in Table 3 and Table 4 the values of $2L_0^{(n,\varepsilon)}$ and $2L_0^{(n,\varepsilon')}$ for different order and precision values. We notice that the values are slightly greater in dimension two. Assuming that the length of the extension has the same order of magnitude as the input length we obtain in dimension one and two the complexities, independent of ε, presented in Table 5. The complexity of the indirect B-spline transform corresponds to the computation of a single interpolated value.

We verified empirically how the computation time of each step depends on the B-spline order n for a homographic transformation. As it depends neither on the extension choice nor on the homography, we took the half-symmetric extension and the identity. In Figure 5 we display the computation times for $\varepsilon = 10^{-6}$. The prefiltering in the larger domain is more costly than the prefiltering in the same domain and the difference increases with the order. However in any case the prefiltering cost remains negligible with respect to the indirect B-spline transform cost. The strong increase between $n = 11$ and $n = 12$ in the indirect B-spline transform cost comes from the non-tabulation of the kernel.

$n \setminus \varepsilon$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
2	8	12	14	16	20	22	24	28	30	32	34
3	12	14	18	22	26	28	32	36	40	42	46
4	18	24	30	36	40	46	52	58	62	68	74
5	22	28	34	42	48	56	62	70	76	84	90
6	30	38	46	56	66	74	82	90	100	110	118
7	32	42	54	64	76	84	96	106	116	126	138
8	40	54	66	78	90	104	116	130	140	154	166
9	46	58	72	90	102	116	130	144	162	172	188
10	54	68	88	104	118	134	152	168	186	202	218
11	58	76	94	110	132	148	168	188	204	222	242

Table 3: Total length of extension $2L_0^{(n,\varepsilon)}$ for unidimensional signals in function of the order n and the precision ε.

$n \setminus \varepsilon$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
2	10	14	16	18	20	24	26	28	32	34	36
3	14	18	22	24	28	32	36	38	42	46	48
4	20	26	30	38	42	48	52	60	64	70	76
5	24	32	38	44	52	58	64	72	80	86	92
6	32	40	48	58	68	76	86	94	106	112	120
7	36	48	56	68	78	88	98	110	120	132	142
8	44	56	70	80	96	110	120	132	144	158	172
9	50	64	78	92	106	122	134	150	164	180	192
10	56	76	90	108	126	140	158	172	190	206	222
11	64	82	100	118	138	154	172	190	210	228	246

Table 4: Total length of extension $2L_0^{(n,\varepsilon')}$ for two-dimensional signals in function of the order n and the precision ε.

	1D	2D
Prefiltering	$O(nK)$	$O(2nKL)$
Indirect B-spline transform	$O(n^2)$	$O(n^3)$

Table 5: Complexity of the B-spline interpolation algorithms for a signal of length K or an image of size $K \times L$ using order n. The complexity of the indirect B-spline transform corresponds to the computation of a single interpolated value.

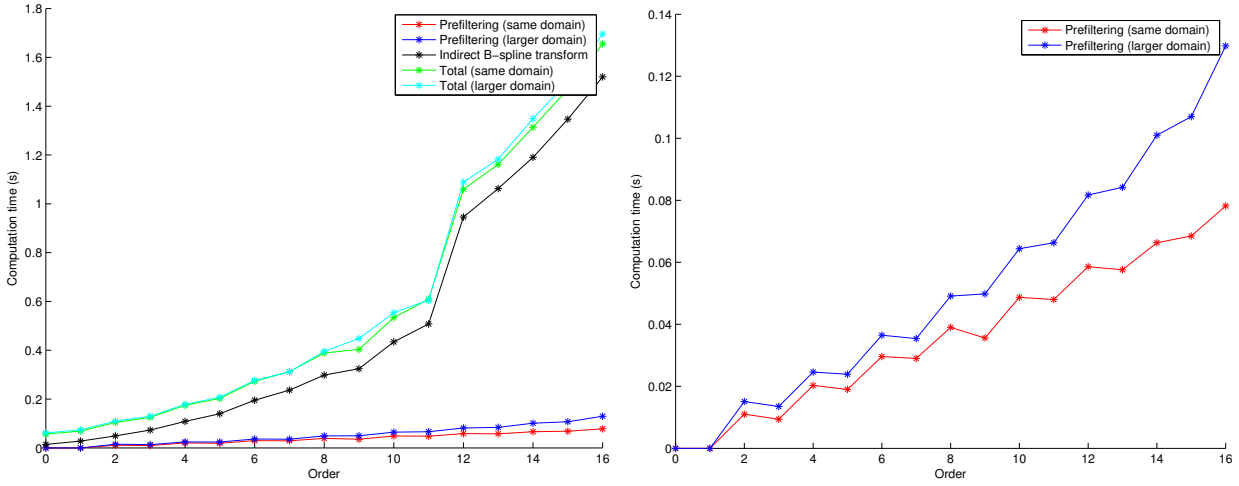
6.2 Computation Error

The computation error committed during the prefiltering step is estimated by checking if the interpolation condition (69) is verified. In practice it is done by comparing the initial image f with its homographic transformation by the identity f_{Id}.

We lead the computations for:

- the four boundary extensions of Table 1,
- any order $n \in \{2, \ldots, 16\}$,
- any precision $\varepsilon \in \{10^{-2}, \ldots, 10^{-12}\}$,
- the two prefiltering algorithms (only on the larger domain for the constant extension).

In all cases the computation error is less than ε i.e. $\|f - f_{\mathrm{Id}}\|_\infty \leq \varepsilon$. It empirically shows that the result of Theorem 2 is verified.



(a) Global (b) Prefiltering

Figure 5: Computation time of the different B-spline interpolation steps for a homographic transformation of the 512×512 gray-level image *Lena* with $\varepsilon = 10^{-6}$. The jump between $n = 11$ and $n = 12$ is due to the kernel not being tabulated in the code above order 11.

Which precision ε should be used? There is no recommended choice for ε because it depends on the context of application. There is clearly a trade-off between precision and computational cost. As the prefiltering step cost is negligible with respect to the indirect B-spline transform cost (see previous section) when a large amount of pixel values are interpolated, the computational cost aspect should not be taken into account by the user. For instance if the interpolated values are stored in single-precision floating-point format, the value $\varepsilon = 10^{-6}$ should be considered.

6.3 Zoom at the Boundary

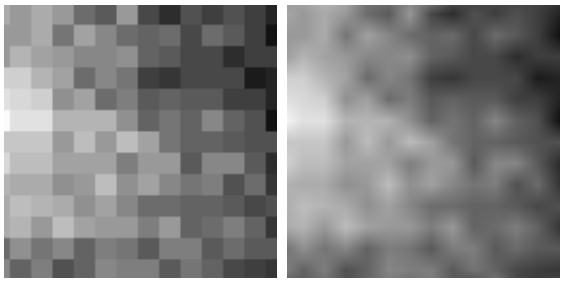
By zooming at one of the boundaries of an image using the B-spline interpolation we are able to highlight the influence of the boundary extension. We performed a zoom by a factor 20 using various orders and boundary conditions. It is computed using $\varepsilon = 10^{-6}$ and the prefiltering is done on a larger domain. In Figure 6 we display a small part of size 256×256 of the zoomed images that corresponds to the center of the right boundary. We see more and more details as the order increases. The boundary condition influence can be seen by comparing the right part of images. As an example we display in Figure 7 the comparison between the small images corresponding to order 16.

6.4 Evolution of the Results with the Order of Interpolation

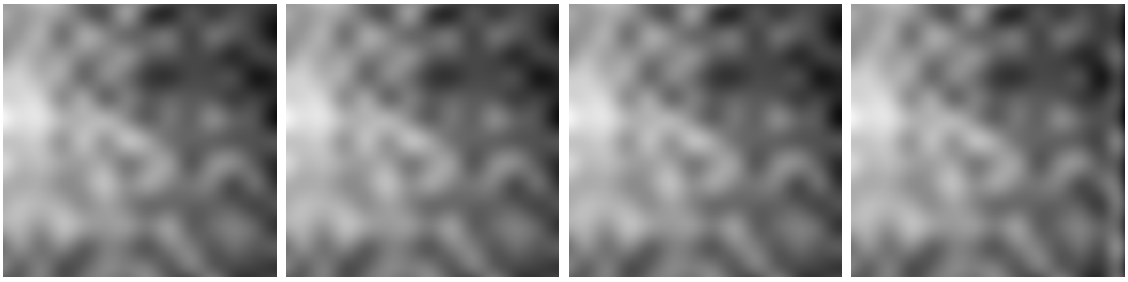
In this part, we analyze experimentally the evolution of the B-spline interpolation results with the order n. It is commonly admitted that the results of the cubic ($n = 3$) B-spline interpolation are sufficient and that increasing the order does not lead to significant improvements. We show the contrary and recommend, except when efficiency is primordial, to use a higher order B-spline interpolation. In the following, the computations are done using $\varepsilon = 10^{-6}$ and the prefiltering on a larger domain.

