Generalized Sampling in Computer Graphics

Diego Nehab[1,2] Hugues Hoppe[2]
[1]IMPA [2]Microsoft Research

IMPA Technical Report E022/2011

Microsoft Research Technical Report MSR-TR-2011-16

February 2011

(Originally submitted to SIGGRAPH 2010)

Generalized Sampling in Computer Graphics

Diego Nehab[1,2] Hugues Hoppe[2]
[1]IMPA [2]Microsoft Research

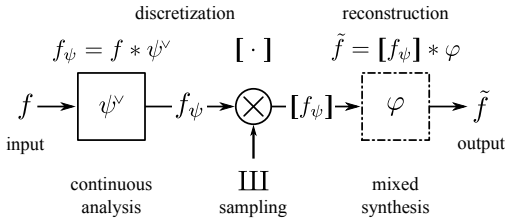


Figure 1: Traditional discretization and reconstruction. Given a continuous signal f, the discretization process convolves it with an analysis filter $\psi^\vee$ before sampling. The reconstruction process applies mixed convolution between the discrete sampling $[f_\psi]$ and a reconstruction kernel φ to obtain the reconstructed output.

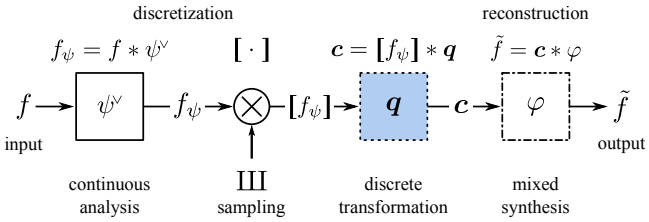


Figure 2: Generalized discretization and reconstruction. A discrete filtering stage is added to the pipeline. The output $[f_\psi]$ of the sampling stage is convolved with a discrete transformation filter $\boldsymbol{q}$. It is the result $\boldsymbol{c}$ of this stage (and not $[f_\psi]$) that is convolved with reconstruction kernel φ to produce the reconstructed signal.

Abstract

Analysis and reconstruction filters are crucial in graphics. The signal processing community has recently developed new filtering strategies based on a generalization of the traditional sampling pipeline. The main idea is to select simple basis functions such as B-splines but to effectively reshape these kernels by adding a discrete transformation filter. This approach is not widely known to graphics practitioners. In this paper we introduce new notation to succinctly summarize important algorithms in generalized sampling. We also present and analyze novel algorithms, including supersampling for antialiased rendering, and image downscaling for mipmap creation. The advantages of generalized sampling are twofold. The non-negativity of B-spline kernels simplifies both importance-based integration and GPU evaluation. And, the broader support of the transformed kernels improves filtering quality. A key challenge is that the discrete transformation often involves inverse convolutions, but fortunately the associated linear systems are banded and can be parallelized.

Keywords: signal processing, discretization and reconstruction, image interpolation, supersampling, antialiasing, downscaling.

1 Introduction

Given the continuous nature of visual information and the discrete nature of computers, it is unsurprising that discretization and reconstruction are fundamental operations in computer graphics. Figure 1 shows the traditional convolution-based sampling and reconstruction pipeline. During discretization (e.g., scene rasterization), a continuous input signal f is passed through an *analysis filter* $\psi^\vee$ (a.k.a. *sampling kernel*, *prefilter*, or *antialiasing filter*) before being sampled. The result is a discrete sequence $[f_\psi]$ (e.g., an image). During reconstruction (e.g., texture interpolation), the continuous approximation $\tilde{f}$ of the original signal is obtained by mixed convolution with a *reconstruction kernel* φ (a.k.a. *generating function*, *basis function*, or *postfilter*). The roles of the analysis filter $\psi^\vee$ and reconstruction kernel φ have traditionally been understood in light of the sampling theorem [Shannon 1949]. Thus, $\psi^\vee$ eliminates high frequencies from f so that the bandlimited f_ψ can be sampled without aliasing. In turn, φ aims to reconstruct $\tilde{f}$ from the sampled values without introducing spurious high frequencies. Ideally, $\tilde{f}$ equals f_ψ or better yet, if f is already bandlimited then $\tilde{f}$ equals f itself. The filters ψ and φ are typically designed to have small support for efficiency reasons, and this hinders their effectiveness.

Over the last 15 years, the signal and image processing communities have made significant progress using a slight generalization of the discretization and reconstruction pipeline [Unser and Aldroubi 1994; Unser et al. 1995a,b; Blu et al. 1999, 2001, 2004; Condat et al. 2005]. As shown in figure 2, the result $[f_\psi]$ of the sampling stage is transformed by a discrete filter $\boldsymbol{q}$ (a.k.a. *correction* or *basis change*) to form a new discrete signal $\boldsymbol{c}$, which is then convolved with φ as usual to reconstruct $\tilde{f}$. By introducing the discrete transformation $\boldsymbol{q}$, we can effectively create analysis and reconstruction filters of the form $\overline{\psi} = \psi * \boldsymbol{q}$ and $\overline{\varphi} = \boldsymbol{q} * \varphi$, which have wider support and thus higher quality. The key to the efficiency of this generalized sampling framework is that the transformation kernels $\boldsymbol{q}$ that arise in practice can be factored into a sequence of compact filters and inverses of compact filters. Even in the case of inverse filters, there are efficient algorithms to compute the required discrete convolution. Thus the correction stage adds negligible cost.

Generalized sampling was initially driven by the development of interpolation strategies based on a more general class of reconstruction kernels [Blu et al. 1999]. Figure 3 shows this application. The sampled image $[f_\psi]$ is processed by a discrete filter $\boldsymbol{q}$ resulting in a new image $\boldsymbol{c}$. This image can then be reconstructed efficiently with a simple cubic B-spline filter β^3. The resulting upsampled image is sharper and more isotropic (i.e., has higher quality) than that produced with the traditional Mitchell-Netravali filter [1988], even though both filters have the same degree and support.

We believe that the benefits of generalized sampling can impact many areas of computer graphics. In section 5 we explore several applications, including antialiased rendering and mipmap downscaling. Figure 4 shows the scenario of antialiasing using supersampling. A continuous signal f (i.e., a scene) is supersampled using the cubic B-spline basis β^3 as an antialiasing filter. The resulting image is then transformed with a discrete filter $\boldsymbol{q}$ that reshapes the antialias-

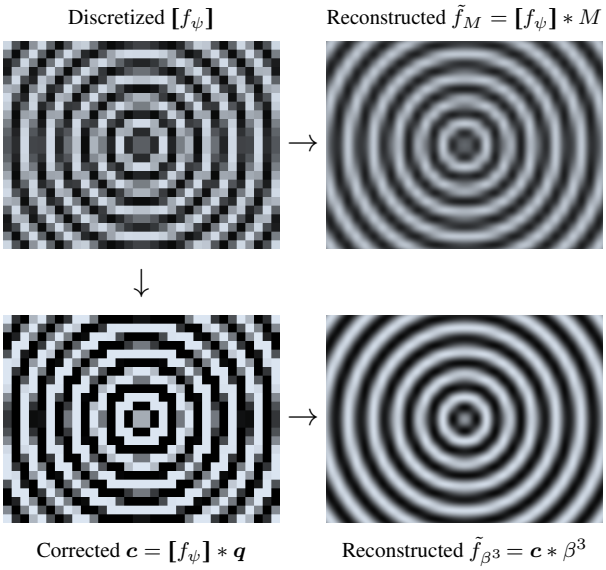


Figure 3: Reconstruction example. The top row shows the result of the traditional cubic Mitchell-Netravali filter M. The bottom row uses the generalized sampling approach, first applying a digital prefilter $\boldsymbol{q}$ as a preprocess, and then reconstructing with the cubic B-spline β^3 — which is less expensive to evaluate on a GPU than M.

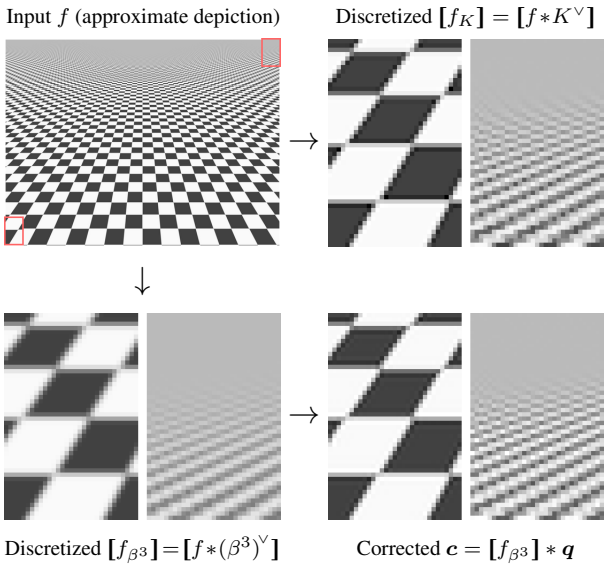


Figure 4: Supersampling example. The top row shows the result of the Keys (Catmull-Rom) filter K. The bottom row shows supersampling with the B-spline basis function β^3, followed by a digital correction filter $\boldsymbol{q}$. The kernels K and β^3 have the same support, and the fact that β^3 is non-negative simplifies importance sampling.

ing kernel *a posteriori*. The resulting image exhibits less ringing and aliasing for a similar computational cost. Moreover, the fact that we can easily draw random samples distributed as β^3 simplifies importance-based Monte Carlo integration.

Our paper aims to broaden access to this theory in the graphics community through the following contributions:

- We introduce novel notation (section 3) that simplifies the derivation of many previous algorithmic results (section 4), as well as our own new algorithms;
- We present and analyze novel algorithms for antialiased rendering and for downscaling of images (section 5);
- We describe the key operation of inverse convolution (of a compactly supported filter) in terms of linear algebra, and leverage matrix techniques to parallelize its computation (section 6);
- We present an off-line algorithm that enforces range constraints on the transformed coefficients $\boldsymbol{c}$, for compatibility with fixed-point image formats (section 6.4);
- We evaluate many reconstruction and antialiasing strategies, using animation tests that reveal subtle visual artifacts (section 8).

2 Previous work

The sampling theorem [Shannon 1949] bootstrapped decades of research by establishing the conditions under which a signal can be sampled and reconstructed without any loss of information. Specifically, a bandlimited signal that is sampled more densely than its Nyquist rate (twice its maximum frequency) can be reproduced exactly with the ideal reconstruction kernel $\varphi = \mathrm{sinc}$. To eliminate high frequencies from a non-bandlimited signal, we again use the ideal low-pass filter $\psi^\vee = \mathrm{sinc}$ for the analysis. The search for finite-support alternatives to the sinc filter led to the development of various windowed-sinc approximations (see [Meijering et al. 1999a] for a comprehensive analysis).

In the context of generalized sampling, the roles of $\psi^\vee$ and φ have been updated (see [Unser 2000] for a historical account). Given a sampling period T, the reconstruction kernel φ defines a signal subspace $V_{\varphi,T}$ containing all functions of the form $\tilde{f}_T(x) = \sum_{k\in\mathbb{Z}} \boldsymbol{c}_T[k]\varphi(x/T - k)$. The analysis and correction filters $\psi^\vee$ and $\boldsymbol{q}$ take an input function and determine the discrete sequence $\boldsymbol{c}_T$. A key principle in the design of φ, $\boldsymbol{q}$, and $\psi^\vee$ comes from approximation theory. The goal is to find combinations that lead the approximation error $\|f - \tilde{f}_T\|_{L_2}$ to quickly vanish as the sampling period decreases. The development of practical formulas that tie the approximation error to the choice of a given reconstruction scheme [Unser 1996] led to a number of results, including improved interpolation strategies [Blu et al. 2001].

Due to the symmetric role of the ideal low-pass filter sinc in discretization and reconstruction (a property that comes from orthogonality [Hummel 1983; Unser and Aldroubi 1994]), kernels designed to approximate sinc are used in both applications. In fact, this duality allows kernels specifically designed for reconstruction [Mitchell and Netravali 1988] to perform well in discretization (section 5).

Nevertheless, the two applications impose different constraints on the designer: reconstruction filters must be efficient to evaluate, whereas analysis filters are sometimes preferred to be non-negative. Research in alternative analysis filters $\psi^\vee$ and reconstruction kernels φ has therefore been conducted mostly independently. We split the description of related work accordingly.

2.1 Reconstruction kernels

We focus on piecewise polynomial reconstruction kernels. Besides enjoying an efficiency advantage over non-polynomial alternatives, these have been shown to match windowed-sinc approximations in quality [Meijering et al. 2001]. Excellent surveys of reconstruction strategies can be found elsewhere [Lehmann et al. 1999; Thévenaz et al. 2000; Meijering 2002]. Like Thévenaz et al. [2000], we use a set of properties that characterize reconstruction kernels to guide us through the bibliography.

The *degree* N of a kernel φ is the maximum degree found in any of its polynomials pieces. The *support* W of φ is the width of the smallest interval outside of which φ vanishes. Increasing either the degree or support of a kernel φ creates more degrees of freedom to the designer, but unfortunately also adds to the runtime computational cost.

To guarantee linear-phase response, most kernels are *symmetric* about the origin (see [Blu et al. 2004] for a curious exception). The *regularity* R measures the smoothness of φ. In other words, a ker-

nel φ is said to be in C^R if it can be differentiated R times. The *order of approximation* L measures the rate at which the approximation error vanishes as the sampling step is reduced:

$$\|f - \tilde{f}_T\|_{L_2} \propto T^L \text{ as } T \to 0.$$

Equivalently, a kernel with approximation order L can reproduce polynomial signals of degree $L-1$. Enforcement of regularity, symmetry, and approximation order in φ removes degrees of freedom from the design process.

