Image Interpolation and Resampling

Philippe Thévenaz, Thierry Blu and Michael Unser
Swiss Federal Institute of Technology—Lausanne
philippe.thevenaz@epfl.ch, thierry.blu@epfl.ch, michael.unser@epfl.ch

***Abstract*—This chapter presents a survey of interpolation and resampling techniques in the context of exact, separable interpolation of regularly sampled data. In this context, the traditional view of interpolation is to represent an arbitrary continuous function as a discrete sum of weighted and shifted synthesis functions—in other words, a mixed convolution equation. An important issue is the choice of adequate synthesis functions that satisfy interpolation properties. Examples of finite-support ones are the square pulse (nearest-neighbor interpolation), the hat function (linear interpolation), the cubic Keys' function, and various truncated or windowed versions of the sinc function. On the other hand, splines provide examples of infinite-support interpolation functions that can be realized exactly at a finite, surprisingly small computational cost. We discuss implementation issues and illustrate the performance of each synthesis function. We also highlight several artifacts that may arise when performing interpolation, such as ringing, aliasing, blocking and blurring. We explain why the approximation order inherent in the synthesis function is important to limit these interpolation artifacts, which motivates the use of splines as a tunable way to keep them in check without any significant cost penalty.**

I. INTRODUCTION

Interpolation is a technique that pervades many an application. Interpolation is almost never the goal in itself, yet it affects both the desired results and the ways to obtain them. Notwithstanding its nearly universal relevance, some authors give it less importance than it deserves, perhaps because considerations on interpolation are felt as being paltry when compared to the description of a more inspiring grand scheme of things of some algorithm or method. Due to this indifference, it appears as if the basic principles that underlie interpolation might be sometimes cast aside, or even misunderstood. The goal of this chapter is to refresh the notions encountered in classical interpolation, as well as to introduce the reader to more general approaches.

1.1. Definition

What is interpolation? Several answers coexist. One of them defines interpolation as an informed estimate of the unknown [1]. We prefer the following—admittedly less concise—definition: model-based recovery of continuous data from discrete data within a known range of abscissa. The reason for this preference is to allow for a clearer distinction between interpolation and extrapolation. The former postulates the existence of a known range where the model applies, and asserts that the deterministically-recovered continuous data is entirely described by the discrete data, while the latter authorizes the use of the model outside of the known range, with the implicit assumption that the model is "good" near data samples, and possibly less good elsewhere. Finally, the three most important hypothesis for interpolation are:

1) *The underlying data is continuously defined;*
2) *Given data samples, it is possible to compute a data value of the underlying continuous function at any abscissa;*
3) *The evaluation of the underlying continuous function at the sampling points yields the same value as the data themselves.*

1.2. Scope

It follows from this definition that interpolation, in some form or another, is needed each time the data to process is known only by discrete samples, which is almost universally the case in the computer era. This ubiquitous applicability is also the rule in biomedical applications. In this chapter, we restrict the discussion to the case where the discrete data are regularly sampled on a Cartesian grid. We also restrict the discussion to exact interpolation, where the continuous model is required to take the same values as the sampled data at the grid locations. Finally, we restrict ourselves to linear methods, such that the sum of two interpolated functions is equal to the interpolation of the sum of the two functions. For this reason, we will only mention Kriging [2, 3] and shape-based interpolation [4, 5] as examples of non-linear interpolation, and quasi-interpolation [6] as an example of inexact interpolation, without discussing them further. In spite of these restrictions, the range of applicability of the interpolation methods discussed here remains large, especially in biomedical imagery, where it is very common to deal with regularly sampled data.

1.3. Applications

Among biomedical applications where interpolation is quite relevant, the most obvious are those where the goal is to modify the sampling rate of pixels (picture elements) or voxels (volume elements). This operation, named rescaling, is desirable when an acquisition device—say, a scanner—has a non-homogeneous resolution, typically a fine within-slice resolution and a coarse across-slice resolution. In this case, the purpose is to change the aspect ratio of voxels in such a way that they correspond to geometric cubes in the physical space [7, 8]. Often, the across-slice resolution is modified to match the within-slice resolution, which is left unchanged. This results in a volumetric representation that is easy to handle (e.g., to visualize or to rotate) because it enjoys homogenous resolution.

A related operation is reslicing [9]. Suppose again that some volume has a higher within-slice than across-slice resolution. In this case, it seems natural to display the volume as a set of images oriented parallel to the slices, which offers its most detailed presentation. Physicians may however be sometimes interested in other views of the same data; for simplicity, they often request that the volume be also displayed as set of images oriented perpendicular to the slices. With respect to the physical space, these special orientations are named axial, coronal and sagittal, and require at most rescaling for their proper display. Meanwhile, interpolation is required to display any other orientation—in this context, this is named reslicing.

The relevance of interpolation is also obvious in more advanced visualization contexts, such as volume rendering. There, it is common to apply a texture to the facets that compose the rendered object [10]. Although textures may be given by models (procedural textures), this is generally limited to computer graphics applications; in biomedical rendering, it is preferable to display a texture given by a map consisting of true data samples. Due to the geometric operations involved (e.g., perspective projection), it is necessary to resample this map, and this resampling involves interpolation. In addition, volumetric rendering also requires the computation of gradients, which is best done by taking the interpolation model into account [11].

A more banal use of interpolation arises with images (as opposed to volumes). There, a physician may both want to inspect an image at coarse scale and to study some detail at fine scale. To this end, interpolation operations like zooming-in and -out are useful [12, 13]. Related operations are (sub-pixel) translation or panning, and rotation [14]. Less ordinary transformations may involve a change of coordinates, for example the polar-to-Cartesian scan conversion function that transforms acquired polar-coordinate data vectors from an ultrasound transducer into the Cartesian raster image needed for the display monitors. Another application of the polar-to-Cartesian transform arises in the three-dimensional reconstruction of icosahedral viruses [15].

In general, almost every geometric transformation requires that interpolation be performed on an image or a volume. In biomedical imaging, this is particularly true in the context of registration, where an image needs to be translated, rotated, scaled, warped, or otherwise deformed, before it can match a reference image or an atlas [16]. Obviously, the quality of the interpolation process has a large influence on the quality of the registration.

The data model associated to interpolation also affects algorithmic considerations. For example, the strategy that goes by the name of multiresolution proposes to solve a problem first at the coarse scale of an image pyramid, and then to iteratively propagate the solution at the next finer scale, until the problem has been solved at the finest scale. In this context, it is desirable to have a framework where the interpolation model is consistent with the model upon which the image pyramid is based. The assurance that only a minimal amount of work is needed at each new scale for refining the solution, is only present when the interpolation model is coherent with the multiresolution model [17].

Tomographic reconstruction by (filtered) back-projection forms another class of algorithms that rely on interpolation to work properly. The principle is as follows: several x-ray images of a real-world volume are acquired, with a different relative orientation for each image. After this physical acquisition stage, one is left with x-ray images (projections) of known orientation, given by data samples. The goal is to reconstruct a numeric representation of the volume from these samples (inverse Radon transform), and the mean is to surmise each voxel value from its pooled trace on the several projections. Interpolation is necessary because the trace of a given voxel does not correspond in general to the exact location of pixels in the projection images. As a variant, some authors have proposed to perform reconstruction with an iterative approach that requires the direct projection of the volume (as opposed to its back-projection) [18]. In this second approach, the volume itself is oriented by interpolation, while in the first approach the volume is fixed and is not interpolated at all, but the projections are.

Interpolation is so intimately associated with its corresponding data model that, even when no resampling operation seems to be involved, it nevertheless contributes under the guise of its data model. For example, we have defined interpolation as the link between the discrete world and the continuous one. It follows that the process of data differentiation (calculating the data derivatives), which is defined in the continuous world, can only be interpreted in the discrete world if one takes the interpolation model into consideration. Since derivatives or gradients are at the heart of many an algorithm (e.g., optimizer, edge detection, contrast enhancement), the design of gradient operators that are consistent with the interpolation model [19, 20] should be an essential consideration in this context.

II. CLASSICAL INTERPOLATION

Although many ways have been designed to perform interpolation, we concentrate here on linear algorithms of the form

$$f(\mathbf{x}) = \sum_{\mathbf{k}\in \boldsymbol{Z}^q} f_{\mathbf{k}}\, \varphi_{\text{int}}(\mathbf{x}-\mathbf{k}) \qquad \forall \mathbf{x} = \left(x_1, x_2, \ldots, x_q\right) \in \boldsymbol{R}^q, \tag{1}$$

where an interpolated value $f(\mathbf{x})$ at some (perhaps non-integer) coordinate $\mathbf{x}$ in a space of dimension q is expressed as a linear combination of the samples $f_{\mathbf{k}}$ evaluated at integer coordinates $\mathbf{k} = \left(k_1, k_2, \ldots, k_q\right) \in \boldsymbol{Z}^q$, the weights being given by the values of the function $\varphi_{\text{int}}(\mathbf{x}-\mathbf{k})$. Typical values of the space dimension correspond to bidimensional images (2D), with $q = 2$, and tridimensional volumes (3D), with $q = 3$. Without loss of generality, we assume that the regular sampling step is unity. Since we restrict this discussion to exact interpolation, we ask the function φ_{int} to satisfy the interpolation property—as we shall soon see, it must vanish for all integer arguments except at the origin, where it must take a unit value. A classical example of the synthesis function φ_{int} is the sinc function, in which case all synthesized functions are band-limited.

As expressed in (1), the summation is performed over all integer coordinates $\mathbf{k} \in \boldsymbol{Z}^q$, covering the whole of the Cartesian grid of sampling locations, irrespective of whether or not there actually exists a physically acquired sample $f_{\mathbf{k}_0}$ at some specific $\mathbf{k}_0$. In practice however, the number of known samples is always finite; thus, in order to satisfy the formal convention in (1), we have to extend this finite number to infinity by setting suitable boundary conditions over the interpolated function, for example using mirror symmetries (see Appendix). Now that the value of any sample $f_{\mathbf{k}}$ is defined (that is, either measured or determined), we can carry the summation all the way to—and from—infinity.

The only remaining freedom lies in the choice of the synthesis function φ_{int}. This restriction is but apparent. Numerous candidates have been proposed: Appledorn [21], B-spline [22], Dodgson [23], Gauss, Hermite, Keys [24, 25], Lagrange, linear, Newton, NURBS [26], o-Moms [27], Rom-Catmull, sinc, Schaum[28], Thiele, and more. In addition to them, a large palette of apodization windows have been proposed for the practical realization of the sinc function, which at first sight is the most natural function for interpolation. Their naming convention sounds like a pantheon of mathematicians [29]: Abel, Barcilon-Temes, Bartlet, Blackman, Blackman-Harris, Bochner, Bohman, Cauchy, Dirichlet, Dolph-Chebyshev, Fejér, Gaussian, Hamming, Hanning, Hanning-Poisson, Jackson, Kaiser-Bessel, Parzen, Poisson, Riemann, Riesz, Tukey, de la Vallée-Poussin, Weierstrass, and more.

2.1. Interpolation Constraint

Consider the evaluation of (1) in the specific case when all coordinates of $\mathbf{x} = \mathbf{k}_0$ are integer

$$f_{\mathbf{k}_0} = \sum_{\mathbf{k}\in \boldsymbol{Z}^q} f_{\mathbf{k}}\, \varphi_{\text{int}}(\mathbf{k}_0-\mathbf{k}) \qquad \forall \mathbf{k}_0 \in \boldsymbol{Z}^q. \tag{2}$$

This equation is known as the interpolation constraint. Perhaps, the single most important point of this whole chapter about interpolation is to recognize that (2) is a **discrete convolution**. Then, we can rewrite Equation (2) as

$$f_{\mathbf{k}_0} = (f * p)_{\mathbf{k}_0} \qquad \forall \mathbf{k}_0 \in \boldsymbol{Z}^q, \tag{3}$$

where we have introduced the notation $p_{\mathbf{k}} = \varphi_{\text{int}}(\mathbf{k})$ to put a heavy emphasis on the fact that we only discuss convolution between sequences that have the crucial property of being discrete. By contrast, (1) is not a convolution in this sense, because there φ_{int} is evaluated at possibly non-integer values. From now on, we shall use $f_{\mathbf{k}}$ to describe the samples of f, and $p_{\mathbf{k}}$ to describe the values taken by φ_{int} for integer argument. Clearly we have that (2) is equivalent to $p_{\mathbf{k}} = \delta_{\mathbf{k}}$, where $\delta_{\mathbf{k}}$ is the Kronecker symbol that is characterized by a central value $\delta_{\mathbf{0}} = 1$, and by zero values everywhere else. This function of integer argument takes the role of the neutral element for a discrete convolution.

III. Generalized Interpolation

As an alternative algorithm, let us now consider the form

$$f(\mathbf{x}) = \sum_{\mathbf{k} \in \boldsymbol{Z}^q} c_{\mathbf{k}}\, \varphi(\mathbf{x} - \mathbf{k}) \qquad \forall \mathbf{x} \in \boldsymbol{R}^q. \tag{4}$$

The crucial difference between the classical formulation (1) and the generalized formulation (4) is the introduction of coefficients $c_{\mathbf{k}}$ in place of the sample values $f_{\mathbf{k}}$. This offers new possibilities, in the sense that interpolation can now be carried in two separate steps: firstly, the determination of coefficients $c_{\mathbf{k}}$ from the samples $f_{\mathbf{k}}$, and secondly, the determination of desired values $f(\mathbf{x})$ from the coefficients $c_{\mathbf{k}}$. The benefit of this separation is to allow for an extended choice of synthesis functions, some with better properties than those available in the restricted classical case where $c_{\mathbf{k}} = f_{\mathbf{k}}$. The apparent drawback is the need for an additional step. We shall see later that this drawback is largely compensated by the gain in quality resulting from the larger selection of synthesis functions to choose from.

In the present approach, the synthesis function is not necessarily finite-support, nor is required to satisfy the interpolation property; in return, it becomes essential to assign a specific value to those samples $f_{\mathbf{k}}$ that are unknown because they are out of the range of our data. In practice, this assignment is implicit, and the unknown data are usually assumed to be mirrored from the known data.

3.1. Determination of the Coefficients

Suppose we want to enforce a condition akin to (2), in the context of generalized interpolation. Considering again only integer arguments $\mathbf{x} = \mathbf{k}_0$, we write

$$f_{\mathbf{k}_0} = \sum_{\mathbf{k} \in \boldsymbol{Z}^q} c_{\mathbf{k}}\, p_{\mathbf{k}_0 - \mathbf{k}} \qquad \forall \mathbf{k}_0 \in \boldsymbol{Z}^q, \tag{5}$$

where $p_{\mathbf{k}} = \varphi(\mathbf{k})$. Given some function φ that is known a priori, this expression is nothing but a linear system of equations in terms of the unknown coefficients $c_{\mathbf{k}}$. According to (5), the dimension of this system is infinite, both with respect to the number of equations (because all arguments $\mathbf{k}_0 \in \boldsymbol{Z}^q$ are considered), and to the number of unknowns (because all indexes $\mathbf{k} \in \boldsymbol{Z}^q$ are considered in the sum). One way to reduce this system to a manageable size is to remember that the number of known samples $\mathbf{k}_0$ is finite in practice (which limits the number of equations), and at the same time to constrain φ to be finite-support (which limits the number of unknowns). We are now faced with a problem of the form

$\mathbf{c} = \mathbf{P}^{-1}\,\mathbf{f}$, and a large part of the literature (e.g., [30]) is devoted to the development of efficient techniques for inverting the matrix $\mathbf{P}$ in the context of specific synthesis functions φ.