6.4.1 Comparison to the Shannon-Whittaker Interpolation

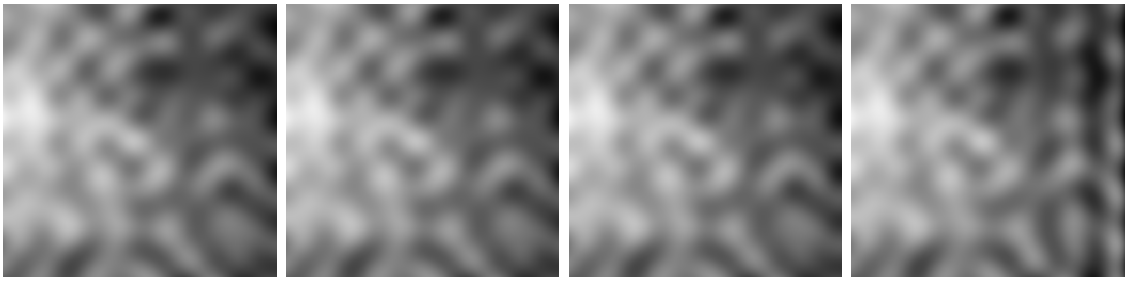
The B-spline interpolation approaches the Shannon-Whittaker interpolation when the order goes to infinity [2]. We empirically highlight this result by comparing the homographic transformations of



(a) Order 0 and order 1

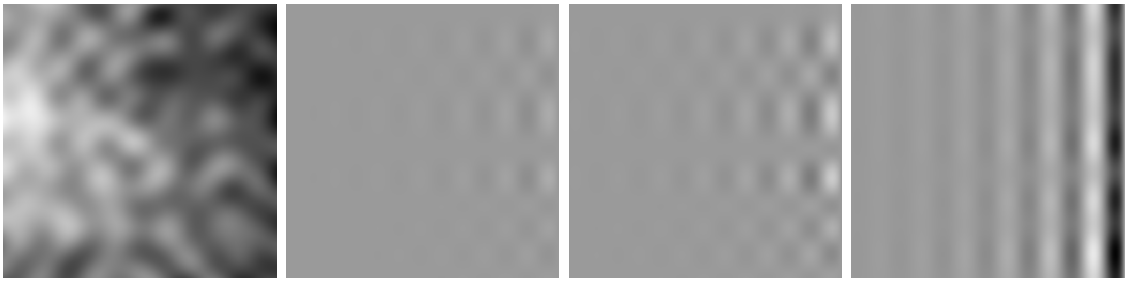


(b) Order 3



(c) Order 16

Figure 6: Zoom by a factor 20 (crop of size 256×256 centered in the middle of the right boundary) using B-spline interpolation for various orders and boundary conditions. It is computed using $\varepsilon = 10^{-6}$ and the prefiltering on a larger domain. From the left to the right we use the constant, half-symmetric, whole-symmetric and periodic extension. For orders 0 and 1 the result are the same with the four extensions. We see more and more details as the order increases. The boundary condition influence can be seen by comparing the right part of images. The affine transformation $y = 9x - 1080$ is applied to these images before visualization.



(a) Half-symmetric (b) Difference with constant (c) Difference with whole-symmetric (d) Difference with periodic

Figure 7: Comparison between the results of Figure 6 for order 16. The difference is made with the half-symmetric extension result. The affine transformation $y = 22x + 154$ is applied to the difference images before visualization.

the *Lena* image by h (defined by (77)) obtained using respectively the B-spline interpolation and the Shannon-Whittaker interpolation. The comparison is done by computing the root mean square error (RMSE) between the central parts of the two resampled images. As the boundary extension choice has no influence we choose the periodic extension. The Shannon-Whittaker interpolate is then a trigonometric polynomial whose coefficients are obtained thanks to the discrete Fourier transform (DFT) coefficients [1]. It is sampled at locations $h^{-1}(i,j)$ using the nonequispaced fast Fourier transform (NFFT) algorithm [10], which is a much slower algorithm than the B-spline interpolation. The decay of the error difference with the order n is visible in Figure 8.

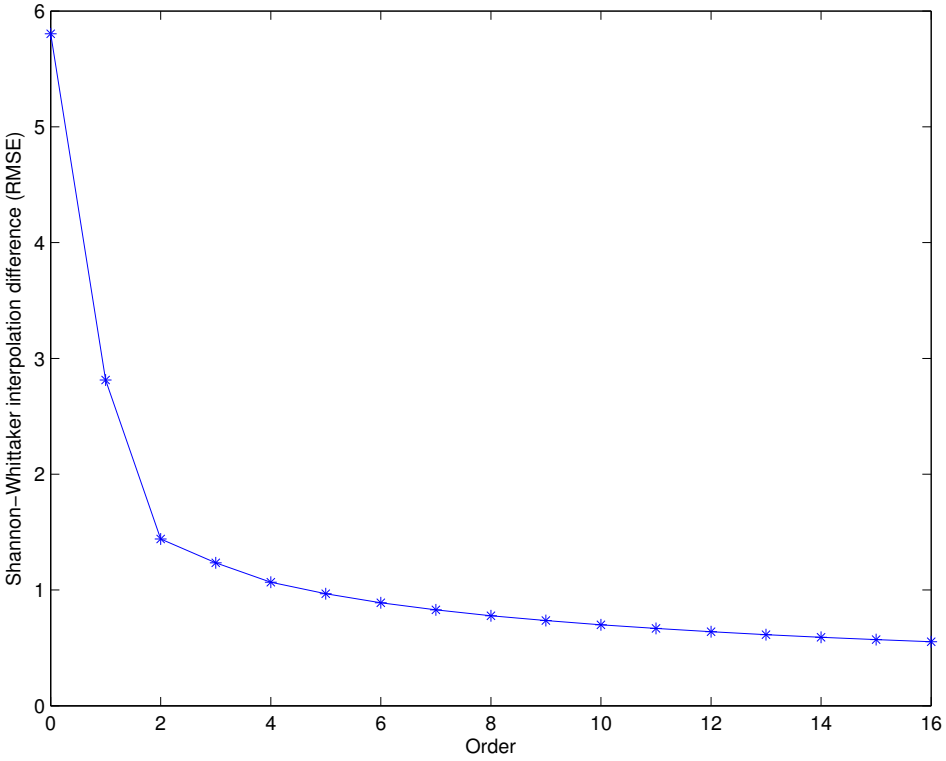


Figure 8: Decay of the difference between the Shannon-Whittaker interpolation and the B-spline interpolation for $n \in \{0,\dots,16\}$.

6.4.2 Consistency of the B-spline Interpolation

In order to study the consistency of the B-spline interpolation we applied ten shifts of 0.1 pixels (in the horizontal direction) and then one shift of -1 pixel. The consistency measurement is then given as the RMSE between the central parts of the initial and output images. As in practice the boundary condition influence on this measurement is negligible, we arbitrarily chose to use the half-symmetric boundary condition. The decay of the consistency measurement with the order n is displayed in Figure 9. It justifies the choice of a high order B-spline interpolation ($n = 11$ for instance) while it is commonly admitted that the cubic B-spline interpolation is sufficient. Note that the computational error is negligible with respect to the model error. In Figure 10 we display for $n \in \{0, 1, 3, 16\}$ the central parts of the difference images and the corresponding discrete Fourier transform modulus. The differences are localized in the high frequencies were the model error is higher. For $n = 0$ it's exactly the gradient of the image.

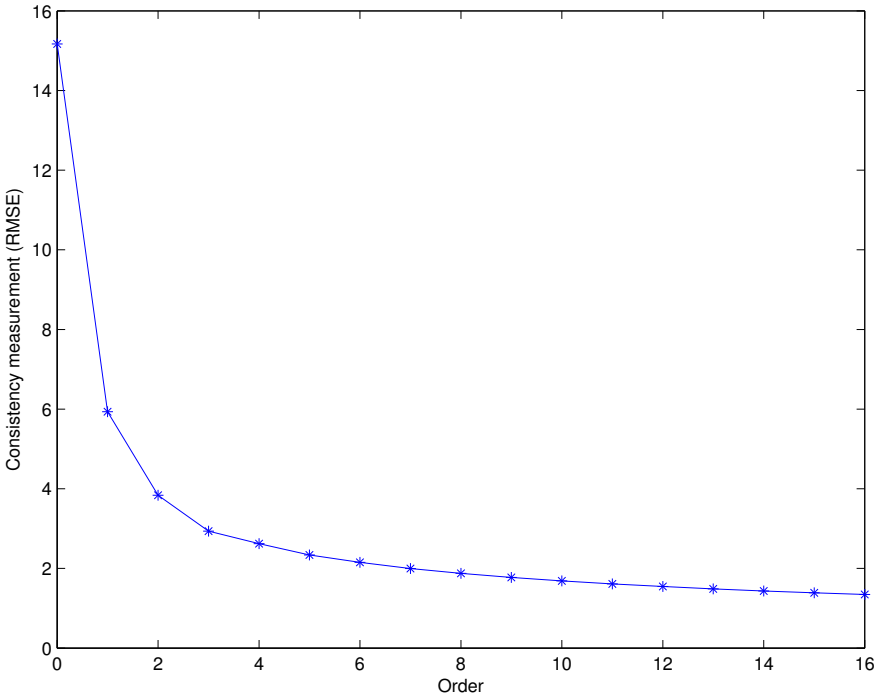
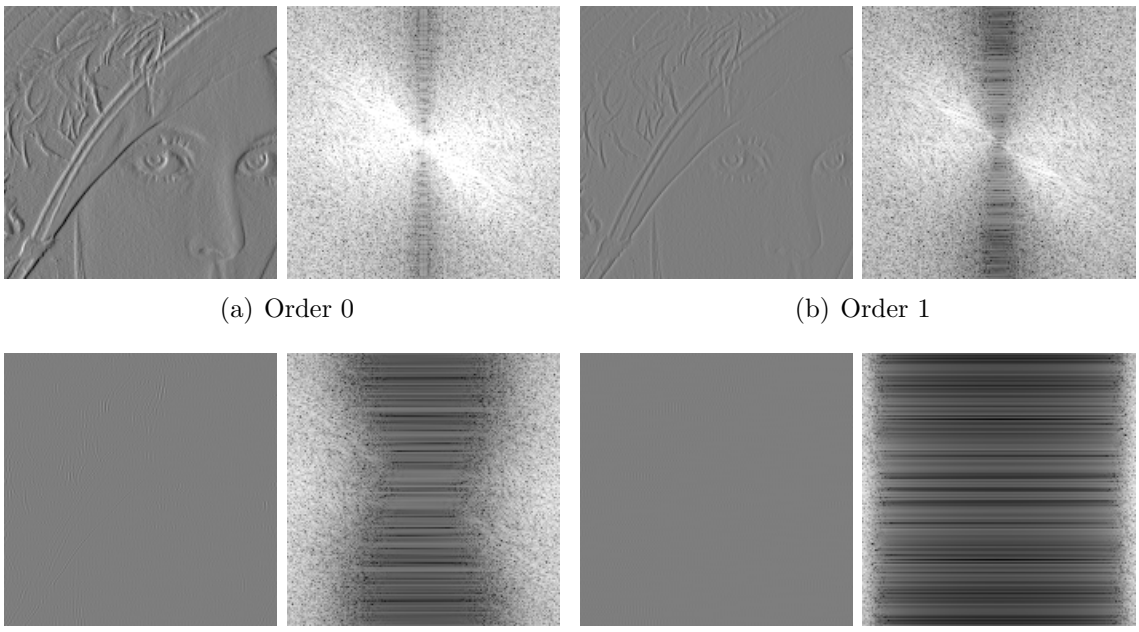