The best approximation order a kernel of degree N can attain is $L=N+1$. This optimal order can be achieved even with a relatively compact support $W=N+1$ [Blu et al. 2001]. Different strategies for setting the remaining degrees of freedom have led to the development of a multitude of reconstruction kernels.

A reconstruction kernel φ is *interpolating* if it satisfies $\varphi(0) = 1$ and $\varphi(k) = 0, k \in \mathbb{Z} \setminus \{0\}$. Naturally, enforcing this property eliminates further degrees of freedom. Popular interpolating kernels include the ubiquitous nearest-neighbor ($L = 1, W = 1$) and linear ($L = 2, W = 2$) interpolation kernels.

Keys [1981] started from the family of differentiable interpolating cubic kernels ($N = 3, R = 1$). Optimizing the remaining degree of freedom for approximation order, he found the Catmull-Rom spline [Catmull and Rom 1974] ($L = 3, W = 4$), and another kernel with wider support and higher approximation order ($L = 4, W = 6$). Following a similar method, which is based on equating the Taylor series expansions of f and $\tilde{f}$, German [1997] designed a quartic kernel characterized by $L = 5, W = 7, R = 1$. Note that the support of these kernels is larger than necessary for their approximation order.

In designing a family of cubic kernels, Mitchell and Netravali [1988] started from $W = 4$, $R = 1$ and $L = 1$. The two remaining degrees of freedom were subjectively evaluated by their effect on the amount of ringing, blur, and anisotropy in upsampled images. Enforcing $L = 2$ left only one degree of freedom, and its value was chosen based on subjective quality.

Dodgson [1997] studied a family of second-order, quadratic reconstruction kernels ($N = 2, W = 3, L = 2$). From this family he identified two interesting kernels: one is C^0-continuous and interpolating; the other is C^1-continuous but non-interpolating.

Inspired by the work of Park and Schowengerdt [1983], who had reached the same cubic kernel as Keys through a different route, Schaum [1993] derived a frequency-domain formula for the mean approximation error over all shifts of the input function:

$$\|f - \tilde{f}_T\|_{L_2} = \left(\int |\hat{f}(\omega)|^2 E(T\omega) \frac{\mathrm{d}\,\omega}{2\pi} \right)^{-\frac{1}{2}}.$$

Interestingly, the expression for the so-called *error kernel* $E(T\omega)$ does not depend on f, but only on φ. Since much of the useful information in images lies in the low frequencies, Schaum designed a family of interpolating kernels whose associated error kernel vanishes at $\omega=0$, along with as many of its derivatives as possible. The result is a family of local Lagrangian interpolators parametrized by degree, each with optimal order and support $L = W = N + 1$.

Meijering et al. [1999b] designed kernels starting from interpolation and regularity constraints. To make the kernels behave like sinc in the low frequencies, the remaining parameters were selected to cancel as many terms as possible in the Maclaurin series of their Fourier transforms. The resulting cubic kernel is the same as that of Keys. Although the supports of the quintic and septic kernels are larger ($W = 6$ and 8, respectively), they have the same approximation order $L = 3$ as the cubic and perform only slightly better.

A breakthrough came from the idea that the responsibility for interpolating the original samples need not be imposed on the continuous kernel φ itself, but can instead be achieved by using a discrete correction prefilter $\boldsymbol{q}$. This was first demonstrated in the context of B-spline interpolation [Hou and Andrews 1978; Unser et al. 1991]. B-splines are the most regular members of a class of functions called MOMS (maximal order, minimum support) [Blu et al. 2001]. The best performing kernels, the O-MOMS (optimal MOMS), trade-off regularity to minimize the leading coefficient C_{int} in the mean interpolation error

$$\|f - \tilde{f}_T\|_{L_2} = C_{\text{int}}\, T^L\, \|f^{(L)}\|_{L_2} \text{ as } T \to 0.$$

For applications needing derivatives, the SO-MOMS (sub-optimal MOMS) minimize C_{int} subject to C^1-continuity. None of these kernels use up degrees of freedom enforcing the interpolation constraint. In contrast, the I-MOMS (interpolating MOMS) are precisely the local Lagrangian interpolators described by Schaum [1993] (which do not perform nearly as well).

As we will see in section 4.1, obtaining the optimal reconstruction error within a signal subspace V_φ requires both access to the original signal f and analysis with the dual $\mathring{\varphi}^\vee$ of φ. Nevertheless, when we only have access to $[\![f_\psi]\!]$, it is still possible to reduce reconstruction error by mimicking the low-frequency behavior of $\mathring{\varphi}^\vee$.

The *shifted* linear interpolation scheme of Blu et al. [2004] gives up on linear phase and uses the additional freedom to minimize the approximation error. *Quasi-interpolation* schemes instead give up on interpolation of $[\![f_\psi]\!]$, so that $\boldsymbol{q}$ is freed of this restriction. Instead, the interpolation property holds only when f is a polynomial of degree less than L (the quasi-interpolation order). Blu and Unser [1999] propose an IIR design for $\boldsymbol{q}$, Condat et al. [2005] an all-pole design, and Dalai et al. [2005] a FIR design. The improvements can be substantial, particularly for low-degree schemes ($N \leq 3$).

2.2 Analysis filters

In computer graphics, analysis filters (called *antialiasing filters*) are used for image synthesis in 2D or 3D rendering. The input signal f is the 2D illustration or projected 3D scene. Compared to many other signal-processing fields, graphics is fortunate in that this continuous function f is available prior to sampling, sometimes even in analytic form.

Computing samples of the filtered signal f_ψ at positions $k \in \mathbb{Z}$ entails evaluating integrals of the form

$$f_\psi(k) = \big(f * \varphi^\vee\big)(k) = \int_{\Omega_\psi} f(x+k)\, \psi(x)\, \mathrm{d}x, \tag{1}$$

where Ω_ψ is the support of ψ. Techniques for computing an exact or approximate analytic solution to (1) are called *prefiltering techniques*. Unfortunately, analytic approximations are often impractical. Instead, one often relies on *supersampling techniques* which approximate (1) using Monte Carlo estimation:

$$\int_{\Omega_\psi} f(x+k)\, \psi(x)\, \mathrm{d}x = \mathbb{E}\big(f(X+k)\, \psi(X)\big) \tag{2}$$

$$\approx \frac{1}{m} \sum_{i=1}^{m} f(x_i+k)\, \psi(x_i). \tag{3}$$

Here $\mathbb{E}$ stands for expectation, and the x_i are samples drawn from the random variable X, distributed in Ω_φ. To reduce the variance of the estimator, and therefore increase its precision, it suffices to increase the number m of samples proportionally.

Adapting the spatial distribution of the samples can improve the rate of convergence of the integral (see *quasi*-Monte Carlo, importance sampling, and stratified sampling), and can also reshape the spectral properties of the estimation error, thereby trading off aliasing for noise (see jittering) [Glassner 1995]. Here we are interested in the antialiasing kernels themselves, and therefore assume that the integrals are evaluated to the maximum output precision.

The quality of an antialiasing filter ψ involves a subjective balance between aliasing, blurring, and ringing. The sweet spot also depends on properties of the output device such as post-aliasing [Mitchell and Netravali 1988]. Some favor the sharpness offered by filters with negative lobes (the *negative lobists* [Blinn 1989]), while others fear the accompanying ringing in dark regions (a problem accentuated by gamma correction). Professional rendering systems offer several alternatives, including box, triangle, Catmull-Rom (Keys), Gaussian, and sinc filters [Pixar 2005].

For both prefiltering and supersampling, it is important for the analysis filter to have small support W to maximize performance. For prefiltering, compact support reduces the complexity of the analytic expression because it involves fewer scene elements. For supersampling, small support improves locality of reference.

3 Notation and basic properties

In this section we introduce some novel notation and a variety of properties (some standard, others less known) that will be used throughout the paper. These tools simplify the derivation of many algorithms and concepts to trivial algebraic manipulations.

We denote discrete sequences by bold letters, or write them enclosed in bold square-brackets $\mathbf{[\,]}$:

$$\boldsymbol{c} \stackrel{\text{def}}{=} \mathbf{[}\ldots, c_{-2}, c_{-1},\ c_0,\ c_1, c_2, \ldots\mathbf{]}, \quad c_k \in \mathbb{R},\ k \in \mathbb{Z}.$$

An element of a sequence is selected by postfixing the sequence with the index in normal square-brackets $[\,]$:

$$\boldsymbol{c}[k] \stackrel{\text{def}}{=} c_k, \quad k \in \mathbb{Z}.$$

The operation of uniformly sampling a function at all multiples of a sampling period T is so fundamental to this paper that we introduce special notation for the resulting sequence:

$$\begin{aligned}\mathbf{[}f\mathbf{]}_T &\stackrel{\text{def}}{=} \mathbf{[}\ldots, f(-2T), f(-T),\ f(0),\ f(T), f(2T), \ldots\mathbf{]}\\ &\stackrel{\text{def}}{=} \mathbf{[}f\mathbf{]},\ \text{in short when } T = 1.\end{aligned}$$

Throughout the text we assume a unit sampling period unless otherwise indicated. Where there is risk of ambiguity or need for added flexibility, we explicitly specify the sampling index, as in:

$$\mathbf{[}f(kT)\mathbf{]}_{k\in\mathbb{Z}} = \mathbf{[}f\mathbf{]}_T.$$

Let "$\cdot$" denote the implicit parameter of a univariate function, e.g.:

$$f(s\cdot + k) \ \stackrel{\text{def}}{=}\ x \mapsto f(s\,x + k).$$

The three flavors of convolution are denoted by the same operator. The discrete and continuous convolutions are defined as

$$\boldsymbol{b} * \boldsymbol{c} \stackrel{\text{def}}{=} \sum_{k\in\mathbb{Z}} \boldsymbol{b}[k]\boldsymbol{c}[\cdot - k] \quad \text{and} \quad f * g \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(x)g(\cdot - x)\,\mathrm{d}x,$$

and mixed convolution is defined as

$$\boldsymbol{c} * \varphi \stackrel{\text{def}}{=} \boldsymbol{c} *_1 \varphi, \quad \text{where} \quad \boldsymbol{c} *_T \varphi \stackrel{\text{def}}{=} \sum_{k\in\mathbb{Z}} \boldsymbol{c}[k]\varphi(\cdot - kT).$$

Commutativity and associativity apply to all combinations of convolutions, including mixed convolutions. Thus parentheses are unnecessary, and we can manipulate convolutions like products:

$$(4)\quad (\boldsymbol{b} * \boldsymbol{c}) * f = \boldsymbol{b} * (\boldsymbol{c} * f) \quad \text{and} \quad (f * g) * \boldsymbol{b} = f * (g * \boldsymbol{b}).$$

Interestingly, sampling a mixed convolution is equivalent to performing a discrete convolution between the sequence and the sampled function (as long as the sampling periods are the same):

$$(5)\qquad \mathbf{[}\boldsymbol{b} *_T f\mathbf{]}_T = \boldsymbol{b} * \mathbf{[}f\mathbf{]}_T.$$

The discrete and continuous unit impulses $\boldsymbol{\delta}$ and δ are uniquely defined by the sampling properties:

$$(6)\qquad \boldsymbol{\delta} * \boldsymbol{c} \stackrel{\text{def}}{=} \boldsymbol{c},\ \forall \boldsymbol{c} \quad \text{and} \quad \delta * f \stackrel{\text{def}}{=} f,\ \forall f.$$

The discrete impulse can also be defined simply as

$$(7)\qquad \boldsymbol{\delta} = \mathbf{[}\ldots, 0, 0, 1, 0, 0, \ldots\mathbf{]} \ \text{ where } \ \boldsymbol{\delta}[0] = 1.$$

The convolution inverse $\boldsymbol{b}^{-1}$ of a sequence $\boldsymbol{b}$, when it exists (see section 6 for details), is uniquely defined by:

$$(8)\qquad \boldsymbol{b}^{-1} * \boldsymbol{b} \stackrel{\text{def}}{=} \boldsymbol{\delta}.$$

Shifted versions of $\boldsymbol{\delta}$ and δ are denoted in short as

$$\boldsymbol{\delta_k} \stackrel{\text{def}}{=} \boldsymbol{\delta}[\cdot - k],\ k \in \mathbb{Z} \quad \text{and} \quad \delta_\tau \stackrel{\text{def}}{=} \delta(\cdot - \tau),\ \tau \in \mathbb{R}.$$

Due to the convolution properties

$$\boldsymbol{b}[\cdot - k] * \boldsymbol{c} = (\boldsymbol{b} * \boldsymbol{c})[\cdot - k] \quad \text{and} \quad f(\cdot - \tau) * g = (f * g)(\cdot - \tau),$$

we can use the shorthand notation for the shifting operations

$$\boldsymbol{\delta_k} * \boldsymbol{c} = \boldsymbol{c}[\cdot - k] \quad \text{and} \quad \delta_\tau * \varphi = \varphi(\cdot - \tau).$$

The reflection of sequences and functions is denoted by

$$\boldsymbol{c}^\vee \stackrel{\text{def}}{=} \boldsymbol{c}[-\cdot] \quad \text{and} \quad f^\vee \stackrel{\text{def}}{=} f(-\cdot),$$

and distributes over all flavors of convolution:

$$(9)\quad (\boldsymbol{b} * \boldsymbol{c})^\vee = \boldsymbol{b}^\vee * \boldsymbol{c}^\vee,\ \ (f * g)^\vee = f^\vee * g^\vee, \text{ and } \ (\boldsymbol{b} * \varphi)^\vee = \boldsymbol{b}^\vee * \varphi^\vee.$$

We also use short notation for the reflected convolution inverse:

$$\boldsymbol{b}^{-\vee} \stackrel{\text{def}}{=} (\boldsymbol{b}^{-1})^\vee = (\boldsymbol{b}^\vee)^{-1}.$$