Another strategy arises once it is recognized that (5) is again a discrete convolution equation that can be written as

$$f_{\mathbf{k}_0} = (c * p)_{\mathbf{k}_0} \qquad \forall \mathbf{k}_0 \in \mathbb{Z}^q. \tag{6}$$

It directly follows that the infinite sequence of coefficients $\{c_{\mathbf{k}}\}$ can be obtained by convolving the infinite sequence $\{f_{\mathbf{k}}\}$ by the convolution-inverse $(p)^{-1}$. The latter is simply a sequence of numbers $\{(p)_{\mathbf{k}}^{-1}\}$ such that $\left(p * (p)^{-1}\right)_{\mathbf{k}} = \delta_{\mathbf{k}}$. This sequence is uniquely defined and does generally exist in the cases of interest. Convolving both sides of (6) by $(p)_{\mathbf{k}}^{-1}$, we get that

$$c_{\mathbf{k}_0} = \left((p)^{-1} * f\right)_{\mathbf{k}_0} \qquad \forall \mathbf{k}_0 \in \mathbb{Z}^q. \tag{7}$$

Since discrete convolution is nothing but a digital filtering operation, this suggests that discrete filtering can be an alternative solution to matrix inversion for the determination of the sequence of coefficients $\{c_{\mathbf{k}}\}$ needed to enforce the desirable constraint (5). A very efficient algorithm for performing this computation for an important class of synthesis functions can be found in [19, 20]; its computational cost for the popular cubic B-spline is two additions and three multiplications per produced coefficient.

3.2. Reconciliation

Comparing (1) with (4), it appears that classical interpolation is a special case of generalized interpolation with $c_{\mathbf{k}} = f_{\mathbf{k}}$ and $\varphi = \varphi_{\mathrm{int}}$. We show now that the converse is also true, since it is possible to interpret the generalized interpolation $f(\mathbf{x}) = \sum c_{\mathbf{k}}\, \varphi(\mathbf{x} - \mathbf{k})$ as a case of classical interpolation $f(\mathbf{x}) = \sum f_{\mathbf{k}}\, \varphi_{\mathrm{int}}(\mathbf{x} - \mathbf{k})$. For that, we have to determine the interpolant φ_{int} from its non-interpolating counterpart φ. From (4) and (7), we write

$$f(\mathbf{x}) = \sum_{\mathbf{k}_1 \in \mathbb{Z}^q} \left((p)^{-1} * f\right)_{\mathbf{k}_1} \varphi(\mathbf{x} - \mathbf{k}_1) = \sum_{\mathbf{k}_1 \in \mathbb{Z}^q} \sum_{\mathbf{k}_2 \in \mathbb{Z}^q} (p)_{\mathbf{k}_2}^{-1} f_{\mathbf{k}_1 - \mathbf{k}_2}\, \varphi(\mathbf{x} - \mathbf{k}_1).$$

We finally determine that the interpolant φ_{int} that is hidden behind a non-interpolating φ is

$$\varphi_{\mathrm{int}}(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^q} (p)_{\mathbf{k}}^{-1}\, \varphi(\mathbf{x} - \mathbf{k}). \tag{8}$$

It is crucial to understand that this equivalence allows the exact and efficient handling of an infinite-support interpolant φ_{int} by performing operations only with a finite-support, non-interpolating function φ. The freedom to select a synthesis function is much larger after the interpolation constraint has been removed; this opens up the use of synthesis functions φ that offer much better performance (for a given computational cost) than any explicit interpolant φ_{int}.

IV. TERMINOLOGY AND OTHER NONSENSE

Since the need for interpolation arises so often in practice, its droning use makes it look like a simple operation. This is deceiving, and the fact that the interpolation terminology is so perplexing hints at hidden difficulties. Let us attempt to befuddle the reader: the cubic B-spline is a piecewise polynomial function of degree three. It does not correspond to what is generally understood as cubic convolution, the latter being a Keys' function made of piecewise polynomials of degree three (like the cubic B-spline) and of maximal order three (contrary to the cubic B-spline, for which the order is four). No B-spline of sufficient degree should ever be used as an interpolant φ_{int}, but a high-degree B-spline makes for a high-quality synthesis function φ. The Appledorn function of degree four is no polynomial and has order zero. There is no degree that would be associated to the sinc function, but its order is infinite. Any polynomial of a given degree can be represented by splines of the same degree, but, when the spline degree increases to infinity, the cardinal spline tends to the sinc function, which can at best represent a polynomial of degree zero. In order to respect isotropy in two dimensions, we expect that the synthesis function itself must be endowed with rotational symmetry; yet, the best possible function (sinc) is not. Kriging is known to be the optimal unbiased linear interpolator, yet it does not belong to the category of linear systems; an example of linear system is the Dodgson synthesis function made of quadratic polynomials. Bilinear transformation and bilinear interpolation have nothing in common. Your everyday image is low-pass, yet its most important features are its edges. And finally, despite more than ten years of righteous claims to the contrary [31], some authors (who deserve no citation) persist in believing that every synthesis function φ built to match Equation (4) can be used in Equation (1) as well, in the place of φ_{int}. Claims that the use of a cubic B-spline blurs data, which are wrongly made in a great majority of image processing textbooks, are a typical product of this misunderstanding.

Confused?

We hope that the definitions of order and degree, given later in this paper, will help to clarify this mess. We also hope to make clear when to use Equation (1) and when to use the generalized approach of Equation (4).

V. Artifacts

In most clinical situations, data are available once only, at a given resolution (or sampling step). Thus, there exist no absolute truth regarding the value of f between its samples $f_{\mathbf{k}}$; moreover, there is no rigorous way to check whether an interpolation model corresponds to the physical reality without introducing at least some assumptions. For these reasons, it is necessary to resort to mathematical analysis for the assessment of the quality of interpolation. The general principle is to define an interpolated function f_h as given by a set of samples that are h units apart and that satisfy

$$f_h(\mathbf{x}) = \sum_{\mathbf{k}\in \boldsymbol{Z}^q} c_{\mathbf{k}}\, \varphi(\tfrac{1}{h}\mathbf{x} - \mathbf{k}) \qquad \forall \mathbf{x} \in \boldsymbol{R}^q,$$

with the interpolation constraint that $f_h(h\mathbf{k}) = f(h\mathbf{k})$ for all $\mathbf{k} \in \boldsymbol{Z}^q$. The difference between $f_h(\mathbf{x})$ and $f(\mathbf{x})$ for all $\mathbf{x} \in \boldsymbol{R}^q$ will then describe how fast the interpolated function f_h converges to the true function f when the samples that define f_h become more and more dense, or, in other words, when the sampling step h becomes smaller and smaller. When this sampling step is unity, we are in the conditions of Equation (4). The details of the mathematical analysis address issues such as how to measure the error between f_h and f, and how to restrict—if desired—the class of admissible functions f. Sometimes, this mathematical analysis allows the determination of a synthesis function with properties that are optimal in a given sense [27]. A less rigorous approach is to perform experiments that involve interpolation and resampling, often followed by visual judgment. Some effects associated to interpolation have been named according to the results of such visual experiments; the most perceptible effects are called ringing, aliasing, blocking, and blurring.

5.1. Resampling

Resampling by interpolation is generally understood as the following procedure:

1) *Take a set of discrete data* $f_{\mathbf{k}}$*;*
2) *Build by interpolation a continuous function* $f(\mathbf{x})$*;*
3) *Perform a geometric transformation* T *that yields* $f(T(\mathbf{x})) = \sum f_{\mathbf{k}_1}\,\varphi(T(\mathbf{x}) - \mathbf{k}_1)$*;*
4) *Summarize the continuous function* $f(T(\mathbf{x}))$ *by a set of discrete data samples* $f(T(\mathbf{k}_2))$.

Often, the geometric transformation results in a change of sampling rate. Irrespective of whether this change is global or only local, it produces a continuous function $f(T(\mathbf{x}))$ that cannot be represented exactly by the specific interpolation model that was used to build $f(\mathbf{x})$ from $f_{\mathbf{k}}$. In general, given an arbitrary transformation T, no set of coefficients $c_{\mathbf{k}}$ can be found for expressing $f(T(\mathbf{x}))$ as an exact linear combination of shifted synthesis functions. By the resampling operation, what is reconstructed instead is the function $g(\mathbf{x}) \neq f(T(\mathbf{x}))$ that satisfies

$$g(\mathbf{x}) = \sum_{\mathbf{k}_2 \in \mathbf{Z}^q} f(T(\mathbf{k}_2))\,\varphi_{\text{int}}(\mathbf{x} - \mathbf{k}_2) \qquad \forall \mathbf{x} \in \boldsymbol{R}^q,$$

where $g(\mathbf{x})$ and $f(T(\mathbf{x}))$ take the same value at the sample locations $\mathbf{x} = \mathbf{k}_2$, but not necessarily elsewhere.

It follows that the resulting continuous function $g(\mathbf{x})$ that could be reconstructed is only an approximation of $f(T(\mathbf{x}))$. Several approaches can be used to minimize the approximation error (e.g., least-squares error over a continuous range of abscissa [32]). Since the result is only an approximation, one should expect artifacts. Those have been classified in the four broad categories called ringing, aliasing, blocking, and blurring.

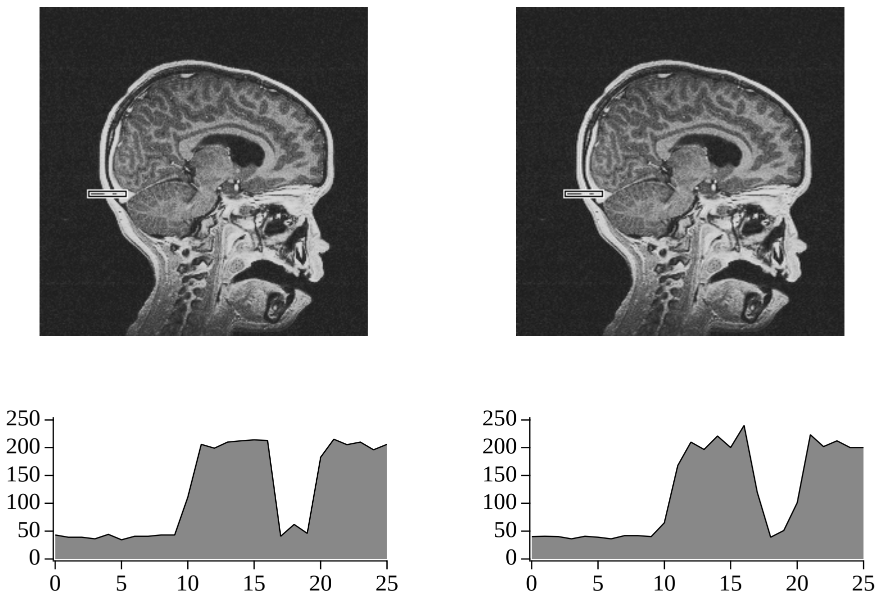


Figure 1: Ringing. Especially with high-quality interpolation, oscillations may appear after horizontal translation by half a pixel. Left: original MRI. Right: translated MRI.

5.2. Ringing

Ringing arises because most good synthesis functions are oscillating. In fact, ringing is less of an artifact—in the sense that it would correspond to a deterioration of the data—than the consequence of the choice of a model: it may sometimes happen (but this is not the rule) that data are represented (e.g., magnified or zoomed) in such a way that the representation, although appearing to be plagued with ringing "artifacts", is nonetheless exact, and allows the perfect recovery of the initial samples. Ringing can also be highlighted by translating by a non-integer amount a signal where there is a localized domain of constant samples bordered by sharp edges. After interpolation, translation and resampling, the new samples do no more exhibit a constant value over the translated domain, but they tend to oscillate. This is known as the Gibbs effect; its perceptual counterpart is the Mach bands phenomena. Figure 1 shows an occurrence of ringing in the outlined area due to the horizontal translation of a high-contrast image by half a pixel.

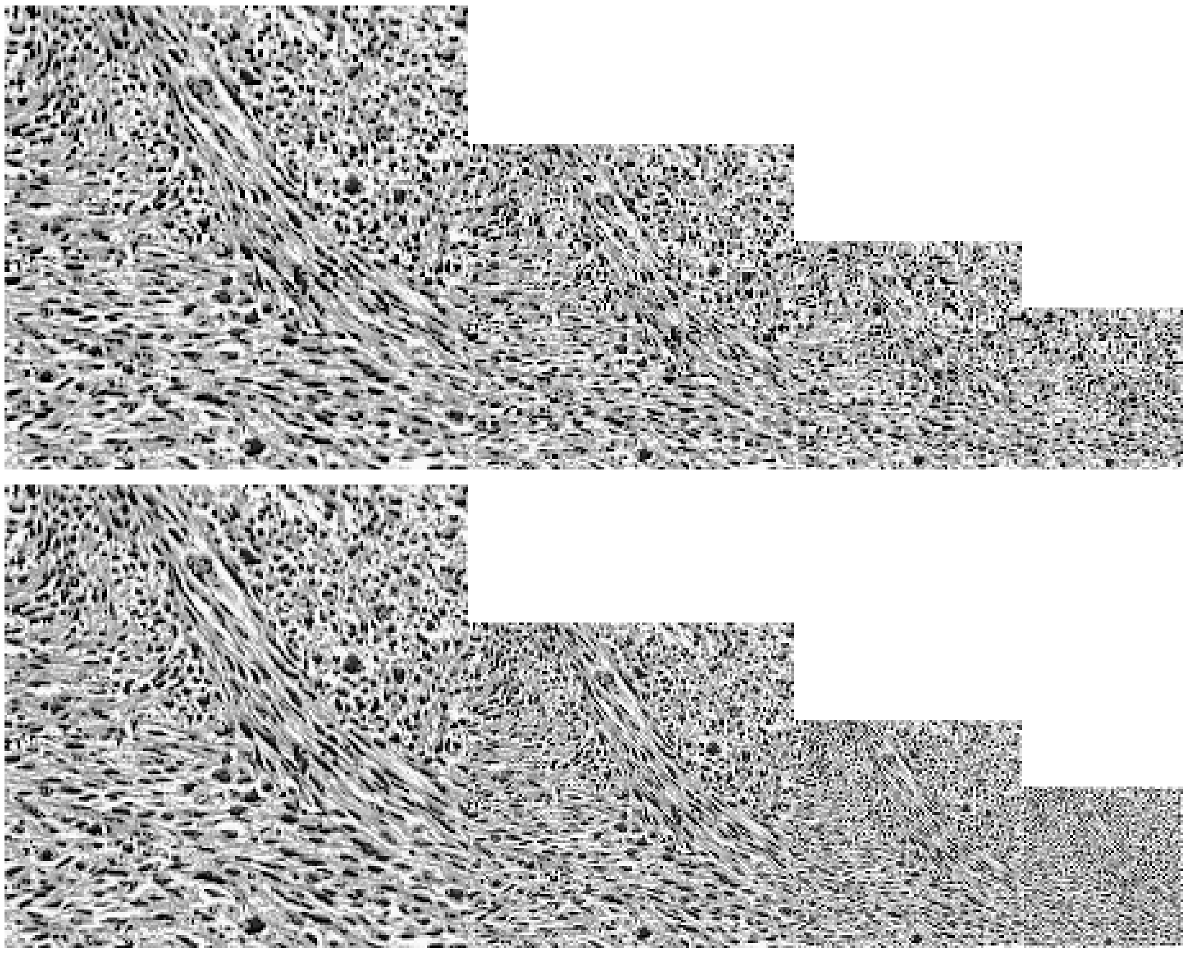


Figure 2: Aliasing. Top: low quality introduces a lot of aliasing. Bottom: better quality results in less aliasing. Both: at too coarse scale, the structural appearance of the bundles of cells is lost.

5.3. Aliasing

Unlike ringing, aliasing is a true artifact because it is never possible to perform exact recovery of the initial data from their aliased version. Aliasing is related to the discrete nature of the samples. When it is desired to represent a coarser version of the data using less samples, the optimal procedure is first to create a precise representation of the coarse data that uses every available sample, and then only to downsample this coarse representation. In some sense, aliasing appears when this procedure is not followed, or when there is a mismatch between the coarseness of the intermediate representation and the degree of downsampling (not coarse enough or too much downsampling). Typical visual signatures of aliasing are moiré effects and the loss of texture. Figure 2 illustrates aliasing.

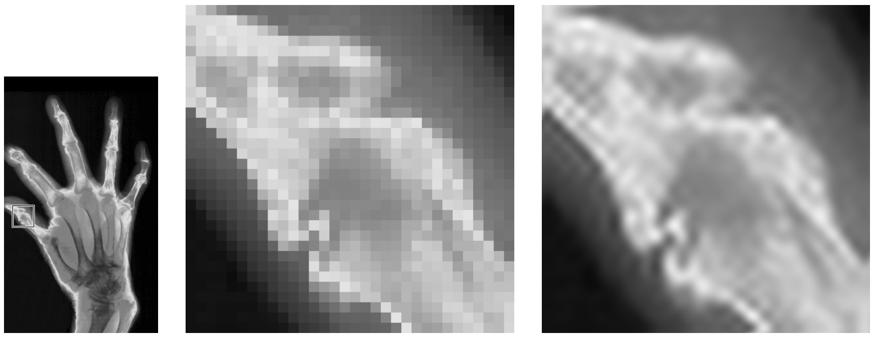


Figure 3: Blocking. After low-quality magnification, the highlighted area (left) appears pixellated (center) Better-quality magnification results in less pixellation (right).

