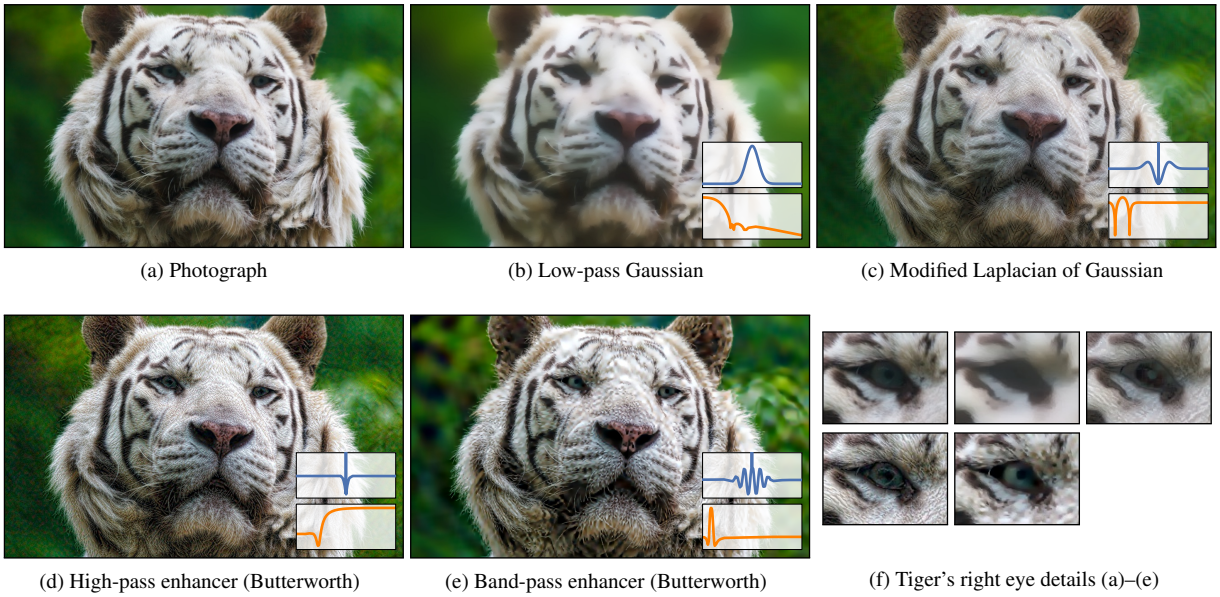
EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

High-Order Recursive Filtering of Non-Uniformly Sampled Signals for Image and Video Processing

Eduardo S. L. Gastal[†] and Manuel M. Oliveira[‡]

Instituto de Informática – UFRGS



(a) Photograph (b) Low-pass Gaussian (c) Modified Laplacian of Gaussian

(d) High-pass enhancer (Butterworth) (e) Band-pass enhancer (Butterworth) (f) Tiger's right eye details (a)–(e)

Figure 1: A variefy of high-order recursive filters applied by our method to the photograph in (a). For these examples, non-uniform sampling positions are computed using an edge-aware transform. Thus, the resulting filters preserve the image structure and do not introduce visual artifacts such as halos around objects. The graphs in the insets show the filter's impulse response in blue, and its frequency response (Bode magnitude plot) in orange.

Abstract

We present a discrete-time mathematical formulation for applying recursive digital filters to non-uniformly sampled signals. Our solution presents several desirable features: it preserves the stability of the original filters; is well-conditioned for low-pass, high-pass, and band-pass filters alike; its cost is linear in the number of samples and is not affected by the size of the filter support. Our method is general and works with any non-uniformly sampled signal and any recursive digital filter defined by a difference equation. Since our formulation directly uses the filter coefficients, it works out-of-the-box with existing methodologies for digital filter design. We demonstrate the effectiveness of our approach by filtering non-uniformly sampled signals in various image and video processing tasks including edge-preserving color filtering, noise reduction, stylization, and detail enhancement. Our formulation enables, for the first time, edge-aware evaluation of any recursive infinite impulse response digital filter (not only low-pass), producing high-quality filtering results in real time.

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Computer Graphics]: Image Processing and Computer Vision—Enhancement

1. Introduction

Digital filters are fundamental building blocks for many image and video processing applications. **Recursive digital filters** are particularly important in this context, as they have several desirable features: they can be evaluated in $O(N)$-time for N pixels, being ideal for real-time applications; they can represent infinite impulse response (*i.e.*, the value of a pixel may contribute to the values of the *entire* image or video frame, which is necessary for several applications, such as recoloring and colorization); and their implementation is relatively straightforward, since they are defined by a difference equation (Eq. 1).

For digital manipulation and processing, continuous signals (often originating from real-world measurements) must be sampled. Traditionally, uniform sampling is preferred, and samples are arranged on a spacetime regular grid (for example, the rows, columns, and frames of a video sequence). However, several applications are better defined using **non-uniform sampling**, such as alias-free signal processing [SS60], global illumination [Jen96], edge-aware image processing [GO11], filtering in asynchronous systems [FBF10], particle counting in physics [PO00], among many others [Eng07]. The main difficulty is that standard operations for filtering—such as fast Fourier transforms (FFT), convolutions, and recursive filters—are commonly formulated with uniform sampling in mind.

We introduce a mathematical formulation for applying recursive digital filters to non-uniformly sampled signals. Our approach is based on simple constructs and provably preserves stability of any digital filter, be it low-pass, high-pass, or band-pass. We also explore relevant aspects and implications to image and video processing applications, such as filter behavior to image edges (Sections 3.2.3, 3.2.4 and 5), and common boundary condition specification (Section 3.2.5). The flexibility of our solution allows for the fine-tuning of the filter response for specific applications (Section 6.2).

Our method is general and works with any non-uniformly sampled signal and any recursive digital filter defined by a difference equation. Since our formulation works directly with the filter coefficients, it works out-of-the-box with existing methodologies for digital filter design. In particular, we illustrate the usefulness of our formulation by using it to integrate higher-order recursive filtering with recent work on edge-aware transforms (Section 6.1). Such an integration allows us to demonstrate, *for the first time ever*, *linear-time edge-aware implementations of arbitrary recursive digital filters*. We show examples of a variety of such filters, including Gaussian, Laplacian of Gaussian, and low/high/band-pass Butterworth and Chebyshev filters (Figure 1).

Our solution produces high-quality results in real time, and works on color images at arbitrary filtering scales. The resulting filters have infinite impulse responses, and their computational costs are not affected by the sizes of the supports of the filters. We demonstrate the flexibility and effectiveness of our solution in various image and video applications, including edge-aware color filtering, noise reduction, stylization, and detail enhancement.

The **contributions** of our work include:

- *A discrete-time $O(N)$ mathematical formulation for applying arbitrary recursive filters to non-uniformly sampled signals (Section 3)*;
- *Two normalization schemes for filtering non-uniformly sampled signals: one based on piecewise resampling (Section 3.2.3), and the other based on spatially-variant scaling (Section 3.2.4). In edge-aware applications, our infinite impulse response (IIR) normalization schemes provide control over the filter's response to signal discontinuities (*i.e.*, edges). This was previously only possible for finite impulse response (FIR) filters*;
- *A general technique to obtain linear-time low-pass, high-pass, and band-pass edge-aware filters (Section 6.1). Our approach allows one to perform all these filters in real time*;
- *The first linear-time edge-aware demonstrations of several low/high/band-pass filters, including Gaussian, Laplacian of Gaussian, and Butterworth (Section 6.2)*;
- *A demonstration of uses of non-uniform filtering in various image and video processing applications (Section 6.2), for which we discuss important details, such as common boundary conditions (Section 3.2.5), and symmetric filtering (Section 3.2.6)*.

2. Background on Recursive Filtering

Recursive filters have been extensively studied in the past 50 years, and a variety of mathematical techniques are available for their design and analysis [PM07]. In computer graphics, recursive filtering has been employed in a large number of applications, including interpolation [BTU99], temporal coherence [FL95], edge-aware image processing [GO11, GO12, Yan12], and efficient GPU filtering [NMLH11]. These filters have several advantages compared to other filtering methods based on brute-force convolution, summed-area-tables, and FFTs. For instance, they have linear-time complexity in the number of input samples; infinite impulse responses; and a relatively straightforward implementation.