Figure 9: Decay of the error for the interpolation order $n \in \{0, \dots, 16\}$ when performing ten successive translations by 0.1 pixel of an image and finally compensating with a -1 pixel translation. It justifies the choice of a high order B-spline interpolation ($n = 11$ for instance) while it is commonly admitted that the cubic B-spline interpolation is sufficient. Note that the computational error is negligible with respect to the model error.



(a) Order 0 (b) Order 1

(c) Order 3 (d) Order 16

Figure 10: Central parts of the difference images for the consistency tests and the corresponding (unnormalized) discrete Fourier transform modulus (in logarithmic scale $u \mapsto \log(1 + u)$) for $n \in \{0, 1, 3, 16\}$. The differences are localized in the high frequencies were the model error is higher. For $n = 0$ it is exactly the gradient of the image. We added 128 to the difference images and multiplied the spectrum modulus by 30 before visualization.

6.4.3 Comparison between Different Orders

We also compared the B-spline interpolation results for different orders. As in Section 6.4.1, we computed the homographic transformations of the *Lena* image by h (defined by (77)) for different orders. The resampled images are compared to the one corresponding to the maximal order available, i.e., $n = 16$. The comparison is done by computing the RMSE between the central parts of the images. As the boundary extension choice has no influence we chose the half-symmetric extension. The decay of the difference with the order n is visible in Figure 11. The average difference for the cubic B-spline is around one gray level and is three times smaller for order 11. In Figure 12 we display for $n \in \{1, 3, 11, 15\}$ the central parts of the resampled images and of the difference images (with the corresponding discrete Fourier transform modulus). As the order increases the resampled image becomes sharper. The interpolation kernel becomes closer to the cardinal sine so that less high-frequency content is attenuated.

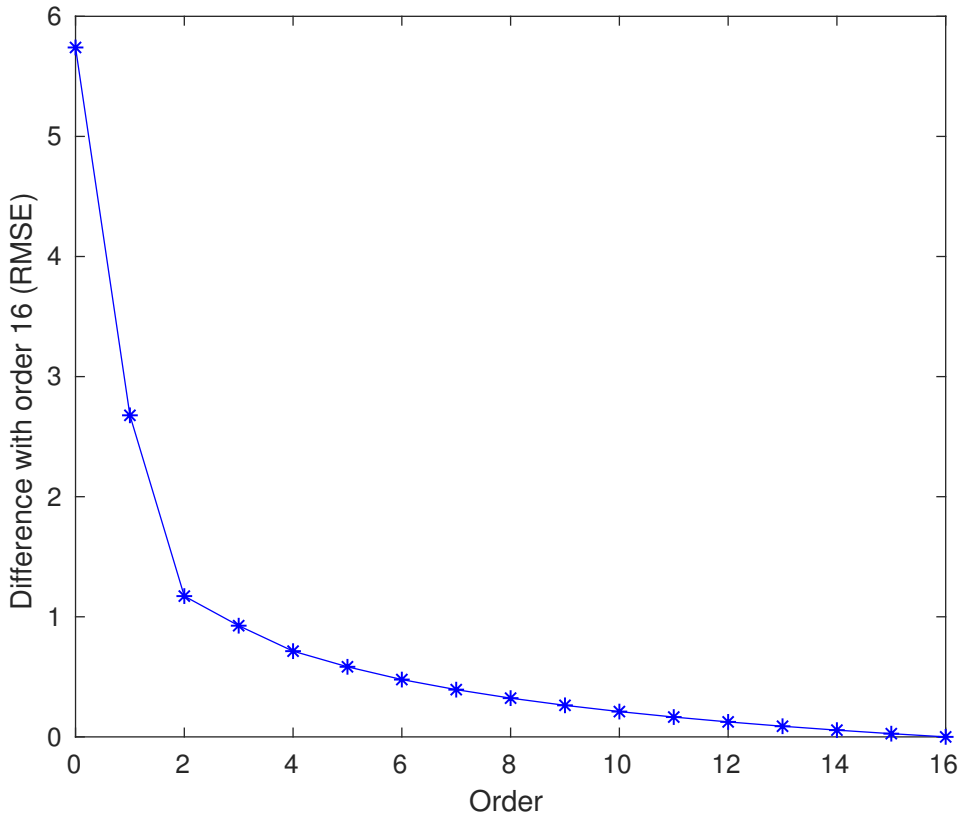
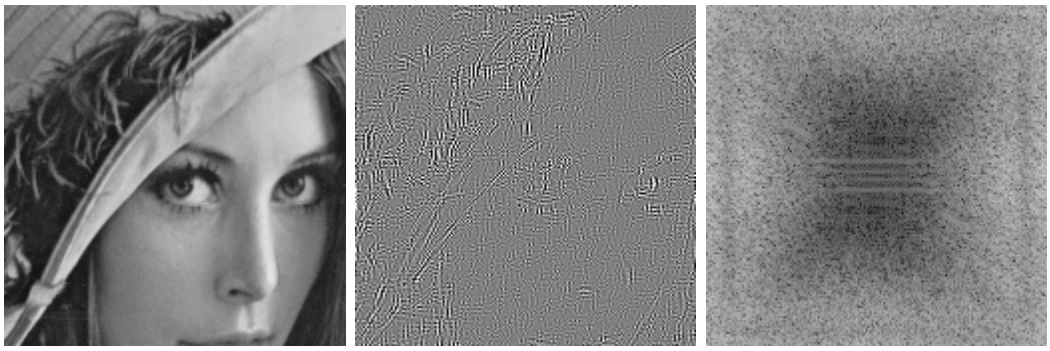


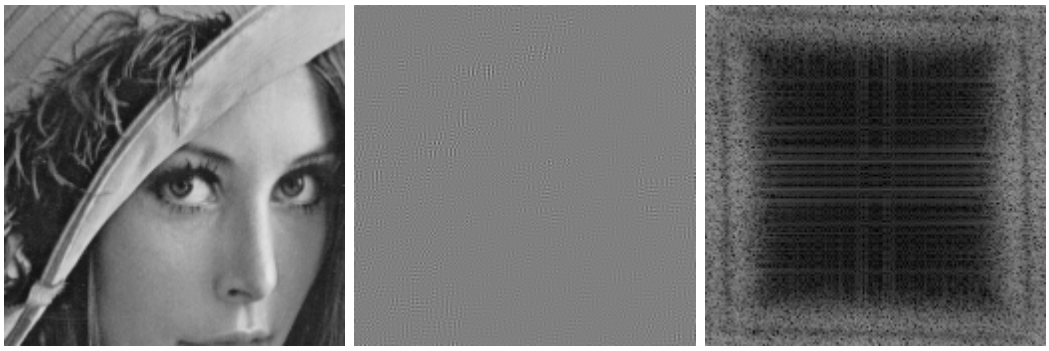
Figure 11: Decay of the difference between the B-spline interpolation of order $n \in \{0, \dots, 16\}$ and the one of order 16. The comparison is done by computing the RMSE between the central parts of the images. As the boundary extension choice has no influence we chose the half-symmetric extension. The average difference for the cubic B-spline is around one gray level and is three times smaller for order 11.



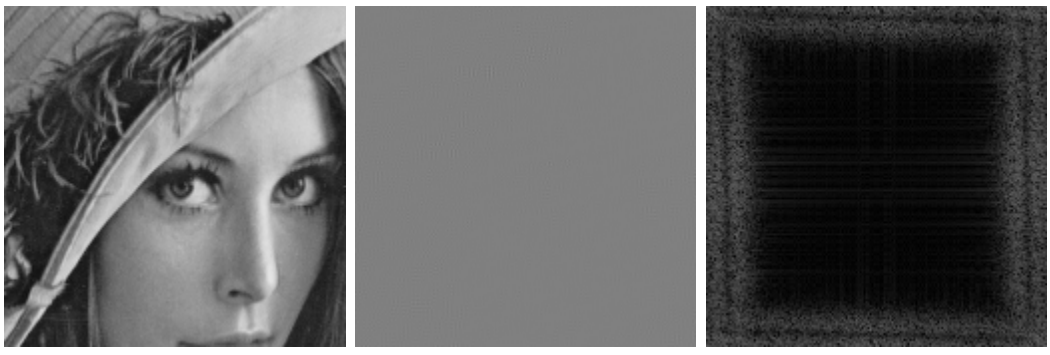
(a) Order 1



(b) Order 3



(c) Order 11



(d) Order 15

Figure 12: Comparison between resampled images obtained using $n \in \{0, 1, 3, 15\}$ and the one using $n = 16$. Only the central parts of the resampled image (left) and of the difference image (center) are shown. The right image is 30 times the (unnormalized) discrete Fourier transform modulus (in logarithmic scale $u \mapsto \log(1 + u)$) of the difference. The affine transformation $y = 50x + 128$ is applied to the difference images before visualization. As the order increases the resampled image becomes sharper. The interpolation kernel becomes closer to the cardinal sine so that less high-frequency content is attenuated.