The cross-correlation of functions f and g is denoted by

$$(10)\qquad a_{f,g} \stackrel{\text{def}}{=} f * g^\vee.$$

In particular, the auto-correlation of f is denoted by

$$(11)\qquad a_f \stackrel{\text{def}}{=} f * f^\vee.$$

The L_2 inner product of real-valued functions f and g is defined as

$$\langle f,\ g\rangle \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(x)g(x)\,\mathrm{d}x.$$

A key relationship between inner products and convolution lets us express in compact notation the operation of sampling a function f with an analysis filter ψ:

$$\begin{aligned}\langle f,\ \psi\rangle &= (f * \psi^\vee)(0) \Rightarrow\\ \langle f,\ \delta_k * \psi\rangle &= (f * \psi^\vee)(k) \Rightarrow\\ \mathbf{[}\langle f,\ \delta_k * \psi\rangle\mathbf{]}_{k\in\mathbb{Z}} &= \mathbf{[}f * \psi^\vee\mathbf{]}.\end{aligned} \qquad (12)$$

Two functions f and g are biorthogonal if they satisfy

$$(13)\qquad \langle f,\ \delta_k * g\rangle = \langle \delta_i * f,\ \delta_j * g\rangle = \begin{cases}1 & i = j\\ 0 & \text{otherwise,}\end{cases}$$

where $k, i, j \in \mathbb{Z}$ and $k = i - j$, or equivalently,

$$(14)\qquad \mathbf{[}a_{f,g}\mathbf{]} = \boldsymbol{\delta}.$$

The Discrete Time Fourier Transform (DTFT) of a sequence and the Fourier Transform of a function are defined respectively by

$$(15)\quad \widehat{\boldsymbol{c}}(\omega) \stackrel{\text{def}}{=} \sum_{k\in\mathbb{Z}} \boldsymbol{c}[k] e^{-i\omega k} \quad \text{and} \quad \hat{f}(\omega) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(x) e^{-i\omega x}\,\mathrm{d}x.$$

Convolution in the time domain becomes a product in the frequency domain. This is also true of mixed convolutions:

$$(16)\qquad \widehat{(\boldsymbol{b} * \boldsymbol{c})} = \widehat{\boldsymbol{c}}\,\widehat{\boldsymbol{b}} \qquad \widehat{(f * g)} = \hat{f}\hat{g}. \qquad \widehat{(f * \boldsymbol{c})} = \hat{f}\,\widehat{\boldsymbol{c}}$$

The B-spline basis functions β^n are defined by the relations:

$$(17)\qquad \beta^0(x) = \begin{cases} 1 & |x| < \frac{1}{2} \\ \frac{1}{2} & |x| = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \quad \beta^n = \beta^{n-1} * \beta^0.$$

Our notation can summarize concisely many of the important properties that kernel functions may have:

Property of kernel φ	Expressed in our notation
Symmetry	$\varphi = \varphi^\vee$
Interpolation	$\left[\varphi\right] = \boldsymbol{\delta}$
Orthogonality	$\left[\varphi * \varphi^\vee\right] = \boldsymbol{\delta}$
Normalization	$\varphi * 1 = 1$
Partition of unity	$\varphi * \left[1\right] = 1$

4 Previous algorithms

In this section we use our new notation to review several previous algorithms involving generalized sampling. To simplify the presentation, we express all computations in one dimension. However, all the results are easily extended to 2D images and 3D volumes by making use of tensor-product basis functions. For example, in 2D we have $\varphi_{\text{2D}}(x, y) = \varphi(x)\,\varphi(y)$.

Space of admissible kernels In formal presentations, it is common to open with admissibility restrictions on the generating function φ. That is, we must ensure that the signal subspace

$$V_{\varphi,T} = \{\tilde{f}_T : \mathbb{R} \to \mathbb{R} \mid \tilde{f}_T = \boldsymbol{c}_T *_T \varphi(\,\cdot/T),\ \forall \boldsymbol{c}_T \in l_2\}$$

is a closed subspace of L_2 before we can discuss approximation in $V_{\varphi,T}$. To do so, we require that the set of shifted generating functions $\varphi(\,\cdot/T - k), k \in \mathbb{Z}$ form a Riesz basis of $V_{\varphi,T}$. In other words, any signal in $V_{\varphi,T}$ is uniquely determined by $\boldsymbol{c}_T$. This restriction is equivalent to the requirement that convolution with the auto-correlation sequence $\left[a_\varphi\right]$ of φ must be an invertible operator from l_2 into itself. Interestingly, if $\boldsymbol{p}$ is an invertible convolution operator and $\overline{\varphi} = \boldsymbol{p} * \varphi$, then $V_{\overline{\varphi}} = V_\varphi$ and therefore $\overline{\varphi}$ and φ are *equivalent generating functions*.

Another common requirement is that $V_{\varphi,T}$ contain elements arbitrarily close to any signal in L_2 as we allow the sampling step T to decrease indefinitely. This is equivalent to enforcing the *partition of unity* property in φ, that is $\sum_{k\in\mathbb{Z}} \varphi(\,\cdot - k) = 1$. It is also equivalent to requiring φ to have approximation order $L \geq 1$.

The important observation here is that these safeguards are satisfied by most generating functions we find in practice. The interested reader is referred to [Aldroubi and Unser 1994; Unser and Aldroubi 1994] for a careful treatment of the subject.

4.1 Orthogonal projection

The closest reconstructed signal $\tilde{f} \in V_\varphi \subset L_2$ to an input signal $f \in L_2$ is the orthogonal projection $P_\varphi f$ [Aldroubi and Unser 1994]. The vector $\boldsymbol{c}$ corresponding to the orthogonal projection can be generated by sampling the convolution of f with $\overline{\psi}^\vee = \mathring{\varphi}^\vee$, where $\mathring{\varphi}$ is the *dual* of φ (see figure 5c). A key property is that $\mathring{\varphi} \in V_\varphi$, and therefore $\mathring{\varphi}$ can be expressed as a mixed convolution with φ, i.e., $\overline{\psi} = \boldsymbol{q} * \varphi$. We derive an expression for $\boldsymbol{q}$ as follows.

The orthogonality condition is

$$(18)\qquad (f - \tilde{f}) \perp V_\varphi \quad \Leftrightarrow \quad \langle f - \tilde{f},\, \delta_k * \varphi \rangle = 0,\ \forall k \in \mathbb{Z},$$

or equivalently by (12), $\left[(f - \tilde{f}) * \varphi^\vee\right] = \mathbf{0}$. Therefore,

$$(19)\qquad \left[f * \varphi^\vee\right] = \left[\tilde{f} * \varphi^\vee\right] \qquad \text{by linearity}$$
$$(20)\qquad = \left[\boldsymbol{c} * \varphi * \varphi^\vee\right] \qquad \text{since } \tilde{f} = \boldsymbol{c} * \varphi$$
$$(21)\qquad = \boldsymbol{c} * \left[\varphi * \varphi^\vee\right] \qquad \text{by (5)}$$
$$(22)\qquad = \boldsymbol{c} * \left[a_\varphi\right] \qquad \text{by (11).}$$

Here we note that $\left[a_\varphi\right] = \left[\varphi * \varphi^\vee\right]$ is the sampled auto-correlation function of φ. Convolving both sides with the inverse of $\left[a_\varphi\right]$,

$$(23)\qquad \boldsymbol{c} = \left[a_\varphi\right]^{-1} * \left[f * \varphi^\vee\right] \qquad \text{by (8)}$$
$$(24)\qquad = \left[f * \left[a_\varphi\right]^{-1} * \varphi^\vee\right] \qquad \text{by (5) and (4)}$$
$$(25)\qquad = \left[f * \left(\left[a_\varphi\right]^{-\vee} * \varphi\right)^\vee\right] \qquad \text{by (9)}$$
$$(26)\qquad = \left[f * \left(\left[a_\varphi\right]^{-1} * \varphi\right)^\vee\right] \qquad a_\varphi \text{ is symmetric.}$$

Therefore, the expression we seek for the dual is

$$(27)\quad \mathring{\varphi} = \left[a_\varphi\right]^{-1} * \varphi, \quad \text{so that} \quad \boldsymbol{q} = \left[a_\varphi\right]^{-1} \quad \text{(see figure 5c,f).}$$

Indeed, φ and $\mathring{\varphi}$ are duals since:

$$(28)\qquad \left[\varphi * \mathring{\varphi}^\vee\right] = \left[\varphi * \left(\left[a_\varphi\right]^{-1} * \varphi\right)^\vee\right] = \left[\varphi * \left[a_\varphi\right]^{-1} * \varphi^\vee\right]$$
$$(29)\qquad = \left[a_\varphi\right]^{-1} * \left[\varphi * \varphi^\vee\right] = \left[a_\varphi\right]^{-1} * \left[a_\varphi\right] = \boldsymbol{\delta},$$

which implies biorthogonality from (14).

Benefit of generalized sampling Here comes the interesting part. Recall from section 2 that for efficiency we desire both φ and ψ to be compactly supported. However, if φ has compact support, its dual $\mathring{\varphi}$ must have infinite support (except for the uninteresting kernels for which $\left[a_\varphi\right] = \boldsymbol{\delta}$). Thus, directly evaluating $\left[f * \mathring{\varphi}^\vee\right]$ is impractical. Instead, we factor it as $\left[f * \mathring{\varphi}^\vee\right] = \left[f * \overline{\psi}\right] = \left[f * \psi\right] * \boldsymbol{q}$, with $\psi = \varphi$ and $\boldsymbol{q} = \left[a_\varphi\right]^{-1}$, both of which are efficient. We will see pattern again in other algorithms.

4.2 Consistent sampling (oblique projection)

Often we have little control over the analysis filter $\psi^\vee$ (e.g., it is part of an acquisition device). Naturally, this prevents us from using the orthogonal projection. When $\psi^\vee$ is known and with a given φ, *consistent sampling* [Unser and Aldroubi 1994] is a strategy that guarantees idempotence of the sampling pipeline. The appropriate discrete filter $\boldsymbol{q}$ is derived as follows:

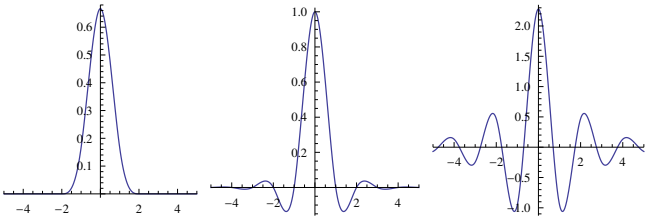
$$(30)\qquad \left[f_\psi\right] = \left[\left(\left[f_\psi\right] * \overline{\varphi}\right) * \psi^\vee\right]$$
$$(31)\qquad = \left[f_\psi\right] * \left[\overline{\varphi} * \psi^\vee\right]$$
$$(32)\qquad = \left[f_\psi\right] * \left[\boldsymbol{q} * \varphi * \psi^\vee\right]$$
$$(33)\qquad = \left[f_\psi\right] * \boldsymbol{q} * \left[a_{\varphi,\psi}\right].$$

Note that $\left[a_{\varphi,\psi}\right] = \left[\varphi * \psi^\vee\right]$ is the sampled cross-correlation of φ and ψ. Since $\boldsymbol{q} * \left[a_{\varphi,\psi}\right]$ leaves the arbitrary $\left[f_\psi\right]$ unchanged,

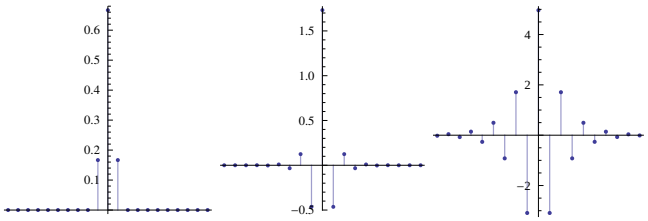
$$(34)\qquad \boldsymbol{q} * \left[a_{\varphi,\psi}\right] = \boldsymbol{\delta} \quad \Rightarrow \quad \boldsymbol{q} = \left[a_{\varphi,\psi}\right]^{-1} \qquad \text{by (7) and (8).}$$

Since $\left[f_\psi\right] = \left[f * \psi^\vee\right]$ and $\left[f_\psi\right] * \overline{\varphi} = \tilde{f}$, we can rewrite (30) as

$$(35)\qquad \left[f * \psi^\vee\right] = \left[\tilde{f} * \psi^\vee\right] \quad \Rightarrow \quad \left[(f - \tilde{f}) * \psi^\vee\right] = \mathbf{0}.$$



(a) Primal φ (compact support) (b) Cardinal φ_{int} (interpolating) (c) Dual $\mathring{\varphi}$ (biorthogonal to φ)



(d) Sampled primal $[\![\varphi]\!]$ ($[\![\frac{1}{6}, \frac{4}{6}, \frac{1}{6}]\!]$ for β^3) (e) Discrete filter $[\![\varphi]\!]^{-1}$ ($\varphi_{\text{int}} = [\![\varphi]\!]^{-1} * \varphi$) (f) Discrete filter $[\![a_\varphi]\!]^{-1}$ ($\mathring{\varphi} = [\![a_\varphi]\!]^{-1} * \varphi$)

Figure 5: Visualization of the functions and sequences associated with the choice $\varphi = \beta^3$ of the cubic B-spline basis.

Thus consistent sampling can be viewed as a form of *oblique projection* since the residual $f - \tilde{f}$ is orthogonal to V_ψ, and not necessarily to V_φ. The notation is $\tilde{f} = P_{\varphi \perp \psi} f$.

The approximation error of the oblique projection is bounded by

$$\|f - P_\varphi f\| \leq \|f - P_{\varphi \perp \psi} f\| \leq (\cos \theta_{\psi,\varphi})^{-1} \|f - P_\varphi f\|, \tag{36}$$

where $\theta_{\psi,\varphi}$ is a measure of the "maximum angle" between the two spaces and can be computed from their *spectral coherence* [Unser and Aldroubi 1994].