A causal infinite impulse response (IIR) linear filter is described by a *difference equation* in the spatial/time domain:

$$g[k] = \sum_{i=0}^{Q} n_i \, f[k-i] + \sum_{i=1}^{P} d_i \, g[k-i], \quad k = 0 \ldots N-1; \quad (1)$$

where $f[k]$ is the input sequence of length N, $g[k]$ is the output sequence, and $\{n_i, d_i\} \in \mathbb{R}$ are the filter coefficients. P

† eslgastal@inf.ufrgs.br
‡ oliveira@inf.ufrgs.br

is called the *feedback order* of the filter. A 0th-order filter is simply a finite impulse response (FIR) one. Since the input sequence is finite and only defined for $k = 0 \dots N-1$, one needs to define the values of $f[-q]$ for $q = 1 \dots Q$ and $g[-p]$ and for $p = 1 \dots P$, called the *initial conditions* of the system. Eq. 1 can be evaluated in $O(N)$ time, and it implements a *causal* filter since its output only depends on previous outputs and current/previous inputs.

The causal system from Eq. 1 is equivalently described by its *transfer function* $H(z)$ in the z-domain [PM07]:

$$H(z) = \frac{G(z)}{F(z)} = \frac{\sum_{i=0}^{Q} n_i \, z^{-i}}{1 - \sum_{i=1}^{P} d_i \, z^{-i}}, \tag{2}$$

where $F(z) = \mathcal{Z}\{f[k]\}$ and $G(z) = \mathcal{Z}\{g[k]\}$ are the z-transforms of the input and output sequences, respectively. The unilateral z-transform of a sequence $x[k]$ is given by

$$X(z) = \mathcal{Z}\{x[k]\} = \sum_{k=0}^{\infty} x[k] \, z^{-k}. \tag{3}$$

The P roots of the denominator in Eq. 2 are the finite *poles* of the transfer function. The output sequence $g[k]$ can be obtained from $H(z)$ and $F(z)$ by computing the inverse z-transform (on both sides) of $G(z) = H(z)F(z)$. This yields $g[k] = (h * f)[k]$, where $*$ is discrete convolution. $h[k]$ is the *impulse response* of the filter described by Eq. 1, given by the inverse z-transform of $H(z)$. Its discrete-time Fourier transform $\hat{h}(\omega)$ is obtained from its z-transform as $\hat{h}(\omega) = H(e^{j\omega})$, where $j = \sqrt{-1}$. Since the filter is causal, the impulse response is zero for negative indices: $h[k] = 0$ for $k < 0$.

2.1. Non-Uniform Sampling

Recursive filtering of non-uniformly sampled signals has been studied by Poulton and Oksman [PO00], and by Fesquet and Bidégaray [FBF10]. They model the filtering process in the s-domain, related to the continuous spatial/time domain by the Laplace transform. The filter then becomes a continuous differential equation, which can be solved numerically using a variable time-step to represent the non-uniformly sampled output. Essentially, the scheme chosen for the numerical solution defines how one transforms the s-domain (where the filter is modeled in continuous-time) to the z-domain (where the filter is evaluated in discrete-time).

Fesquet and Bidégaray [FBF10] review several numerical integration approaches to solve the s-domain differential equation using variable time-steps. They conclude that the semi-implicit bilinear method of [PO00] is possibly the best option in terms of complexity and stability. However, this transform is not defined for systems with poles at $z = -1$, and may be ill-conditioned for systems with poles very close to $z = -1$ (some high pass filters) [MAT14b].

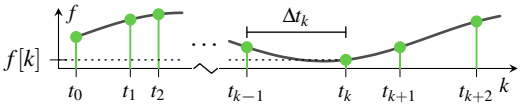


Figure 2: Example of a non-uniformly sampled signal.

3. Recursive Filtering of Non-Uniformly Sampled Signals

We introduce an alternative formulation for recursive filtering of non-uniformly sampled signals. For this, we extend Eq. 1 to work directly in the non-uniform discrete domain. As a result, we can directly apply an arbitrary discrete-time filter $H(z)$ to any non-uniform signal. In the upcoming discussions, we focus on properties and applications relevant to image and video processing. For instance, we discuss different normalization schemes (Sections 3.2.3 and 3.2.4) that may lend to better-suited filters for specific tasks (Section 5). We also describe the integration of high-order recursive digital filters with recent work on edge-aware transforms (Section 6.1), enabling, for the first time, edge-aware evaluation of a variety of filters, such as those illustrated in Figure 1.

Our approach takes as **input** a discrete-time filter defined by its transfer function $H(z)$, an input sequence $f[k]$ of length N, and a set of positive values $\{\Delta t_k\}$ which define the distance (or time delay) between subsequent samples (Figure 2). The distance values Δt_k are commonly obtained from real measurements at the time of sampling [FBF10], or computed in other ways [GO11]. From an initial position t_0, we compute the exact position t_k of the k-th input sample using the recurrence $t_k = t_{k-1} + \Delta t_k$. Our **output** is a sequence $g[k]$ of length N, containing the filtered input values.

3.1. The Naive Approach

One might be tempted to directly apply Eq. 1 to the input sequence $f[k]$, while ignoring the distance values Δt_k. Certainly, this does not produce the desired output, since it treats the sequence $f[k]$ as if it were a uniformly-sampled sequence. The underlying problem lies in the filter $H(z)$ (Eq. 2), which has a hidden dependency on a *constant* sampling interval T. One can intuitively see this dependency through its discrete-time Fourier transform, obtained from $H(z)$ by letting $z = e^{j\omega}$: note that the frequency parameter $\omega \in [-\pi, \pi]$ is *normalized* relative to the sampling interval T. That is, ω is measured in radians per *sample*. This means that the sequence being filtered should have been sampled using the same interval T, otherwise the response of $H(z)$ cannot be effectively characterized in the frequency domain.

One naive solution for filtering non-uniformly sampled sequences is to perform sample-rate conversion to bring the sampling rate to a constant value. However, this is impractical as it introduces severe overhead to the filtering process: depending on the values of the intervals Δt_k, sample-

rate conversion to a constant interval T could require large amounts of memory and time. Another interesting solution is the non-uniform extension of the FFT [Mar01]. However, its performance is still superlinear $O(NlogN)$ in the number of pixels, and its implementation somewhat complex.

Our approach, described in the following sections, addresses all of these limitations.

3.2. Our Approach

This section presents the main contribution of our work: *a mathematical formulation required to apply an arbitrary recursive filter to non-uniformly sampled signals*. We solve this problem by first decomposing a P-th order filter into a set of 1st-order ones (Section 3.2.1), then deriving the equations for the individual 1st-order filters in a non-uniform domain (Sections 3.2.2–3.2.6), and finally applying them separately to the input data (Eq. 6).

3.2.1. Decomposition into 1st-Order Filters

Let $H(z)$ be a P-th order filter whose P poles $b_1, b_2, \ldots, b_P$ are all distinct. Then, through partial-fraction expansion [PM07], $H(z)$ can be decomposed into a *sum* of P 1st-order filters and one 0th-order FIR filter:

$$H(z) = \sum_{i=1}^{P} \frac{a_i}{1 - b_i\, z^{-1}} + \sum_{i=0}^{Q-P} c_i\, z^{-i}, \quad \{a_i, b_i, c_i\} \in \mathbb{C}. \tag{4}$$

The i-th 1st-order filter $H_i(z) = \frac{a_i}{1-b_i\, z^{-1}}$ is described in the spatial domain by the difference equation

$$g_i[k] = a_i\, f[k] + b_i\, g_i[k-1], \tag{5}$$

or equivalently by the convolution of the input sequence $f[k]$ with its (causal) impulse response $h_i[k] = a\, b^k$:

$$g_i[k] = (h_i * f)[k].$$

Due to the linearity of Eq. 3, the original filter $H(z)$ can then be computed in *parallel* in $O(N)$-time as the summed response of all g_i, plus a convolution with the FIR filter:

$$g[k] = \sum_{i=1}^{P} g_i[k] + \sum_{i=0}^{Q-P} c_i\, f[k-i]. \tag{6}$$

If $H(z)$ contains a multiple-order pole of order $m > 1$ (*i.e.*, $b_i = b_{i+1} = \ldots = b_{i+m-1}$), its partial-fraction expansion (Eq. 4) will also contain terms of orders 1 through m:

$$\begin{aligned} &H_i(z) + H_{i+1}(z) + \cdots + H_{i+m-1}(z) \\ &= \frac{a_{i,1}}{1 - b_i\, z^{-1}} + \frac{a_{i,2}\, z^{-1}}{(1 - b_i\, z^{-1})^2} + \cdots + \frac{a_{i,m}\, z^{-(m-1)}}{(1 - b_i\, z^{-1})^m}. \end{aligned} \tag{7}$$

However, any term of order $l = 1 \ldots m$ can be decomposed into a *product* (in the z-domain) of 'l' 1st order terms:

$$\frac{a_{i,l}\, z^{-(l-1)}}{(1 - b_i\, z^{-1})^l} = \frac{a_{i,l}}{1 - b_i\, z^{-1}} \prod_{1}^{l-1} \frac{z^{-1}}{1 - b_i\, z^{-1}}.$$

In the spatial domain, this is the application in *sequence* of 'l' 1st order filters, which is also performed in $O(N)$-time.