7 Conclusion

In this paper we presented the theory and practice to perform B-spline interpolation for any order, in particular in the case of images. It is based on the seminal two-step method proposed by Unser et al. in 1991 that uses linear filtering and for which the computational error is not controlled and the boundary extension is fixed.

The two proposed prefiltering algorithms require additional computations to handle correctly any boundary extension. We proved theoretically and experimentally that the computational errors are controlled (up to dimension two). The first algorithm is general and works for any boundary extension while the second is applicable under specific assumptions. The global interpolation algorithm remains efficient because the computational cost increases slowly with the precision (which can be set to the single precision in most of the applications).

In an experimental part we showed that increasing the order of the B-spline interpolation improves the interpolation quality. When efficiency is not primordial, a high order B-spline interpolation must be preferred to cubic B-spline interpolation.

In addition, we provide a detailed description, and the corresponding implementation, of how to evaluate the B-spline kernel and to compute the B-spline interpolator parameters. As a fundamental application we also provide an implementation of homographic transformation of images using B-spline interpolation.

Acknowledgements

Work partly founded by the Office of Naval research by grant N00014-17-1-2552, ANR-DGA project ANR-12-ASTR-0035, Centre National d'Études Spatiales (CNES, MISS and TOSCA AHA and SMOS-HR Projects), and the European Research Council (advanced grant Twelve Labours n246961).

The authors would also like to thank Prof. Jean-Michel Morel for his support, suggestions, and many fruitful discussions.

Image Credits

 Standard test image

 Provided by the authors, CC-BY

A Practical Computation of the Poles and the Normalization Constant

A.1 Computation of the Polynomial Coefficients

As expressed in (25), the prefiltering step requires the knowledge of the normalization constant $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ and of the poles of order n introduced in (13). Theses poles correspond to the roots in $]-1,0[$ of the (palindromic) polynomial $\tilde{B}^{(n)}$ of degree $2\tilde{n}$ defined for $z \in \mathbb{C}$ by

$$\tilde{B}^{(n)}(z) = z^{\tilde{n}} B^{(n)}(z) = b_0^{(n)} z^{\tilde{n}} + \sum_{i=1}^{\tilde{n}} b_i^{(n)} (z^{\tilde{n}+i} + z^{\tilde{n}-i}). \tag{78}$$

The coefficients $(b_k^{(n)})_{0\le k\le \tilde{n}} = \left(\beta^{(n)}(k)\right)_{0\le k\le \tilde{n}}$ being given, the poles can be approximated numerically using a polynomial equation solver. The coefficients themselves can be computed directly thanks to the explicit formula given in (27) but a more efficient computation based on a recursive formula which only involves simple additions and multiplications is preferred. Indeed, for $m \ge 1$ β^m verifies the following identity (proven in Section A.3) for $x \in \mathbb{R}$,

$$m\beta^{(m)}(x) = \left(\frac{m+1}{2} + x\right)\beta^{(m-1)}\left(x + \frac{1}{2}\right) + \left(\frac{m+1}{2} - x\right)\beta^{(m-1)}\left(x - \frac{1}{2}\right). \tag{79}$$

Setting for simplicity $d_k^{(m)} = \beta^{(m)}\left(k + \frac{1}{2}\right)$ for $k \in \mathbb{Z}$, we have

$$m b_k^{(m)} = \left(\frac{m+1}{2} + k\right) d_k^{(m-1)} + \left(\frac{m+1}{2} - k\right) d_{k-1}^{(m-1)} \tag{80}$$

$$m d_k^{(m)} = \left(\frac{m+2}{2} + k\right) b_{k+1}^{(m-1)} + \left(\frac{m}{2} - k\right) b_k^{(m-1)}. \tag{81}$$

Define $\tilde{m} = \left[\frac{m}{2}\right]$. As $b_{\tilde{m}+1}^{(m)} = 0$ and $d_{-1}^{(m)} = d_0^{(m)}$ we can compute the coefficients $(b_k^{(n)})_{0\le k\le \tilde{n}}$ recursively using Algorithm 9. In particular, it is possible to obtain an explicit expression of $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ as a

Algorithm 9: Polynomial coefficients computation

Input : The B-spline order n
Output: The coefficients $(b_k^{(n)})_{0\le k\le \tilde{n}}$ of $\tilde{B}^{(n)}$

Initialize with $b_0^{(0)} = 1$ and $d_0^{(0)} = \frac{1}{2}$
for $m = 1$ **to** $n-1$ **do**
  Define $\tilde{m} = \left[\frac{m}{2}\right]$
  for $k = 0$ **to** $\tilde{m}$ **do**
    Compute $b_k^{(m)}$ using (80)
    Compute $d_k^{(m)}$ using (81)
  end
end
for $k = 0$ **to** $\tilde{n}$ **do**
  Compute $b_k^{(n)}$ using (80)
end

function of n.

A.2 Explicit Expression of the Normalization Constant

Proposition 2. *We have*

$$\gamma^{(n)} = \begin{cases} 2^n n! & n \text{ even} \\ n! & n \text{ odd.} \end{cases} \tag{82}$$

Proof. Assuming $n \geq 2$ and applying (80) to $k = \tilde{n}$ and $m = n$ we get

$$b_{\tilde{n}}^{(n)} = \frac{1}{n}\left(\frac{n+1}{2} - \tilde{n}\right) d_{\tilde{n}-1}^{(n-1)}. \tag{83}$$

Similarly, applying (81) to $k = \tilde{n} - 1$ and $m = n - 1$ we get

$$d_{\tilde{n}-1}^{(n-1)} = \frac{1}{n-1}\left(\frac{n+1}{2} - \tilde{n}\right) b_{\tilde{n}-1}^{(n-2)}. \tag{84}$$

As $\tilde{n} - 1 = \widetilde{n-2}$ we obtain the recursive equation

$$b_{\tilde{n}}^{(n)} = \frac{1}{n(n-1)}\left(\frac{n+1}{2} - \tilde{n}\right)^2 b_{\widetilde{n-2}}^{(n-2)}. \tag{85}$$

Now let $n \geq 0$. Noting that $b_{\tilde{0}}^{(0)} = b_{\tilde{1}}^{(1)} = 1$ and

$$\frac{m+1}{2} - \tilde{m} = \begin{cases} \frac{1}{2} & m \text{ even} \\ 1 & m \text{ odd,} \end{cases} \tag{86}$$

we finally have the following explicit expression for $b_{\tilde{n}}^{(n)}$,

$$b_{\tilde{n}}^{(n)} = \begin{cases} \frac{1}{2^n n!} & n \text{ even} \\ \frac{1}{n!} & n \text{ odd.} \end{cases} \tag{87}$$

□

It is a direct consequence of Proposition 2 that $\gamma^{(n)}\tilde{B}^{(n)}$ is a polynomial with integer coefficients. Indeed, let $j \in \mathbb{Z}$. Combining (27) at location $x = j$ with (82), we obtain $b_j^{(n)} = b_{\tilde{n}}^{(n)} a_j^{(n)}$ where

$$a_j^{(n)} = \begin{cases} \sum_{i=0}^{n+1} \binom{n+1}{i} (-1)^i \left(2(j-i) + n + 1\right)_+^n, & n \text{ even} \\ \sum_{i=0}^{n+1} \binom{n+1}{i} (-1)^i \left(j - i + \frac{n+1}{2}\right)_+^n, & n \text{ odd.} \end{cases} \tag{88}$$

Thus $a_j^{(n)} = \frac{b_j^{(n)}}{b_{\tilde{n}}^{(n)}} \in \mathbb{Z}$. Actually it is a non-negative integer since $\beta^{(n)}$ is non-negative. This property justifies why the expression of $B^{(n)}$ in Table 2 only involves integers. However it is not used in practice because the renormalization by $\gamma^{(n)}$ may introduce numerical errors for n large.