4.3 Interpolation

In section 2, we saw that relieving φ from the restrictive interpolation constraints $[\![\varphi]\!] = \boldsymbol{\delta}$ lets us use degrees of freedom to optimize for other desirable properties. Here we derive the discrete filter $\boldsymbol{q}$ that causes the reconstruction of $\boldsymbol{c}$ with φ to interpolate $[\![f_\psi]\!]$. Since this filtering operation can be performed during preprocessing, we pay no performance penalty during the reconstruction stage. In the words of Blu et al. [1999], "higher-quality at no additional cost". From the interpolation assumption,

$$[\![f_\psi]\!] = [\![\tilde{f}\,]\!] = [\![\boldsymbol{c} * \varphi]\!] = \boldsymbol{c} * [\![\varphi]\!] = [\![f_\psi]\!] * \boldsymbol{q} * [\![\varphi]\!]. \tag{37}$$

Here we note that $\boldsymbol{q} * [\![\varphi]\!]$ leaves the arbitrary $[\![f_\psi]\!]$ unchanged, so

$$\boldsymbol{q} * [\![\varphi]\!] = \boldsymbol{\delta} \;\Rightarrow\; \boldsymbol{q} = [\![\varphi]\!]^{-1} \qquad \text{by (6) and (8).} \tag{38}$$

It is possible to reach the same results starting from the consistent sampling conditions. Simply set $\psi = \delta$ in (34):

$$\boldsymbol{q} = [\![a_{\varphi,\psi}]\!]^{-1} = [\![\varphi * \psi^\vee]\!]^{-1} = [\![\varphi * \delta\,]\!]^{-1} = [\![\varphi]\!]^{-1}, \tag{39}$$

as expected. This preprocessing approach is equivalent to reconstruction with an effective basis function $\overline{\varphi} = \boldsymbol{q} * \varphi$ which is called the *cardinal* generating function φ_{int} of V_φ (see figure 5b,e):

$$\varphi_{\text{int}} = [\![\varphi]\!]^{-1} * \varphi. \tag{40}$$

It is easy to verify that φ_{int} is indeed an interpolating kernel:

$$[\![\varphi_{\text{int}}]\!] = [\![[\![\varphi]\!]^{-1} * \varphi]\!] = [\![\varphi]\!]^{-1} * [\![\varphi]\!] = \boldsymbol{\delta}. \tag{41}$$

4.4 Least-squares translation

The naïve way of translating a discrete signal $[\![f_\psi]\!]$ by an offset τ is to first reconstruct it as $\tilde{f} = \boldsymbol{c} * \varphi \in V_\varphi$ in some way, and then sample the translated reconstruction $\delta_\tau * \tilde{f}$:

$$[\![\delta_\tau * \tilde{f}\,]\!] = [\![\delta_\tau * \boldsymbol{c} * \varphi]\!] = \boldsymbol{c} * [\![\delta_\tau * \varphi]\!]. \tag{42}$$

The operation reduces to a discrete convolution between the coefficient array $\boldsymbol{c}$ and the sampled, translated basis function. In a sense, this operation samples $\delta_\tau * \tilde{f}$ without any filtering. The idea behind the least-squares translation operator described in [Unser et al. 1995b] is to form the orthogonal projection $P_\varphi(\delta_\tau * \tilde{f}) = \boldsymbol{c}_\tau * \varphi$:

$$\boldsymbol{c}_\tau = [\![\delta_\tau * \tilde{f} * \mathring{\varphi}^\vee]\!] = [\![\delta_\tau * \boldsymbol{c} * \varphi * \mathring{\varphi}^\vee]\!] = \boldsymbol{c} * [\![\delta_\tau * \varphi * \mathring{\varphi}^\vee]\!] \tag{43}$$

$$= \boldsymbol{c} * [\![\delta_\tau * \varphi * [\![a_\varphi]\!]^{-1} * \varphi^\vee]\!] \tag{44}$$

$$= \boldsymbol{c} * [\![a_\varphi]\!]^{-1} * [\![\delta_\tau * a_\varphi]\!]. \tag{45}$$

Therefore, the sampled least-squares translation is

$$[\![P_\varphi(f * \delta_\tau)]\!] = \boldsymbol{c} * [\![a_\varphi]\!]^{-1} * [\![\delta_\tau * a_\varphi]\!] * [\![\varphi]\!], \tag{46}$$

which can be computed as three consecutive discrete convolutions.

Unser et al. observe from rewriting (44) into the form

$$\boldsymbol{c}_\tau = \boldsymbol{c} * [\![\delta_\tau * (a_\varphi)_{\text{int}}]\!] \qquad \text{by (40)} \tag{47}$$

that least-squares translation in space V_φ is essentially the same as naïve translation in the space V_{a_φ}. For example, if $\varphi = \beta^n$ is the B-spline of degree n (and order $L = n + 1$), then $a_\varphi = \beta^{2n+1}$ is the B-spline of degree $2n + 1$ (and order $L = 2n + 2$). Therefore, in this case least-squares translation is equivalent to performing the naïve translation in a space that has twice the approximation order.

4.5 Least-squares downscaling

Image resizing involves resampling its signal at a different rate. An extremely naïve downscaling algorithm is to sample the reconstructed function with a different period s to obtain the sequence

$$[\![\tilde{f}\,]\!]_{\frac{1}{s}} = [\![\tilde{f}_s]\!], \tag{48}$$

where $\tilde{f}_s \stackrel{\text{def}}{=} \tilde{f}(\cdot/s)$ denotes the scaled version of $\tilde{f}$. Here aliasing is a great concern, because we are likely sampling $\tilde{f}_s$ below its Nyquist rate. To address this, one should instead apply an appropriately scaled analysis filter $\frac{1}{s}\overline{\psi}_s = \boldsymbol{p} *_s \psi_s$, to produce

$$\tfrac{1}{s}[\![\tilde{f} * (\overline{\psi}_s)^\vee]\!]_s. \tag{49}$$

Here, the factor $\frac{1}{s}$ simply renormalizes the scaled filter. When s is an integer, we usually rely on a discrete approximation:

$$[\![\tilde{f} * (\overline{\psi}_s)^\vee]\!]_s \approx [\![[\![\tilde{f}\,]\!] * [\![(\overline{\psi}_s)^\vee]\!]]\!]_s = \boldsymbol{p} * [\![[\![\tilde{f}\,]\!] * [\![(\psi_s)^\vee]\!]]\!]_s \tag{50}$$

As in the previous section, Unser et al. [1995a] present a least-squares construction that operates in the continuous domain. The idea is to compute the least-squares scaled version of $\tilde{f}$ by taking its orthogonal projection from space V_φ to space $V_{\varphi,s}$. Let $(\varphi_s)^\circ$ denote the dual of φ_s.[1] The orthogonal projection $P_{\varphi_s}\tilde{f} = \boldsymbol{c}_s *_s \varphi_s$ can be computed as follows:

$$\boldsymbol{c}_s = [\![\tilde{f} * ((\varphi_s)^\circ)^\vee]\!]_s = [\![\boldsymbol{c} * \varphi * ((\varphi_s)^\circ)^\vee]\!]_s \tag{51}$$

$$= [\![\boldsymbol{c} * \varphi * ([\![a_{\varphi_s}]\!]_s)^{-1} *_s (\varphi_s)^\vee]\!]_s \qquad \text{by (27)} \tag{52}$$

$$= ([\![a_{\varphi_s}]\!]_s)^{-1} * [\![\boldsymbol{c} * \varphi * (\varphi_s)^\vee]\!]_s \qquad \text{by (5)} \tag{53}$$

[1] Note that $(\varphi(\cdot/s))^\circ = \frac{1}{s}\mathring{\varphi}(\cdot/s)$.

and since $\llbracket a_{\varphi_s} \rrbracket_s = s \llbracket a_\varphi \rrbracket$,

$$\boldsymbol{c}_s = \tfrac{1}{s} \llbracket a_\varphi \rrbracket^{-1} * \left\llbracket \boldsymbol{c} * \varphi * (\varphi_s)^\vee \right\rrbracket_s \tag{54}$$

$$= \tfrac{1}{s} \llbracket a_\varphi \rrbracket^{-1} * \llbracket \boldsymbol{c} * a_{\varphi,\varphi_s} \rrbracket_s, \tag{55}$$

where a_{φ,φ_s} is the (continuous) cross-correlation of φ and φ_s. In words, convolve $\boldsymbol{c}$ with a_{φ,φ_s}, sample with step s, and apply the discrete correction filter $\frac{1}{s} \llbracket a_\varphi \rrbracket^{-1}$ to the result.

Note that since the mixed convolution and the sampling operation use different step sizes in the term $\llbracket \boldsymbol{c} * a_{\varphi,\varphi_s} \rrbracket_s$ of (55), relation (5) does not apply. Therefore, in the general case we may have to evaluate the cross-correlation function a_{φ,φ_s} at the arbitrary positions $s\,i - j$, with $i, j \in \mathbb{Z}$. Fortunately, when $s \in \mathbb{Z}$ (corresponding to downscaling by an integer factor), the operation becomes much simpler: compute $\boldsymbol{c} * \llbracket a_{\varphi,\varphi_s} \rrbracket$ and decimate the results by s. Better yet, we can compute only the elements in the discrete convolution that remain after the integer decimation operation. Either way, $\llbracket a_{\varphi,\varphi_s} \rrbracket$ has a compact support and the few required nonzero elements can be precomputed.

When general scaling factors are necessary, certain approximations can make the problem more tractable (though still not suitable for real-time applications). Restricting φ to the family of B-splines, Unser et al. [1995a] provide explicit formulas for the piecewise-constant $\varphi = \beta^0$ and piecewise-linear $\varphi = \beta^1$ cases, and show that for higher orders the cross-correlation function a_{β^n,β^n_s} quickly converges to a Gaussian. In contrast, Lee et al. [1998] approach the problem from the oblique projection perspective. The idea is to reconstruct $\tilde{f}$ with φ and then analyze it with a simpler $(\psi_s)^\vee$. Since the target reconstruction space is spanned by φ_s, the correction filter $\llbracket a_{\psi_s,\varphi_s} \rrbracket^{-1}$ completes the oblique projection. Simple and efficient algorithms are provided for $\psi_s = \beta^0_s$ piecewise-constant and $\varphi = \beta^n$ B-splines.

5 New algorithms

We apply the theory of generalized sampling to other scenarios in graphics. We first describe three new algorithms to improve and simplify scene antialiasing based on supersampling, and compare their results. We then describe an algorithm for high-quality image downscaling. A common theme in all these techniques is to perform signal analysis using a simple kernel, and then modify the sampled signal *a posteriori*.

5.1 Least-squares supersampling

We have already seen that the best approximation for a signal $f \in L_2$ in space $V_\varphi \subset L_2$ is given by its orthogonal projection $P_\varphi f$. Unlike in most other signal-processing applications, in rendering we have access to the continuous function f. Therefore we can accurately compute this orthogonal projection, and sample the result to form an image. The direct approach would be:

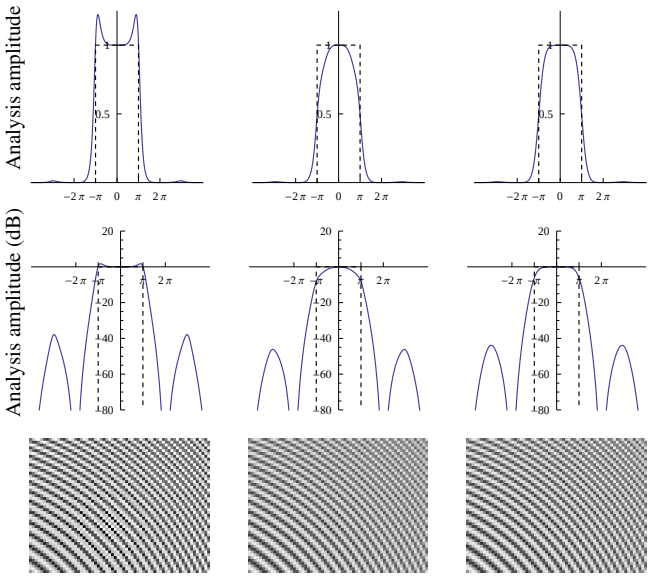
$$\llbracket P_\varphi f \rrbracket = \left\llbracket \llbracket f * \mathring{\varphi}^\vee \rrbracket * \varphi \right\rrbracket. \tag{56}$$

While φ is typically chosen to have local support, its dual $\mathring{\varphi}$ usually has infinite support. Therefore, direct evaluation of (56) requires a continuous integral over a large extent, which can be impractical. Fortunately, we can rewrite (56) as:

$$\llbracket P_\varphi f \rrbracket = \llbracket f * \mathring{\varphi}^\vee \rrbracket * \llbracket \varphi \rrbracket = \left\llbracket f * \llbracket a_\varphi \rrbracket^{-1} * \varphi^\vee \right\rrbracket * \llbracket \varphi \rrbracket \tag{57}$$

$$= \llbracket f * \varphi^\vee \rrbracket * \llbracket a_\varphi \rrbracket^{-1} * \llbracket \varphi \rrbracket. \tag{58}$$

In words, we filter with the primal (compact and perhaps even non-negative) and then apply a discrete correction filter. This correction is the composition of $\llbracket a_\varphi \rrbracket^{-1}$ (the inverse of a compactly supported filter), and $\llbracket \varphi \rrbracket$ (compactly supported). Therefore, the correction step can be computed efficiently with the techniques of section 6.