5.4. Blocking

Blocking arises when the support of the interpolant is finite. In this case, the influence of any given pixel is limited to its surroundings, and it is sometimes possible to discern the boundary of this influence zone. Synthesis functions with sharp transitions, such as those in use with the method named nearest-neighbor interpolation, exacerbate this effect. Figure 3 presents a typical case of blocking.

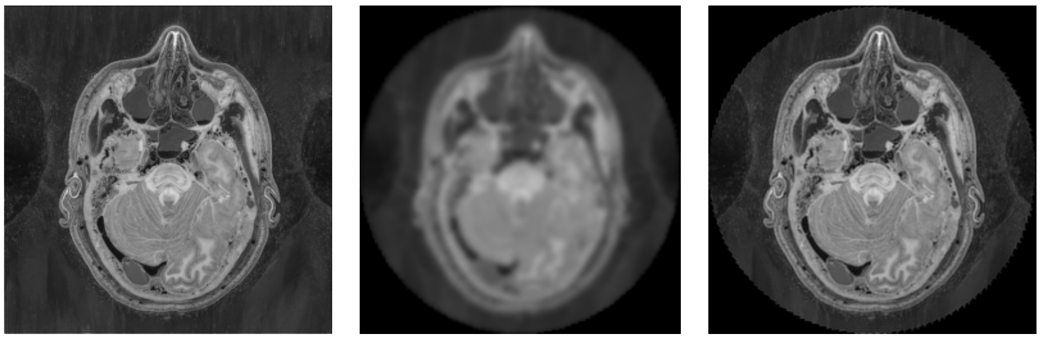


Figure 4: Blurring. Iterated rotation may lose many small-scale structure when the quality of interpolation is insufficient (center). Better quality results in less loss (right). Left: original.

5.5. Blurring

Finally, blurring is related to aliasing in the sense that it is also a mismatch between an intermediate data representation and their final downsampled or oversampled version. In this case, the mismatch is such that the intermediate data is too coarse for the task. This results in an image that appears to be out of focus. When the filter associated to the synthesis function φ or φ_{int} is very different from an ideal filter, aliasing and blurring can occur simultaneously (they usually do). Note that, both for aliasing and blurring considerations, the intermediate representation need not be explicitly available or computed. To highlight blurring, it is enough to iterate the same interpolation operation several times, thus effectively magnifying the effect. Figure 4 has been obtained by the compound rotation of an image by 36 steps of $10°$ each.

VI. Desirable Properties

The quality of geometric operations on images or volumes is very relevant in the context of biomedical data analysis. For example, the comparison of images taken with two different conditions requires that the geometric operation that aligns one with the other be high-quality in order to allow a detailed analysis of their differences. That is to say, it is not enough that the geometric alignment be correctly specified; it is also crucial that it be correctly performed. Typical instances of that class of problems involve functional Magnetic Resonance Imaging (fMRI), where the relative amplitude of the difference between two conditions (e.g., active vs. inactive) is very small. High quality (fidelity to the original) is also a desirable trait in common display operations such as the reslicing of anisotropic data or the change of scale and orientation of many pictorial representations. Typically, a physician will request that rotating or zooming (magnifying) a region of interest of an image does introduce no spurious features, while keeping all the minute details he might be interested in.

The price to pay for high-quality interpolation is computation time. For this reason, it is important to select a synthesis function that offers the best trade-off. There are several aspects to consider. The most important deals with the support of φ_{int} or φ, which is a measure of the interval in which we have that $\varphi(\mathbf{x}) \neq 0$. The larger the support, the more the computation time. Another important aspect is the quality of approximation inherent in the synthesis function. Often, the larger the support, the better the quality; but it should be noted that the quality of synthesis functions with identical support may vary. Other aspects involve the ease of analytical manipulation (when useful), the ease of computation, and the efficiency of the determination of the coefficients $c_{\mathbf{k}}$ when φ is used instead of φ_{int}.

6.1. Separability

Consider Equations (1) or (4) in multidimensions, with $q > 1$. To simplify the situation, we restrict the interpolant φ_{int} or the non-interpolating φ to be finite-support. Without loss of generality, we assume that this support is of size S^q (e.g., a square with side S in two dimensions, a cube in three dimensions). This means that the equivalent of 1D interpolation with a synthesis function requiring, say, 5 evaluations, would require as much as 125 function evaluations in 3D. Figure 5 shows what happens in the intermediate 2D situation. This large computational burden can only be reduced by imposing restrictions on φ. An easy and convenient way is to ask that the synthesis function be separable, as in

$$\varphi_{\text{sep}}(\mathbf{x}) = \prod_{i=1}^{q} \varphi(x_i) \quad \forall \mathbf{x} = \left(x_1, x_2, \ldots, x_q\right) \in \boldsymbol{R}^q .$$

The very beneficial consequence of this restriction is that the data can be processed in a separable fashion, line-by-line, column-by-column, and so forth. In particular, the determination of the interpolation coefficients needed for generalized interpolation is separable, too, because the form (4) is linear. In the previous example, the $5^3 = 125$ evaluations needed in 3D reduce to $3 \times 5 = 15$ evaluations of a 1D function when it is separable. We show in Figure 5 how the $5^2 = 25$ evaluations needed in the 2D non-separable case become $2 \times 5 = 10$ in the separable case. For the rest of this paper, we concentrate on separable synthesis functions; we describe them and analyze them in one dimension, and we use the expression above to implement interpolation efficiently in a multidimensional context.

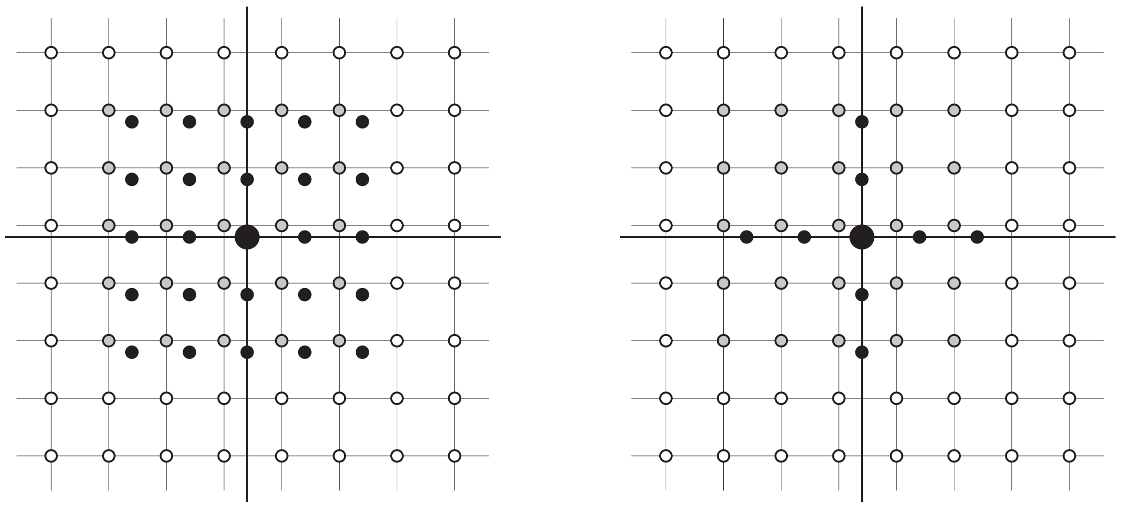


Figure 5 left: Non-separable 2D interpolation with a square support of side 5 *(*25 *function evaluations). Large central black dot: coordinate* $\mathbf{x}$ *at which* $f(\mathbf{x})=\sum c_{\mathbf{k}}\,\varphi(\mathbf{x}-\mathbf{k})$ *is computed. All black dots: coordinates* $\mathbf{x}$ *corresponding to the computation of* $\varphi(\mathbf{x}-\mathbf{k})$.

Figure 5 right: Separable 2D interpolation (10 *function evaluations). Large central black dot: coordinate* $\mathbf{x}$ *where the value* $f(\mathbf{x})=\sum\left(\sum c_{k_1,k_2}\,\varphi(x_1-k_1)\right)\varphi(x_2-k_2)$ *is computed. All black dots: coordinates* x_i *corresponding to the computation of* $\varphi(x_i-k_i)$.

Figure 5 left and right: The white and gray dots give the integer coordinates $\mathbf{k}$ *where the coefficients* $c_{\mathbf{k}}$ *are defined. Gray dots: coefficients* $c_{\mathbf{k}}$ *that contribute to the interpolation.*

6.2. Symmetry

Preserving spatial relations is a crucial issue for any imaging system. Since interpolation can be interpreted as the filtering (or equivalently, convolution) operation proposed in Equation (3), it is important that the phase response of the involved filter does not result in any phase degradation. This consideration translates into the well-known and desirable property of symmetry such that $\varphi(\mathbf{x})=\varphi(-\mathbf{x})$ or $\varphi_{\mathrm{int}}(\mathbf{x})=\varphi_{\mathrm{int}}(-\mathbf{x})$. Symmetry is satisfied by all synthesis functions considered here, at the possible minor and very localized exception of nearest-neighbor interpolation. Symmetry implies that the only coefficients $c_{\mathbf{k}}$ that are needed in Figure 5 are those that are closest to the coordinates $\mathbf{x}$ corresponding to the computation of $\varphi(\mathbf{x}-\mathbf{k})$. In the specific case of Figure 5, there are 25 of them, both for a separable φ_{sep} and a non-separable φ.

6.3. Partition of Unity

How can we assess the inherent quality of a given synthesis function? We answer this question gradually, developing it more in the next section, and we proceed at first more by intuition than by a rigorous analysis. Let us consider that the discrete sequence of data we want to interpolate is made of samples that all take exactly the same value $f_{\mathbf{k}}=f_{\mathbf{0}}$ for any $\mathbf{k}\in\boldsymbol{Z}^q$. In this particular case, we intuitively expect that the interpolated continuous function $f(\mathbf{x})$ should also take a constant value (preferably the same $f_{\mathbf{0}}$) for all arguments $\mathbf{x}\in\boldsymbol{R}^q$. This desirable property is called the reproduction of the constant. Its relevance is particularly high in image processing because the spectrum of images is very often concentrated towards low frequencies. From (1), we derive

$$1=\sum_{\mathbf{k}\in\boldsymbol{Z}^q}\varphi_{\mathrm{int}}(\mathbf{x}-\mathbf{k})\qquad\forall\mathbf{x}\in\boldsymbol{R}^q.$$

This last equation is also known as the partition of unity. It is equivalent to impose that its Fourier transform satisfies some sort of interpolation property in the Fourier domain (see Appendix). The reproduction of the constant is also desirable for a non-interpolating synthesis function φ, which is equivalent to ask that it satisfies the partition of unity condition, too. To see why, we remember that the set of coefficients $c_{\mathbf{k}}$ used in the reconstruction equation (4) is obtained by digital filtering of the sequence of samples $\{\ldots, f_{\mathbf{0}}, f_{\mathbf{0}}, f_{\mathbf{0}}, \ldots\}$. Since the frequency representation of this sequence is exclusively concentrated at the origin, filtering will simply result in another sequence of coefficients $\{\ldots, c_{\mathbf{0}}, c_{\mathbf{0}}, c_{\mathbf{0}}, \ldots\}$ that also have a constant value. We have that

$$\frac{1}{c_{\mathbf{0}}} = \sum_{\mathbf{k}\in \boldsymbol{Z}^q} \varphi(\mathbf{x}-\mathbf{k}) \qquad \forall \mathbf{x} \in \boldsymbol{R}^q .$$

It is common practice to impose that φ be given in a normalized (or scaled) form, such that $f_{\mathbf{0}} = 1$ implies that $c_{\mathbf{0}} = 1$.

VII. Approximation Theory

So far, we have only two types of synthesis functions: those that do reproduce the constant, and those that do not. We introduce now more categories. Firstly, we perform the following experiment:

1) *Take some arbitrary square-integrable function* $f(\mathbf{x})$ *and select a sampling step* $h > 0$*;*
2) *Create a set of samples* $f(h\mathbf{k})$*;*
3) *From this sequence, using either (1) or (4), build an interpolated function* $f_h(\mathbf{x}) = \sum c_{\mathbf{k}}\, \varphi(\frac{1}{h}\mathbf{x}-\mathbf{k})$*;*
4) *Compare* f *and* f_h *using some norm, for example the mean-square (or* L_2*) norm* $\varepsilon^2(h) = \|f - f_h\|_{L_2}^2 = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left(f(\mathbf{x}) - f_h(\mathbf{x})\right)^2 \,\mathrm{d}x_1 \cdots \mathrm{d}x_q$.

When the sampling step h gets smaller, more details of f can be captured; it is then reasonable to ask that the approximation error $\varepsilon(h)$ gets smaller, too. The fundamental question are: how much smaller, what influence has the free choice of the arbitrary function f, and what role does play the synthesis function φ that is used to build f_h?

We respond to these questions by introducing a formula for predicting the approximation error in the Fourier domain [33, 34, 35]

$$\eta^2(h) = \tfrac{1}{2\pi} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left|\hat{f}(\omega)\right|^2 E(\omega h) \,\mathrm{d}\omega_1 \cdots \mathrm{d}\omega_q , \tag{9}$$

where $\hat{f}(\omega)$ is the Fourier transform of the arbitrary function $f(\mathbf{x})$ (see Appendix), and where E is an error kernel that depends on the synthesis function only, and that is given by

$$E(\omega) = \left(\left| \sum_{\mathbf{k}\in \boldsymbol{Z}_*^q} \hat{\varphi}(\omega + 2\pi\mathbf{k}) \right|^2 + \sum_{\mathbf{k}\in \boldsymbol{Z}_*^q} \left| \hat{\varphi}(\omega + 2\pi\mathbf{k}) \right|^2 \right) \Big/ \left| \sum_{\mathbf{k}\in \boldsymbol{Z}^q} \hat{\varphi}(\omega + 2\pi\mathbf{k}) \right|^2 . \tag{10}$$

The equivalence $\varepsilon = \eta$ holds for band-limited functions. For those functions that do not belong to that class, the estimated error $\eta(h)$ must be understood as the average error over all possible sets of samples $f(h\mathbf{k} + \Delta)$, where $\Delta = (\Delta_1, \Delta_2, \ldots, \Delta_q)$ is some phase term with $\Delta_i \in [0, h[$. When $q = 1$, for band-limited functions f and when the synthesis function φ_{int} is interpolating, this error kernel reduces to the kernel proposed in [36].

On the face of Equation (9), a decrease in the sampling step h will result in a decrease of the argument of E. Since the function f is arbitrary, and since it is desired that the approximation error $\varepsilon(h)$ vanishes for a vanishing sampling step h, the error kernel itself must also vanish at the origin. It is thus interesting to develop E in a Mac-Laurin series around the origin (for simplicity, we consider only the 1D case here). Since this function is even (i.e., symmetric), only even factors need be considered, and the Mac-Laurin development is

$$E(\omega) = \sum_{n \in \boldsymbol{N}} \frac{E^{(2n)}(0)}{(2n)!}\,\omega^{2n},$$

where $E^{(2n)}$ is the $(2n)$-th derivative of the error kernel. By definition, the order of differentiation L for which $E^{(2L)}(0) \neq 0$ and $E^{(2n)}(0) = 0 \quad \forall n \in [0, L-1]$, is called the approximation order of φ. Thus, for a synthesis function of order L, the infinite Mac-Laurin expansion is given by

$$E(\omega) = \left(C_\varphi\right)^2 \omega^{2L} + \left(\sum_{n=L+1}^{\infty} \frac{E^{(2n)}(0)}{(2n)!}\,\omega^{2n}\right),$$

where the constant C_φ depends on φ only. When the sampling step is small enough, we can neglect the high-order terms of this expansion. The introduction of the resulting expression of E into (9) yields

$$\eta^2(h) = \left(C_\varphi\right)^2 h^{2L} \left(\tfrac{1}{2\pi}\int_{-\infty}^{\infty} \left|\omega^L \hat{f}(\omega)\right|^2 \mathrm{d}\omega\right) \text{ as } h \to 0,$$

where the parenthesized expression is recognized as being the norm of the L-th derivative of the smooth function f we started from.