3.2.2. 1st-Order Filtering in a Non-Uniform Domain

Let $H(z) = a/(1 - b\, z^{-1})$ be a 1st-order filter with coefficients $\{a, b\} \in \mathbb{C}$, which is described in the spatial domain by Eq. 5. Without loss of generality, from here on we will assume this filter has been designed for a constant and unitary sampling interval $T = 1$. Suppose the input sequence is zero ($f[k] = 0$) for all k between some positive integers k_0 and k_1, where $k_0 < k_1$. Then, if we unroll the recurrence relation in Eq. 5, the response of the system for $g[k_1]$ can be written as

$$g[k_1] = a\, f[k_1] + b^{k_1 - k_0}\, g[k_0].$$

Note that $k_1 - k_0$ is the spatial distance (or elapsed time) between the k_0-th and k_1-th samples, due to the unitary sampling interval. However, if we take into account the non-uniform distribution of the sequence $f[k]$, the distance $k_1 - k_0$ is incorrect, and should be replaced by $t_{k_1} - t_{k_0}$:

$$g[k_1] = a\, f[k_1] + b^{t_{k_1} - t_{k_0}}\, g[k_0].$$

Thus, let $k_0 = k - 1$ and $k_1 = k$; and note that $\Delta t_k = t_k - t_{k-1}$ (see Figure 2). Eq. 5 is written in a non-uniform domain as

$$g[k] = a\, f[k] + b^{\Delta t_k}\, g[k-1]. \tag{8}$$

This equation correctly propagates the value of $g[k-1]$ to $g[k]$ according to the sampling distance Δt_k. However, it fails to preserve the normalization of the filter.

A normalized filter has unit gain at some specified frequency ω_*. For example, a low-pass filter commonly has unit gain at $\omega_* = 0$, which is equivalent to saying that its discrete impulse response should sum to one: $\sum h[k] = 1$. Normalizing a filter is done by scaling its impulse response (in practice, its numerator coefficients $\{n_i\}$ from Eq. 2) by an appropriate factor γ. To find γ one uses the discrete-time Fourier transform[†], which *assumes a constant sampling interval*. Indeed, non-uniform sampling breaks this assumption, meaning that there does not exist a single scaling factor γ which makes the filter everywhere normalized.

Assuming the input filter $H(z)$ (originally designed for uniformly-sampled signals) is normalized, *we present two ways of correcting Eq. 8 to preserve normalization when filtering non-uniformly sampled signals*. The first approach is *based on piecewise resampling* (Section 3.2.3), while the second is *based on spatially-variant scaling* (Section 3.2.4). Each approach produces a different response for the filter (see discussion in Section 5), while maintaining its defining characteristics. Appendix A proves the stability of our filtering equations.

[†] Let $\omega_* \in [-\pi, \pi]$ be the normalized frequency parameter. The gain $|\gamma|$ at frequency ω_* of a filter $U(z)$ is $|\gamma| = |U(e^{j\omega_*})| = |\hat{u}(\omega_*)|$. Thus, $H(z) = U(z)/|\gamma|$ is a filter normalized to unit-gain at ω_*.

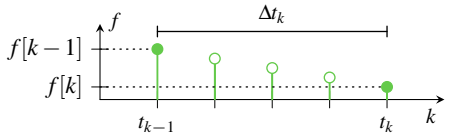


Figure 3: Piecewise linear unitary resampling between the $(k-1)$-th and k-th samples. In this example, $\Delta t_k = 4$.

3.2.3. Normalization-preserving piecewise resampling

Recall that, in the z-domain, a digital filter is designed and normalized assuming a *constant* sampling interval T (Section 3.1). To work with non-uniform sampling, it is impractical to perform sample-rate conversion on the full input sequence due to time and memory costs. In this section, we show how one can perform *piecewise* resampling in a very efficient way. In particular, we show that *it is possible to express this resampling process using a closed-form expression* (*i.e.*, one does not have to actually create and filter new samples, nor store them in memory). Thus, we are able to preserve normalization and maintain the $O(N)$-time performance of the filter, even when dealing with non-uniformly sampled signals.

Without loss of generality, assume $T = 1$. Also assume, for the time being, that the non-uniform distances between samples are positive integers (*i.e.*, $\Delta t_k \in \mathbb{N}$). This restriction will be removed later. To compute the output value $g[k]$, we will use the known previous output value $g[k-1]$ and create new samples between $f[k-1]$ and $f[k]$ to obtain uniform and unitary sampling. This process is illustrated in Figure 3, where new input samples (shown as green outlined circles) are linearly interpolated from the actual samples (green circles at times t_{k-1} and t_k). While a linear interpolator does not obtain ideal reconstruction of the underlying continuous signal [PM07], it is computationally efficient and produces good results for image and video processing [GO11]. Other polynomial interpolators such as Catmull-Rom [CR74] can be used at the expense of additional computation.

Since we are working with a causal filter, the newly interpolated samples will contribute to the value of $g[k]$, but not to $g[k-1]$. As expected, this contribution is simply the convolution of the interpolated samples with the filter's impulse response $h[k]$. Since the convolution result will be added to the value of $g[k]$ (the k-th sample), it is evaluated at position t_k of the domain. The end result is an additional summation term Φ in Eq. 8:

$$g[k] = a\,f[k] + b^{\Delta t_k}\,g[k-1] + \underbrace{\sum_{i=1}^{\Delta t_k - 1} h[\Delta t_k - i]\,\tilde{f}_k[i]}_{\Phi}. \qquad (9)$$

$\tilde{f}_k[i]$ is the i-th sample interpolated from $f[k-1]$ and $f[k]$:

$$\tilde{f}_k[i] = \frac{i}{\Delta t_k}\,(f[k] - f[k-1]) + f[k-1]. \qquad (10)$$

Closed-form solution One can evaluate the summation Φ into a closed-form expression by substituting Eq. 10 and the 1st-order impulse response $h[k] = a\,b^k$ into it:

$$\Phi = \left(\frac{b^{\Delta t_k} - 1}{r_0\,\Delta t_k} - r_1\,b\right) f[k] - \left(\frac{b^{\Delta t_k} - 1}{r_0\,\Delta t_k} - r_1\,b^{\Delta t_k}\right) f[k-1], \qquad (11)$$

where $r_0 = (b-1)^2/(a\,b)$ and $r_1 = a/(b-1)$. This formula can be evaluated in *constant time* regardless of the number of new interpolated samples. Furthermore, despite our initial assumption, Eq. 11 works correctly for non-integer values of the non-uniform distances between samples (*i.e.*, $\Delta t_k \in \mathbb{R}$).

3.2.4. Renormalization by spatially-variant scaling

We can avoid the need for reconstructing the underlying continuous signal through spatially-variant scaling. By unrolling the recurrence in Eq. 8, one obtains a representation of the filtering process as a brute-force convolution:

$$g[k] = \sum_{n=-\infty}^{k} a\,b^{t_k - t_n} f[n]. \qquad (12)$$

Eq. 12 can be *interpreted* as a (causal) linear *spatially-variant* system acting on a uniform sequence $f[k]$ and producing another uniform sequence $g[k]$. The frequency characteristics of this system are spatially-variant since its impulse response is spatially variant. Therefore, such a system can only be normalized to unit gain using a spatially-variant scaling factor γ_k. In this way, the sequence resulting from Eq. 8 is used to build a normalized output sequence $g'[k]$ as

$$g'[k] = g[k] \,/\, |\gamma_k|\,, \quad k = 0 \ldots N-1. \qquad (13)$$

The computation of the values $\{\gamma_k\}$ depends on how one interprets the infinite sum in Eq. 12, as it references input samples $f[k]$ for negative indices k. Two interpretations exist:

Interpretation #1: *Input samples $f[k]$ do not exist for $k < 0$ and, thus, it makes no sense to reference their values.* This is common when working with time-varying signals, such as videos, and filtering along time. Thus, to reference only valid data, the convolution in Eq. 12 should start at zero:

$$g[k] = \sum_{n=0}^{k} a\,b^{t_k - t_n} f[n]. \qquad (14)$$

The gain of this system for frequency ω_* is measured by its response to a complex sinusoid oscillating at ω_*. Thus, when processing the k-th sample, the gain $|\gamma_k|$ at frequency ω_* for the filter in Eq. 14 is

$$|\gamma_k| = \left| \sum_{n=0}^{k} a\,b^{t_k - t_n}\,e^{-j\,\omega_*\,n} \right|. \qquad (15)$$

Algorithm detail Computing γ_k for all k directly from the summation in Eq. 15 results in quadratic $O(N^2)$-time complexity. Linear $O(N)$-time performance can be obtained by

expressing the value of γ_k in terms of the previous value γ_{k-1}. This results in the recurrence relation

$$|\gamma_k| = \left| a\, e^{-j\,\omega_*\,k} + b^{\Delta t_k}\,\gamma_{k-1} \right|, \quad k = 1 \ldots N-1, \qquad (16)$$

with initial value $\gamma_0 = a$.