A.3 Proof of the Recursive Identity (79)

For $n = 1$, the verification of (79) is straightforward using the explicit expression

$$\beta^{(1)}(x) = \begin{cases} 1 - |x|, & |x| \leq 1 \\ 0, & |x| > 1. \end{cases} \tag{89}$$

To proceed with a recursive proof, assume the identity is verified up to $n \geq 1$. Then we can write

$$\beta^{(n+1)}(x) = \int_{-1/2}^{1/2} \beta^{(0)}(y)\beta^{(n)}(x-y)\,dy, \tag{90}$$

and integrating by parts we get

$$\begin{aligned}\beta^{(n+1)}(x) &= \left[y\beta^{(n)}(x-y)\right]_{-1/2}^{1/2} + \int_{-1/2}^{1/2} y\beta^{(n)'}(x-y)\,dy \\ &= \frac{1}{2}\left(\beta^{(n)}(x+\frac{1}{2}) + \beta^{(n)}(x-\frac{1}{2})\right) + \int_{\mathbb{R}} \beta^{(0)}(y)y\beta^{(n)'}(x-y)\,dy. \end{aligned}\tag{91}$$

For $n = 1$, $\beta^{(1)'}$ is defined everywhere except at $x \in \{-1, 0, 1\}$, but from the point of view of integration we can ignore this defect and write

$$\begin{aligned}\beta^{(1)'}(x) &= \begin{cases} 1, & -1 < x < 0 \\ -1, & 0 < x < 1 \\ 0, & |x| > 1, \end{cases} \\ &= \beta^{(0)}(x+\frac{1}{2}) - \beta^{(0)}(x-\frac{1}{2}). \end{aligned}\tag{92}$$

Now for $n \geq 2$, using (1) we have

$$\begin{aligned}\beta^{(n)'}(x) &= \left(\beta^{(n-2)} * \beta^{(1)'}\right)(x) = \int \beta^{(n-2)}(y)\beta^{(1)'}(x-y)\,dy \\ &= \int \beta^{(n-2)}(y)\beta^{(0)}(x+\frac{1}{2}-y)\,dy - \int \beta^{(n-2)}(y)\beta^{(0)}(x-\frac{1}{2}-y)\,dy \\ &= \beta^{(n-1)}(x+\frac{1}{2}) - \beta^{(n-1)}(x-\frac{1}{2}), \end{aligned}\tag{93}$$

so that according to (92), this equation is also valid for $n = 1$.[7] Therefore

$$\int \beta^{(0)}(y)y\beta^{(n)'}(x-y)\,dy = \int \beta^{(0)}(y)y\left(\beta^{(n-1)}(x+\frac{1}{2}-y) - \beta^{(n-1)}(x-\frac{1}{2}-y)\right)\,dy. \tag{94}$$

But the recursivity assumption at location $x - y$ can be rewritten

$$\begin{aligned} y\left(\beta^{(n-1)}(x+\frac{1}{2}-y) - \beta^{(n-1)}(x-\frac{1}{2}-y)\right) &= x\left(\beta^{(n-1)}(x+\frac{1}{2}-y) - \beta^{(n-1)}(x-\frac{1}{2}-y)\right) \\ &\quad + \frac{n+1}{2}\left(\beta^{(n-1)}(x+\frac{1}{2}-y) + \beta^{(n-1)}(x-\frac{1}{2}-y)\right) \\ &\quad - n\beta^{(n)}(x-y). \end{aligned}\tag{95}$$

Combining (94) and (95), and then coming back to (91), we can write

$$\begin{aligned}\beta^{(n+1)}(x) &= \left(\frac{n+1}{2} + \frac{1}{2}\right)\left(\beta^{(n)}(x+\frac{1}{2}) + \beta^{(n)}(x-\frac{1}{2})\right) \\ &\quad + x\beta^{(n)}(x+\frac{1}{2}) - x\beta^{(n)}(x-\frac{1}{2}) - n\beta^{(n+1)}(x), \end{aligned}\tag{96}$$

which yields after rearrangement of the terms

$$(n+1)\beta^{(n+1)}(x) = \left(\frac{(n+1)+1}{2} + x\right)\beta^{(n)}(x+\frac{1}{2}) + \left(\frac{(n+1)+1}{2} - x\right)\beta^{(n)}(x-\frac{1}{2}), \tag{97}$$

exactly (79) at $n + 1$.

[7] This could also be shown more concisely using distributions by the observation that $\beta^{(0)'} = \delta_{-1/2} - \delta_{1/2}$.

B Explicit Formula for the B-spline Function

We show by induction on order n the explicit formula (27) of the B-spline function $\beta^{(n)}$. For $n = 1$, we have the four cases for the right hand side of (27):

$$\begin{cases} 0+0+0=0 & \text{if } x<-1 \text{ since } x-i+1<0 \text{ for } i=0,1,2.\\ (x+1)_+ +0+0=x+1 & \text{if } -1\le x\le 0.\\ (x+1)-2x+0=1-x & \text{if } 0\le x\le 1.\\ (x+1)-2x+(x-1)=0 & \text{if } 1\le x. \end{cases}$$

We recognize the function $\max(0, 1-|x|)$. On the other hand, we have

$$\beta^{(1)}(x)=\int \beta^{(0)}(y)\beta^{(0)}(x-y)\,dy=\left(\int_{\max(-1/2,x-1/2)}^{\min(1/2,x+1/2)} dy\right)_+ =(1+\min(0,x)-\max(0,x))_+=(1-|x|)_+,$$

which justifies (27) for $n = 1$. To proceed with the induction, let us first consider the function $p_n(x) = (x)_+^n$ and compute

$$p_n * \beta^{(0)}(x)=\int_0^{+\infty} y^n\beta^{(0)}(x-y)\,dy=\int_{(x-1/2)_+}^{(x+1/2)_+} y^n\,dy=\frac{1}{n+1}\left((x+\frac12)_+^{n+1}-(x-\frac12)_+^{n+1}\right).$$

Now, assuming (27) holds for some $n \ge 1$, we get

$$\begin{aligned}
\beta^{(n+1)}(x) =& \frac{1}{n!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i}p_n*\beta^{(0)}\left(x-i+\frac{n+1}{2}\right)\\
=&\frac{1}{(n+1)!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i}\left(\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1}-\left(x-(i+1)+\frac{(n+1)+1}{2}\right)_+^{n+1}\right)\\
=&\frac{1}{(n+1)!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i}\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1}+\\
&\frac{1}{(n+1)!}\sum_{i=1}^{n+2}(-1)^i\binom{n+1}{i-1}\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1}\\
=&\frac{1}{(n+1)!}(-1)^0\binom{n+2}{0}\left(x-0+\frac{(n+1)+1}{2}\right)_+^{n+1}+\\
&\frac{1}{(n+1)!}\sum_{i=0}^{n+1}(-1)^i\left(\binom{n+1}{i}+\binom{n+1}{i-1}\right)\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1}+\\
&\frac{1}{(n+1)!}(-1)^{n+2}\binom{n+2}{n+2}\left(x-(n+2)+\frac{(n+1)+1}{2}\right)_+^{n+1}\\
=&\frac{1}{(n+1)!}\sum_{i=0}^{n+2}(-1)^i\binom{n+2}{i}\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1}.
\end{aligned}$$

The third equality is obtained by changing index i to $i+1$, and the last one using the identity $\binom{n+1}{i}+\binom{n+1}{i-1}=\binom{n+2}{i}$. Thus, we get the explicit formula (27) at index $n+1$.

C Truncation Indices

C.1 Proof of Theorem 1

In Algorithm 4 and Algorithm 5, the output error comes from the truncation of the initialization sums during the application of the exponential filters. Each truncation introduces an error that propagates to the following computations. To prove Theorem 1 we state and prove a more general theorem which provides a control of the error incurring after each application of an exponential filter.

Given N, a non-negative integer, $L_{\text{ini}} < L_{\text{end}}$ two integers, and $s \in \mathbb{R}^{\mathbb{Z}}$, we define the truncated signal $T_{N,L_{\text{ini}},L_{\text{end}}}(s)$ by

$$\forall k \in \mathbb{Z}, \quad T_{N,L_{\text{ini}},L_{\text{end}}}(s)_k = \begin{cases} s_k & L_{\text{ini}} - N \leq k \leq L_{\text{end}} + N, \\ 0 & \text{otherwise.} \end{cases} \tag{98}$$

We recall that f denotes the input finite signal of length K and $\varepsilon > 0$ the output precision. Instead of computing the intermediate filtered signals $(c^{(i)})_{0 \leq i \leq \tilde{n}}$, in Algorithm 4 and Algorithm 5 we compute the $(p^{(i)})_{0 \leq i \leq \tilde{n}}$ defined by

$$\begin{cases} p^{(0)} = f, \\ p^{(i)} = h^{(z_i)} * T_{N^{(i,\varepsilon)},L_{\text{ini}}^{(i,\varepsilon)},L_{\text{end}}^{(i,\varepsilon)}}(p^{(i-1)}) \quad \text{for } 1 \leq i \leq \tilde{n}, \end{cases} \tag{99}$$

where

$$L_{\text{ini}}^{(i,\varepsilon)} = \begin{cases} -L_i^{(n,\varepsilon)} & \text{in Algorithm 4,} \\ 0 & \text{in Algorithm 5,} \end{cases} \tag{100}$$

and

$$L_{\text{end}}^{(i,\varepsilon)} = \begin{cases} K - 1 + L_i^{(n,\varepsilon)} & \text{in Algorithm 4,} \\ K - 1 & \text{in Algorithm 5.} \end{cases} \tag{101}$$

For $0 \leq i \leq \tilde{n}$, we can write $p^{(i)} = c^{(i)} + e^{(i)}$ where $e^{(i)}$ is the error committed at step i. Finally the computed coefficients are $p = \gamma^{(n)} p^{(\tilde{n})} = c^{(\tilde{n})} + \gamma^{(n)} e^{(\tilde{n})}$. Note that the filtered signals are implicitly extended outside their computational domains (by zeros in Algorithm 4 and by the boundary condition in Algorithm 5).