Finally, for a synthesis function of approximation order L, we get that

$$\eta(h) = \left\|f - f_h\right\|_{L_2} = C_\varphi\, h^L \left\|f^{(L)}\right\|_{L_2} \text{ as } h \to 0. \qquad (11)$$

This result expresses that we can associate to any φ a number L and a constant C_φ such that the error of approximation ε predicted by η decreases like h^L, when h is sufficiently small. Since this decrease can be described as being $O(h^L)$, the number L is called the approximation order of the synthesis function φ. This process happens without regard to the specific function f that is first being sampled with step h, and then reincarnated as f_h.

• Thesis

This analysis is relevant to image processing because most images have an essentially low-pass characteristic, which is equivalent to say that they are oversampled, or in other words, that the sampling step h is small with respect to the spatial scale over which image variations occur. In this sense, the number L that measures the approximation order is a relevant parameter.

• Antithesis

Is approximation order really important? After all, the effect of a high exponent L kicks in only when the sampling step h gets small enough; but, at the same time, common sense dictates that the amount of data to process grows like h^{-1}. The latter consideration (efficiency, large h) often wins over the former (quality, small h) when it comes to settle a compromise. Moreover, the important features of an image reside in its edges which, by definition, are very localized, thus essentially high-pass. Most of the time anyway, there is no debate at all because h is simply imposed by the acquisition hardware. Thus, the relevance of L is moot when efficiency considerations lead to critical sampling.

• Synthesis

Equations (9) and (10) describe the evolution of the error for every possible sampling step h; thus, the error kernel E is a key element when it comes to the comparison of synthesis functions, not only near the origin, but over the whole Fourier axis. In a certain mathematical sense, the error kernel E can be understood as a way to predict the approximation error when φ is used to interpolate a sampled version of the infinite-energy function $f(x)=\sin(\omega x)$. Being a single number, but being also loaded with relevant meaning, the approximation order is a convenient summary of this whole curve.

7.1. Strang-Fix Equivalence

Suppose we are interested in just the approximation order L of a synthesis function φ, without caring much about the details of E. In this case, the explicit computation of (10) is not necessary. Instead, Strang-Fix [37] have proposed a series of conditions that are equivalent to (11). The interest of these equivalent conditions is that they can be readily tested. They are valid for all synthesis functions with sufficient decay—sinc is one of the very rare cases where these conditions are not satisfied. We mention three equivalent 1D Strang-Fix conditions as

1) *L-th order zeros in the Fourier domain*

$$\begin{cases} \hat{\varphi}(0)=1 \\ \hat{\varphi}^{(n)}(2\pi k)=0 \qquad k\in \boldsymbol{Z}_* \qquad n\in[0,L-1] \end{cases};$$

2) *Reproduction of all monomials of degree* $n\le N=L-1$

$$\forall n\in[0,N] \quad \exists\left\{\dots,c_{-1}^{(n)},c_{0}^{(n)},c_{1}^{(n)},\dots\right\} \quad \Big| \quad \sum_{k\in\boldsymbol{Z}} c_k^{(n)}\,\varphi(x-k)=x^n.$$

3) *Discrete moments*

$$\sum_{k\in\boldsymbol{Z}}(x-k)^n\,\varphi(x-k)=\mu_n \qquad \forall n\in[0,L-1],$$

where μ_n *depends on* n *only.*

Under mild hypothesis, any of these conditions is equivalent to $\varepsilon(h)\le \text{Const}\times h^L\,\left\|f^{(L)}\right\|_{L_2}$.

7.2. Reproduction of the Polynomials

We have already discussed the fact that the approximation order L is relevant only when the data are oversampled, and we mentioned that this is indeed broadly true for typical images. The new light brought by the Strang-Fix theory is that it is equivalent to think in terms of frequency contents or in terms of polynomials—apart from some technical details. Intuitively, it is reasonable to think that, when the data is smooth at scale h, we can model it by polynomials without introducing too much error. This has been formalized as the Weierstrass approximation theorem. What the Strang-Fix theory tells us is that there exist a precise relation between the approximation order and the maximal degree of the polynomials that the synthesis function can reproduce exactly. For example, we started this discussion on the theoretical assessment of the quality of any synthesis function by investigating the reproduction of the constant. We have now completed a full circle and are back to the reproduction of polynomials, of which the constant is but a particular case.

7.3. Regularity

Consider sampling a smooth function f. From the samples $f(hk)$, and by interpolation, reconstruct a function f_h that is an approximation of f. Since f is smooth, intuition tells us that it is desirable that f_h be smooth as well; in turn, intuition dictates that the only way to ensure this smoothness is that φ be smooth, too. These considerations could lead one to the following syllogism:

1) *The order of approximation L requires the reproduction of monomials of degree $L-1$;*
2) *Monomials of degree $N = L-1$ are functions that are at least N-times differentiable;*
3) *An N-times differentiable synthesis function is required to reach the L-th order of approximation.*

Intuition is sometimes misleading.

A function that is at least n-times continuously differentiable is said to have regularity C^n. A continuous, but otherwise non-differentiable function is labeled C^0, while a discontinuous function is said to possess no regularity. Some authors insist that the regularity of the synthesis function is an important issue [38]. This may be true when the differentiation of f_h is needed, but differentiating data more than, say, once or twice, is uncommon in everyday applications. Often, at most the gradient of an image is needed; thus, it is not really necessary to limit the choice of synthesis functions to those that have a high degree of regularity. In fact, the conclusion of the syllogism above is incorrect, and a synthesis function does not need to be N-times differentiable to have an $N+1$ order of approximation.

For example, Schaum [28] has proposed a family of interpolants inspired by Lagrangian interpolation. A member of this family is shown at Figure 6; it is made of pieces of quadratic polynomials patched together. Despite the fact that this function is discontinuous, it possesses an approximation order $L = 3$. Thus, a linear sum of those shifted functions with well-chosen coefficients is able to exactly reproduce a constant, a ramp, and a quadratic as well. In this specific case, the synthesis function is interpolating; thus, we have that

$$\forall k \in \boldsymbol{Z} \quad \begin{cases} f_k = 1 & \Rightarrow \quad f(x) = 1 \qquad \forall x \in \boldsymbol{R} \\ f_k = k & \Rightarrow \quad f(x) = x \qquad \forall x \in \boldsymbol{R}. \\ f_k = k^2 & \Rightarrow \quad f(x) = x^2 \qquad \forall x \in \boldsymbol{R} \end{cases}$$

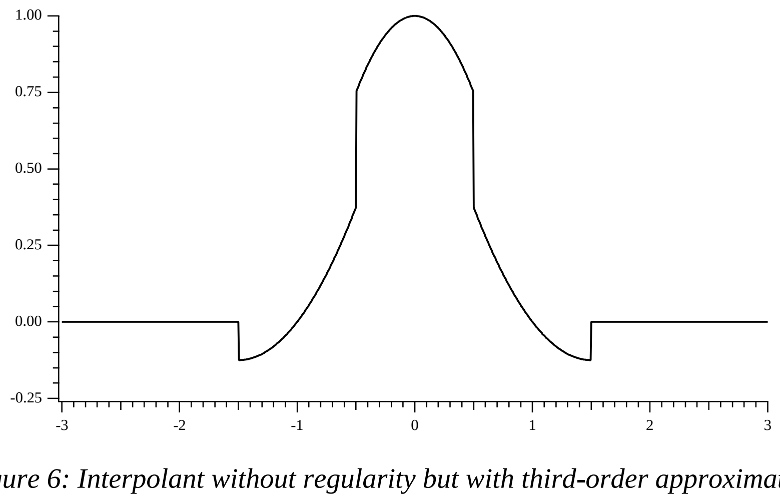


Figure 6: Interpolant without regularity but with third-order approximation

7.4. Approximation Kernel Example

Comparing the sinc to the nearest-neighbor interpolation provides a good test case to grasp the predictive power of E. Since the Fourier transform of the sinc function is simply a rectangular function that takes a unit value in $[-\pi, \pi]$ and is zero elsewhere, the denominator in (10) is unity for any frequency ω (because a single term of the main domain contributes to the infinite sum). On the other hand, since the summation is performed over non-null integers $k \in \boldsymbol{Z}_*$, there is no contributing term on the numerator side and E is zero in the main domain $[-\pi, \pi]$. By a similar reasoning, it takes value two outside of the main domain. This corresponds to the well known fact that a sinc synthesis function can represent a band-limited function with no error, and at the same time suffers a lot from aliasing when the function is not band-limited.

Nearest-neighbor interpolation is characterized by a rectangular synthesis function, the Fourier transform of which is a sinc function (this situation is the converse of the previous case). Unfortunately, Expression (10) is now less manageable. We can nevertheless plot a numeric estimate of (9). The resulting approximation kernels are represented at Figure 7.

At the origin, when $\omega = 0$, it is apparent from Figure 7 that both sinc and nearest-neighbor interpolation produce no error; thus, they reproduce the constant. More generally, the degree of "flatness" at the origin (the number of vanishing derivatives) directly gives the approximation order of the synthesis function—being a straight line in the sinc case, this flatness is infinite, and so is the approximation order. When ω grows, interpolation by nearest-neighbor is less good than sinc interpolation, the latter being perfect up to Nyquist's frequency. Less known, but apparent from Figure 7, is the fact that nearest-neighbor interpolation is indeed better than sinc for some (not all) of the part (if any) of the function to interpolate that is not band-limited.

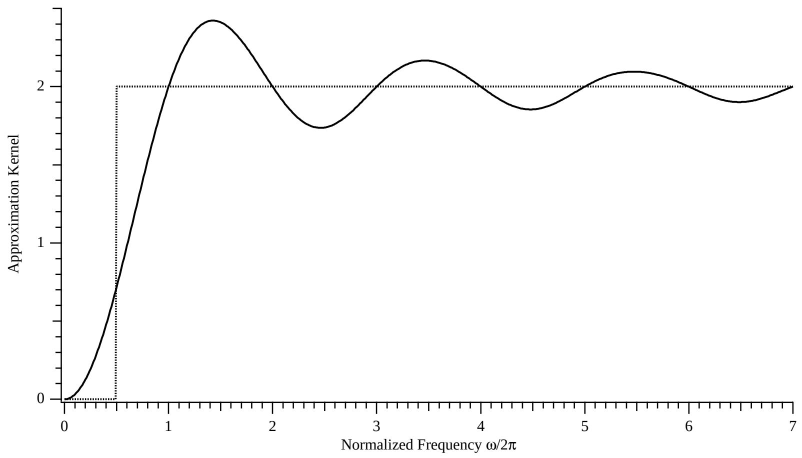


Figure 7: Approximation kernel in Fourier. Solid line: nearest-neighbor. Dotted line: sinc

VIII. Specific Examples

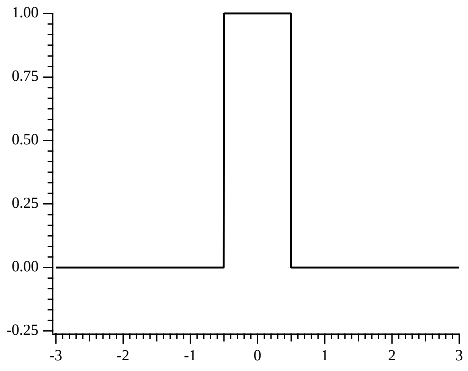
In this section, we present several synthesis functions and discuss some of their properties. We give their explicit form, their support, we graph their shape and show their equivalent interpolating form when they are no interpolant themselves. We also give their approximation order and their approximation kernel, along with their regularity.

8.1. Nearest-Neighbor

The synthesis function associated to nearest-neighbor interpolation is the simplest of all, since it is made of a square pulse. Its support is unity; it is an interpolant, and satisfies the partition of unity, provided a slight asymmetry is introduced at the edges of the square pulse. The approximation order is one (it reproduces at most the constant). It is discontinuous, thus has no regularity; its expression is given by

$$\varphi^0(x) = \begin{cases} 0 & x < \frac{-1}{2} \\ 1 & \frac{-1}{2} \le x < \frac{1}{2} \\ 0 & \frac{1}{2} \le x \end{cases}.$$

The main interest of this synthesis function is its simplicity, which results in the most efficient of all implementations. In fact, for any coordinate $\mathbf{x}$ where it is desired to compute the value of the interpolated function f, there is only one sample $f_{\mathbf{k}}$ that contributes, no matter how many dimensions q are involved. The price to pay is a severe loss of quality.



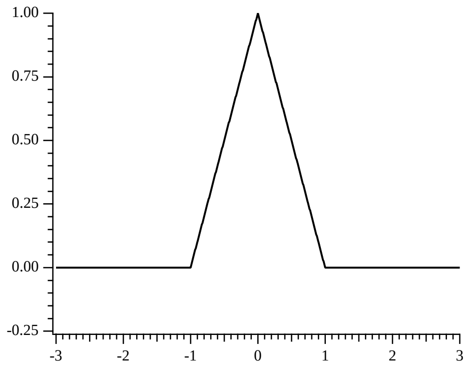


Figure 8: Synthesis functions. Left: nearest-neighbor. Right: linear

8.2. Linear

The linear interpolant enjoys a large popularity because the complexity of its implementation is very low, just above that of the nearest-neighbor; moreover, some consider that it satisfies Occam's razor principle by being the simplest interpolant one can think of that builds a continuous function f out of a sequence of discrete samples $\{f_k\}$. It is made of the (continuous-signal) convolution of a square pulse with itself, which yields a triangle, sometimes also named a hat or a tent function. Its support covers two units; it is an interpolant, and its approximation order is two (it reproduces straight lines of any finite slope). Its regularity is C^0, which expresses that it is continuous but not differentiable. In 1D, this interpolant requires at most two samples to produce an interpolated value. In 2D, also called bilinear interpolation, its separable implementation requires four samples, and six in 3D (eight in the non-separable case, where it is called trilinear interpolation [9, 39]). The expression for the 1D linear synthesis function is

$$\beta^1(x) = \begin{cases} 1 - |x| & |x| < 1 \\ 0 & 1 \le x \end{cases}.$$

8.3. B-splines

There is a whole family of synthesis functions made of B-splines. By convention, their symbolic representation is β^n, where $n \in \boldsymbol{N}$ is not a power, but an index called the degree of the spline. These functions are piecewise polynomials of degree n; they are globally symmetric and $(n-1)$-times continuously differentiable. Thus, their regularity is C^{n-1}. They are given by

$$\beta^0(x) = \begin{cases} 1 & |x| < \frac{1}{2} \\ \frac{1}{2} & |x| = \frac{1}{2} \\ 0 & |x| > \frac{1}{2} \end{cases}$$

and

$$\beta^n(x) = \sum_{k=0}^{n+1} \frac{(-1)^k\,(n+1)}{(n+1-k)!\,k!} \left(\tfrac{n+1}{2} + x - k\right)_+^n \qquad \forall x \in \boldsymbol{R}, \quad \forall n \in \boldsymbol{N}_*,$$

where by definition

$$(x)_+^n = \left(\max(0, x)\right)^n \qquad n > 0.$$

Both the support and the approximation order of these functions is one more than their degree. They enjoy many other interesting properties that fall outside the scope of this chapter, except perhaps the fact that a B-spline derivative can be computed recursively by

$$\frac{\mathrm{d}}{\mathrm{d}x}\beta^n(x) = \beta^{n-1}(x + \tfrac{1}{2}) - \beta^{n-1}(x - \tfrac{1}{2}) \qquad n > 0.$$

Then, computing the exact gradient of a signal given by a discrete sequence of interpolation coefficients $\{c_k\}$ can be done as follows:

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \sum_{k \in \boldsymbol{Z}} c_k \frac{\mathrm{d}}{\mathrm{d}x}\beta^n(x-k) = \sum_{k \in \boldsymbol{Z}} \left(c_k - c_{k-1}\right)\beta^{n-1}(x - k + \tfrac{1}{2}),$$

where the n-times continuous differentiability of B-splines ensures that the resulting function is smooth when $n \geq 3$, or at least continuous when $n \geq 2$.