(a) Least-squares: $\beta^3 * \llbracket a_{\beta^3} \rrbracket^{-1} * \llbracket \beta^3 \rrbracket$ (b) Consistent: $\beta^e * \llbracket a_{K,\beta^3} \rrbracket^{-1} * \llbracket a_K \rrbracket$ (c) Generalized: $\beta^3 * \llbracket \beta^3 \rrbracket^{-1}$

Figure 6: Frequency response of the effective analysis filter $\overline{\psi}$ for three supersampling strategies. (a) The least-squares strategy of section 5.1 magnifies high-frequencies and is aliased. (b) The consistent strategy of section 5.2 produces similar results to the Keys kernel as expected (compare the spectrum to that of figure 10d). (c) The generalized strategy of section 5.3 produces the best results.

Unfortunately, using the orthogonal projection can be perceptually unsatisfactory. This should come as no surprise. The orthogonal projection of a discontinuous signal f into the space of bandlimited functions V_{sinc}, for example, contains strong undesirable ringing artifacts due to the Gibbs phenomenon (see figure 6). Since most rendering applications encounter discontinuous signals, this serious concern motivates the following alternative techniques.

5.2 Consistent supersampling

Let us consider the case that we would like to supersample with an analysis filter ξ, but for efficiency reasons we prefer to use a simpler analysis filter ψ. Discretization with the analysis filter $\xi^\vee$ can be interpreted as the orthogonal projection into space V_ξ, where the resulting discrete sequence is to be reconstructed with $\overline{\varphi} = \xi^\circ$, the dual of ξ. We can use the machinery of consistent sampling (section 4.2) to sample the oblique projection from V_ψ into V_{ξ°:

$$\llbracket P_{\xi^\circ \perp \psi} f \rrbracket = \llbracket f_\psi \rrbracket * \boldsymbol{q}, \quad \text{where} \tag{59}$$

$$\boldsymbol{q} = \llbracket a_{\xi^\circ,\,\psi} \rrbracket^{-1} = \llbracket \xi^\circ * \psi^\vee \rrbracket^{-1} = \left\llbracket \llbracket a_\xi \rrbracket^{-1} * \xi * \psi^\vee \right\rrbracket^{-1} \tag{60}$$

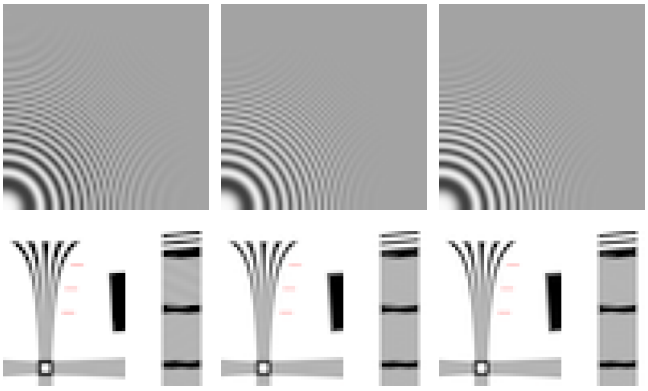
$$= \llbracket a_\xi \rrbracket * \llbracket \xi * \psi^\vee \rrbracket^{-1} = \llbracket a_\xi \rrbracket * \llbracket a_{\xi,\psi} \rrbracket^{-1}. \tag{61}$$

A special case worthy of mention is when $\psi = \beta^n$ is a member of the B-spline family. One reason we may prefer β^n as an analysis filter is that it is a probability density function (i.e., non-negative). Moreover it is trivial to draw random samples from β^n. Because β^n is the n-fold convolution $\beta^0 * \beta^0 * \cdots * \beta^0$, it is the probability density function of the sum of $n+1$ random variables distributed as β^0, which itself is just uniform distribution in the interval $[-\frac{1}{2}, \frac{1}{2}]$.

Now assume $\xi = K$ is the Keys (Catmull-Rom) kernel. Using the consistent supersampling strategy, we can filter with β^3 then apply the discrete correction filter

$$\boldsymbol{q} = \llbracket a_K \rrbracket * \llbracket a_{K,\beta^3} \rrbracket^{-1}. \tag{62}$$

Once again, with the help of section 6, this correction step can be implemented efficiently.



(a) Traditional Keys (Catmull-Rom) (b) Consistent: $\beta^3 * [a_{K,\beta^3}]^{-1} * [a_K]$ (c) Generalized: $\beta^3 * [\beta^3]^{-1}$

Figure 7: Comparisons of supersampling results using various strategies, on a zonal plate function and resolution chart.

5.3 Generalized supersampling

The trick we used to filter with the dual is of course a special case of a more general property that lets us indirectly sample a function f with any of the equivalent generating functions $\overline{\psi} = \boldsymbol{p} * \psi$ of a signal subspace:

$$[f * \overline{\psi}^\vee] = [f * (\boldsymbol{p} * \psi)^\vee] = [f * \boldsymbol{p}^\vee * \psi^\vee] = [f * \psi^\vee] * \boldsymbol{p}^\vee. \tag{63}$$

Motivated by the fact that the step response of high-quality interpolation strategies contains mild ringing artifacts (see figure 12), we investigate the quality of sampling with cardinal B-splines filters. In particular, the choice $\overline{\psi} = (\beta^3)_{\text{int}}$ leads to particularly good results (see section 5.4) while requiring a very simple correction filter:

$$\left[f * \left((\beta^3)_{\text{int}}\right)^\vee\right] = [f * (\beta^3)^\vee] * [\beta^3]^{-\vee}. \tag{64}$$

In other words, we simply set $\psi = \beta^3$ and $\boldsymbol{q} = [\beta^3]^{-\vee}$. This was in fact the technique used to produce the result in figure 4.

5.4 Supersampling experiments

Figure 7 compares results using different supersampling strategies on the zonal plate function $f = \sin(x^2 + y^2)$ as in Mitchell and Netravali [1988], as well as close-ups of a vector graphics ISO 12233 test chart. The quality of the results are similar. Traditional Keys filtering shows slightly more aliasing.

5.5 Generalized downscaling

Just as in the case of least-squares supersampling, the least-squares downscaling technique of section 4.5 gives rise to ringing artifacts (figure 8a,b). We may therefore wish to take advantage of the perceptually superior filtering properties of an equivalent generating function $\overline{\psi} = \boldsymbol{p} * \psi$. To do that, we follow the steps of section 4.5. In words, we reconstruct a signal $\tilde{f}$ in V_φ, and then sample $\tilde{f}_{\psi_s}$. In this derivation, let $\boldsymbol{f}$ denote our discrete input, so that $\tilde{f} = \boldsymbol{f} * \overline{\varphi}$, with $\overline{\varphi} = \boldsymbol{r} * \varphi$. It follows that:

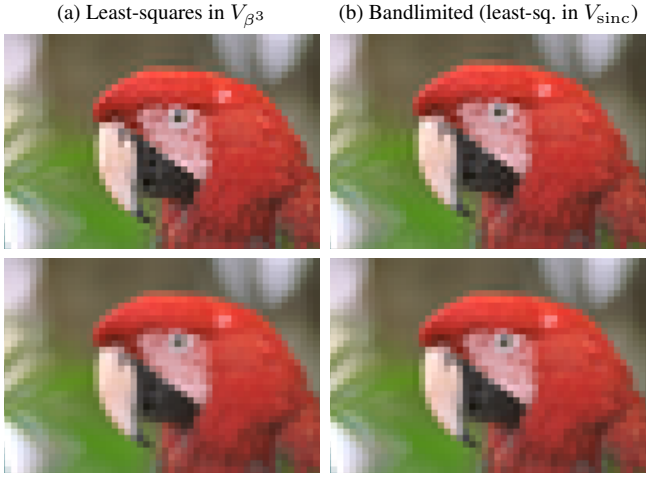
$$[\tilde{f}_{\overline{\psi}_s}]_s = [\tilde{f} * (\tfrac{1}{s}\,\overline{\psi}_s)^\vee]_s = [\boldsymbol{f} * \overline{\varphi} * (\tfrac{1}{s}\,\psi_s)^\vee]_s \tag{65}$$

$$= [\boldsymbol{f} * \overline{\varphi} * (\tfrac{1}{s}\,\boldsymbol{p} *_s \psi_s)^\vee]_s = \tfrac{1}{s}\,\boldsymbol{p}^\vee * [\boldsymbol{f} * \overline{\varphi} * (\psi_s)^\vee]_s \tag{66}$$

$$= \tfrac{1}{s}\,\boldsymbol{p}^\vee * [\boldsymbol{f} * \boldsymbol{r} * \varphi * (\psi_s)^\vee]_s \tag{67}$$

$$= \tfrac{1}{s}\,\boldsymbol{p}^\vee * [\boldsymbol{f} * \boldsymbol{r} * a_{\varphi,\psi_s}]_s. \tag{68}$$

Equation (68) describes a family of downscaling algorithms parametrized by the choices of the analysis and reconstruction kernels $\overline{\psi}$ and $\overline{\varphi}$. For example, setting $\overline{\psi} = \overline{\varphi} = (\beta^3)_{\text{int}}$, and specializing to $s = 2$, we reach the following algorithm for the recursive



(a) Least-squares in V_{β^3} (b) Bandlimited (least-sq. in V_{sinc}) (c) Using $\psi = K$ (Keys) (d) Using $\overline{\psi} = \overline{\varphi} = (\beta^3)_{\text{int}}$

Figure 8: Comparison of downscaling strategies. The images show mipmap level 3 (corresponding to downscaling by $s = 8$). (a,b) Least-squares downscaling (section 4.5) in V_{β^3} and V_{sinc} (bandlimited scaling in frequency space) are the sharpest, but contain ringing artifacts. (c) The popular Keys (Catmull-Rom) kernel performs well. (d) The images produced by our generalized downscaling strategy in equation (69) are sharper and show negligible ringing.

Figure 9: Comparison of aliasing characteristics of two fast downscaling strategies. Given the input image (left), downscaling by a factor of 8 with the traditional Keys filter (70) results in more aliasing (middle) than with generalized downscaling (71) (right).

generation of level i in an image pyramid, starting from an input image $\boldsymbol{f}_0$:

$$\boldsymbol{f}_i = \tfrac{1}{2}[\beta^3]^{-\vee} * \left[\boldsymbol{f}_{i-1} * [\beta^3]^{-1} * [a_{\beta^3,\beta_2^3}]\right]_2. \tag{69}$$

Here, sequence $[a_{\beta^3,\beta_2^3}]$ is symmetric, has support 12, and is used in a direct convolution. The remaining operations are convolutions with the inverse of kernel $[\beta^3]$, which is symmetric and has support 3. Section 6 shows how to perform these operations efficiently.

Figure 8 compares the result of (69) with traditional Keys (Catmull-Rom) downscaling, i.e. setting $\overline{\psi} = K$ in equation (49) to obtain

$$\boldsymbol{f}_i = \tfrac{1}{2}\left[\boldsymbol{f}_{i-1} * [(K_2)^\vee]\right]_2. \tag{70}$$

The generalized downscaling result is slightly sharper.

In real-time applications (such as mipmap construction), we recommend generalized downscaling with $\overline{\psi} = S^3$ (the cubic interpolator of Schaum [1993]) and $\overline{\varphi} = \beta^0$ (nearest-neighbor). The resulting sequence $[a_{S^3,\beta_2^0}] = \frac{1}{192}[\text{-7,-9,53,155,155,53,-9,-7}]$ has only 8 nonzero elements and there are no inverse convolutions:

$$\boldsymbol{f}_i = \tfrac{1}{2}\left[\boldsymbol{f}_{i-1} * [a_{S^3,\beta_2^0}]\right]_2. \tag{71}$$

Thus it is as efficient as traditional Keys downscaling (70), and as shown in figure 9, it aliases less.

5.6 Summary of algorithms

The following table provides a summary of the generalized sampling algorithms. It shows the setting in which they are applicable, and the associated discrete transforms $\boldsymbol{q}$.

Algorithm	Setting	Discrete correction $\boldsymbol{q}$
Orth. projection	$\psi = \varphi$	$\left[a_\varphi\right]^{-1}$
Consist. sampling	ψ, φ given	$\left[a_{\psi,\varphi}\right]^{-1}$
Interpolation	$\psi = \delta$?	$\left[\varphi\right]^{-1}$
L.S. translation	$\left[P_\varphi(\tilde{f} * \delta_\tau)\right]$	$\left[a_\varphi\right]^{-1} * \left[\delta_t * a_\varphi\right] * \left[\varphi\right]$
L.S. downscaling†	$\left[P_{\varphi_s} \tilde{f}\right]$	$s\left[a_\varphi\right]^{-1} * \left[\varphi\right]$
L.S. supersampling	$\left[P_\varphi f\right]$	$\left[a_\varphi\right]^{-1} * \left[\varphi\right]$
Consist. supersampling‡*	$\left[P_{\xi \circ \perp \psi} f\right]$	$\left[a_\xi\right] * \left[a_{\xi,\psi}\right]^{-1}$
Gen. supersampling‡	$\left[f * \overline{\psi}^\vee\right]$	$\boldsymbol{p}^\vee, \ \overline{\psi} = \boldsymbol{p} * \psi$
Gen. downscaling†	$\left[\tilde{f} * (\overline{\psi}_s)^\vee\right]_s$	$\frac{1}{s}\boldsymbol{p}^\vee, \ \overline{\psi} = \boldsymbol{p} * \psi$

†The downscaling operations also involve an additional filter before downsampling and correction. ‡In the supersampling algorithms, the result of the discrete correction $\boldsymbol{q}$ is the sampled reconstruction $\left[\tilde{f}\right]$ rather than the sequence $\boldsymbol{c}$. *The function $\xi \notin V_\psi$ is the desired analysis filter.

6 Efficient inverse convolution

For mathematical convenience, discrete convolutions are defined over infinite sequences. In practice however, we always deal with finite sequences. For the purposes of this paper, when we compute a convolution $\boldsymbol{f} = \boldsymbol{b} * \boldsymbol{c}$, the sequences either have finite support or we restrict our attention to an n-element span.