Interpretation #2: *Input samples $f[k]$ exist for $k < 0$ and their values are defined by application-specific initial (or boundary) conditions*. This is common when working with signals defined in space, such as images and 3D volumes. Section 3.2.5 discusses some choices of boundary conditions. For this case, we have no option but to work with an infinite sum to compute the gain:

$$|\gamma_k| = \left| \sum_{n=-\infty}^{k} a\, b^{t_k - t_n}\, e^{-j\,\omega_*\,n} \right|. \qquad (17)$$

Luckily, the recurrence relation in Eq. 16 is still valid for this infinite sum, but with a different initial value γ_0. To compute this new γ_0, one can arbitrarily choose the sampling positions t_k for $k < 0$ (as part of the definition of the boundary conditions). The obvious choice is a unitary and uniform sampling, which implies that $t_k = t_0 + k$ for $k < 0$. Given this choice and Eq. 17, one obtains

$$\gamma_0 = \sum_{n=-\infty}^{0} a\, b^{-n}\, e^{-j\,\omega_*\,n} = \frac{a}{1 - b\, e^{j\,\omega_*}}, \quad |b| < 1. \qquad (18)$$

The convergence condition $|b| < 1$ is always true for any stable 1st-order filter since b is a pole of its transfer function [PM07].

Note on normalization When implementing a P-th order filter, its 1st-order component filters (obtained through partial-fraction expansion in Section 3.2.1) should be normalized using the gain of the original P-th order filter (*i.e.*, *they should **not** be normalized separately*). Thus, let $H(z)$ be a P-th order filter from Eq. 4. Its gain (and normalization factor) $|\gamma_k|$ at frequency ω_*, evaluated at k, is given by combining the gains $|\gamma_{i,k}|$ $(i = 1 \ldots P)$ of all its composing 1st-order filters:

$$|\gamma_k| = \left| \sum_{i=1}^{P} \gamma_{i,k} + \sum_{n=0}^{Q-P} c_n\, e^{-j\,\omega_*\,n} \right|.$$

The rightmost summation represents the gain contribution of the 0th-order term in Eq. 4. Each $\gamma_{i,k}$ is computed replacing the i-th filter coefficients $\{a_i, b_i\}$ into Eq. 16, and choosing an initial value $\gamma_{i,0}$ according to Interpretation #1 or #2.

If $H(z)$ contains a multiple-order pole b_i of order m, the gain contribution of the terms of orders 1 through m (Eq. 7) is given by the sum

$$\left| \sum_{l=1}^{m} a_{i,l}\, e^{-j\,\omega_*\,(l-1)}\, \left(\gamma^*_{i,k}\right)^l \right|.$$

$|\gamma^*_{i,k}|$ is the gain for the filter $\frac{1}{1-b_i z^{-1}}$, computed by the substitutions $a \to 1$ and $b \to b_i$ into Eq. 16.

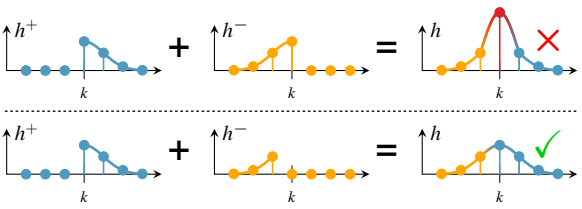


Figure 4: *(Top)* Central sample counted by both causal h^+ and anti-causal h^- filters, resulting in an incorrect impulse response $h = h^+ + h^-$. *(Bottom)* Central sample counted only by the causal h^+ filter, resulting in a correct impulse response $h = h^+ + h^-$.

3.2.5. Initial Conditions

To compute the value $g[0]$ of the first output sample for the filters in Eq. 9 and Eq. 13, one needs to define the values of $f[-1]$ and $g[-1]$: the initial (or *boundary*) conditions of the system. A *relaxed* initial condition is obtained by setting both values to zero: $f[-1] = g[-1] = 0$. However, when filtering images and videos, one frequently *replicates the initial input sample* (*i.e.*, $f[-1] = f[0]$). The corresponding initial value $g[-1]$ is found by assuming a constant output sequence for $k < 0$, where all its samples have a constant value β. This constant is found by solving the system's difference equation: defining a uniform and unitary sampling for $k < 0$, the 1st-order system's equation $g[-1] = a\,f[-1] + b\,g[-2]$ becomes $\beta = a\,f[0] + b\,\beta$. Solving for β gives

$$g[-1] = \beta = \frac{a}{1-b}\, f[0].$$

3.2.6. Non-Causal and Symmetric Filters

For image and video processing, one is usually interested in filters with *non-causal* response. That is, filters for which the output value of a pixel p depends on the values of *pixels to left and pixels to right* of p (see Section 6.1.2 on how we define 2D filters). A non-causal symmetric response is usually achieved by applying the filter in two passes: a causal (left-to-right) pass and an anti-causal (right-to-left) pass. This combination can be done either in series [VVYV98, GO11] or in parallel [Der93].

If done in parallel, one must be careful not to count the central sample twice when designing the filter [Der93]; otherwise, the resulting impulse response may be incorrect (Figure 4, top). A simple way to avoid this problem is to include the central sample only on the causal pass (Figure 4, bottom). For a 1st order filter with causal transfer function given by $H^+(z) = \frac{a}{1-b\,z^{-1}}$, the respective anti-causal transfer function which does not count the central sample is $H^-(z) = \frac{a\,b\,z}{1-b\,z}$, and its corresponding difference equation is

$$g^-[k] = a\,b\,f[k+1] + b\,g^-[k+1]. \qquad (19)$$

For non-uniform domains, Eq. 19 should be rewritten as

$$g^-[k] = a\, b^{\Delta t_k}\, f[k+1] + b^{\Delta t_k}\, g^-[k+1], \qquad (20)$$

and it must be normalized as described in Section 3.2.3 and Section 3.2.4.

4. Designing Digital Filters

Our method can be used to filter any non-uniformly sampled signal using any recursive digital filter defined by a difference equation. Since our formulation uses the filter coefficients, it directly works with existing methodologies for IIR digital filter design. For example, both MATLAB [MAT14a] and the open-source SciPy library [JOP*] provide routines that compute the coefficients of well-known filters such as Butterworth, Chebyshev, and Cauer. They also implement the partial-fraction expansion described in Section 3.2.1 through the routine `residuez()`.

It is also easy to create new filters by combining and modifying existing ones. For example, the high-pass enhancer filter from Figure 1(d) was created by combining a scaled high-pass Butterworth filter with an all-pass filter: $2\,H_{high}(z) + 1$; the filter from Figure 1(c) was similarly created from a band-pass Laplacian of Gaussian (LoG): $1 - 2.5\,H_{LoG}(z)$.

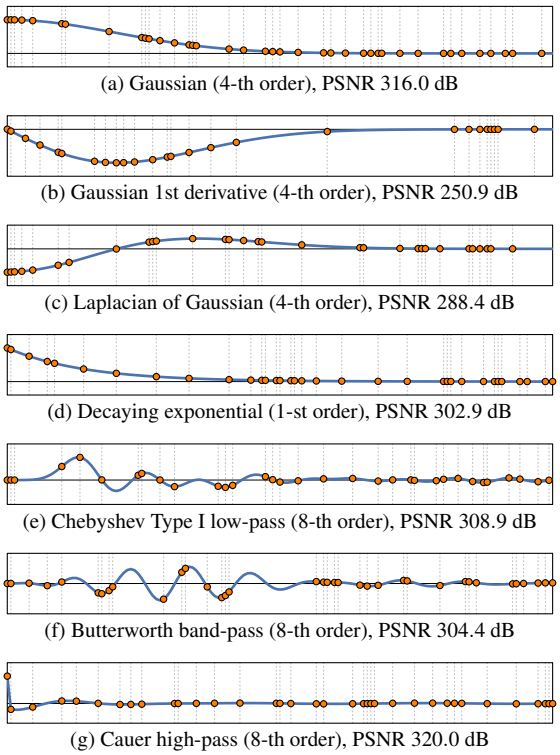
Recursive digital filters can also be designed to approximate *in linear-time* other IIR filters which are commonly superlinear in time. For example, Deriche [Der93] and Van Vliet et al. [VVYV98] show how to approximate a Gaussian filter and its derivatives, and Young et al. [YVVvG02] implement recursive Gabor filtering.