• Degree $n = 0$

The B-spline of smallest degree $n = 0$ is almost identical to the nearest-neighbor synthesis function. They differ from one another only at the transition values, where we ask that β^0 be symmetrical with respect to the origin (φ^0 is not), and where we ask that β^0 satisfies the partition of unity. Thus, contrary to the nearest-neighbor case, it happens in some exceptional cases (evaluation at half-integers) that interpolation with β^0 requires the computation of the average between two samples. Otherwise, this function is indistinguishable from nearest-neighbor.

• Degree $n = 1$

The B-spline function of next degree β^1 is exactly identical to the linear case.

• Degrees $n > 1$

No spline β^n of degree $n > 1$ benefits from the property of being interpolating; thus, great care must be exerted never to use one in the context of Equation (1). Equation (4) must be used instead. Unfortunately, some authors have failed to observe this rule, and this violation lead to disastrous experimental results that were absurd. To help settle this issue, we give in Appendix an efficient routine that transforms the sequence of data samples $\{f_k\}$ into coefficients $\{c_k\}$ by the way of in-place recursive filtering.

A cubic B-spline is often used in practice. Its expression is given by

$$\beta^3(x)=\begin{cases}\tfrac{2}{3}-\tfrac{1}{2}|x|^2\,(2-|x|) & 0\le|x|<1\\ \tfrac{1}{6}(2-|x|)^3 & 1\le|x|<2.\\ 0 & 2\le|x|\end{cases}$$

This synthesis function is not interpolant. As explained at Equation (8), it is nonetheless possible to exhibit an infinite-support synthesis function $\varphi_{\mathrm{int}}=\beta^3_{\mathrm{card}}$ that allows one to build exactly the same interpolated function f. To give a concrete illustration of this fact, we show at Figure 9 both the non-interpolating cubic B-spline β^3 along with its interpolating equivalent synthesis function. The latter is named a cubic cardinal spline β^3_{card}. Graphically, the B-spline looks similar to a Gaussian; this is not by chance, since a B-spline can be shown to converge to a Gaussian of infinite variance when its degree increases. Already for a degree as small as $n=3$, the match is amazingly close since the maximal relative error between a cubic B-spline and a Gaussian with identical variance is only about 3.5%. On the right side of Figure 9, the cardinal spline displays decaying oscillations, which is reminiscent of a sinc function. This is not by chance, since a cardinal spline can be shown to converge to a sinc function when its degree increases [40, 41]. We give in Figure 10 the approximation kernel for B-splines of several degrees. Clearly, the higher the degree, the closer to a sinc is the equivalent cardinal spline, and the better is the performance.

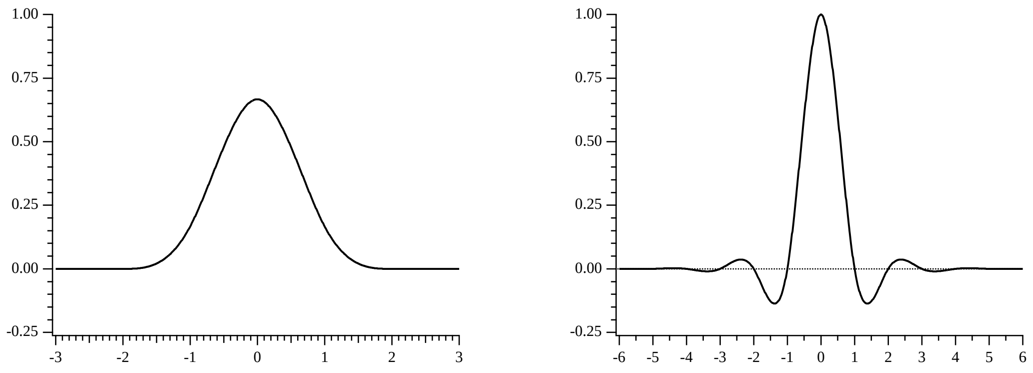


Figure 9: B-spline of third degree. Left: function shape. Right: equivalent interpolant

The B-spline functions enjoy the maximal order of approximation for a given integer support; conversely, they enjoy the minimal support for a given order of approximation. Thus, they belong to the family of functions that enjoy Maximal Order and Minimal Support, or Moms. It can be shown that any of these functions can be expressed as the weighted sum of a B-spline and its derivatives [27].

$$\mathrm{Moms}^n(x)=\beta^n(x)+\sum_{m=1}^{n}c_m\frac{\mathrm{d}^m}{\mathrm{d}x^m}\beta^n(x).$$

B-splines are those Moms functions that are maximally differentiable. We present below two other members of this family. The o-Moms functions are such that their least-squares approximation constant C_φ is minimal, while Schaum's functions are interpolating but have a suboptimal approximation constant that is worse than both o-Moms and B-splines.

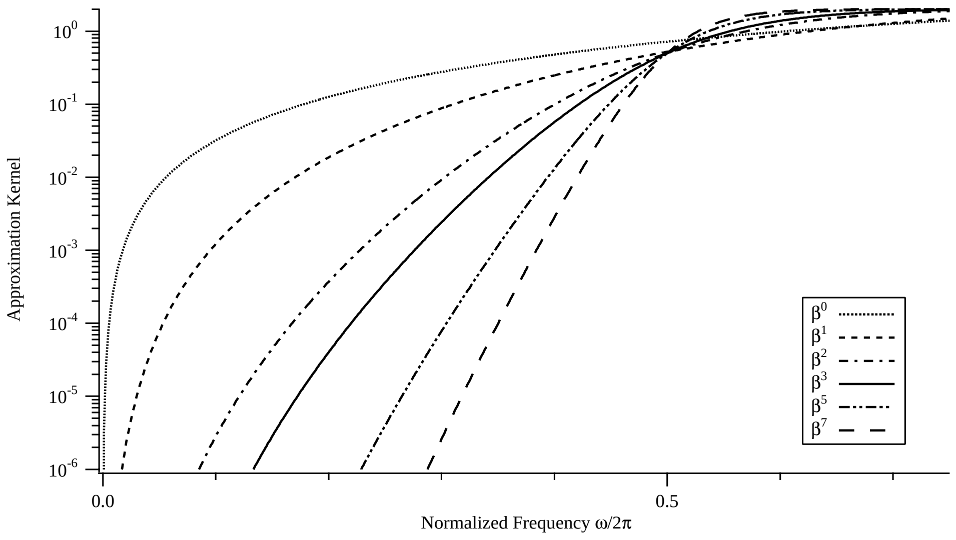


Figure 10: B-spline synthesis function. Approximation kernel for several degrees

8.4. o-Moms

The o-Moms functions are indexed by their polynomial degree n. They are symmetric and their knots are identical to those of the B-spline they descend from. Moreover, they have the same support as β^n, that is, $W = n+1$; this support is the smallest achievable for a synthesis function with approximation order $L = n+1$. Although their order is identical to that of a B-spline of same degree, their approximation error constant C_φ is much smaller. In fact, the o-Moms functions are such that their least-squares constant reaches its smallest possible value. In this sense, they are asymptotically optimal approximators, being the shortest for a given support, with the highest approximation order, and the smallest approximation error constant [27].

These functions are not interpolating; thus, we need a way to compute the sequence of coefficients $\{c_k\}$ required for the implementation of Equation (4). Fortunately, the same routine than for the B-splines can be used (see Appendix).

The o-Moms functions of degree zero and one are identical to β^0 and β^1, respectively. The o-Moms functions of higher degree can be determined recursively in Fourier [27]; we give here the expression for $n = 3$

$$\mathrm{oMoms}^3(x) = \beta^3(x) + \tfrac{1}{42}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\beta^3(x) = \begin{cases} \tfrac{1}{2}\,|x|^3 - |x|^2 + \tfrac{1}{14}\,|x| + \tfrac{13}{21} & 0 \le |x| < 1 \\ \tfrac{-1}{6}\,|x|^3 + |x|^2 - \tfrac{85}{42}\,|x| + \tfrac{29}{21} & 1 \le |x| < 2. \\ 0 & 2 \le |x| \end{cases}$$

As a curiosity, we point out in Figures 11 and 12 that this synthesis function has a slope discontinuity at the origin; thus, its regularity is C^0 (in addition, it has other slope discontinuities for $|x| = 1$ and $|x| = 2$.) It is nevertheless optimal in the sense described.

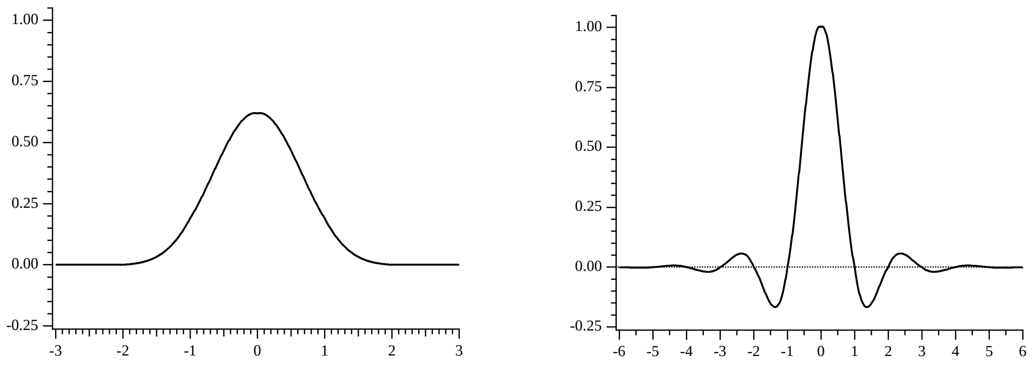


Figure 11: o-Moms of third degree. Left: function shape. Right: equivalent interpolant

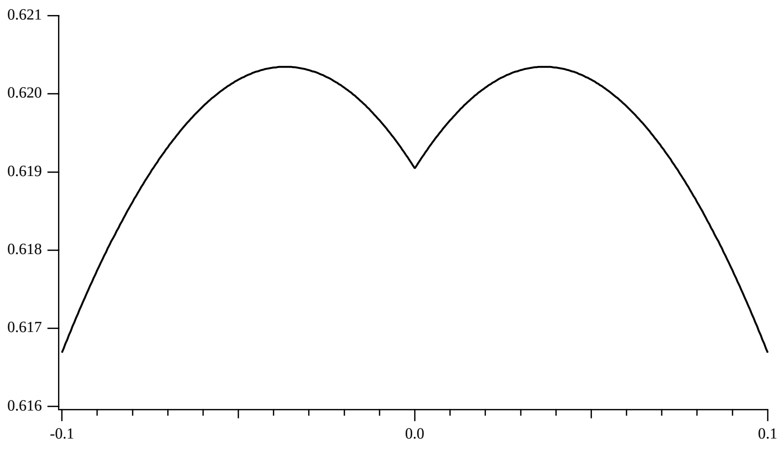


Figure 12: o-Moms of third degree (central part)

8.5. Schaum's functions

Like the o-Moms, the pseudo-Lagrangian kernels proposed by Schaum in [28] can also be represented as a weighted sum of B-splines and their even-order derivatives. They have same order and same support as B-splines and o-Moms. Their main interest is that they are interpolant. Their main drawback with respect to both o-Moms and B-splines is a worse approximation constant C_φ: for the same approximation order $L = 4$, the minimal value is reached by o-Moms with $C_\varphi = 0.000627$; the constant for the cubic spline is more than twice with $C_\varphi = 0.00166$, while the cubic Schaum loses an additional order of magnitude with $C_\varphi = 0.01685$. They have no regularity (are discontinuous) for even degrees, and are C^0 for odd degrees. Figure 6 shows the quadratic member of that family.

8.6. Keys' function

The principal reason for the popularity enjoyed by the family of Keys' functions [24] is the fact that they perform better than linear interpolation, while being interpolating. Thus, they do not require the determination of interpolation coefficients, and the classical Equation (1) can be used. These functions are made of piecewise cubic polynomials and depend on a parameter a. Their general expression is

$$\mathrm{u}_a(x) = \begin{cases} (a+2)\,|x|^3 - (a+3)\,|x|^2 + 1 & 0 \le |x| < 1 \\ a\,|x|^3 - 5a\,|x|^2 + 8a\,|x| - 4a & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}.$$

Comparing this expression to that of the cubic spline, it is apparent that both require the computation of piecewise polynomials of the same support. However, their approximation order differ: the best order that can be reached by a Keys' function is three, for the special value $a = \frac{-1}{2}$, while the cubic spline has order four. This extra order for β^3 comes at the cost of the computation of a sequence $\{c_k\}$, for use in Equation (4). However, by using a recursive filtering approach, this cost can be made negligible. Altogether, the gain in speed offered by Keys' function is not enough to counterbalance the loss in quality when comparing β^3 and $u_{\frac{-1}{2}}$. Moreover, the regularity of Keys is C^1, which is one less than that of the cubic spline.

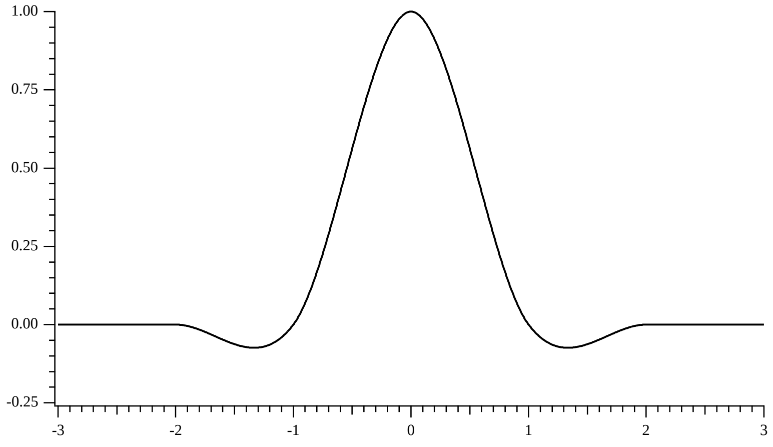


Figure 13: Keys' interpolation with $a = \frac{-1}{2}$

8.7. Sinc

For a long time, sinc interpolation—which corresponds to ideal filtering—has been the Graal of geometric operations. Nowadays, researchers acknowledge that, while sinc interpolation can be realized under special circumstances (e.g., translation of a periodic signal by discrete Fourier transform operations), in general it can only be approximated, thus reintroducing a certain amount of aliasing and blurring, depending on the quality of the approximation. Another drawback of the sinc function is that it decays only slowly, which generates a lot of ringing. In addition, there is no way to tune the performance to any specific application: it is either a sinc (or approximation thereof), or it is something else.

The sinc function provides error-free interpolation of the band-limited functions. There are two difficulties associated with this statement. The first one is that the class of band-limited functions represents but a tiny fraction of all possible functions; moreover, they often give a distorted view of the physical reality in an imaging context—think of the transition air/matter in a CT scan: as far as classical physics is concerned, this transition is abrupt and cannot be expressed as a band-limited function. Further, there exist obviously no way at all to perform any kind of anti-aliasing filter on physical matter (before sampling). Most patients would certainly object to any attempt of the sort.

The second difficulty is that the support of the sinc function is infinite. An infinite support is not too bothering, even in the context of Equation (8), provided an efficient algorithm can be found to implement interpolation with another equivalent synthesis function that has a finite support. This is exactly the trick we used with B-splines and o-Moms. Unfortunately, no function can be at the same time band-limited and finite-support, which precludes any hope to find a finite-support synthesis function φ for use in Equation (8). Thus, the classical solution is simply to truncate sinc itself by multiplying it with a finite-support window; this process is named apodization. A large catalog of apodizing windows is available in [29], along with their detailed analysis.

By construction, all these apodized functions are interpolants. While the regularity of the non-truncated sinc function is infinite, in general the regularity of its truncated version depends on the apodization window. In particular, regularity is maximized by letting the edges of the window support coincide with a pair of zero-crossings of the sinc function. This results in reduced blocking artifacts. In theory, any apodization window is admissible; including, say, a window w_u such that $w_u(x)\operatorname{sinc}(x) = u_{\frac{-1}{2}}(x)$, where $u_{\frac{-1}{2}}(x)$ is the Keys' function. In practice, the only windows that are considered have all a broadly Gaussian appearance and are often built with trigonometric polynomials. We investigate two of them below.