Say $\boldsymbol{b}$ has finite support and $\boldsymbol{c_n}$ is an n-element span of $\boldsymbol{c}$. To obtain an n-element output vector $\boldsymbol{f_n} = \boldsymbol{b} * \boldsymbol{c_n}$, we must specify what happens past the boundaries of $\boldsymbol{c_n}$. There are several alternatives. The simplest is to assume that $\boldsymbol{c}$ is zero outside of $\boldsymbol{c_n}$ (zero padding). A better option is to interpret $\boldsymbol{c_n}$ as one of the periods of $\boldsymbol{c}$, which is assumed to be periodic[2]. An even better option, to avoid discontinuities at the boundaries, is to work with even-periodic extensions[3]. For example, the even-periodic extension of a 4-element sequence $\boldsymbol{c}_4 = \left[1, 2, 3, 4\right]$ is:

$$\boldsymbol{c} = \left[\ldots, 3, 4, 4, 3, 2, 1, \underbrace{1, 2, 3, 4}_{\boldsymbol{c}_4}, 4, 3, 2, 1, 1, 2 \ldots\right].$$

In any case, discrete convolution by $\boldsymbol{b}$ is a linear operator and as such can be expressed in matrix form:

$$\boldsymbol{f_n} = \boldsymbol{b} * \boldsymbol{c_n} = B_n \boldsymbol{c_n}.$$

This observation clarifies the meaning of the inverse operator:

$$\boldsymbol{c_n} = \boldsymbol{b}^{-1} * \boldsymbol{f_n} = B_n^{-1} \boldsymbol{f_n}.$$

In other words, inverting a discrete convolution is equivalent to solving a linear system. Fortunately, matrix B has regular structure. For even-periodic extensions, it is *almost* Toeplitz. Efficient algorithms exist to solve the associated linear systems [Boisvert 1991]. Here we describe an algorithm adapted from Malcolm and Palmer [1974] for the common case where $\boldsymbol{b_3} = \left[p, q, p\right]$ is symmetric with three nonzero elements. The resulting even-periodic matrix is

$$B = \begin{bmatrix} p+q & p & & & & \\ p & q & p & & & \\ & \ddots & \ddots & \ddots & & \\ & & p & q & p & \\ & & & p & q+p \end{bmatrix},$$

[2]Periodicity is used by circular convolutions, and is implied by the DFT.

[3]Even-periodicity is implied by the type-II DCT. It is available as the *mirror* texture address mode in Direct3D, and as the *reversed* or *symmetric* array padding strategies in Mathematica or Matlab, respectively.

and its LU-decomposition takes the form

$$B = p \begin{bmatrix} 1 & & & & \\ l_0 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_{n-3} & 1 & \\ & & & l_{n-2} & 1 \end{bmatrix} \begin{bmatrix} l_0^{-1} & 1 & & & \\ & l_1^{-1} & 1 & & \\ & & \ddots & \ddots & \\ & & & l_{n-2}^{-1} & 1 \\ & & & & v \end{bmatrix}.$$

Interestingly, when $|q| > 2|p|$ (i.e., when B is diagonally dominant), we do not need to compute or store the entire factorization. This is because the sequence $\left[l_0, l_1, \ldots\right]$ quickly converges to a limit $(l_\infty = (q - \sqrt{q^2 - 4p^2})/2p)$, as it forms a convergent sequence of periodic continued fractions. Thus we can precompute and store the prefix of the sequence l until convergence. Moreover the sequence values are independent of n. The last diagonal element $v = 1 + 1/l_\infty$ of U must be handled separately due to boundary effects.

As an example, we provide the LU-decomposition arising from the cubic B-spline interpolation problem, where $\boldsymbol{b_3} = \frac{1}{6}\left[1, 4, 1\right]$:

```
const float l[] = {
  0.2f, 0.26315789f, 0.26760563f, 0.26792453f,
  0.26794742f, 0.26794907f, 0.26794918f, 0.26794919f };
const int ln = sizeof(l)/sizeof(l[0]);
const float p = 0.16666667f, v = 4.73205078f;
```

Only 8 coefficients are needed before the sequence l converges to single-precision floating-point. The forward and back-substitution are so simple that we include the source code for completeness:

```
void solve(float* f, int n) {
  // First pass: solve L c' = f in-place
  for (int i=1; i<ln; i++) f[i] = f[i] - l[i-1]*f[i-1];
  const float l_∞ = l[ln-1];
  for (int i=ln; i<n; i++) f[i] = f[i] - l_∞*f[i-1];
  // Second pass: solve U c = c' in-place
  f[n-1] = p^-1/v*f[n-1];
  for (int i=n-2; i>=ln-1; i--) f[i] = l_∞*(p^-1*f[i]-f[i+1]);
  for (int i=ln-2; i>=0; i--) f[i] = l[i]*(p^-1*f[i]-f[i+1]);
}
```

From the algorithm, it is clear that $\boldsymbol{c}_n = \boldsymbol{b_3}^{-1} * \boldsymbol{f}_n$ can be computed with only $3n$ products and $2n$ additions. Coincidentally, this is the same cost as computing $\boldsymbol{f}_n = \boldsymbol{b_3} * \boldsymbol{c_n}$. An important difference is that each element in $\boldsymbol{b_3} * \boldsymbol{c_n}$ can be computed independently, whereas the computation $\boldsymbol{b_3}^{-1} * \boldsymbol{f_n}$ involves chains of output dependencies. If one can make do with a scaled version $p\,\boldsymbol{c}_n$ of the solution, such as for the image between the horizontal and vertical solvers, then the computation requires n fewer products.

6.1 Connection to recursive digital filters

It is also possible to approach discrete inverse convolution from the digital filtering perspective [Hou and Andrews 1978; Unser et al. 1991]. The basic idea is to note that $\boldsymbol{b}$ is an all-zero filter. The inverse filter $\boldsymbol{b}^{-1}$ is an all-pole filter that can be factored into a causal part and an anticausal part, each to be realized as a recursive filter. Unsurprisingly, these are largely equivalent to the LU-decomposition approach we describe (L^{-1} playing the role of the causal part, U^{-1} the anticausal part). The connection is clearer when the input size is conceptually infinite, in which case B becomes bi-infinite Toeplitz, and only the limits l_∞ and d_∞ appear in L and U, which become Toeplitz themselves [McGill 1991].

With a pair of recursive digital filters, Unser et al. [1991] present an algorithm for solving $\boldsymbol{b_3}^{-1} * \boldsymbol{f_n}$ using only $2n$ products and $2n$ additions (i.e., one fewer product per element), to produce a scaled solution $p\,\boldsymbol{c_n}$. As we saw in the previous section, this insight is not specific to the recursive-filter interpretation.

In our opinion, viewing convolution as a matrix operation has a number of benefits. The finite matrix B includes boundary conditions explicitly. Also, it lets us take advantage of the vast literature on parallel solution of banded linear systems.

6.2 Connection to deconvolution

Many physical processes such as motion blur and defocus can be formulated as convolution between an ideal image and a kernel which may be spatially varying and even unknown. The problem of recovering the ideal image is referred to as deconvolution. In our context we use the term inverse convolution to avoid confusion, because the operation may be used to transform an ideal image into a new basis, as in figure 3.

6.3 Parallelization

When dealing with images, we solve one linear system per row, and then one per column. Because a typical image has at least 512 rows and columns, processing all rows concurrently and then all columns concurrently provides significant parallelism for multicore CPUs.

Nonetheless, we use an additional strategy, recursive doubling, to expose parallelism within each linear system. The idea is as follows. The dependency of each output value on the previous one in a tight loop such as

```
for (int i = 1; i < n; i++) f[i] -= l∞*f[i-1];
```

gives little freedom to the processor. We can rewrite the loop to evaluate small *batches* of values per iteration. Here is an example using batches of two values:

```
float p = 0, l = l∞, l² = l*l;
for (int i = 0; i <= n-2; i += 2) {
    float a = f[i+0], b = f[i+1];
    float a1 = a-l*p, b1 = b-l*a;
    f[i+0] = a1; p = f[i+1] = b1+l²*p;
}
```

Within each batch, the algorithm uses the recursive-doubling strategy of Stone [1973] to expose instruction-level parallelism. This enables a superscalar core to execute multiple arithmetic operation per clock-cycle and to reorder instructions to hide memory latency. The performance improvement can be significant (up to 50% on smaller images).

Our CPU implementation of discrete inverse convolution operates in-place on a single-precision floating-point image. It makes two streaming passes over the image data. The first pass sweeps down over the image, applying the substitution matrices L and U over the image rows (left-to-right and right-to-left respectively). In addition, as it sweeps down, it also applies the forward-substitution matrix L over the resulting image columns. (Interleaving the horizontal computation with the vertical sweep reduces total bandwidth to main memory.) The second pass sweeps through the image from bottom to top, applying the back-substitution matrix U over the image columns. For both the vertical and horizontal computations, we distribute bands of image columns or rows over several parallel threads for multiprocessing. And, pixels are processed in groups of 4 with vector SSE instructions, using recursive doubling to expose parallelism. For a 3-channel 1-megapixel image, and the $\boldsymbol{q} = \lfloor \beta^3 \rceil^{-1}$ cubic B-spline interpolation kernel, we obtain a processing rate of 200 megapixels per second on a 2.66GHz Intel Core i7 quad-core processor.

6.4 Range constraints

An important consideration is that a signal f with values in a bounded range $[0, 1]$ may lead to a solution vector $\boldsymbol{c}$ with values outside this range, especially when f has high-frequency content. While this is likely acceptable for floating-point or HDR image representations, it is a concern for common 8-bit fixed-point formats.

To enable quantization of $\boldsymbol{c}$ within an 8-bit image format, we set its quantized range to a somewhat larger interval $[-0.5, 1.5]$, and solve the constrained optimization

$$\arg\min_{\boldsymbol{c}_n} \|B_n \boldsymbol{c}_n - \boldsymbol{f}_n\|^2 \text{ s.t. } -0.5 \le \boldsymbol{c}_n[k] \le 1.5,\ k = 1..n. \quad (72)$$

This is a least squares problem with simple bounds constraints, and can be solved efficiently, for instance with the Matlab function lsqlin. It may be possible to implement this constrained optimization in real-time, but we have not yet explored this.

One might expect that the sampled reconstruction $\lfloor \tilde{f} \rceil$ would only interpolate a given input f with 7 bits of precision, because the quantized $\boldsymbol{c}$ values now span twice the range. Surprisingly, this is not the case for natural images. With the bicubic B-spline basis, each pixel is reconstructed by a weighted combination of 16 coefficients of $\boldsymbol{c}$. This combination effectively provides sub-quantum precision. Even with the simple constrained optimization (72), which is unaware of the subsequent quantization process, a reconstruction of the original image of figure 8 from an 8-bit-quantized vector $\boldsymbol{c}$ has an rms error of 0.10% and a maximum error of 1.2%, with over 93% of pixel values reproduced exactly. Intuitively, the reconstruction B-spline φ gives more precision to low frequencies.

As future work one could explore a combinatorial optimization to let the 8-bit-quantized $\boldsymbol{c}$ values approximate an image with more than 8 bits of precision. Such super-precision may be possible only in low-frequency regions, but these smooth regions are precisely where precision is most perceptible.

7 GPU reconstruction

GPU texture samplers have custom hardware to perform bilinear reconstruction, essentially at no cost. With traditional filtering, moving to higher-quality reconstruction incurs a large performance penalty because the dedicated bilinear units are unhelpful. For instance, implementation of the bicubic Mitchell-Netravali filter requires $4 \times 4 = 16$ texture reads, versus 1 read for bilinear filtering.

As shown by Sigg and Hadwiger [2005], even though the bicubic B-spline basis has the same 4×4 support, the fact that it is non-negative allows it to be evaluated by combining just 4 bilinear reads at appropriately computed locations. Therefore, the use of B-splines for reconstruction in the lower row of Figure 3 is a significant benefit. Ruijters et al. [2008] also describe a CUDA implementation of cubic B-spline evaluation in both 2D and 3D.

8 Experiments and analysis

To help sort through dozens of discretization and reconstruction strategies, we present quantitative and qualitative comparisons. Our reconstruction tests include traditional evaluations of the effect of repeated rotations and translations applied to an image. For discretization, we present qualitative comparisons based on conventional challenging test patterns. We also include a variety of animation tests revealing subtle artifacts that would otherwise be hard to measure quantitatively. Note that due to space constraints, we only include in the paper a small subset of the strategies tested. The remaining results can be found in the supplemental materials.

8.1 Frequency and transient responses

One of the most valuable tools in assessing the quality of analysis and reconstruction filters is in the form of plots of their frequency and transient behavior. In this section, we review how to extend this analysis to the generalized sampling pipeline.

Frequency response The frequency response of an analysis filter ψ reveals important information about the relationship between the input signal f and the filtered f_ψ. Conversely, the frequency response of a reconstruction kernel φ describes the connection between the discretized $\lfloor f_\psi \rceil$ and its reconstruction $\tilde{f}$.