Lemma 1. *Let $s \in \mathbb{R}^{\mathbb{Z}}$ and $-1 < \alpha < 0$. Let $g \in \mathbb{R}^{\mathbb{Z}}$ be a perturbation. Let N be a non-negative integer and $L_{ini} < L_{end}$ be two integers. Define $s' = T_{N,L_{ini},L_{end}}(s+g)$ and the error $e = h^{(\alpha)} * (s' - s)$. Then,*

$$\|e\|_\infty \leq C_\alpha \left((1-\alpha) \|g\|_\infty + |\alpha|^{N+1} \left(1 + |\alpha|^{L_{end} - L_{ini}} \right) \|s\|_\infty \right), \tag{102}$$

where

$$C_\alpha = \frac{-\alpha}{(1-\alpha^2)(1+\alpha)}. \tag{103}$$

Proof of Lemma 1. With (41) we can write

$$e = \frac{\alpha}{\alpha^2 - 1} \left(\underbrace{k^{(\alpha)} * (s' - s)}_{\text{causal part}} + \underbrace{l^{(\alpha)} * (s' - s)}_{\text{anticausal part}} - (s' - s) \right) \tag{104}$$

Now, let us evaluate e_i for $L_{\text{ini}} \leq i \leq L_{\text{end}}$. First, by definition of s', $(s' - s)_i = g_i$. Then we deal with the causal and anti-causal part of (104).

Causal part. With the adequate change of indices we have

$$\left(k^{(\alpha)} * (s' - s)\right)_i = \sum_{j=0}^{\infty} \alpha^j (s' - s)_{i-j} \tag{105}$$

$$= \sum_{j=-\infty}^{i} \alpha^{i-j} (s' - s)_j \tag{106}$$

$$= \alpha^i \left(\sum_{j=L_{\text{ini}}-N}^{i} \alpha^{-j} g_j - \sum_{j=-\infty}^{L_{\text{ini}}-N-1} \alpha^{-j} s_j \right). \tag{107}$$

Anticausal part. Similarly we have

$$\left(l^{(\alpha)} * (s' - s)\right)_i = \sum_{j=0}^{\infty} \alpha^j (s' - s)_{i+j} \tag{108}$$

$$= \sum_{j=i}^{\infty} \alpha^{j-i} (s' - s)_j \tag{109}$$

$$= \alpha^{-i} \left(\sum_{j=i}^{L_{\text{end}}+N} \alpha^j g_j - \sum_{j=L_{\text{end}}+N+1}^{\infty} \alpha^j s_j \right). \tag{110}$$

Finally,

$$e_i = \frac{\alpha}{\alpha^2 - 1} \left(\underbrace{\alpha^i \sum_{j=L_{\text{ini}}-N}^{i-1} \alpha^{-j} g_j + \alpha^{-i} \sum_{j=i}^{L_{\text{end}}+N} \alpha^j g_j}_{\text{previous error}} - \underbrace{\alpha^i \sum_{j=-\infty}^{L_{\text{ini}}-N-1} \alpha^{-j} s_j - \alpha^{-i} \sum_{j=L_{\text{end}}+N+1}^{\infty} \alpha^j s_j}_{\text{initialization error}} \right). \tag{111}$$

By the triangular inequality we obtain the four upper-bounds

$$\left| \sum_{j=L_{\text{ini}}-N}^{i-1} \alpha^{-j} g_j \right| \leq |\alpha|^{N-L_{\text{ini}}} \frac{1 - |\alpha|^{-(i-L_{\text{ini}}+N)}}{1 - |\alpha|^{-1}} \|g\|_\infty = -\alpha \frac{|\alpha|^{-i} - |\alpha|^{N-L_{\text{ini}}}}{1 + \alpha} \|g\|_\infty, \tag{112}$$

$$\left| \sum_{j=i}^{L_{\text{end}}+N} \alpha^j g_j \right| \leq |\alpha|^i \frac{1 - |\alpha|^{L_{\text{end}}+N-i+1}}{1 - |\alpha|} \|g\|_\infty = \frac{|\alpha|^i - |\alpha|^{L_{\text{end}}+N+1}}{1 + \alpha} \|g\|_\infty, \tag{113}$$

$$\left| \sum_{j=-\infty}^{L_{\text{ini}}-N-1} \alpha^{-j} s_j \right| \leq \frac{|\alpha|^{N+1-L_{\text{ini}}}}{1 + \alpha} \|s\|_\infty, \tag{114}$$

$$\left| \sum_{j=L_{\text{end}}+N+1}^{\infty} \alpha^j s_j \right| \leq \frac{|\alpha|^{N+1+L_{\text{end}}}}{1 + \alpha} \|s\|_\infty. \tag{115}$$

Thus,

$$|e_i| \leq C_\alpha \left(\|g\|_\infty \left[1 - \alpha \left(1 - |\alpha|^{N-L_{\text{ini}}+i} - |\alpha|^{N+L_{\text{end}}-i} \right) \right] + \|s\|_\infty |\alpha|^{N+1} \left(|\alpha|^{i-L_{\text{ini}}} + |\alpha|^{L_{\text{end}}-i} \right) \right) \tag{116}$$

Using the relation $|\alpha|^{i-L_{\text{ini}}} + |\alpha|^{L_{\text{end}}-i} \leq 1 + |\alpha|^{L_{\text{end}}-L_{\text{ini}}}$ and by removing the negative terms we get the following upper-bound that is independent of i,

$$\|e\|_\infty \leq C_\alpha \left((1 - \alpha) \|g\|_\infty + |\alpha|^{N+1} \left(1 + |\alpha|^{L_{\text{end}}-L_{\text{ini}}} \right) \|s\|_\infty \right). \tag{117}$$

□

For $0 \leq i \leq \tilde{n}$, define $D^{(i)}$ and $\varepsilon^{(i)}$ by

$$D^{(i)} = \prod_{j=1}^{i} \frac{-z_j}{(1+z_j)^2}, \tag{118}$$

and

$$\varepsilon^{(i)} = \varepsilon \rho^{(n)} D^{(i)} \prod_{j=i+1}^{\tilde{n}} \mu_j. \tag{119}$$

We recall that the $(\mu_j)_{1 \leq j \leq \tilde{n}}$ are defined in (50). Lemma 1 provides a control of the error when an exponential filter is applied to a signal whose values may be perturbed by a small error and where the initialization sums are truncated. We deduce from it the following theorem.

Theorem 3. *Assume $n \leq 16$ and $K \geq 4$. For $0 \leq i \leq \tilde{n}$, we have*

$$\|e^{(i)}\|_\infty \leq \varepsilon^{(i)} \|f\|_\infty. \tag{120}$$

To prove this theorem we need the two following lemmas.

Lemma 2. *For $0 \leq l \leq \tilde{n}$, we have*

$$\|c^{(l)}\|_\infty \leq D^{(l)} \|f\|_\infty. \tag{121}$$

Proof of Lemma 2. Let $s \in \mathbb{R}^{\mathbb{Z}}$ and $-1 < \alpha < 0$. Applying the triangular inequality in (42) we get

$$\|h^{(\alpha)} * s\|_\infty \leq \frac{\alpha}{\alpha^2 - 1} \frac{1-\alpha}{1+\alpha} |s|_\infty = \frac{-\alpha}{(1+\alpha)^2} \|s\|_\infty. \tag{122}$$

The result is proven by successively applying this inequality to $c^{(j)}$ and z_j for $1 \leq j \leq l$. □

Define $\theta : (x, m) \in]-1, 0[\times \mathbb{N} \mapsto -\frac{\log(1+|x|^m)}{\log|x|}$.

Lemma 3. *For $x > -0.75$ and $m \geq 4$ we have $\theta(x, m) < 1$.*

Proof of Lemma 3. θ is a decreasing function with respect to its variables and $\theta(-0.75, 4) < 1$. □

Proof of Theorem 3. The result is proved by induction.

- **Base case.** For $i = 0$ we have $e^{(0)} = 0$ and because $\mu_1 = 0$, $\varepsilon^{(0)} = 0$. Thus, $\|e^{(0)}\|_\infty = \varepsilon^{(0)} \|f\|_\infty$.
- **Inductive step.** For $1 \leq i \leq \tilde{n}$, assume $\|e^{(i-1)}\|_\infty \leq \varepsilon^{(i-1)} \|f\|_\infty$. Then we prove that $\|e^{(i)}\|_\infty \leq \varepsilon^{(i)} \|f\|_\infty$ as follows.

  Applying Lemma 1 to $s = c^{(i-1)}$, $\alpha = z_i$, $g = e^{(i-1)}$, $N = N^{(i,\varepsilon)}$, $L_{\text{ini}} = L_{\text{ini}}^{(i,\varepsilon)}$ and $L_{\text{end}} = L_{\text{end}}^{(i,\varepsilon)}$ we get

  $$\|e^{(i)}\|_\infty \leq C_{z_i} \left((1-z_i) \|e^{(i-1)}\|_\infty + |z_i|^{N^{(i,\varepsilon)}+1} \left(1 + |z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}} \right) \|c^{(i-1)}\|_\infty \right). \tag{123}$$

  Using the induction hypothesis and Lemma 2 with $l = i - 1$ we have

  $$\|e^{(i)}\|_\infty \leq \eta^{(i,\varepsilon)} \|f\|_\infty, \tag{124}$$

  where

  $$\eta^{(i,\varepsilon)} = C_{z_i} \left((1-z_i) \varepsilon^{(i-1)} + |z_i|^{N^{(i,\varepsilon)}+1} \left(1 + |z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}} \right) D^{(i-1)} \right). \tag{125}$$

Noting that for $n \leq 16$ the poles are greater than -0.75, we have by definition of $N^{(i,\varepsilon)}$ and with Lemma 3 applied to $x = z_i > -0.75$ and $m = L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)} \geq 4$,

$$\begin{aligned} N^{(i,\varepsilon)} + 1 &\geq \frac{\log\left(\varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j\right)}{\log|z_i|} + 1 && (126)\\ &\geq \frac{\log\left(\varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j\right)}{\log|z_i|} + \theta\left(z_i, L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}\right) && (127)\\ &\geq \frac{\log\left(\varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j\right) - \log\left(1+|z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}}\right)}{\log|z_i|}. && (128)\end{aligned}$$

The previous relation is equivalent to

$$\begin{aligned} |z_i|^{N^{(i,\varepsilon)}+1}\left(1+|z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}}\right) &\leq \varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j && (129)\\ &= \frac{(1-z_i)\varepsilon^{(i)}}{D^{(i)}} - \frac{(1-z_i)\varepsilon^{(i-1)}}{D^{(i-1)}}. && (130)\end{aligned}$$

Finally,

$$\eta^{(i,\varepsilon)} \leq C_{z_i}\frac{D^{(i-1)}}{D^{(i)}}(1-z_i)\varepsilon^{(i)} = \frac{-z_i(1+z_i)^2(1-z_i)}{-z_i(1-z_i^2)(1+z_i)}\varepsilon^{(i)} = \varepsilon^{(i)}, \quad (131)$$

and using (124) we obtain $\|e^{(i)}\|_\infty \leq \varepsilon^{(i)}\|f\|_\infty$.