To illustrate the use of our approach to obtain non-uniform filtering equations, Appendix B shows the derivation of a recursive non-uniform $O(N)$-time Gaussian filter with normalization preserved by piecewise resampling.

5. Evaluation and Discussion

The supplementary materials (available at `http://inf.ufrgs.br/~eslgastal/NonUniformFiltering`) include an implementation of our method, together with various examples of using it to process synthetic data, as well as several images and video.

Accuracy Figure 5 illustrates the accuracy of our approach when computing the impulse response of several IIR filters using non-uniform sampling. The filters are defined by their coefficients in the z-domain, and are included in the supplementary materials. Each plot shows the corresponding analytical ground-truth impulse response (solid blue line), together with the output samples (orange dots) obtained by filtering a non-uniformly sampled impulse. The sampling positions (indicated by vertical dotted lines) were generated randomly. The impulse was represented by a 1 at the origin, followed by 0's at the sampling positions.



(a) Gaussian (4-th order), PSNR 316.0 dB

(b) Gaussian 1st derivative (4-th order), PSNR 250.9 dB

(c) Laplacian of Gaussian (4-th order), PSNR 288.4 dB

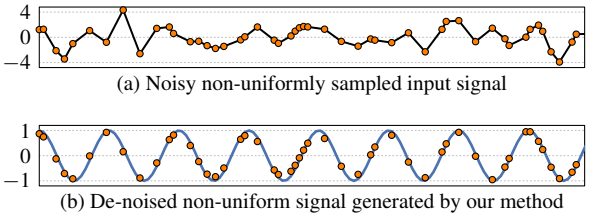
(d) Decaying exponential (1-st order), PSNR 302.9 dB

(e) Chebyshev Type I low-pass (8-th order), PSNR 308.9 dB

(f) Butterworth band-pass (8-th order), PSNR 304.4 dB

(g) Cauer high-pass (8-th order), PSNR 320.0 dB

Figure 5: Accuracy of our approach when filtering an impulse with several IIR filters using non-uniform sampling. The solid blue lines are the analytical ground-truth impulse responses. The small orange dots are the output samples.



(a) Noisy non-uniformly sampled input signal

(b) De-noised non-uniform signal generated by our method

Figure 6: A noisy non-uniformly sampled sinusoid in (a) is filtered by the band-pass Butterworth filter from Figure 5(f) using our approach. The filtered samples are shown in (b), superimposed on the original noiseless signal (in blue).

Figure 5 shows that our results are numerically accurate and visually indistinguishable from ground-truth, with PSNR consistently above 250 dB (note that a PSNR above 40 dB is already considered indistinguishable visual difference in image processing applications). Furthermore, since the impulse response uniquely characterizes the filter, this experiment guarantees the accuracy of our approach in filtering general non-impulse signals. This conclusion is illustrated in Figure 6, where we use the band-pass filter from Figure 5(f) to denoise a non-uniformly sampled signal.

Performance We implemented our approach in C++. Filtering one million samples using a 1st-order filter and 128-bit complex floating point precision takes 0.007 seconds on a single core of an i7 3.6 GHz CPU. This performance scales linearly with the number of samples. It also scales linearly with the order of the filter, which for common applications is rarely larger than 10. All the effects shown in Figure 1 were generated using 4th-order filters.

Our approach is highly parallelizable. A high-order filter is decomposed as a sum of *independent* 1st-order filters which can be computed in parallel (Section 3.2.1). Each 1st-order filter can also be parallelized internally using the approach described in [NMLH11].

Implementation Details The latest CPUs have extremely fast instructions for evaluating the exponentiations in Eqs. 9 and 13. Our C++ code calls `std::pow(`b`,`Δt_k`)` directly. For older CPUs, one can use precomputed tables to further improve filtering times. Other constants dependent on the filter coefficients, such as r_0 and r_1, should be precomputed outside the main filtering loop.

Image and video processing applications use filters that take real inputs and produce real outputs (*i.e.*, $f, g \in \mathbb{R}$). One property of real filters is that any complex coefficient in its partial-fraction expansion must have a complex-conjugate pair [PM07]. Thus, in practice, we only have to compute the filter response for one coefficient in each complex-conjugate pair, multiply the result by two, and drop the imaginary part.

Other Approaches The continuous-space method described by Poulton and Oksman [PO00] may be used for filtering non-uniformly sampled signals. However, their method requires mapping between discrete and continuous space using the bilinear transform, which can become ill-conditioned for very large or small sampling intervals [Bru11], especially in high-pass filters [BPS14]. This has a direct impact on the accuracy and quality of the filter. Furthermore, their approach does not allow control over normalization schemes (Sections 3.2.3 and 3.2.4), and their work does not explore details which become important when filtering images and videos, such as boundary conditions (Section 3.2.5) and the construction of symmetric filters (Section 3.2.6).

Gastal and Oliveira [GO11] describe a simple 1st-order decaying-exponential low-pass filter which works in a non-uniform domain. It can be shown that such a filter is the simplest special case of our more general method: their filter can be obtained from our equations by (i) using a zero-order-hold ($f_k^{zoh}[i] = f[k]$) instead of the linear interpolator in Eq. 9; and by (ii) noticing that the coefficients of a normalized 1st-order low-pass real filter *must* satisfy $a = 1 - b$. This results in the filtering equation $g[k] = (1 - b^{\Delta t_k})\, f[k] + b^{\Delta t_k}\, g[k-1]$. Additionally, their formulation applies causal/anticausal filters in series, and does not lend to a truly symmetric filter in non-uniform domains. Our formulation using filters in parallel addresses this limitation.

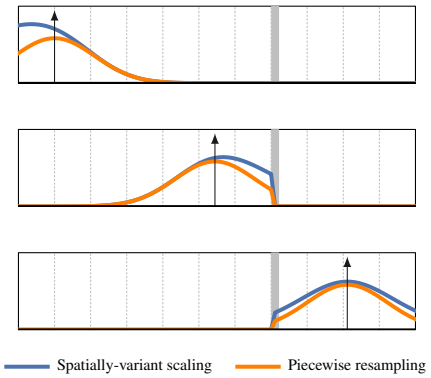


Figure 7: An impulse (upward-pointing arrow) travels from the left to the right of the domain. This domain contains a simulated discontinuity close to its center, shown as a vertical gray line (in a real signal, like an image, this could be an edge from an object—see Section 6.1). The impulse is filtered using our approach to deal with the discontinuity, and a Gaussian kernel. We normalize the filter either by spatially-variant scaling (impulse response shown in blue), or by piecewise resampling (impulse response shown in orange). Note how each normalization scheme results in a different response to the discontinuity and boundary in the domain. Thus, in edge-aware applications we are able to control the filter's response to the edges in the signal [GO11].

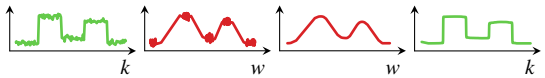
Finally, our IIR normalization schemes are related to the FIR convolution operators defined by [GO11]: our spatially-variant scaling provides the same response as normalized convolution [KW93], and our piecewise resampling generates the same response as interpolated convolution. Thus, in edge-aware applications, our IIR normalization schemes provide control over the filter's response to the edges (see the plots in Figure 7). This was previously only possible for the FIR filters of [GO11].

6. Applications

This Section demonstrates the usefulness of our formulation to various tasks in image and video processing. In particular, we show how to integrate high-order recursive filtering with recent work on edge-aware transforms. Thus, we demonstrate *the first linear-time edge-aware implementations of several recursive digital filters, including Gaussian, Butterworth, and other general low/high/band-pass filters*.