To calculate the frequency response of a piecewise polynomial kernel, we can use the method of Thévenaz et al. [2000] or rely on symbolic manipulation programs. In the generalized sampling context, we must also take into account the effect of the discrete

filtering stage. This is a simple matter, since

$$\widehat{\boldsymbol{q} * \varphi} = \widehat{\boldsymbol{q}}\,\hat{\varphi} \qquad \text{and} \qquad \widehat{\boldsymbol{q}^{-1}} = 1/\widehat{\boldsymbol{q}}. \tag{73}$$

Usually, $\boldsymbol{q}$ can be factored into components that have compact support or compactly supported inverses, so that an explicit and finite expression for the DTFT of $\boldsymbol{q}$ comes directly from definition (15) and property (16). We illustrate with the B-spline family, for which the formulas are particularly simple, since

$$\widehat{\beta^n} = \operatorname{sinc}^{n+1}, \qquad \widehat{(\beta^n)_{\text{int}}} = (\operatorname{sinc}^{n+1}) / \widehat{\boldsymbol{[}\beta^n\boldsymbol{]}}, \tag{74}$$

$$\widehat{(\beta^n)^\circ} = (\operatorname{sinc}^{n+1}) / \widehat{\boldsymbol{[}a_{\beta^n}\boldsymbol{]}}, \qquad a_{\beta^n} = \beta^{2n+1}. \tag{75}$$

The frequency response of $(\beta^3)_{\text{int}}$, for example, is

$$\widehat{(\beta^3)_{\text{int}}} = \frac{\operatorname{sinc}^4}{\frac{1}{6}\widehat{\boldsymbol{[}1,4,1\boldsymbol{]}}} \;\Rightarrow\; \widehat{(\beta^3)_{\text{int}}}(\omega) = \frac{(\frac{\omega}{2})^{-4}\sin^4(\frac{\omega}{2})}{\frac{1}{6}\big(4+2\cos(\omega)\big)}. \tag{76}$$

Figures 10 show the amplitude responses of a variety of kernels. (The phase responses are all linear and have been omitted.) It has been shown that the spectrum of cardinal B-splines (and of their duals) converges to that of the ideal low-pass filter (shown in dashed lines) as n increases [Aldroubi and Unser 1994], and this is clearly visible from figures 10a–f. The plots also indicate that cubic B-spline interpolation should perform substantially better than the popular Mitchell-Netravali and Keys (Catmull-Rom) cubic kernels (compare with figures 10d–e), and even match the quality of Lanczos' windowed-sinc approximation with support $W = 6$. Using the same support, we could instead use the quintic B-spline interpolation and obtain even higher quality. These results are confirmed by the experiments in section 8.

The reconstruction response does not tell the whole story because we are often more interested in the relationship between the sampled and resampled signals $\boldsymbol{[}f_\psi\boldsymbol{]}$ and $\boldsymbol{[}\tilde{f}\,\boldsymbol{]}$. To qualify this *resampling response*, Parker et al. [1983] suggest evaluating the effect of the naïve translation operation for varying offsets τ (see section 4.4). Once again, we must take into account the discrete filter $\boldsymbol{q}$, so we analyze the spectrum of the discrete filter $\boldsymbol{q} * \boldsymbol{[}\delta_\tau * \varphi\boldsymbol{]}$. Since $\boldsymbol{[}\delta_\tau * \varphi\boldsymbol{]}$ has compact support, the DTFTs can again be computed explicitly from (15) and (16). To consolidate information of all offsets τ into a single plot, each plot shows a shaded the region between the minimum and maximum values over all offsets $\tau \in [-\frac{1}{2}, \frac{1}{2}]$. This means that a horizontal line at amplitude one is the ideal resampling amplitude response, whereas a horizontal line at zero is the ideal resampling phase error. The size and shape of the shaded regions therefore depict the extent to which each resampling strategy deviates from these ideals.

The results in figure 11 show that all resampling strategies attenuate and displace high frequency detail (at least for certain offsets). Unlike the interpolating strategies, Mitchell and Netravali [1988] reconstruction attenuates high frequencies even at zero offset. Also apparent is the improvement in the resampling spectrum of B-spline interpolation as the degree increases. Other conclusions are similar to those drawn from figure 10.

Transient response The impulse and step responses of a reconstruction strategy reveal transient properties that cannot be deduced from the frequency-space analysis just discussed. The impulse response is the result of reconstructing the unit impulse sequence $\boldsymbol{\delta}$, and depicts the kernel $\overline{\varphi} = \boldsymbol{q} * \varphi$ itself. The step response is the result of reconstructing the unit step sequence $\boldsymbol{[}\ldots, 0, 1, \ldots\boldsymbol{]}$ (i.e., a sharp edge).

Figure 12 shows the impulse and step responses of selected kernels. The most prominent transient artifacts visible in many plots is *ringing* — an oscillatory overshoot near image edges. Any kernel containing negative lobes produces some amount of ringing, and minimizing these artifacts was one of the goals in the design of the Mitchell-Netravali filter. The significant oscillations in the impulse and step response of figure 12 have often scared users away from B-spline interpolation, even though the amount of ringing seems to be comparable to the popular Lanczos reconstruction.

Here we note that properly antialiased images seldom contain sharp discontinuities such as the unit impulse and step. Instead, discontinuities are smoothed out into more gradual changes, such as the ramp in sequence $\boldsymbol{[}\ldots, 0, 0, \frac{1}{2}, 1, 1, \ldots\boldsymbol{]}$. As seen in figure 12, such milder transitions generate significantly smaller oscillations. This is why images reconstructed with B-spline interpolation do not suffer from the severe ringing artifacts one might expect.

8.2 Quantitative experiments

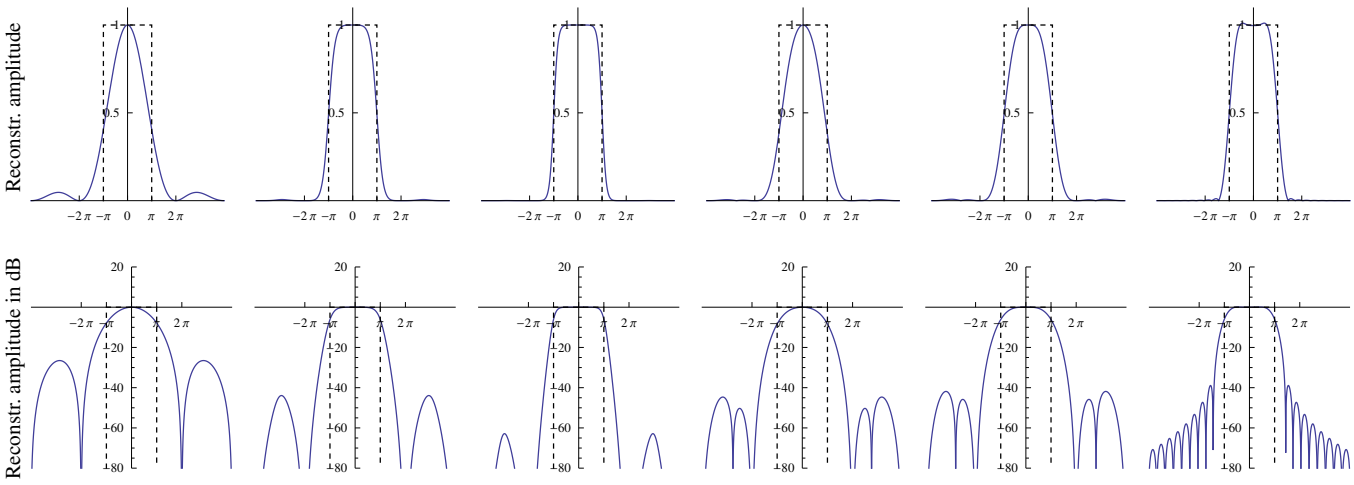
It is common practice in the literature to evaluate the quality of a reconstruction technique by analyzing the effect of repeated operations (such as translations and rotations). This type of test greatly magnifies any artifacts introduced by a given reconstruction scheme to more easily reveal which method performs best. Moreover, if we rotate or translate back to the initial alignment, the original image provides ground truth. The results in table 1 show reconstruction accuracy results, using the mean structural similarity (MSSIM) metric of Wang et al. [2004], which is closer to the perceptual characteristics of the human visual system than mean squared error. We use 32 successive translations, cycling over $\{_{(0.5,0.5),(0.5,-0.5),(-0.5,-0.5),(-0.5,0.5)}\}$, and 31 successive rotations by angle $2\pi/31$ about the image center. Four test images are taken from the Kodak lossless true color image suite, and "CIR" is the radial function $f = 1 + \cos(\frac{4\pi r}{5} 4^{-r/n})$. In supplemental material we provide a larger table with a superset of filters. We also include results of rms error (PSNR), in which the quality rankings of the filters are very similar.

8.3 Animation tests

In animation sequences, the effect of the resampling frequency response of section 8.1 is often visible. Phase errors cause high frequencies to oscillate in position, whereas amplitude errors cause high frequencies to oscillate in brightness. To demonstrate these effects, we use simple animation sequences in which each frame is the result of translating an input image around a circle with a 5 pixel radius, according to the naïve scheme of section 4.4. These results are available in the supplemental material.

9 Conclusions

Generalized sampling is an exciting approach for filtering signals in graphics. Our extensive analysis and experiments confirm that it enables significantly higher quality in upsampling and resampling operations. We have drawn from many concepts of generalized sampling to derive a variety of supersampling techniques, and shown that these offer comparable quality to the best conventional techniques, but with greater freedom in decoupling the integration filter from the desired spectral properties. Similarly, we have explored a new family of downscaling techniques, some of which outperform traditional algorithms in terms of blurring, ringing, and aliasing. Generalized sampling does require the solution of discrete inverse filters, but this problem can be solved efficiently and scales well to multicore architectures. One of our lasting contributions may be the new parameterless notation used throughout the paper, which algebraically joins discrete and continuous functions. By freeing us from tedious index manipulations, it lets us reason more intuitively about otherwise complex operations.



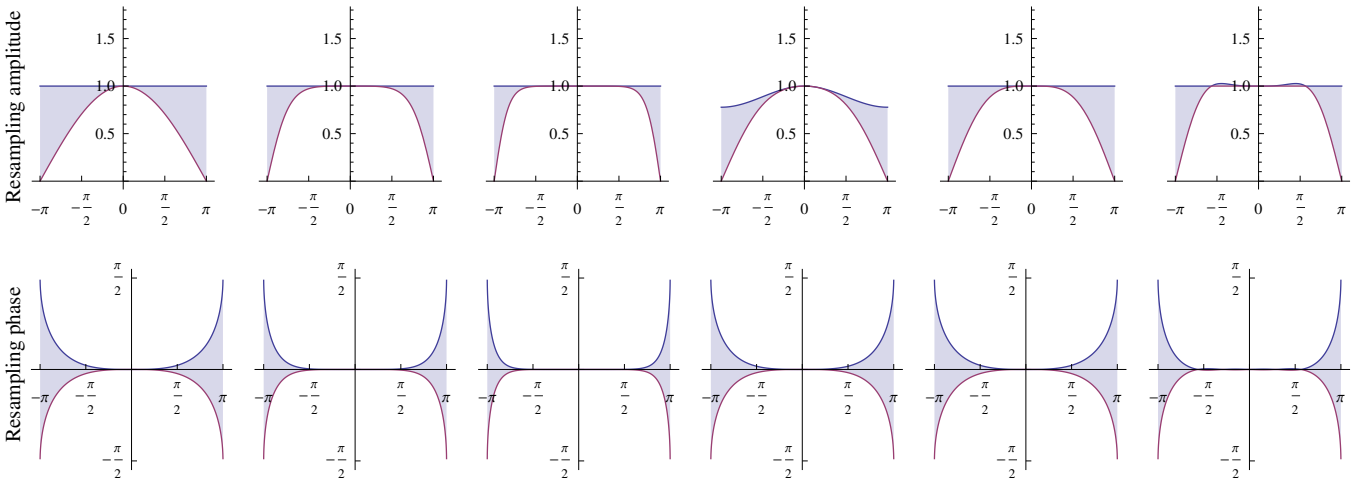
(a) $(\beta^1)_{\text{int}}$ (linear) (b) $(\beta^3)_{\text{int}}$ ($W = 4$) (c) $(\beta^5)_{\text{int}}$ ($W = 6$) (d) Mitchell ($W = 4$) (e) Keys ($W = 4$) (f) Lanczos ($W = 6$)

Figure 10: Reconstruction frequency responses. (a–c) The amplitude response of B-spline interpolation approaches that of the ideal low-pass filter as the degree increases. (b) $(\beta^3)_{\text{int}}$ outperforms the popular cubic kernels (d) [Mitchell and Netravali 1988] and (e) [Keys 1981], and has quality comparable to that of the more expensive Lanczos kernel (f). Using the same support, $(\beta^5)_{\text{int}}$ in (c) performs even better.

Table 1: Quantitative analysis of reconstruction quality. Kernel properties are degree N, width W, and approximation order L. The main columns report the mean structural similarity (MSSIM) between five reference images (circles of Figure 3, four Kodak benchmarks) and their reconstructions for three types of experiments (repeated translations or rotations, and single upsampling). The kernels are sorted in descending order of average quality across all experiments. The interpolating B-splines `bspline*i` consistently outperform the more traditional filters for the same N and W. The O-MOMS kernels `omoms*` offer even slightly higher quality but at the expense of differentiability. The quasi-interpolant `condat2`, which has degree 2 and support 3, also performs remarkably well.