□

Theorem 3 is more general than Theorem 1 because it provides a control of the error at each step.

Lemma 4. *We have*

$$\rho^{(n)}D^{(\tilde{n})} = \frac{1}{\gamma^{(n)}}. \quad (132)$$

Proof of Lemma 4. With the recursive definition of $\beta^{(n)}$ in (1) we have for $n \geq 1$,

$$\sum_{k\in\mathbb{Z}}\beta^{(n)}(k) = \sum_{k\in\mathbb{Z}}\int_{\mathbb{R}}\beta^{(n-1)}(x)\beta^{(0)}(k-x)dx = \int_{\mathbb{R}}\beta^{(n-1)}(x)dx = 1. \quad (133)$$

Noticing that $\sum_{k\in\mathbb{Z}}\beta^{(n)}(k) = B^{(n)}(1)$ and with (14) it can be rewritten has

$$\gamma^{(n)} = \prod_{j=1}^{\tilde{n}}\frac{(1-z_j)^2}{-z_j}. \quad (134)$$

By definition of $D^{(\tilde{n})}$ and $\rho^{(n)}$ we have the result. □

From Lemma 4 we deduce that $\varepsilon^{(\tilde{n})} = \frac{\varepsilon}{\gamma^{(n)}}$. In particular with $i = \tilde{n}$ in Theorem 3, we have $\|e^{(\tilde{n})}\| \leq \varepsilon^{(\tilde{n})}\|f\|_\infty = \frac{\varepsilon}{\gamma^{(n)}}\|f\|_\infty$ which proves Theorem 1.

Remarks:

- In practice the error decreases exponentially with the distance to the boundaries. Thus, the upper-bounds in the proof of Theorem 3 are not tight. The theoretical precision overestimates the experimental error. In addition, in Algorithm 4 we do not initialize at the same places so that the propagation error between steps could be neglected.
- For really small values of ε (e.g. $\varepsilon < 10^{-12}$), the machine precision should be considered. Thus the experimental error may be higher in this case.
- In our implementation the order is limited to 16 because of the poles computation. For higher order Theorem 3 may remain true provided the signals are large enough.

C.2 Proof of Theorem 2

Theorem 2 can be proven as a direct consequence of Theorem 1.

We recall that f denotes the input finite image of size at most 4 along each dimension and $\varepsilon > 0$ the output precision. Instead of computing $c_{\text{col}}(f)$ by applying Algorithm 4 or Algorithm 5 on the columns of f with truncation indices $N^{(n,\varepsilon')}$, we actually compute $p_{\text{col}}(f) = c_{\text{col}}(f) + e_{\text{col}}(f)$ where $e_{\text{col}}(f) \in \mathbb{R}^{\mathbb{Z}^2}$ is an unidimensional prefiltering error. Then, by applying Algorithm 4 or Algorithm 5 on the rows of $p_{\text{col}}(f)$ we obtain

$$p = c + e_{\text{row}} + e_{\text{row+col}}, \tag{135}$$

where e_{row} is the error that comes from the prefiltering along the rows of $p_{\text{col}}(f)$ and $e_{\text{row+col}}$ is the prefiltering along the rows of $e_{\text{col}}(f)$. The output error $e = e_{\text{row}} + e_{\text{row+col}}$ is proven to be smaller than ε as follows. On the one hand,

$$\|e_{\text{row+col}}\|_\infty \underset{Lemma\ 2}{\leq} \gamma^{(n)} D^{(\tilde{n})} \|e_{\text{col}}\|_\infty \underset{Theorem\ 1}{\leq} \varepsilon' \gamma^{(n)} D^{(\tilde{n})} \|f\|_\infty. \tag{136}$$

On the other hand,

$$\|e_{\text{row}}\|_\infty \underset{Theorem\ 1}{\leq} e' \|c_{\text{col}}(f)\|_\infty \underset{Lemma\ 2}{\leq} \varepsilon' \gamma^{(n)} D^{(\tilde{n})} \|f\|_\infty. \tag{137}$$

Finally,

$$\|e\|_\infty \leq 2\varepsilon' \gamma^{(n)} D^{(\tilde{n})} \underset{Lemma\ 4}{=} \varepsilon, \tag{138}$$

which proves Theorem 2.

C.3 Choice of the μ_i

To prove Theorem 3 we did not use the value of μ_i for $2 \leq i \leq \tilde{n}$. Any $(\mu_j)_{2 \leq j \leq \tilde{n}} \in]0,1[^{\tilde{n}-1}$ is acceptable to guarantee the output precision ε but the choice has an influence on the truncation indices $N^{(i,\varepsilon)}$ and thus on the complexity of the algorithms. Indeed, a small value for μ_i leads to less computations for the step i but more computations for steps $j < i$. We set $\mu_1 = 0$ because $e^{(0)} = 0$.

We select $\mu = (\mu_j)_{2 \leq j \leq \tilde{n}} \in]0,1[^{\tilde{n}-1}$ that minimizes the function

$$\Psi : \nu = (\nu_j)_{2 \leq j \leq \tilde{n}} \in]0,1[^{\tilde{n}-1} \mapsto \sum_{i=1}^{\tilde{n}} \frac{\log\left((1-\nu_i)\prod_{j=i+1}^{\tilde{n}} \nu_j\right)}{\log|z_i|}, \tag{139}$$

with the notation $\nu_1 = 0$. In other words it is selected so that $\sum_{i=1}^{\tilde{n}} N^{(i,\varepsilon)}$ is small[8]. Ψ is a derivable function that admits a unique minimizer μ that is given by the Euler condition $\nabla\Psi(\mu) = 0$. For $2 \leq k \leq \tilde{n}$ we have

$$\frac{\partial\Psi}{\partial\nu_k}(\mu) = \frac{-1}{1-\mu_k}\frac{1}{\log|z_k|} + \frac{1}{\mu_k}\sum_{i=1}^{k-1}\frac{1}{\log|z_i|} = 0, \tag{140}$$

i.e.

$$\frac{1}{\mu_k} = 1 + \frac{1}{\log|z_k|\sum_{i=1}^{k-1}\frac{1}{\log|z_i|}}, \tag{141}$$

which corresponds to the definition given in (50). The values of $(\mu_i)_{2\leq i\leq\tilde{n}}$ are displayed in Table 6 for $4 \leq n \leq 9$.

n	$(\mu_i)_{2\leq i\leq\tilde{n}}$
4	$\mu_2 = 0.8081702588338142$
5	$\mu_2 = 0.7886523126940346$
6	$\mu_2 = 0.7775037872839968$ $\mu_3 = 0.9217057449487258$
7	$\mu_2 = 0.7705847640302491$ $\mu_3 = 0.9069526580525736$
8	$\mu_2 = 0.7660491039752506$ $\mu_3 = 0.8982276825918423$ $\mu_4 = 0.9583935084163903$
9	$\mu_2 = 0.7628638545450653$ $\mu_3 = 0.8921921530329509$ $\mu_4 = 0.9478524258426756$

Table 6: Values of $(\mu_i)_{2\leq i\leq\tilde{n}}$ for $4 \leq n \leq 9$.

D Exact Expressions for Initialization Values

Consider an extension that can be expressed as a boundary condition and that is transmitted during the prefiltering (as in Section 3.2.2). Let s be a finite signal of length K which is extended to $\mathbb{Z}$ and $-1 < \alpha < 0$. The initialization values in (38) and (44) may admit exact expressions that depends on K, α and on the extension. In the following we obtain exact expressions for the three boundary conditions proposed in Table 1 that are transmitted.

D.1 Causal Initialization

Accordingly to the case we obtain an exact expression of the causal initialization value $(k^{(\alpha)} * s)_0$ as follows. Note that it only depends on $(s_k)_{0\leq k\leq K-1}$.

• **Periodic extension.** Grouping terms by blocks of length K we have

$$(k^{(\alpha)} * s)_0 = \sum_{i=0}^{+\infty}\alpha^i s_{-i} = \sum_{l=0}^{+\infty}\alpha^{lK}\sum_{i=0}^{K-1}\alpha^i s_{-i-lK}. \tag{142}$$

[8]The minimality is not guaranteed because of the integral part in the definition of $N^{(i,\varepsilon)}$.

Using the K-periodicity of the extended signal we obtain the relation $s_{-i-lK} = s_{K-i}$ for all $(i,l) \in \mathbb{Z}^2$. Finally, we have

$$(k^{(\alpha)} * s)_0 = \sum_{l=0}^{+\infty} \alpha^{lK} \sum_{i=0}^{K-1} \alpha^i s_{K-i} \tag{143}$$

$$= \frac{1}{1-\alpha^K} \sum_{i=0}^{K-1} \alpha^i s_{K-i}, \tag{144}$$

where we recall that $s_K = s_0$.