8.8. Dirichlet Apodization

Dirichlet apodization is perhaps the laziest approach, since the window of total width W is simply an enlarged version of β^0, which requires no more computational effort than a test to indicate support membership. The apodized synthesis function is given by

$$\operatorname{sinc}_W^D(x) = \frac{\sin(\pi x)}{\pi x} \beta^0(\frac{x}{W}),$$

where W is an even integer. The price to pay for laziness is bad quality. First, the regularity of this function is low since it is not differentiable. More important, its approximation order is as bad as $L = 0$, and this function does not even satisfy the partition of unity. This means that a reduction of the sampling step does not necessarily result in a reduction of the interpolation error. Instead of Equation (11), we have that

$$\|f - f_h\|_{L_2} \to C_\varphi \|f^{(L)}\|_{L_2} \text{ as } h \to 0.$$

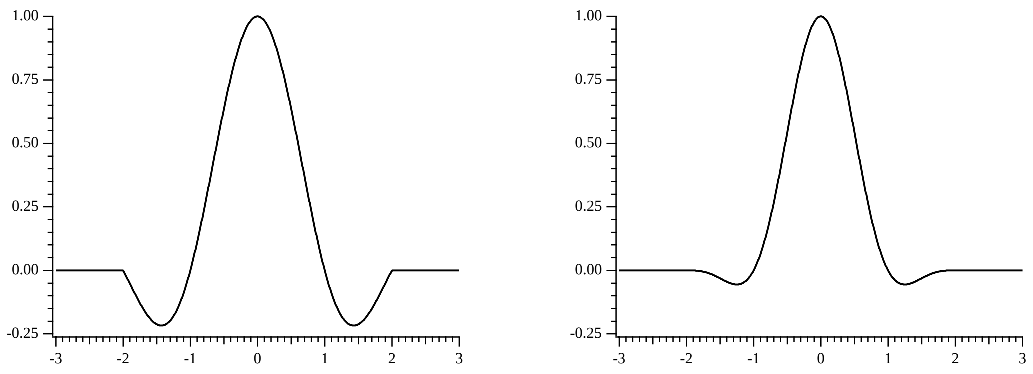


Figure 14: Sinc apodization with $W = 4$. *Left: Dirichlet. Right: Hanning*

8.9. Hanning Apodization

Apodization, being defined as the multiplication of a sinc function by some window, corresponds in Fourier to a convolution-based construction. The Hanning window is one out of several attempts to design a window that has favorable properties in Fourier. The result is

$$\mathrm{sinc}_W^H(x) = \mathrm{sinc}_W^D(x)\left(\tfrac{1}{2} + \tfrac{1}{2}\cos(\frac{2\pi x}{W})\right).$$

With $L = 0$, the order of approximation of Hanning interpolation is no better than that of Dirichlet interpolation; the constant C_φ is significantly improved, though. Whereas it was $C_\varphi = 0.1076$ for sinc_4^D, it is now $C_\varphi = 0.0153$ for sinc_4^H. Being continuously differentiable, Hanning is also more regular than Dirichlet.

IX. COST-PERFORMANCE ANALYSIS

As seen in Figure 5, the single most influential parameter that dictates the computational cost is the size W of the support of the synthesis function φ. Second to it, we find the cost of evaluating $\varphi(x-k)$ for a series of arguments $(x-k)$. Lastly, there is a small additional cost associated to the computation of interpolation coefficients $c_{\mathbf{k}}$ in the context of Equation (4). We want to mention here that the importance of this overhead is negligible, especially in the practical case where it needs be computed once only before several interpolation operations are performed. This situation arises often in the context of iterative algorithms, and in the context of interactive imaging; moreover, it disappears altogether when the images are stored directly as a set of coefficients $\{c_{\mathbf{k}}\}$ rather than a set of samples $\{f_{\mathbf{k}}\}$. Thus, we shall ignore this overhead in the theoretical performance analysis that follows.

9.1. Cost

Let us assume that we want to compute the interpolated value $f(\mathbf{x})$ of an image f at argument $\mathbf{x}$, using a separable synthesis functions of finite-support W. For each output value, we first need to perform W multiplications and W additions to compute $c_{k_2} = \sum c_{k_1,k_2}\,\varphi(x_1 - k_1)$, with an inner loop over k_1. This computation is embedded in a similar outer loop over k_2 that is executed W times and that involves the computation of $f(\mathbf{x}) = \sum c_{k_2}\,\varphi(x_2 - k_2)$. Finally, we need W^2 mult-and-adds in 2D; more generally, we need $2\,W^q$ operations in q dimensions, where we consider that a multiplication is worth an addition.

To this cost, one must add (qW)-times the cost of the evaluation of the separable synthesis function. When the latter is piecewise polynomial, on average we need $W/4$ tests to determine which of the polynomial piece to use. Once a polynomial is selected, evaluation by the Horner's scheme further requires $(W-1)$ mult-and-adds. Putting all that together, the magnitude of the global cost of all operations for a piecewise polynomial synthesis function is $O(W^q)$, more precisely $2\,W^q + q\,W\left(\tfrac{9}{4}W - 2\right)$.

In the case of the sinc family, the synthesis function is no polynomial. Then, each evaluation requires the computation of a transcendental function and the multiplication by the apodization window. This cost does not depend on the support W; hence, the magnitude of the global cost of all operations for an apodized sinc synthesis function is also $O(W^q)$, more precisely $2\,W^q + \lambda\,q\,W$, where $\lambda = 12$ operations are spent in the evaluation of a Hanning apodization window (we consider that the transcendental functions sine or cosine are worth two multiplications each), $\lambda = 9$ for a Bartlet window and $\lambda = 6$ in the Dirichlet case.

It follows from these theoretical considerations that the support for which a sinc-based synthesis function (e.g., Hanning) comes at a lesser computational cost than a polynomial-based one, is about $W = 6$ in two dimensions. For images or volumes, where $q > 1$, it is important to realize that this result does not imply that the use of sinc functions is more efficient than that of polynomials, because sinc typically requires a larger support than polynomials to reach the same quality. Since the global cost is $O(W^q)$, and since $q > 1$, any increase in W dominates over the other terms.

9.2. Performance

We present at Figure 15 a comparison of the error kernel for several synthesis functions of same support $W = 4$. It includes cubic B-spline, cubic o-Moms, and cubic Schaum as examples of polynomial functions, and Dirichlet and Hanning as examples of apodized sinc.

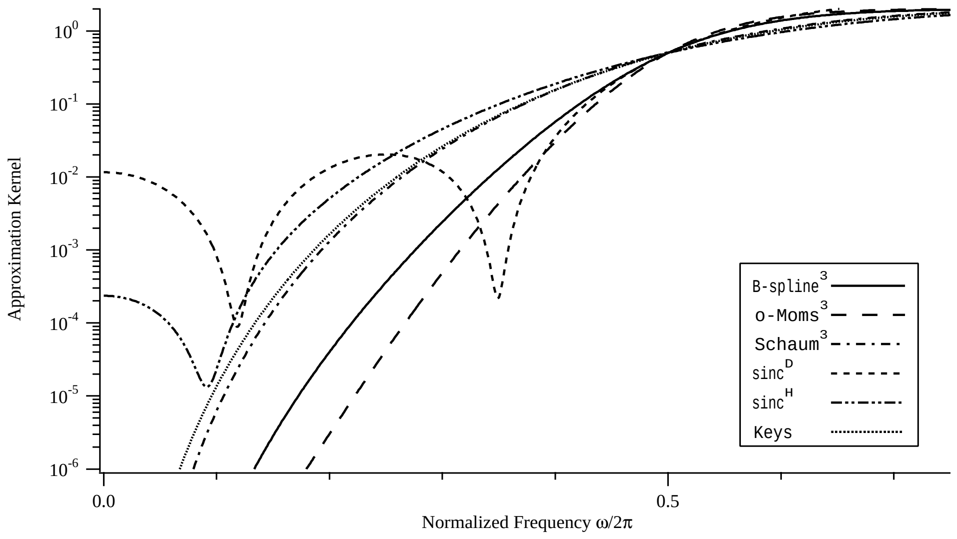


Figure 15: Comparison of synthesis functions of same support $W = 4$

We observe that the sinc-based synthesis functions do not reproduce the constant. Since most of the energy of virtually any image is concentrated at low frequencies, it is easy to predict that these functions will perform poorly when compared to polynomial-based synthesis functions. We shall see in the experimental section that this prediction is fulfilled; for now, we limit our analysis to that of the more promising polynomial cases.

On the grounds of Equation (9), we can select a specific function f to sample-and-interpolate, and we can predict the amount of resulting squared interpolation error. As a convenient simplification, we now assume that this function f has a constant-value power spectrum; in this case, it is trivial to obtain the interpolation error by integrating the curves in Figure 15. Table 1 gives the resulting values as a signal-to-noise ratio expressed in dB, where the integration has been performed numerically over the domain $\omega \in [-\pi, \pi]$. These results have been obtained by giving the same democratic weight to all frequencies up to Nyquist's rate; if low frequencies are considered more important than high frequencies, then the order of approximation L and its associated constant C_φ are the most representative quality indexes.

9.3. Trade-off

Figure 16 presents the combination of the theoretical results of computation time and of those of interpolation quality. In order to show the versatility of the approximation kernel, we have changed the function f from white noise (Table 1) to a band-limited function that satisfies a first-order Markov model [42, p.34] parametrized by $\rho = 0.9$. This model is representative of a large variety of real images.

Synthesis function	ε^2 (dB)	L
β^7	16.10	8
β^6	15.54	7
β^5	14.88	6
β^4	14.14	5
oMoms^3	14.03	4
β^3	13.14	4
u_{-1}	12.33	1
β^2	12.11	3
$u_{-\frac{1}{2}}$	11.02	3
cubic Schaum	10.98	4
$u_{-\frac{1}{4}}$	10.14	1
Dodgson	9.98	1
β^1	9.23	2
φ^0	5.94	1

Table 1: Performance index for white noise

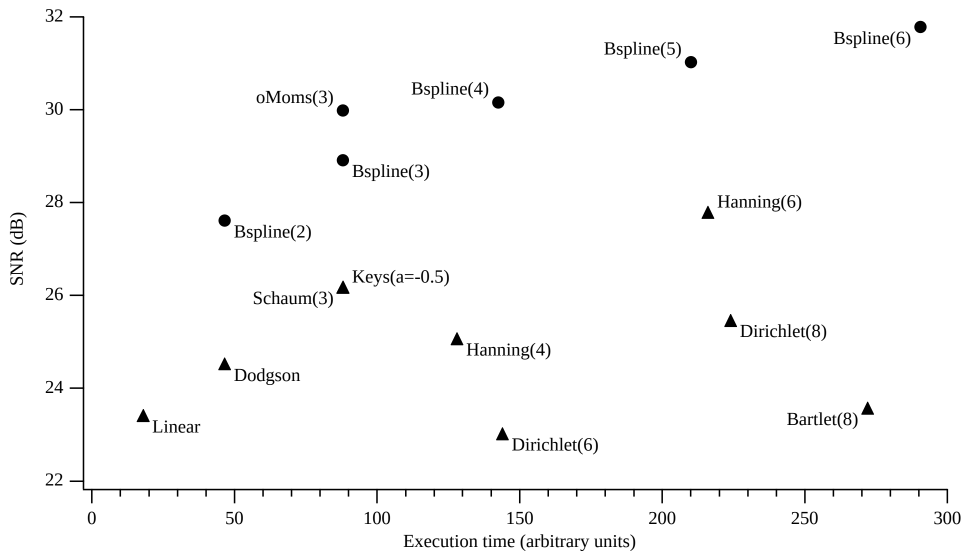


Figure 16: Theoretical performance index for a first-order Markov model with $\rho = 0.9$. Triangles: interpolating functions. Circles: non-interpolating functions.

X. Experiments

To magnify the degradation that results from interpolation, we adopt the following strategy that has for goal the highlighting of—as opposed to the avoidance of—the deleterious effects of interpolation: we apply a succession of $r = 15$ rotations of $\frac{2\pi}{15} = 24$ each to some image, such that the output of any given step is used as input for the next step; then, we compare the initial image with the final output. To limit potential boundary effects, we perform the final comparison on the central square of the image only. Also, we avoid any interaction between interpolation and quantization by performing all computations in a floating-point format. Figure 17 shows the image we want to rotate. It is made of a radial sinusoidal chirp with a spatial frequency that decreases from center to edge.

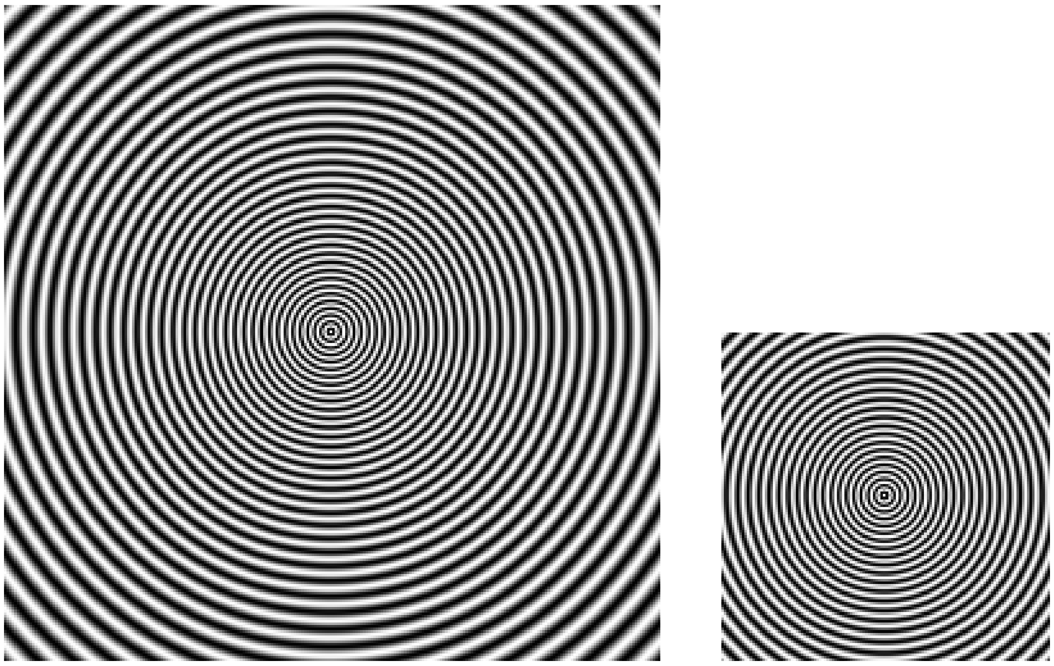


Figure 17: Synthetic image. Left: whole. Right: central square

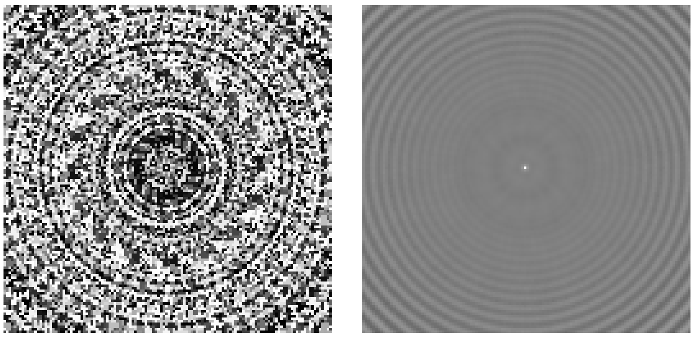


Figure 18: Some visual results. Left: nearest-neighbor interpolation. Right: linear interpolation

10.1. Nearest-Neighbor and Linear Interpolation

Figure 18 shows the result of this experiment when using the two most commonly found interpolants: nearest-neighbor and linear. Clearly, nearest-neighbor interpolation results in a lot of data shuffling; as a curiosity, we mention that perfect unshuffling is sometimes possible under special circumstances, such that reversing the order of operations restores the initial data without error [43]. No other interpolation methods proposed here is reversible. For example, linear interpolation results in strong attenuation of high frequencies, as can be inferred from the visual comparison of Figure 17 with Figure 18. This loss cannot be compensated. It corresponds to the prediction made in Figure 10, according to which linear interpolation, or β^1, performs poorly when compared to other cases.