6.1. General Edge-Aware Filtering

An *edge-aware* filter transforms the content of an image while taking into account its structure. For example, an edge-aware smoothing filter can remove low-contrast variations in the image while preserving the high-contrast edges; and



(a) $f[k]$ (b) $f[t^{-1}(w)]$ (c) $g[t^{-1}(w)]$ (d) $g[k]$

Figure 8: Edge-preserving low-pass filtering using the domain transform. *(a)* Input sequence $f[k]$. *(b)* Input sequence after the warping defined by the domain transform $w = t(k)$. *(c)* Input sequence after warping and filtering with a low-pass Gaussian filter. *(d)* Output sequence $g[k]$ obtained after un-warping (c).

an edge-aware enhancement filter can increase local contrast without introducing visual artifacts such as halos around objects. Due to these properties, edge-aware filters are important components of several image and video processing applications [DD02, LLW04, LFUS06, Fat09, FFL10].

Recently, Gastal and Oliveira [GO11] showed how any filtering kernel can be made edge-aware by adaptively warping the input sequence using a *domain transform*. Conceptually, they warp the input image (signal) along orthogonal 1-D curves while preserving the distances among pixels, as measured in higher-dimensional spaces. In such a warped domain, pixels (samples) are *non-uniformly spaced*. Applying a linear filter in this warped domain and then reversing the warp results in an edge-aware filter of the original samples. This process is illustrated in Figure 8 for a low-pass filter applied to a 1-D signal. In practice, there is no need to explicitly warp and un-warp the signal, and the entire operation is performed on-the-fly in a single step.

The technique of Gastal and Oliveira [GO11] is fast and lends to good results. However, its solution (in linear time) has only been demonstrated on two simple filters: an iterated box filter and a recursive 1st-order decaying-exponential filter (both low-pass filters). *Using our formulation of non-uniform filtering, we are able to generalize their approach to work on recursive filters of any order, and in linear time*, which allows practically unlimited control over the shape of the filtering kernel. In other words, with our generalization we can *transform any recursive linear filter h* (described by Eq. 1) *into a recursive edge-aware filter h_{EA}* (described by either Eq. 9 or Eq. 13). The resulting filter is non-linear, and maintains the characteristics of the original filter. Thus, for instance, if h is a low-pass filter, h_{EA} will be a low-pass edge-aware filter. Furthermore, since h_{EA} is also described by a difference equation, it will filter an input sequence of length N in $O(N)$ time.

Next we review the domain transform and show how it integrates with our method. Section 6.2 shows various examples of applications that use this integration.

6.1.1. Review of the Domain Transform

Assuming a unitary sampling interval along the rows (or columns) of an image, the imagespace distance between two samples $f[k]$ and $f[k+\delta]$ is δ, for $\delta \in \mathbb{N}$. Using a domain transform $t(k)$, the *warped-space* distance between the same samples is $t(k+\delta) - t(k)$. By definition (see Eq. 21), the domain transform is a monotonically increasing function (*i.e.*, $t(k+\delta) - t(k) \geq \delta$).

Gastal and Oliveira [GO11] obtain their domain transform using the ℓ_1 norm to compute distances over the image manifold. We instead use the ℓ_2 norm. This results in a (discrete) domain transform given by

$$t(k) = \sum_{i=1}^{k} \sqrt{1 + \left(\frac{\sigma_s}{\sigma_r}\right)^2 \sum_{c=1}^{d} (f_c[i] - f_c[i-1])^2}. \qquad (21)$$

Here, $f_c[i]$ is the i-th element in the sequence of N samples obtained from the c-th *channel* of the signal, from a total of 'd' channels (for example, an RGB image has $d = 3$ channels: red, green, and blue). σ_s and σ_r are parameters of the edge-aware filter. σ_s controls the imagespace size of the filter kernel, and σ_r controls its range size (*i.e.*, how strongly edges affect the resulting filter). We refer the reader to [GO11] for further details.

6.1.2. Using the Domain Transform with Our Method

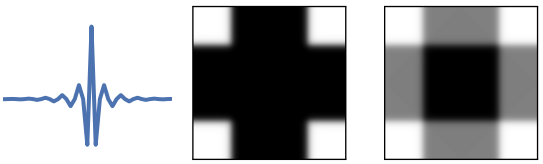
Eq. 21 defines new non-uniform positions for each sample in the spatial domain. Consequently, the warped-space distance between adjacent samples $f[k-1]$ and $f[k]$ is given by

$$\Delta t_k = \sqrt{1 + \left(\frac{\sigma_s}{\sigma_r}\right)^2 \sum_{c=1}^{d} (f_c[k] - f_c[k-1])^2} \qquad (22)$$

The values $\{\Delta t_k\}$ can be precomputed for all $k = 0 \ldots N-1$, and then substituted into the filtering equations (Eqs. 9 and/or 13) for evaluating the filter. In this way, we obtain an *edge-aware* implementation of arbitrary recursive filters. For example, using the non-uniform Gaussian filter derived in Appendix B, we obtain a $O(N)$-time edge-aware Gaussian.

Filtering 2D Signals As described by [GO11], we filter 2D images by performing a horizontal pass along each image row, and a vertical pass along each image column. How the horizontal and vertical passes are combined depend on the desired frequency response of the filter. Low-pass filters are better applied in *sequence*: assuming the horizontal pass is performed first, the vertical pass is applied to the result produced by the horizontal one. High and band-pass filters are usually better applied in *parallel*: the horizontal and vertical passes are performed independently, and their result added at the end. This suggestion is simply a design choice: each option (sequence/parallel) will result in a filter with different 2D frequency response. In our specific case, we apply low-pass filters in sequence since a greater amount of high-frequencies are "removed" from the signal, and we apply high-pass filters in parallel since a greater amount of high-frequencies are preserved in the signal (see Figure 9). This is the strategy we used for filtering the images shown

(a) Low-pass filter (b) 2D DFT, sequence (c) 2D DFT, parallel



(d) High-pass filter (e) 2D DFT, sequence (f) 2D DFT, parallel

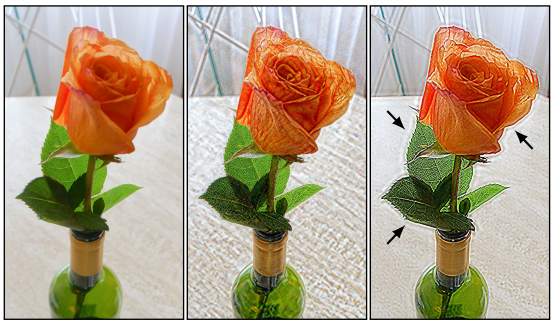
Figure 9: The 1D low-pass filter with impulse response shown in (a) may be applied to the rows and columns of an image to obtain a 2D filter. The corresponding horizontal and vertical passes may be combined either in sequence or in parallel, each option resulting in a filter with different 2D frequency response. This is illustrated by the 2D discrete Fourier transforms (DFT) shown in (b) and (c), where white represents a gain of one and black a gain of zero, and the zero-frequency has been shifted to the center of the images. It is clear that applying the filter from (a) in sequence removes a greater amount of high-frequencies from the signal. Thus, this low-pass filter is better applied in sequence, as in (b). The opposite is true for high-pass filters. The high-pass filter from (d) is better applied in parallel, as in (f), since a greater amount of high-frequencies are preserved.

in the paper. Filtering higher-dimensional signals such as 3D volumes is performed analogously.

6.2. Image and Video Processing Examples

Detail Manipulation Our formulation for non-uniform recursive digital filters enables for the first time the direct application of general filters in edge-aware applications. For example, one can perform general frequency-domain manipulations without introducing artifacts such as halos around objects, as shown in Figure 10. This is possible due to the non-uniform sampling of the image pixels defined by Eq. 22.

Figure 1 shows several examples of high-order IIR filters used to manipulate the details of the photograph shown in (a). In (b), a low-pass Gaussian smoothes small variations while preserving large-scale features. For the image shown in (c), we used a modified band-stop Laplacian of Gaussian to create a stylized look for the image. For the result shown in (d), we used a high-pass Butterworth to enhance fine details in the tiger's fur and whiskers. The image in (e) was obtained with a band-pass Butterworth to improve local contrast by enhancing medium-scale details. For (e),



(a) Photograph (b) Non-uniform (c) Uniform

Figure 10: Detail enhancement using a high-pass filter. Non-uniform sampling (b) avoids the common halo artifacts (black arrows) in traditional uniform sampling (c).

(a) Photograph with scribbles (b) Edited output

Figure 11: Turning bronze into gold using our approach. See the text for details.

an edge-aware low-pass post-filter was applied to obtain the final result, as recommended in [GO11]. This is necessary because 2D filtering using the domain transform sometimes introduces axis-aligned artifacts in the filtered image. Our supplementary materials show these filters applied to many other images and to a video.