- **Half-symmetric extension.** Grouping terms by blocks of length $2K$ we have

$$(k^{(\alpha)} * s)_0 = s_0 + \alpha \sum_{i=0}^{+\infty} \alpha^i s_{-i-1} = s_0 + \alpha \sum_{l=0}^{+\infty} \alpha^{2lK} \sum_{i=0}^{2K-1} \alpha^i s_{-i-2lK-1}. \tag{145}$$

Using the $2K$-periodicity of the extended signal we obtain the relation $s_{-i-2lK} = s_{-i}$ for all $(i,l) \in \mathbb{Z}^2$ which gives

$$\sum_{i=0}^{2K-1} \alpha^i s_{-i-2lK-1} = \sum_{i=0}^{2K-1} \alpha^i s_{-i-1} = \sum_{i=0}^{K-1} \alpha^i s_{-i-1} + \alpha^K \sum_{i=0}^{K-1} \alpha^i s_{-i-K-1}. \tag{146}$$

For $i \in \{0, \ldots, K-1\}$ the symmetry around $-\frac{1}{2}$ can be written as $s_{-i-1} = s_i$ and the symmetry around $-K-\frac{1}{2}$ can be written as $s_{-i-K-1} = s_{-K+i} = s_{-(K-1-i)-1} = s_{K-1-i}$. Noting that $\sum_{l=0}^{\infty} \alpha^{2lK} = \frac{1}{1-\alpha^{2K}}$, we finally have

$$(k^{(\alpha)} * s)_0 = s_0 + \frac{\alpha}{1-\alpha^{2K}} \sum_{i=0}^{K-1} \alpha^i (s_i + \alpha^K s_{K-1-i}). \tag{147}$$

- **Whole-symmetric extension.** Using the symmetry around 0 and grouping terms by blocks of length $2K-1$ we have

$$(k^{(\alpha)} * s)_0 = \sum_{i=0}^{+\infty} \alpha^i s_{-i} = \sum_{i=0}^{+\infty} \alpha^i s_i = \sum_{l=0}^{+\infty} \alpha^{(2K-1)l} \sum_{i=0}^{2K-2} \alpha^i s_{i+(2K-1)l}. \tag{148}$$

Using the $(2K-1)$-periodicity of the extended signal we obtain the relation $s_{i+(2K-1)l} = s_i$ for all $(i,l) \in \mathbb{Z}^2$ which gives

$$\sum_{i=0}^{2K-2} \alpha^i s_{i+(2K-1)l} = \sum_{i=0}^{2K-2} \alpha^i s_i = \sum_{i=0}^{K-1} \alpha^i s_i + \alpha^K \sum_{i=0}^{K-2} \alpha^i s_{K+i}. \tag{149}$$

For $i \in \{0, \ldots, K-2\}$ the symmetry around $K-1$ can be written as $s_{K+i} = s_{K-2-i}$. Noting that $\sum_{l=0}^{\infty} \alpha^{(2K-1)l} = \frac{1}{1-\alpha^{2K-1}}$, we finally have

$$(k^{(\alpha)} * s)_0 = \frac{\alpha}{1-\alpha^{2K-1}} \left(\sum_{i=0}^{K-1} \alpha^i s_i + \alpha^K \sum_{i=0}^{K-2} \alpha^i s_{K-2-i} \right). \tag{150}$$

D.2 Anti-causal Initialization

Exact expressions for the anti-causal initialization values $(h^{(\alpha)} * s)_{K-1}$ are provided in Section 3.2.2 for the two symmetric extensions. For the periodic extension we obtain an exact expression by performing similar computations as in Section 3.2.2 but without truncating the sums. Note that it only depends on $(s_k^{(\alpha)})_{0 \leq k \leq K-1}$.

Assume s is a K-periodic signal. Grouping terms by blocks of length K we have

$$\left(l^{(\alpha)} * s^{(\alpha)}\right)_{K-1} = \sum_{i=0}^{+\infty} \alpha^i s_{K-1+i}^{(\alpha)} \tag{151}$$

$$= s_{K-1}^{(\alpha)} + \alpha \sum_{i=0}^{+\infty} \alpha^i s_{K+i}^{(\alpha)} \tag{152}$$

$$= s_{K-1}^{(\alpha)} + \alpha \sum_{l=0}^{+\infty} \alpha^{lK} \sum_{i=0}^{K-1} \alpha^i s_{lK+i}^{(\alpha)}. \tag{153}$$

As $s^{(\alpha)} = k^{(\alpha)} * s$ is also K-periodic we have for all $(i,l) \in \mathbb{Z}^2$, $s_{lK+i}^{(\alpha)} = s_i^{(\alpha)}$. Noting that $\sum_{l=0}^{\infty} \alpha^{lK} = \frac{1}{1-\alpha^K}$ and $h^{(\alpha)} = -\alpha l^{(\alpha)}$, we finally have

$$(h^{(\alpha)} * s)_{K-1} = -\alpha \left(s_{K-1}^{(\alpha)} + \frac{\alpha}{1-\alpha^K} \sum_{i=0}^{K-1} \alpha^i s_i^{(\alpha)} \right). \tag{154}$$

References

[1] R. Abergel and L. Moisan, *The Shannon Total Variation*, Journal of Mathematical Imaging and Vision, (2017), pp. 1–30. http://dx.doi.org/10.1007/s10851-017-0733-5.

[2] A. Aldroubi, M. Unser, and M. Eden, *Cardinal Spline Filters: Stability and Convergence to the Ideal Sinc Interpolator*, Signal Processing, 28 (1992), pp. 127–138. http://dx.doi.org/10.1016/0165-1684(92)90030-Z.

[3] E.W. Cheney, *Approximation theory III*, vol. 12, Academic Press New York, 1980. ISBN 9780121710507.

[4] P. Dierckx, *Curve and surface fitting with splines*, Oxford University Press, 1995. ISBN 9780198534402.

[5] W. Dorn, *Generalizations of Horner's rule for polynomial evaluation*, IBM Journal of Research and Development, 6 (1962), pp. 239–245. https://doi.org/10.1147/rd.62.0239.

[6] P. Getreuer, *Linear Methods for Image Interpolation*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.g_lmii.

[7] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2nd edition, 2004. ISBN 9780521540513.

[8] H. Hou and H. Andrews, *Cubic Splines for Image Interpolation and Digital Filtering*, IEEE Transactions on Acoustics, Speech and Signal Processing, 26 (1978), pp. 508–517. http://dx.doi.org/10.1109/TASSP.1978.1163154.

[9] E.I. Jury, *Theory and Application of the Z-Transform Method*, Wiley, 1964. ISBN 9780882751221.

[10] J. Keiner, S. Kunis, and D. Potts, *Using NFFT 3—a software library for various nonequispaced fast Fourier transforms*, ACM Transactions on Mathematical Software (TOMS), 36 (2009), p. 19. http://dx.doi.org/10.1145/1555386.1555388.

[11] M. Liou, *Spline fit made easy*, IEEE Transactions on Computers, 100 (1976), pp. 522–527. http://dx.doi.org/10.1109/TC.1976.1674640.

[12] G. Milovanović and Z. Udovičić, *Calculation of Coefficients of a Cardinal B-spline*, Applied Mathematics Letters, 23 (2010), pp. 1346–1350. http://dx.doi.org/10.1016/j.aml.2010.06.029.

[13] L. Moisan, P. Moulon, and P. Monasse, *Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers*, Image Processing On Line, 2 (2012), pp. 56–73. https://dx.doi.org/10.5201/ipol.2012.mmm-oh.

[14] I.J Schoenberg, *Cardinal Interpolation and Spline Functions*, Journal of Approximation theory, 2 (1969), pp. 167–206. http://dx.doi.org/10.1016/0021-9045(69)90040-9.

[15] ——, *Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions*, in I.J Schoenberg Selected Papers, Springer, 1988, pp. 3–57. http://dx.doi.org/10.1007/978-1-4899-0433-1_1.

[16] C.E. Shannon, *Communication in the Presence of Noise*, Proceedings of the IRE, 37 (1949), pp. 10–21. http://dx.doi.org/10.1109/JRPROC.1949.232969.

[17] L. Simon and J-M. Morel, *Influence of Unknown Exterior Samples on Interpolated Values for Band-limited Images*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 152–184. http://dx.doi.org/10.1137/140978338.

[18] P. Thévenaz, T. Blu, and M. Unser, *Interpolation Revisited [Medical Images Application]*, IEEE Transactions on Medical Imaging, 19 (2000), pp. 739–758. http://dx.doi.org/10.1109/42.875199.

[19] M. Unser, *Splines: A Perfect Fit for Signal and Image Processing*, IEEE Signal Processing Magazine, 16 (1999), pp. 22–38. http://dx.doi.org/10.1109/79.799930.

[20] ——, *Sampling-50 Years After Shannon*, Proceedings of the IEEE, 88 (2000), pp. 569–587. https://doi.org/10.1109/5.843002.

[21] M. Unser, A. Aldroubi, and M. Eden, *Fast B-spline Transforms for Continuous Image Representation and Interpolation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13 (1991), pp. 277–285. http://dx.doi.org/10.1109/34.75515.

[22] ——, *B-spline Signal Processing. I. Theory*, IEEE Transactions on Signal Processing, 41 (1993), pp. 821–833. http://dx.doi.org/10.1109/78.193220.

[23] ——, *B-spline Signal Processing. II. Efficiency Design and Applications*, IEEE Transactions on Signal Processing, 41 (1993), pp. 834–848. http://dx.doi.org/10.1109/78.193221.