10.2. Cubic Interpolation

Figure 19 proposes three synthesis functions of identical support which have essentially the same computational cost. On the left, despite the use of the optimal parameter $a = \frac{-1}{2}$, Keys offers the poorest visual performance since the central part of the figure is blurred. In addition, close inspection (particularly on a monitor screen) discloses blocking artifacts that betray themselves as moiré patterns. Those are absent with cubic spline and cubic o-Moms interpolation, although patterns unrelated to interpolation may eventually be present on paper, in reason of the dithering process inherent in printing these figures. More important, cubic spline interpolation results in less blurring, and cubic o-Moms in even less.

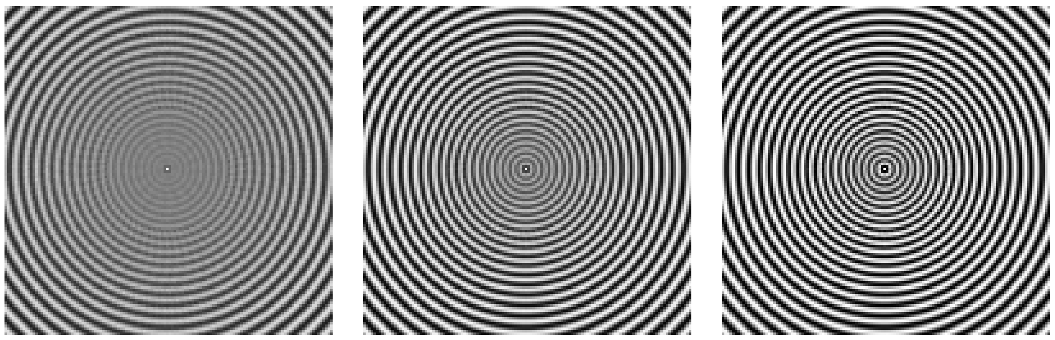


Figure 19: Some visual results. Left: Keys. Center: cubic spline. Right: cubic o-Moms

10.3. Sinc-Based Interpolation

Figure 20 shows the result of using two different truncated and apodized approximations of sinc, where the support is the same as in the functions of Figure 19. The test image of Figure 17 has been built with a non-null average; since an apodized version of sinc does not reproduce this constant value faithfully, each incremental rotation results in a small drift of the average value of the image. This would be true of any interpolation function that would not satisfy the partition of unity. This drift manifests itself as images that appear too dark or too light. We conclude that sinc performs poorly when compared to other synthesis functions of the same support, and not only drift of the mean value, but also both blocking and excessive blurring artifacts are present.

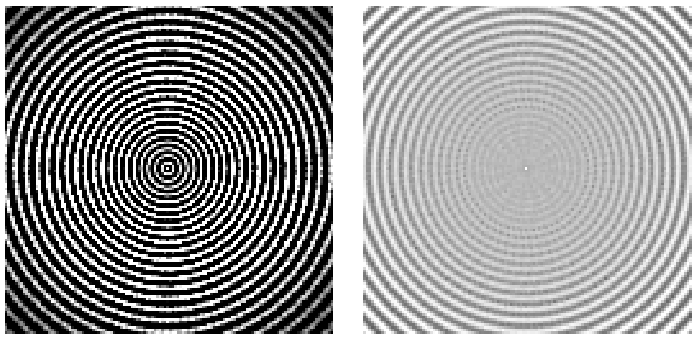


Figure 20: Some visual results. Left: Dirichlet($W = 4$). Right: Hanning ($W = 4$)

10.4. Discussion

Table 2 presents in succinct form the numeric results of these experiments, along with some additional ones. In particular, we also provide the results for the standard Lena test image. The execution time is given in seconds; it corresponds to the duration of a single rotation of a square image 512 pixels on a side. The computer is a Power Macintosh 9600/350. The column ε^2 shows the signal-to-noise ratio between the central part of the initial image of Figure 17 and the result of each experiment, wile the column "Lena" gives qualitatively similar results for this standard test image. The measure of signal-to-noise ratio is defined as

$$\mathrm{SNR} = 10\log\Big(\sum_{\mathbf{k}\in\mathbf{Z}^2} f_{\mathbf{k}}^2 \Big/ \sum_{\mathbf{k}\in\mathbf{Z}^2} \left(f_{\mathbf{k}} - g_{\mathbf{k}}\right)^2\Big),$$

where f is the original data and where g is given by the r-times chaining of the rotation.

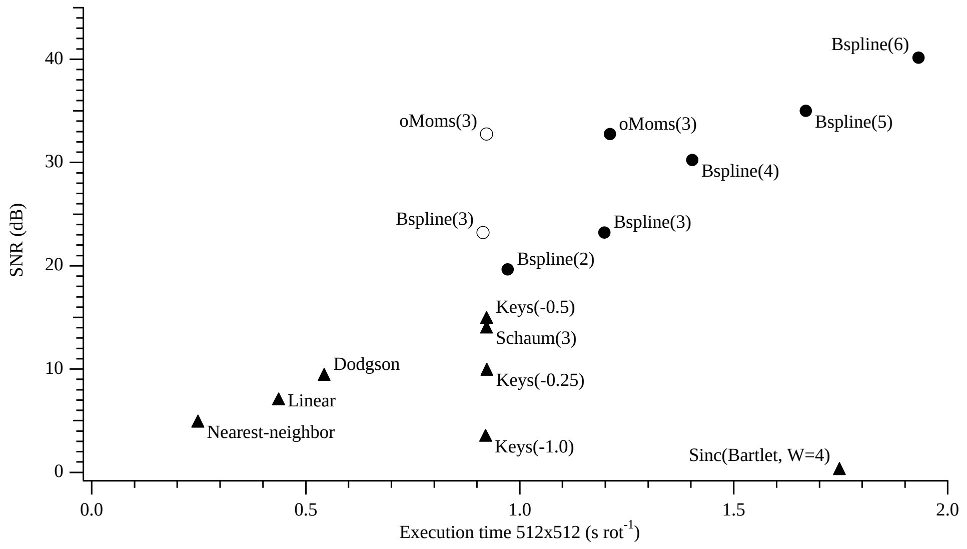


Figure 21: Summary of the main experimental results for the circular pattern. Triangles: interpolating functions. Circles: non-interpolating functions. Hollow circles: accelerated implementation.

These results point out some of the difficulties associated to the analysis of the performance of a synthesis function φ. For example, the computation time should ideally depend on the number of mathematical operations only. In reality, the optimization effort put into implementing each variation with one synthesis function or another, has also some influence. For instance, our faster implementation of the cubic spline and the cubic o-Moms runs in shorter time than reported in Table 2 (namely, 0.91 seconds instead of 1.19). We have nevertheless shown the result of the slower implementation because it corresponds to a somewhat unified level of optimization in all considered cases.

Figure 21 proposes a graphic summary of the most interesting results (circular pattern, quality better than 0 dB and execution time shorter than 2 seconds). It is interesting to compare this figure to Figure 16; the similarity between them confirms that our theoretical ranking of synthesis functions was justified. The difference between the interpolation methods is more pronounced in the experimental case because it has been magnified by the number of rotations performed.

Synthesis Function	Time (s)	Circular (dB)	Lena (dB)
φ^0	0.24	4.96	18.57
β^1	0.43	7.13	21.98
Dodgson	0.54	9.49	24.23
u_{-1}	0.92	3.60	19.33
$u_{-\frac{1}{2}}$	0.92	15.00	28.16
$u_{-\frac{1}{4}}$	0.92	10.06	24.73
cubic Schaum	0.92	14.09	27.61
β^2	0.97	19.65	30.56
β^3	1.19	23.22	31.98
oMoms^3	1.21	32.76	34.29
β^4	1.40	30.25	33.90
sinc Dirichlet $W = 4$	1.71	-0.54	0.34
β^5	1.66	35.01	34.81
sinc Bartlet $W = 4$	1.74	0.38	0.41
β^6	1.93	40.17	35.54
sinc Hamming $W = 4$	2.10	12.58	17.66
sinc Hanning $W = 4$	2.10	7.13	6.76
sinc Dirichlet $W = 6$	2.10	-13.62	-15.24
β^7	2.23	44.69	36.05
sinc Bartlet $W = 6$	2.53	1.03	1.08
sinc Hamming $W = 6$	3.08	22.11	24.06
sinc Hanning $W = 6$	3.08	18.59	19.32

Table 2: Experimental results in numerical form

XI. Conclusion

We have presented two important methods for the exact interpolation of data given by regular samples: in classical interpolation, the synthesis functions must be interpolants, while non-interpolating synthesis functions are allowed in generalized interpolation. We have tried to dispel the too commonly hold belief according to which non-interpolating functions (typically, cubic B-splines) should be avoided. This misconception, present in many a book or report on interpolation, arises because practitioners failed to recognize the difference between classical and generalized interpolation, and attempted to use in the former setting synthesis functions that are fit to the latter only. We have provided a unified framework for the theoretical analysis of the performance of both interpolating and non-interpolating methods. We have applied this analysis to specific cases that involve piecewise polynomial functions as well as sinc-based interpolants. We have performed 2D experiments that support the 1D theory.

We conclude from both theoretical and practical concerns that the most important index of quality is the approximation order of the synthesis function, its support being the most important parameter with respect to efficiency. Thus, the class of functions with Maximal Order and Minimal Support, or Moms, stands apart as the best achievable compromise between quality and speed. We have observed that many formerly-proposed synthesis functions, such as Dodgson, Keys, and any of the apodized versions of a sinc, do not belong to this class. Experiments have confirmed that these synthesis functions do indeed perform poorly. In particular, no sinc-based interpolation results in an acceptable quality with regard to its computational demand. In addition, this family of synthesis functions is difficult to handle analytically, which leads to unnecessary complications for simple operations such as differentiation or integration.

The more favorable class of Moms functions can be further divided into subclasses, the most relevant being B-splines, Schaum and o-Moms. Of those three, the Schaum's functions are the only representatives that are interpolating. Nevertheless, experiments have shown that this strong constraint is detrimental to the performance; we observe that the time spent in computing the interpolation coefficients required by B-splines and o-Moms is a small, almost negligible investment that offers a high pay-off in terms of quality. For this reason, we discourage the use of Schaum and promote generalized interpolation instead, with non-interpolating synthesis functions such as B-splines and o-Moms. For a better impact, we include in the Appendix an efficient routine for the computation of the required interpolation coefficients.

Finally, comparing B-splines to o-Moms, we conclude that the lack of continuity of the latter makes them less suitable than B-splines for imaging problems that require the computation of derivatives, for example to perform operations such as edge detection or image-based optimization of some sort (e.g., snake contouring, registration). These operations are very common in medical imaging. Thus, despite a poorer objective result than o-Moms, B-splines are very good candidates for the construction of an image model. Moreover, they enjoy additional properties, such as easy analytical manipulation, several recursion relations, the m-scale relation (of great importance for wavelets, a domain that has strong links with interpolation [44, 45]), minimal curvature for cubic B-splines, easy extension to inexact interpolation (smoothing splines, least-squares [6]), simplicity of their parametrization (a single number—their degree—is enough to describe them), and possible generalization to irregular sampling, to cite a few.

A property that has high relevance in the context of interpolation is the convergence of the cardinal spline to the non-truncated sinc for high degrees [40, 41]. Throughout the paper, we have given numerous reasons why it is more profitable to approximate a true sinc by the use of non-interpolating synthesis functions rather than by apodization. Even for moderate degrees, the spline-based approximations offers a sensible quality improvement over apodization for a given computational budget.

None of the other synthesis functions, such as Schaum, Dodgson, Keys or sinc-based, offers enough gain in quality to be considered. We note however that the study presented in Table 2 and Figure 21 relies on two images only; moreover, these images are not truly representative of your genuine biomedical data. In addition, the comparison criterion is mean-square, which is precisely the form for which o-Moms are optimal. Perhaps other conclusions would be obtained by the use of more varied images or a different criterion, for example a psycho-visual one.

APPENDIX

A.1. Fourier Transform and Fourier Series

By convention of notation, $\hat{f}(\omega)$ is the continuous Fourier transform of $f(x)$ and is defined by

$$\hat{f}(\omega)=\int_{-\infty}^{\infty} f(x)\,e^{-j\omega x}\,\mathrm{d}x \quad \text{and} \quad f(x)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{f}(\omega)\,e^{j\omega x}\,\mathrm{d}\omega .$$

The Fourier-series decomposition of a 1-periodic function $s(x)=s(x+1)$ is

$$s(x)=\sum_{k\in\boldsymbol{Z}} S_k\,e^{j2\pi k x}, \quad \text{where} \quad S_k=\int_{\frac{-1}{2}}^{\frac{1}{2}} s(x)e^{-j2\pi k x}\,\mathrm{d}x .$$

A.2. Partition of Unity

Let $\varphi(x)$ be a continuous function and let us define

$$s(x)=\sum_{k\in\boldsymbol{Z}}\varphi(x-k) \quad \forall x\in[0,1].$$

Clearly, the resulting function s is periodic and satisfies $s(x)=s(x+1)$. Thus, under wide conditions, its Fourier series can be expressed as

$$s(x)=\sum_{n\in\boldsymbol{Z}} S_n\,e^{j2\pi n x}$$

with

$$S_n=\int_{\frac{-1}{2}}^{\frac{1}{2}}\sum_{k\in\boldsymbol{Z}}\varphi(x-k)e^{-j2\pi n x}\,\mathrm{d}x=\int_{-\infty}^{\infty}\varphi(x)e^{-j2\pi n x}\,\mathrm{d}x=\hat{\varphi}(2\pi n).$$

By substitution into the Fourier series, we have that

$$s(x)=\sum_{n\in\boldsymbol{Z}}\hat{\varphi}(2\pi n)e^{j2\pi n x},$$

from which it is possible to deduce the equivalence between $s(x)=1$ and $\hat{\varphi}(2\pi n)=\delta_n$. Note that a rigorous demonstration of this equivalence requires that $s(x)$ converges uniformly.