While edge-aware detail manipulation has been performed by previous approaches [FAR07, FFLS08, PHK11], all of them work by computing differences between the outputs of a fixed type of low-pass filter. By providing the ability to experiment with the design and composition of new digital filters, our method has the potential do enable a greater variety of effects.

Localized Editing Filtering in non-uniform edge-aware domains can also be used for localized manipulation of pixel colors. In Figure 11(a), color scribbles define two regions of interest in the underlying photograph. For each region, we generate an influence map using our low-pass non-uniform

(a) Photograph (b) Lightness of (a)

(c) 3% of pixels from (a) (d) Reconstructed from (b), (c)

Figure 12: Example of data-aware interpolation using non-uniform filtering. A full-color image is reconstructed from the lightness channel and only 3% of the pixels from the original image, shown in (a). The pixels in (c) were importance-sampled using the gradient magnitude of the lightness channel. PSNR of (d) vs (a) is 31.17 dB.

Gaussian filter (see [LFUS06] for details). The influence map for the region of interest is then normalized by the sum of influence maps for all regions, which defines a soft-segmentation mask. This mask is used to restrict recoloring to certain parts of the image.

Data-aware Interpolation Propagating sparse data across the image space also benefits from an edge-aware operator [LLW04]. For example, in Figure 12 our low-pass non-uniform Gaussian filter is used to propagate the color of a small set of pixels, shown in (c), to the whole image. This generates the full-color image shown in (d). The non-uniform domain is defined by the domain transform applied to the lightness image in (b).

Denoising By grouping pixels based on high-dimensional neighborhoods, we can define a fast and simple denoising algorithm, as illustrated in Figure 13. We cluster pixels from the noisy photo in (a) based on their proximity on the high dimensional non-local means space [BCM05]. For this example, we generate 30 disjoint clusters using k-means, which are color-coded in (c) for visualization. The pixels belonging to the same cluster define a *non-uniformly sampled signal* in the image space. We apply a non-uniform Gaussian filter only to the pixels belonging to the same clusters, averaging-out the zero-mean noise. This is followed by a second non-uniform edge-aware Butterworth low-pass filter on the hole image (adhering to the edges of (a)), with the goal of removing quantization borders which originate from the discrete clusters. The resulting denoised photograph is shown in (b).

Using our formulation, for an image with N pixels, filtering together only pixels belonging to the same clusters is done in $O(N)$ time *for all clusters*. That is, the time complexity is independent of the number of clusters. Without our formulation, for K clusters, one would have to separate the image into K uniformly-sampled N-pixel images for filtering, which would result in $O(NK)$ complexity. Note also that acceleration techniques such as the Adaptive Manifolds [GO12] can perform non-local means filtering extremely fast. However, its time complexity is $O(N/\sigma_s)$, which may lead to slow filtering performance for filters with small values of σ_s (*i.e.*, small imagespace kernel sizes).

Stylization The same idea behind the denoising algorithm above can be used for stylization. In Figure 14(a), we cluster pixels based only on their RGB-proximity. Filtering only pixels (of the input image) belonging to the same clusters with a non-uniform Gaussian, and then superimposing edges computed using the Canny algorithm applied to the filtered image, one obtains a soft cartoon-like look (b).

7. Conclusion

We presented a discrete-time mathematical formulation for applying recursive digital filters to non-uniformly sampled signals. Our method is general and works with any non-uniformly sampled signal and any recursive digital filter defined by a difference equation. We have used our formulation to obtain general low/band/high-pass edge-aware filters. We have demonstrated the effectiveness of such filters applied to non-uniformly sampled signals in various image and video tasks, including edge-preserving color filtering, noise reduction, stylization, and detail enhancement. By providing a simple and natural way to experiment with the design and composition of new digital filters, our method has the potential do enable a great variety of new image and video effects.

8. Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. This work was sponsored by CNPq-Brazil (fellowships and grants 158666/2010-0, 557814/2010-3, 308936/2010-8, and 482271/2012-4) and CAPES. Input photograph from Figure 1(a) is from publicdomainpictures.net (image 94418).

References

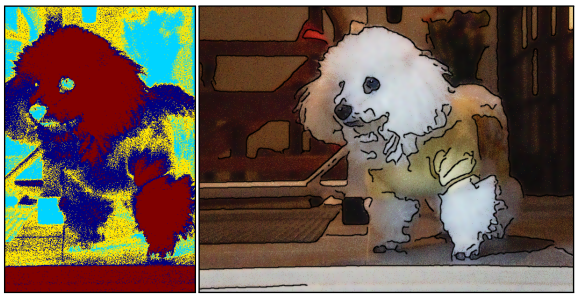
[BCM05] BUADES A., COLL B., MOREL J.: A non-local algorithm for image denoising. In *CVPR* (2005), vol. 2, pp. 60–65. 11

[BPS14] BRUSCHETTA M., PICCI G., SACCON A.: A variational integrators approach to second order modeling and identification of linear mechanical systems. *Automatica 50*, 3 (2014), 727 – 736. 8

(a) Noisy photograph (b) Denoised

(c) Clustering (d) Detail from (a) (e) Detail from (b)

Figure 13: Denoising using non-uniformly sampled pixel groups defined by k-means clustering. See text for details.



(a) Clustering (b) Stylized

Figure 14: Stylization using non-uniform filtering.

[Bru11] BRUSCHETTA M.: *A variational integrators approach to second order modeling and identification of linear mechanical systems*. PhD thesis, Università Degli Studi Di Padova, 2011. 8

[BTU99] BLU T., THÉVENAZ P., UNSER M.: Generalized interpolation: Higher quality at no additional cost. In *IEEE ICIP* (1999), pp. 667–671. 2

[CR74] CATMULL E., ROM R.: A class of local interpolating splines. *Computr aided geometric design 74* (1974), 317–326. 5

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. In *SIGGRAPH '02* (2002), pp. 257–266. 9

[Der93] DERICHE R.: Recursively implementating the gaussian and its derivatives, 1993. 6, 7, 13

[Eng07] ENG F.: *Non-Uniform Sampling in Statistical Signal Processing*. PhD thesis, Linköping universitet, 2007. 2

[FAR07] FATTAL R., AGRAWALA M., RUSINKIEWICZ S.: Multiscale shape and detail enhancement from multi-light image collections. *ACM TOG 26* (2007), 51:1–51:9. 10

[Fat09] FATTAL R.: Edge-avoiding wavelets and their applications. *ACM TOG 28*, 3 (2009), 22. 9

[FBF10] FESQUET L., BIDÉGARAY-FESQUET B.: IIR digital filtering of non-uniformly sampled signals via state representation. *Signal Processing 90*, 10 (2010), 2811 – 2821. 2, 3

[FFL10] FARBMAN Z., FATTAL R., LISCHINSKI D.: Diffusion maps for edge-aware image editing. *ACM TOG 29*, 6 (2010), 145. 9

[FFLS08] FARBMAN Z., FATTAL R., LISCHINSKI D., SZELISKI R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM TOG 27*, 3 (2008), 67. 10

[FL95] FLEET D. J., LANGLEY K.: Recursive filters for optical flow. *IEEE TPAMI 17*, 1 (Jan. 1995), 61–67. 2

[GO11] GASTAL E. S. L., OLIVEIRA M. M.: Domain transform for edge-aware image and video processing. *ACM TOG 30*, 4 (2011), 69:1–69:12. 2, 3, 5, 6, 8, 9, 10

[GO12] GASTAL E. S. L., OLIVEIRA M. M.: Adaptive manifolds for real-time high-dimensional filtering. *ACM TOG 31*, 4 (2012), 33:1–33:13. Proceedings of SIGGRAPH 2012. 2, 11

[Jen96] JENSEN H. W.: Global Illumination using Photon Maps. In *Rendering Techniques*. Springer, 1996, pp. 21–30. 2

[JOP*] JONES E., OLIPHANT T., PETERSON P., ET AL.: SciPy: Open source scientific tools for Python — scipy.signal documentation, 2001–. `http://docs.scipy.org/doc/scipy/reference/signal.html#filter-design`. 7

[KW93] KNUTSSON H., WESTIN C.-F.: Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data. In *CVPR* (1993), pp. 515–523. 8

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R.: Interactive local adjustment of tonal values. *ACM TOG 25*, 3 (2006), 646–653. 9, 11

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM TOG 23* (2004), 689–694. 9, 11

[Mar01] MARVASTI F.: *Nonuniform sampling: theory and practice*, vol. 1. Springer, 2001. 4