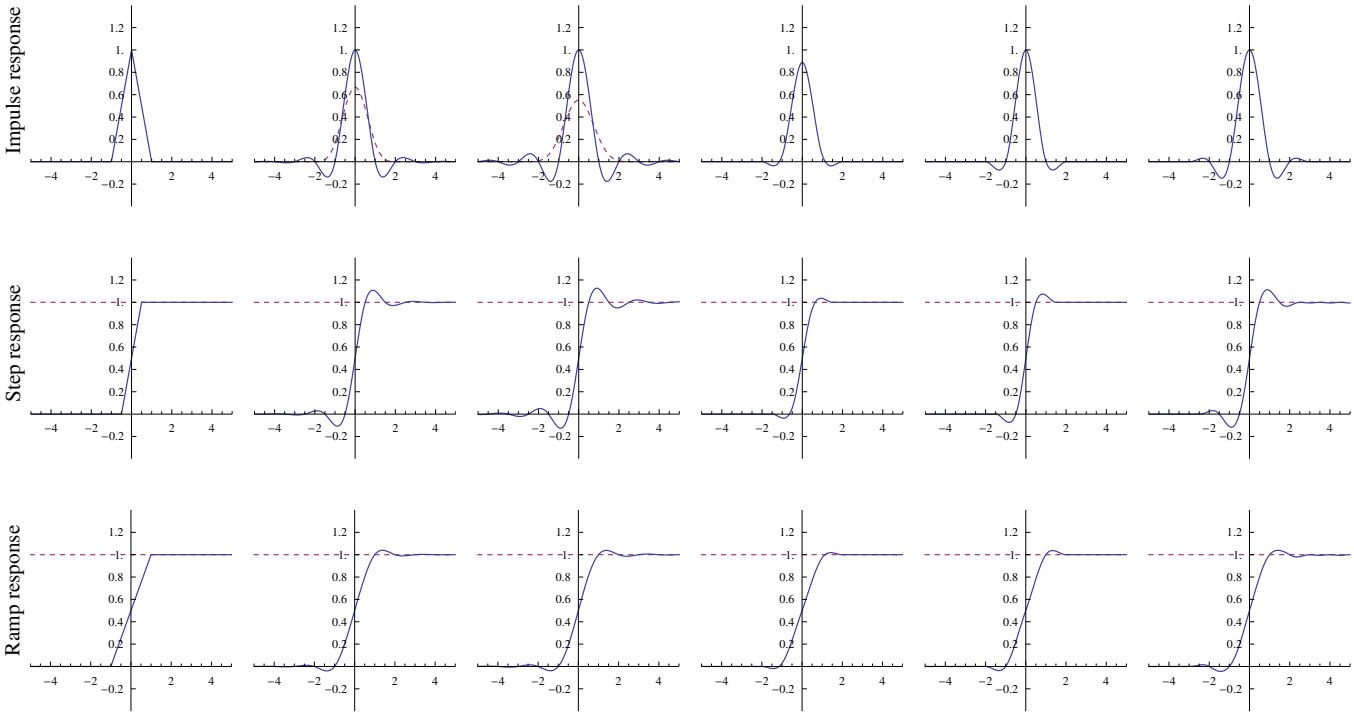
				Comparison against ground truth (MSSIM)											
	Properties			Repeated translations					Repeated rotations					Upscaling	Average
Kernel	N	W	L	CIR	K05	K08	K19	K23	CIR	K05	K08	K19	K23	CIR	AVG
`omoms5`	5	6	6	0.993	0.975	0.945	0.957	0.984	0.999	0.986	0.977	0.980	0.990	0.886	0.979
`bspline5i`	5	6	6	0.990	0.966	0.931	0.947	0.981	0.998	0.983	0.972	0.976	0.989	0.886	0.973
`omoms3`	3	4	4	0.981	0.949	0.905	0.929	0.975	0.997	0.980	0.968	0.973	0.987	0.886	0.964
`quasiblu35`	3	4	4	0.965	0.926	0.874	0.907	0.969	0.994	0.975	0.961	0.968	0.986	0.885	0.946
`condat3`	3	4	4	0.962	0.921	0.867	0.903	0.967	0.991	0.971	0.956	0.964	0.984	0.885	0.943
`hamming6`	–	6	1	0.960	0.918	0.862	0.901	0.966	0.977	0.956	0.937	0.951	0.979	0.885	0.941
`bspline3i`	3	4	4	0.948	0.904	0.846	0.888	0.963	0.977	0.958	0.938	0.952	0.980	0.885	0.935
`condat2`	2	3	3	0.930	0.884	0.822	0.872	0.957	0.989	0.967	0.951	0.961	0.983	0.884	0.927
`lanczos6`	–	6	1	0.964	0.788	0.787	0.827	0.909	0.987	0.959	0.946	0.958	0.978	0.884	0.910
`bspline2i`	2	3	3	0.901	0.854	0.787	0.848	0.949	0.955	0.939	0.916	0.936	0.975	0.884	0.906
`omoms2`	2	3	3	0.814	0.774	0.713	0.793	0.929	0.961	0.944	0.921	0.940	0.976	0.885	0.876
`schaum2`*	2	3	3	0.822	0.782	0.710	0.792	0.930	0.921	0.914	0.885	0.916	0.967	0.877	0.864
`lanczos4`	–	4	1	0.822	0.782	0.710	0.792	0.930	0.896	0.902	0.871	0.906	0.963	0.882	0.857
`keys`†	3	4	3	0.822	0.782	0.710	0.792	0.930	0.894	0.900	0.869	0.905	0.963	0.882	0.857
`schaum3`‡	3	4	4	0.822	0.782	0.710	0.792	0.930	0.876	0.891	0.858	0.897	0.960	0.883	0.852
`dalai1`	1	2	2	0.657	0.652	0.601	0.715	0.896	0.956	0. 938	0.915	0.936	0.974	0.865	0.828
`linrev`	1	2	2	0.686	0.667	0.587	0.710	0.898	0.960	0.926	0.903	0.924	0.960	0.864	0.822
`condat1`	1	2	2	0.651	0.648	0.597	0.713	0.895	0.947	0.933	0.909	0.931	0.972	0.866	0.824
`hamming4`	–	4	1	0.663	0.657	0.603	0.716	0.897	0.822	0.859	0.826	0.875	0.951	0.879	0.787
`mitchell`	3	4	2	0.581	0.599	0.554	0.685	0.881	0.625	0.761	0.733	0.810	0.921	0.881	0.715
`linear`	1	2	2	0.391	0.480	0.449	0.623	0.847	0.540	0.721	0.698	0.787	0.908	0.864	0.644
`nearest`	0	1	1	0.042	0.102	0.087	0.367	0.560	0.547	0.654	0.633	0.731	0.851	0.586	0.457

*Same as IMOMS-2. †Same as Catmull-Rom. ‡Same as IMOMS-3. The kernels `omoms*` are from [Blu et al. 2001]; `bspline*i` are the cardinal B-splines β_{int}; `schaum*` are from [Schaum 1993]; `linrev` is from [Blu et al. 2004]; `condat*` are from [Condat et al. 2005]; `dalai1` is from [Dalai et al. 2005].



(a) $(\beta^1)_{\text{int}}$ (linear) (b) $(\beta^3)_{\text{int}}$ $(W = 4)$ (c) $(\beta^5)_{\text{int}}$ $(W = 6)$ (d) Mitchell $(W = 4)$ (e) Keys $(W = 4)$ (f) Lanczos $(W = 6)$

Figure 11: Resampling frequency responses. The shaded regions indicate the range of amplitude and phase responses when resampling under all possible shifts of the input signal. These results corroborate those of figure 10. The cubic B-spline interpolation (b) performs better than other cubic kernels (d,e), and the quintic B-spline (c) performs better than Lanczos (f).



(a) $(\beta^1)_{\text{int}}$ (linear) (b) $(\beta^3)_{\text{int}}$ $(W = 4)$ (c) $(\beta^5)_{\text{int}}$ $(W = 6)$ (d) Mitchell $(W = 4)$ (e) Keys $(W = 4)$ (f) Lanczos $(W = 6)$

Figure 12: Transient responses. Any reconstruction kernel containing negative lobes results in a certain amount of ringing (b–f). Mitchell and Netravali [1988] designed their kernel to minimize such problems (d), and this can be seen in the comparison with the impulse and step responses of other kernels, in particular B-spline interpolation (b,c) and Lanczos (f). The ramp response, which is more characteristic of properly antialiased images, predicts significantly reduced ringing across all kernels.

References

ALDROUBI, A. and UNSER, M. 1994. Sampling procedures in function spaces and asymptotic equivalence with Shannon's sampling theory. *NFAO*, 15(1–2):1–21.

BLINN, J. F. 1989. Return of the jaggy. *IEEE CG&A*, 9(2):82–89.

BLU, T., THÉNAVAZ, P., and UNSER, M. 1999. Generalized interpolation: Higher quality at no additional cost. In *IEEE ICIP*, volume 3, pages 667–671.

BLU, T., THÉVENAZ, P., and UNSER, M. 2001. MOMS: Maximal-order interpolation of minimal support. *IEEE TIP*, 10(7):1069–1080.

BLU, T., THÉVENAZ, P., and UNSER, M. 2004. Linear interpolation revitalized. *IEEE TIP*, 13(5):710–719.

BLU, T. and UNSER, M. 1999. Quantitative Fourier analysis of approximation techniques: Part I—Interpolators and projectors. *IEEE TSP*, 47(10):2783–2795.

BOISVERT, R. F. 1991. Algorithms for special tridiagonal systems. *SIAM J. Sci. Stat. Comput.*, 12(2):423–442.

CATMULL, E. and ROM, R. 1974. A class of local interpolating splines. In *Computer Aided Geometric Design*, pages 317–326.

CONDAT, L., BLU, T., and UNSER, M. 2005. Beyond interpolation: optimal reconstruction by quasi-interpolation. In *IEEE ICIP*, volume 1, pages 33–36.

DALAI, M., LEONARDI, R., and MIGLIORATI, P. 2005. Efficient digital pre-filtering for least-squares linear approximation. In *VLBV LNCS 3893/2006*, pages 161–169.

DODGSON, N. A. 1997. Quadratic interpolation for image resampling. *IEEE TIP*, 6(9):1322–1326.

GERMAN, I. 1997. Short kernel fifth-order interpolation. *IEEE TSP*, 45(5):1355–1359.

GLASSNER, A. S. 1995. *Principles of Digital Image Synthesis, volumes 1 and 2*. Morgan Kauffmann.

HOU, H. S. and ANDREWS, H. C. 1978. Cubic splines for image interpolation and digital filtering. *IEEE ASSP*, 25(6).

HUMMEL, R. 1983. Sampling for spline reconstruction. *SIAM Journal on Applied Mathematics*, 43(2):278–288.

KEYS, R. G. 1981. Cubic convolution interpolation for digital image processing. *IEEE ASSP*, 29(6):1153–1160.

LEE, C., EDEN, M., and UNSER, M. 1998. High-quality image resizing using oblique projection operators. *IEEE TIP*, 7(5):679–692.

LEHMANN, T. M., GÖNNER, C., and SPITZER, K. 1999. Survey: Interpolation methods in medical image processing. *IEEE TMI*, 18(11):1049–1075.

MALCOLM, M. A. and PALMER, J. 1974. A fast method for solving a class of tridiagonal linear systems. *CACM*, 17(1):14–17.

MCGILL, K. C. 1991. A storage-efficient method for solving banded Toeplitz systems. *IEEE TSP*, 39(10):2351–2353.

MEIJERING, E. H. W. 2002. A chronology of interpolation: From ancient astronomy to modern signal processing. *Proceedings of the IEEE*, 90(3):319–342.

MEIJERING, E. H. W., NIESSEN, W. J., PLUIM, J. P. W., and VIERGEVER, M. A. 1999. Quantitative comparison of sinc-approximating kernels for medical image interpolation. In *MICCAI LNCS 1679/1999*, pages 210–217.

MEIJERING, E. H. W., NIESSEN, W. J., and VIERGEVER, M. A. 2001. Quantitative evaluation of convolution-based methods for medical image interpolation. *Medical Image Analysis*, 5(2):111–126.

MEIJERING, E. H. W., ZUIDERVELD, K. J., and VIERGEVER, M. A. 1999. Image reconstruction by convolution with symmetrical piecewise nth-order polynomial kernels. *IEEE TIP*, 8(2):192–201.

MITCHELL, D. P. and NETRAVALI, A. N. 1988. Reconstruction filters in computer graphics. *Computer Graphics (Proceedings of ACM SIGGRAPH 1988)*, 22(4):221–228.

PARK, S. K. and SCHOWENGERDT, R. A. 1983. Image reconstruction by parametric cubic convolution. *CVGIP*, 23(3):258–272.

PARKER, J. A., KENYON, R. V., and TROXEL, D. E. 1983. Comparison of interpolating methods for image resampling. *IEEE TMI*, MI-2(1):31–39.

PIXAR. 2005. *The RenderMan Interface*. version 3.2.1.

RUIJTERS, D., TER HAAR ROMENY, B. M., and SUETENS, P. 2008. Efficient GPU-based texture interpolation. *Journal of Graphics, GPU & Game Tools*, 13(4):61–69.

SCHAUM, A. 1993. Theory and design of local interpolators. *CVGIP*, 55(6):464–481.

SHANNON, C. E. 1949. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21.

SIGG, C. and HADWIGER, M. 2005. Fast third-order texture filtering. In *GPU Gems 2*, chapter 20, pages 313–329.

STONE, H. S. 1973. An efficient parallel algorithm for the solution of a tridiagonal linear system of equations. *JACM*, 20(1):27–38.

THÉVENAZ, P., BLU, T., and UNSER, M. 2000. Interpolation revisited. *IEEE TMI*, 19(17):739–758.

UNSER, M. 1996. Approximation power of biorthogonal wavelet expansions. *IEEE TSP*, 44(3):519–527.

UNSER, M. 2000. Sampling—50 years after Shannon. *Proceedings of the IEEE*, 88(4):569–587.

UNSER, M. and ALDROUBI, A. 1994. A general sampling theory for nonideal acquisition devices. *IEEE TSP*, 42(11):2915–2925.

UNSER, M., ALDROUBI, A., and EDEN, M. 1991. Fast B-spline transforms for continuous image representation and interpolation. *IEEE PAMI*, 13(3):277–285.

UNSER, M., ALDROUBI, A., and EDEN, M. 1995. Enlargement or reduction of digital images with minimum loss of information. *IEEE TIP*, 4(3):247–258.

UNSER, M., THÉNEVAZ, P., and YAROSLAVSKY, L. 1995. Convolution-based interpolation for fast, high-quality rotation of images. *IEEE TIP*, 4(10):1371–1381.

WANG, Z., BOVIK, A., SHEIKH, H., and SIMONCELLI, E. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. on Image Processing*, 13(4).