A.3. Recursive Filtering

The routine below performs the in-place determination of a 1D sequence of interpolation coefficients $\{c_k\}$ from a sequence of data samples $\{f_k\}$. The returned coefficients $\{c_k\}$ satisfy

$$f_k=\sum_{k\in\boldsymbol{Z}} c_k\,\varphi(x-k) \qquad \forall k\in\boldsymbol{Z},$$

where the synthesis function φ is represented by its poles. The values of these poles for B-splines of degree $n\in\{2,3,4,5\}$ and for cubic o-Moms are available in Table 3. (The B-spline poles of any degree $n>1$ can be shown to be real and lead to a stable implementation.) This routine implicitly assumes that the finite sequence of physical samples, known on the support $\boldsymbol{X}=[0,N-1]$, is extended to infinity by the following set of boundary conventions:

$$\begin{cases} f(x) = f(-x) & \forall x \in \boldsymbol{R} \\ f(N-1+x) = f(-x+N-1) & \forall x \in \boldsymbol{R} \end{cases}.$$

This scheme is called mirror (or symmetric) boundaries. The first relation expresses that the extended signal is symmetric with respect to the origin, and the second mirrors the extended signal around $x = N-1$. Together, these relations associate to any abscissa $y \in \boldsymbol{R} \setminus \boldsymbol{X}$ some specific value $f(y) = f(x)$ with $x \in \boldsymbol{X}$. The resulting extended signal is $(2N-1)$-periodic, and has no additional discontinuities with respect to those that might already exist in the finite-support version of f. This is generally to be preferred to both zero-padding and simple periodization of the signal, because these latter introduce discontinuities at the signal extremities.

	z_1	z_2
β^2	$\sqrt{8}-3$	N / A
β^3	$\sqrt{3}-2$	N / A
β^4	$\sqrt{664-\sqrt{438976}}+\sqrt{304}-19$	$\sqrt{664+\sqrt{438976}}-\sqrt{304}-19$
β^5	$\frac{1}{2}\left(\sqrt{270-\sqrt{70980}}+\sqrt{105}-13\right)$	$\frac{1}{2}\left(\sqrt{270+\sqrt{70980}}-\sqrt{105}-13\right)$
$\mathrm{oMoms}^3(x)$	$\frac{1}{8}\left(\sqrt{105}-13\right)$	N / A

Table 3: Value of the B-spline poles required for the recursive filtering routine

The routine below is in a reality a digital filter. The z-transform of its function transfer is given by

$$B^{-1}(z) = \prod_{i=1}^{\lfloor n/2 \rfloor} \frac{z(1-z_i)(1-z_i^{-1})}{(z-z_i)(z-z_i^{-1})}.$$

As is apparent from this expression, the poles of this filter are power-conjugate, which implies that every second pole is outside the unit circle. To regain stability, this meromorphic filter is realized as a cascade of causal filters that are implemented as a forward recursion, and anti-causal filters that are implemented as a backward recursion. More details are available in [19, 20].

```
#include    <math.h>

/*--------------------------------------------------------------------------*/
void    GetInterpolationCoefficients
        (
            double  c[],        /* input samples --> output coefficients */
            int     DataLength, /* number of samples or coefficients */
            double  z[],        /* poles */
            int     NbPoles,    /* number of poles */
            double  Tolerance   /* admissible relative error */
        )

{
    double  Lambda = 1.0;
    int     n, k;

    /* special case required by mirror boundaries */
    if (DataLength == 1)
        return;
    /* compute the overall gain */
    for (k = 0; k < NbPoles; k = k + 1)
        Lambda = Lambda * (1.0 - z[k]) * (1.0 - 1.0 / z[k]);
    /* apply the gain */
    for (n = 0; n < DataLength; n = n + 1)
        c[n] = c[n] * Lambda;
    /* loop over all poles */
    for (k = 0; k < NbPoles; k = k + 1) {
            /* causal initialization */
            c[0] = InitialCausalCoefficient(c, DataLength, z[k], Tolerance);
            /* causal recursion */
            for (n = 1; n < DataLength; n = n + 1)
                c[n] = c[n] + z[k] * c[n-1];
            /* anticausal initialization */
            c[DataLength-1] = InitialAntiCausalCoefficient(c, DataLength, z[k]);
            /* anticausal recursion */
            for (n = DataLength-2; n >= 0; n = n - 1)
                c[n] = z[k] * (c[n+1] - c[n]);
        }
    }

    /*--------------------------------------------------------------------------*/
    double  InitialCausalCoefficient
            (
                double  c[],        /* coefficients */
                int     DataLength, /* number of coefficients */
                double  z,          /* actual pole */
                double  Tolerance   /* admissible relative error */
            )

    {
        double  Sum, zn, z2n, iz;
        int     n, Horizon;
        int     TruncatedSum;

        /* this initialization corresponds to mirror boundaries */
        TruncatedSum = 0;
        if (Tolerance > 0.0) {
            Horizon = (int)ceil(log(Tolerance) / log(fabs(z)));
            TruncatedSum = (Horizon < DataLength);
        }
        if (TruncatedSum) {
            /* accelerated loop */
            zn = z;
            Sum = c[0];
            for (n = 1; n < Horizon; n = n + 1) {
                Sum = Sum + zn * c[n];
                zn = zn * z;
            }
            return(Sum);
        }
        else {
            /* full loop */
            zn = z;
            iz = 1.0 / z;
            z2n = pow(z, DataLength-1);
            Sum = c[0] + z2n * c[DataLength-1];
            z2n = z2n * z2n * iz;
            for (n = 1; n <= DataLength - 2; n = n + 1) {
                Sum = Sum + (zn + z2n) * c[n];
                zn = zn * z;
                z2n = z2n * iz;
            }
            return(Sum / (1.0 - zn * zn));
        }
    }

    /*--------------------------------------------------------------------------*/
    double  InitialAntiCausalCoefficient (
                double  c[],        /* coefficients */
                int     DataLength, /* number of samples or coefficients */
                double  z           /* actual pole */
            )

    {
        /* this initialization corresponds to mirror boundaries */
        return((z / (z * z - 1.0)) * (z * c[DataLength-2] + c[DataLength-1]));
    }
```

Interpolation coefficients. (Hopefully) didactic and efficient C-code. This code is available for download at the following URL: "`http://bigwww.epfl.ch/`*".*

A.4. Some Synthesis Functions

Name	Expression	W	L	Regularity	Interpol.	Fourier Transform
Nearest-Neighbor	$1 \quad x \in [\frac{-1}{2}, \frac{1}{2}[$	1	1	None	yes	$\mathrm{sinc}(\omega/2\pi)$
Linear	$\beta^1(x) = 1 - \lvert x \rvert \quad x \in]-1,1[$	2	2	C^0	yes	$(\mathrm{sinc}(\omega/2\pi))^2$
Dodgson	$2\beta^2(x) - \frac{1}{2}(\beta^1(x-\frac{1}{2}) + \beta^1(x+\frac{1}{2}))$	3	2	C^0	yes	$(2\,\mathrm{sinc}(\omega/2\pi) - \cos(\omega/2))(\mathrm{sinc}(\omega/2\pi))^2$
Keys ($a = \frac{-1}{2}$)	$3\beta^3(x) - (\beta^2(x-\frac{1}{2}) + \beta^2(x+\frac{1}{2}))$	4	3	C^1	yes	$(3\mathrm{sinc}(\omega/2\pi) - 2\cos(\omega/2))(\mathrm{sinc}(\omega/2\pi))^3$
Cubic Schaum	$\beta^3(x) - \frac{1}{6}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\beta^3(x)$	4	4	C^0	yes	$(1 + \frac{1}{6}\omega^2)(\mathrm{sinc}(\omega/2\pi))^4$
Cubic B-spline	$\beta^3(x) = \begin{cases} \frac{2}{3} - \frac{1}{2}\lvert x \rvert^2 (2 - \lvert x \rvert) & 0 \le \lvert x \rvert < 1 \\ \frac{1}{6}(2 - \lvert x \rvert)^3 & 1 \le \lvert x \rvert < 2 \end{cases}$	4	4	C^2	**no**	$(\mathrm{sinc}(\omega/2\pi))^4$
Cubic o-Moms	$\beta^3(x) + \frac{1}{42}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\beta^3(x)$	4	4	C^0	**no**	$(1 - \frac{1}{42}\omega^2)(\mathrm{sinc}(\omega/2\pi))^4$
B-spline ($n > 1$)	$\int_{x-\frac{1}{2}}^{x+\frac{1}{2}} \beta^{n-1}(t)\,\mathrm{d}t$	$n+1$	$n+1$	C^{n-1}	**no**	$(\mathrm{sinc}(\omega/2\pi))^{n+1}$
Sinc	$\sin(\pi x)/(\pi x)$	∞	∞	C^∞	yes	$1 \quad \omega \in [-\pi, \pi[$
Dirichlet	$\beta^0(\frac{x}{W})\sin(\pi x)/(\pi x)$	W	0	C^0	yes	$W\int_{-\pi}^{\pi} \mathrm{sinc}(W(\omega - \vartheta)/2\pi)\,\mathrm{d}\vartheta$

BIBLIOGRAPHY

[1] D.F. Watson, *Contouring: A Guide to the Analysis and Display of Spatial Data*. New York: Pergamon Press, 1992.

[2] D.E. Myers, "Kriging, Cokriging, Radial Basis Functions and the Role of Positive Definiteness," *Computers and Mathematics with Applications*, vol. 24, pp. 139–148, 1992.

[3] M.R. Stytz and R.W. Parrot, "Using Kriging for 3D Medical Imaging," *Computerized Medical Imaging and Graphics*, vol. 17, pp. 421–442, November-December, 1993.

[4] A. Goshtasby, D.A. Turner and L.A. Ackerman, "Matching of Tomographic Slices for Interpolation," *IEEE Transactions on Medical Imaging*, vol. 11, pp. 507–516, December, 1992.

[5] G.J. Grevera and J.K. Udupa, "Shape-Based Interpolation of Multidimensional Grey-Level Images," *IEEE Transactions on Medical Imaging*, vol. 15, pp. 881–892, December, 1996.

[6] M. Unser and I. Daubechies, "On the Approximation Power of Convolution-Based Least Squares Versus Interpolation," *IEEE Transactions on Signal Processing*, vol. 45, pp. 1697–1711, July, 1997.

[7] M. Haddad and G. Porenta, "Impact of Reorientation Algorithms on Quantitative Myocardial SPECT Perfusion Imaging," *Journal of Nuclear Medicine*, vol. 39, pp. 1864-1869, 1998.

[8] B. Migeon and P. Marche, "In Vitro 3D Reconstruction of Long Bones Using B-Scan Image Processing," *Medicine and Biological Engineering and Computers*, vol. 35, pp. 369-372, July, 1997.

[9] J.L. Ostuni, A.K.S. Santha, V.S. Mattay, D.R. Weinberger, R.L. Levin and J.A. Frank, "Analysis of Interpolation Effects in the Reslicing of Functional MR Images," *Journal of Computer Assisted Tomography*, vol. 21, pp. 803–810, 1997.

[10] F.M. Weinhaus and V. Devarajan, "Texture Mapping 3D Models of Real-World Scenes," *ACM Computer Surveys*, vol. 29, pp. 325–365, December, 1997.

[11] T. Möller, R. Machiraju, K. Mueller and R. Yagel, "Evaluation and Design of Filters Using a Taylor Series Expansion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pp. 184–199, April-June, 1997.

[12] M.R. Smith and S.T. Nichols, "Efficient Algorithms for Generating Interpolated (Zoomed) MR Images," *Magnetic Resonance in Medicine*, vol. 7, pp. 156–171, 1988.

[13] M. Unser, A. Aldroubi and M. Eden, "Enlargement or reduction of Digital Images with Minimum Loss of Information," *IEEE Transactions on Image Processing*, vol. 4, pp. 247–258, March, 1995.

[14] P. Thévenaz and M. Unser, "Separable Least-Squares Decomposition of Affine Transformations," in Proc. *IEEE International Conference on Image Processing*, Santa Barbara, California, U.S.A., October 26-29, 1997, vol. CD paper No. 200,

[15] S.D. Fuller, S.J. Butcher, R.H. Cheng and T.S. Baker, "Three-Dimensional Reconstruction of Icosahedral Particles—The Uncommon Line," *Journal of Structural Biology*, vol. 116, pp. 48–55, January-February, 1996.

[16] U.E. Ruttimann, P.J. Andreason and D. Rio, "Head Motion during Positron Emission Tomography: Is It Significant?," *Psychiatry Research: Neuroimaging*, vol. 61, pp. 43–51, 1995.

[17] P. Thévenaz, U.E. Ruttimann and M. Unser, "A Pyramid Approach to Sub-Pixel Registration Based on Intensity," *IEEE Transactions on Image Processing*, vol. 7, pp. 27–41, January, 1998.

[18] E.V.R. Di Bella, A.B. Barclay, R.L. Eisner and R.W. Schafer, "A Comparison of Rotation-Based Methods for Iterative Reconstruction Algorithms," *IEEE Transactions on Nuclear Science*, vol. 43, pp. 3370–3376, December, 1996.

[19] M. Unser, A. Aldroubi and M. Eden, "B-Spline Signal Processing: Part I—Theory," *IEEE Transactions on Signal Processing*, vol. 41, pp. 821–832, February, 1993.

[20] M. Unser, A. Aldroubi and M. Eden, "B-Spline Signal Processing: Part II—Efficient Design and Applications," *IEEE Transactions on Signal Processing*, vol. 41, pp. 834–848, February, 1993.

[21] C.R. Appledorn, "A New Approach to the Interpolation of Sampled Data," *IEEE Transactions on Medical Imaging*, vol. 15, pp. 369–376, June, 1996.

[22] I.J. Schoenberg, "Contribution to the Problem of Approximation of Equidistant Data by Analytic Functions," *Quarterly of Applied Mathematics*, vol. 4, pp. 45–99, 112–141, 1946, 1946.

[23] N.A. Dodgson, "Quadratic Interpolation for Image Resampling," *IEEE Transactions on Image Processing*, vol. 6, pp. 1322–1326, September, 1997.

[24] R.G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP, pp. 1153–1160, 1981.

[25] S.K. Park and R.A. Schowengerdt, "Image Reconstruction by Parametric Cubic Convolution," *Computer Vision, Graphics, and Image Processing*, vol. 23, pp. 258–272, 1983.

[26] L. Piegl and W. Tiller, *The NURBS Book*. Berlin: Springer-Verlag, 1997.

[27] T. Blu, P. Thévenaz and M. Unser, "Minimum Support Interpolators with Optimum Approximation Properties," in Proc. *IEEE International Conference on Image Processing*, Chicago, Illinois, U.S.A., October 4–7, 1998, pp. WA06.10.

[28] A. Schaum, "Theory and Design of Local Interpolators," *CVGIP: Graphical Models and Image Processing*, vol. 55, pp. 464–481, November, 1993.

[29] F.J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE*, vol. 66, pp. 51–83, January, 1978.

[30] M.-L. Liou, "Spline Fit Made Easy," *IEEE Transactions on Computers*, vol. C-25, pp. 522–527, May, 1976.

[31] E. Maeland, "On the Comparison of Interpolation Methods," *IEEE Transactions on Medical Imaging*, vol. 7, pp. 213–217, September, 1988.

[32] M. Unser, M.A. Neimark and C. Lee, "Affine Transformations of Images: A Least-Squares Formulation," in Proc. *First IEEE International Conference on Image Processing*, Austin, Texas, U.S.A., November 13-16, 1994, vol. 3, of 3, pp. 558–561.

[33] T. Blu and M. Unser, "Approximation Error for Quasi-Interpolators and (Multi)-Wavelet Expansions," *Applied and Computational Harmonic Analysis*, vol. 6, pp. 219–251, March, 1999.

[34] T. Blu and M. Unser, "Quantitative Fourier Analysis of Approximation Techniques: Part I—Interpolators and Projectors," *to appear in IEEE Transactions on Signal Processing*, 1999.

[35] T. Blu and M. Unser, "Quantitative Fourier Analysis of Approximation Techniques: Part II—Wavelets," *to appear in IEEE Transactions on Signal Processing*, 1999.

[36] S.K. Park and R.A. Schowengerdt, "Image Sampling, Reconstruction, and the Effect of Sample-Scene Phasing," *Applied Optics*, vol. 21, pp. 3142–3151, September, 1982.

[37] G. Strang and G. Fix, "A Fourier Analysis of the Finite Element Variational Method," in *Constructive Aspect of Functional Analysis*, Rome, Italy: Edizioni Cremonese, pp. 796–830, 1971.

[38] E.H.W. Meijering, K.J. Zuidervel and M.A. Viergever, "Image Reconstruction with Symmetrical Piecewise nth-Order Polynomial Kernels," *IEEE Transactions on Image Processing*, vol. 8, pp. 192–201, February, 1999.

[39] R.W. Parrot, M.R. Stytz, P. Amburn and D. Robinson, "Towards Statistically Optimal Interpolation for 3-D Medical Imaging," *IEEE Engineering in Medicine and Biology*, vol. 12, pp. 49–59, September/October, 1993.

[40] A. Aldroubi and M. Unser, "Sampling Procedures in Function Spaces and Asymptotic Equivalence with Shannon's Sampling Theorem," *Numerical Function Analysis and Optimization*, vol. 15, pp. 1–21, 1994.

[41] A. Aldroubi, M. Unser and M. Eden, "Cardinal Spline Filters: Stability and Convergence to the Ideal Sinc Interpolator," *Signal Processing*, vol. 28, pp. 127–138, 1992.

[42] A.K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, New Jersey, U.S.A.: Prentice-Hall, Inc., 1989.

[43] M. Unser, P. Thévenaz and L. Yaroslavsky, "Convolution-Based Interpolation for Fast, High-Quality Rotation of Images," *IEEE Transactions on Image Processing*, vol. 4, pp. 1371–1381, October, 1995.

[44] M. Unser, "Ten Good Reasons for Using Spline Wavelets," in Proc. *Proc. SPIE, Wavelet Applications in Signal and Image Processing V*, San Diego, California, U.S.A., July 30–August 1, 1997, vol. 3169, pp. 422–431.

[45] M. Unser and A. Aldroubi, "Polynomial Splines and Wavelets—A Signal Processing Perspective," in *Wavelets—A Tutorial in Theory and Applications*, C.K. Chui, Ed., San Diego: Academic Press, pp. 91–122, 1992.