[MAT14a] MATLAB: version 8.3 (R2014a) — Digital Filter Design documentation, 2014. `http://www.mathworks.com/help/signal/digital-filter-design.html`. 7

[MAT14b] MATLAB: version 8.3 (R2014a) — `d2c` documentation, 2014. `http://www.mathworks.com/help/control/ref/d2c.html`. 3

[NMLH11] NEHAB D., MAXIMO A., LIMA R. S., HOPPE H.: Gpu-efficient recursive filtering and summed-area tables. *ACM TOG 30* (2011), 176:1–176:12. 2, 8

[PHK11] PARIS S., HASINOFF S. W., KAUTZ J.: Local laplacian filters: Edge-aware image processing with a laplacian pyramid. *ACM TOG 30*, 4 (2011), 68:1–68:12. 10

[PM07] PROAKIS J. G., MANOLAKIS D. K.: *Digital Signal Processing: Principles, Algorithms, and Applications*. Pearson Education India, 2007. 2, 3, 4, 5, 6, 8, 13

[PO00] POULTON D., OKSMAN J.: Digital filters for non-uniformly sampled signals. In *Nordic Signal Processing Symposium* (2000), pp. 421–424. 2, 3, 8

[SS60] SHAPIRO H., SILVERMAN R.: Alias-free sampling of random noise. *Journal of the Society for Industrial & Applied Mathematics 8*, 2 (1960), 225–248. 2

[VVYV98] VAN VLIET L. J., YOUNG I. T., VERBEEK P. W.: Recursive gaussian derivative filters. In *ICPR* (1998), vol. 1, pp. 509–514. 6, 7

[Yan12] YANG Q.: Recursive bilateral filtering. In *ECCV 2012*, vol. 7572. 2012, pp. 399–413. 2

[YVVvG02] YOUNG I., VAN VLIET L., VAN GINKEL R.: Recursive gabor filtering. *IEEE TSP 50*, 11 (2002), 2798–2805. 7

Appendix A: Proof of Stability

A necessary and sufficient condition for a digital filter to be stable is that all its poles lie inside the unit circle $|z| < 1$ in the z domain [PM07]. The analytical stability of Eqs. 9 and 13 is proven below.

Proposition 1 For all real $\Delta t_k > 0$ and complex $a, b \neq 0$; if a filter $H(z) = \frac{a}{1-bz^{-1}}$ is stable (*i.e.*, $|b| < 1$), the filter $H'(z)$ derived from $H(z)$ in the way defined by Eq. 9 is also stable.

Proof. Replace the summation in Eq. 9 by the closed-form formula from Eq. 11. The z-domain transfer function of the resulting difference equation has the form

$$H'(z) = \frac{(a+R_0) - R_1 z^{-1}}{1 - b^{\Delta t_k} z^{-1}}.$$

The single pole of this equation is $b^{\Delta t_k}$. For all $\Delta t_k > 0$, we have that $|b^{\Delta t_k}| < 1$ since $|b| < 1$. Thus, $b^{\Delta t_k}$ lies inside the unit circle, and the filter defined by Eq. 9 is stable. □

In the same way one can easily show the analytical stability of Eq. 8 and consequently Eq. 13, since the value of $|\gamma_k|$ is non-zero for all k. Numerically, singularities in the sampling rate ($\Delta t_k \to 0$) may lead to instabilities, since the pole $b^{\Delta t_k}$ gets too close to the unit circle. However, for the applications shown in the paper, we did not experience any numerical issues. Nonetheless, we recommend using 64-bit floating point precision for computations.

Appendix B: Derivation of an $O(N)$-time, non-uniform Gaussian filter with normalization preserved by piecewise resampling.

B.1. Uniform Recursive Gaussian Filtering

Deriche [Der93] gives the following approximation to the positive region ($x \geq 0$) of a unit-height Gaussian of standard deviation σ:

$$u^+(x) = \mathrm{Re}\left\{\alpha_0 \exp\left(-\frac{\lambda_0}{\sigma}x\right) + \alpha_1 \exp\left(-\frac{\lambda_1}{\sigma}x\right)\right\}, \quad (23)$$

where $\mathrm{Re}\{\cdot\}$ denotes the real part of a complex number, and

$$\alpha_0 = \quad 1.6800 + 3.7350\,j, \quad \lambda_0 = 1.783 + 0.6318\,j,$$
$$\alpha_1 = -0.6803 + 0.2598\,j, \quad \lambda_1 = 1.723 + 1.9970\,j.$$

The symmetric kernel is built by combining the positive half u^+ with the negative one $u^-(x) = u^+(-x)$, yielding an undistinguishable approximation to a Gaussian (mean squared error under 2.5×10^{-8}):

$$e^{\frac{-x^2}{2\sigma^2}} \approx u(x) = \begin{cases} u^+(x) & x \geq 0, \\ u^-(x) & x < 0. \end{cases}$$

In his work, Deriche implements a filter with kernel $u(x)$ by expanding the complex exponentials in Eq. 23 into their composing sines and cosines, and extracting the real part. This yields a causal 4th-order recursive system for u^+ and an anti-causal one for u^-.

Note that this recursive Gaussian filter, as described by Deriche, only works for uniformly sampled signals. Next, we use our mathematical formulation to generalize the filter to work in non-uniform domains.

B.2. Non-Uniform Recursive Gaussian Filtering

Different from Deriche, we work directly with the complex exponentials in Eq. 23, and extract the real part after filtering. The causal and anti-causal (complex) filter transfer functions are, respectively,

$$U^+(z) = \frac{\alpha_0}{1 - e^{-\lambda_0/\sigma} z^{-1}} + \frac{\alpha_1}{1 - e^{-\lambda_1/\sigma} z^{-1}}, \text{ and}$$
$$U^-(z) = \frac{\alpha_0\, e^{-\lambda_0/\sigma} z}{1 - e^{-\lambda_0/\sigma} z} + \frac{\alpha_1\, e^{-\lambda_1/\sigma} z}{1 - e^{-\lambda_1/\sigma} z};$$

which are already decomposed into 1st-order filters. Note that U^- is designed to ignore the central sample, as described in Section 3.2.6.

Since the Gaussian is a low-pass filter, it should be normalized to unit gain at zero-frequency ($\omega_* = 0$), which is equivalent to saying its kernel should have unit-area. However, the kernel defined by Eq. 23, and implemented by $U(z) = U^+(z) + U^-(z)$, is not unit-area. A unit-area filter $H(z)$ is obtained as $H(z) = U(z)/|\gamma|$ where $|\gamma|$ is the gain of filter $U(z)$ at zero frequency, given by:

$$|\gamma| = |U(e^{j\omega_*})|\Big|_{\omega_*=0} = \alpha_0 \frac{1 + e^{-\lambda_0/\sigma}}{1 - e^{-\lambda_0/\sigma}} + \alpha_1 \frac{1 + e^{-\lambda_1/\sigma}}{1 - e^{-\lambda_1/\sigma}}.$$

Using our methodology described in Section 3.2, the difference equation which implements our recursive non-uniform $O(N)$-time Gaussian filter with normalization preserved by piecewise resampling is:

$$g[k] = \sum_{i=0}^{1} \mathrm{Re}\left\{g_i^+[k] + g_i^-[k]\right\},$$

where

$$a_i = \alpha_i/|\gamma|,$$
$$b_i = e^{-\lambda_i/\sigma},$$
$$g_i^+[k] = a_i f[k] + b_i^{\Delta t_k} g_i^+[k-1] + \Phi_{k-1,k}(\Delta t_k),$$
$$g_i^-[k] = a_i b_i^{\Delta t_{k+1}} f[k+1] + b_i^{\Delta t_{k+1}} g_i^-[k+1] + \Phi_{k+1,k}(\Delta t_{k+1}),$$
$$\Phi_{j,k}(\delta) = \left(\frac{b_i^\delta - 1}{r_0\,\delta} - r_1\, b_i\right) f[k] - \left(\frac{b_i^\delta - 1}{r_0\,\delta} - r_1\, b_i^\delta\right) f[j].$$

Relaxed boundary condition is obtained by setting out-of-bound values to zero: $f[-1] = f[N] = g_i^+[-1] = g_i^-[N] = 0$. Alternatively, replicated-boundary condition is given by:

$$f[-1] = f[0], \qquad g_i^+[-1] = \frac{a_i}{1 - b_i} f[0],$$
$$f[N] = f[N-1], \qquad g_i^-[N] = \frac{a_i\, b_i}{1 - b_i} f[N-1].$$