Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 89170, 9 pages
doi:10.1155/2007/89170

Research Article

A Fast Mellin and Scale Transform

Antonio De Sena[1] and Davide Rocchesso[2]

[1] *Dipartimento di Informatica, Università di Verona, Strada Le Grazie, 15-37134 Verona, Italy*
[2] *Dipartimento di Arti e Disegno Industriale, Università Iuav di Venezia, Dorsoduro 2206, 30123 Venezia, Italy*

Received 24 August 2006; Revised 30 December 2006; Accepted 5 March 2007

Recommended by Jar-Ferr Kevin Yang

A fast algorithm for the discrete-scale (and β-Mellin) transform is proposed. It performs a discrete-time discrete-scale approximation of the continuous-time transform, with subquadratic asymptotic complexity. The algorithm is based on a well-known relation between the Mellin and Fourier transforms, and it is practical and accurate. The paper gives some theoretical background on the Mellin, β-Mellin, and scale transforms. Then the algorithm is presented and analyzed in terms of computational complexity and precision. The effects of different interpolation procedures used in the algorithm are discussed.

1. INTRODUCTION

The Mellin transform, and the particular version called scale transform, can represent a signal in terms of scale. The scale can be interpreted, similarly to frequency, as a physical attribute of signals. The proposed fast (subquadratic) implementation allows this transform to be used in practical applications. The algorithm can compute the Mellin transform

$$M_f(p) = \int_0^\infty f(t)t^{p-1}\mathrm{d}t, \tag{1}$$

in the complex variable $p = -jc + \beta$, with $\beta \in \mathbb{R}$ fixed parameter and $c \in \mathbb{R}$ independent variable. We call this family of transforms the β-Mellin transform. It is indeed a restriction of the Mellin transform, as the real part of the complex variable p is parameterized. The β parameter allows to select among: (i) a scale-invariant transform ($\beta = 1/2$, scale transform); (ii) a compression/expansion invariant transform ($\beta = 0$); (iii) a shape-invariant transform ($\beta = -1$, the ratio between the maximum of the function and its extension is a constant).

The proposed algorithm is based on the well-known relation between the Mellin and Fourier transforms. While methods that exploit such relation have been proposed long ago [1, 2], efficiency and practicality are still remarkable objectives to be achieved.

Mellin and scale transforms are important in vision and image processing. In particular, a so-called Fourier-Mellin transform can be used for pattern recognition for its invariance to shift, scale, and rotation. In [3], various techniques have been presented for the implementation of the Fourier-Mellin transform, including a polar-log coordinates remapping. In [4], the problem of estimation of scale and orientation differences between objects in images has been approached using the analytical Fourier-Mellin transform [3].

Other approaches to the Mellin transform implementation have been taken by J. Bertrand et al. [5–7]. In their studies, the authors tackled the transform in the frequency domain by considering analytic signals. An implementation based on exponential resampling in the time domain should give a solution to a few practical problems. Namely, a starting point near 0 implies an impossible exponential resampling, and if the signal support in time is very small compared to the starting point of the signal, the exponential sampling becomes a quasiuniform sampling. An implementation that follows this idea has been made by Gonçalvés and Lemoine (http://gdr-isis.org/tftb/refguide/node32.html), but the algorithm appears to be quadratic in complexity. The authors have searched for other implementation of J. Bertrand et al. fast Mellin transform idea, but no sub-quadratic implementation has been found.

In our work, that proceeds in the time domain by creating parallels with Fourier-based theories, we tried to find practical solutions for exponential sampling, while simultaneously keeping the whole framework as simple as possible. In particular, the resampling process does not pose problems regarding the relative small length of the signal because there

is no prebuilt exponential grid, and the exponential warping is adapted to the signal under analysis.

We are mainly interested in applications of the scale transform in the realm of speech and audio processing, where it can be used for various purposes, like scale normalization, signal analysis in the scale domain [8] (scale can be considered as a joint time-frequency attribute), audio manipulation in scale domain [9], and vowel recognition [10].

In Section 2, an introduction to the Mellin and scale transforms will be given along with the definitions of *scale periodicity* and an interpretation of the scale transform. Analogies and relations with the Fourier transform will also be provided. An exponential sampling theorem (an extension of the one provided in [11]) will be presented. This section is the collection of known concepts and new definitions and extensions useful as support for the β-Mellin transform implementation. In Section 3, the base theory for the implementation of the fast Mellin transform will be provided. Exponential sampling will be introduced and sinc, cardinal spline, and spline interpolations will be discussed. In Section 4, the implemented algorithm, its computational cost, and an error analysis will be described.

2. THE MELLIN TRANSFORM

Originally developed by Robert Hjalmar Mellin (1854–1933) for the study of the gamma function, hypergeometric functions, Dirichlet series, the Riemann zeta function and for the solution of partial differential equations, the Mellin transform was also used in electrical engineering, for example for studying motor control systems [12]. In [8], Cohen introduced the "scale transform." This transform is said to be scale-invariant (the Fourier transform is shift-invariant), thus meaning that the signals differing just by a scale transformation (compression or expansion with energy preservation) have the same transform magnitude distribution. Cohen showed that the scale transform is a restriction of the Mellin transform on the vertical line $p = -jc + 1/2$, with $c \in \mathbb{R}$.

2.1. Definition and existence of Mellin transform

The Mellin transform of a function f is defined as in (1), where $p \in \mathbb{C}$ is the Mellin variable.

The existence of the Mellin transform (1) depends on convergence of the transform integral,

$$\int_0^\infty |f(t)|\, t^{p-1} \mathrm{d}t < \infty. \tag{2}$$

This is a general sufficient condition for the existence of the transform. Further considerations [5] can be made using the fact that $p = -jc + \beta$, and different or simpler forms of (2) can be derived.

2.2. Definition of scale transform

The scale transform [8] is a particular restriction of the Mellin transform on the vertical line $p = -jc + 1/2$, with $c \in \mathbb{R}$, just as the Fourier transform can be seen as a restriction of the Laplace transform on the imaginary axis. Thus, the scale transform is defined[1] as

$$D_f(c) = \frac{1}{\sqrt{2\pi}} \int_0^\infty f(t) \mathrm{e}^{(-jc-1/2)\ln t} \mathrm{d}t. \tag{3}$$

The scale inverse transform is given by

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^\infty D_f(c) \mathrm{e}^{(jc-1/2)\ln t} \mathrm{d}c. \tag{4}$$

The key property of the scale transform is its scale invariance. This means that if f is a function and g is a scaled version of f, the transform magnitude of both functions is the same. A scale modification is a compression or expansion of the time axis of the original function that preserves signal energy. Thus, a function $g(t)$ can be obtained with a scale modification from a function $f(t)$ if $g(t) = \sqrt{\alpha} f(\alpha t)$, with $\alpha \in \mathbb{R}^+$. When $\alpha < 1$, we get a scale expansion, when $\alpha > 1$ we get a scale compression. Given a scale modification with parameter α, the scale transforms of the original and scaled signals are related by

$$D_g(c) = \alpha^{jc} D_f(c). \tag{5}$$

This property derives from a similar property of the Mellin transform. In fact, if $h(t) = f(\alpha t)$, then

$$M_h(p) = \alpha^{-p} M_f(p). \tag{6}$$

In both (5) and (6), scaling is reflected by a multiplicative factor for the transforms, and for (5) such factor reduces to a pure phase contribution. So, the scale transform magnitudes of the original and scaled signals are the same,

$$|D_g(c)| = |D_f(c)|. \tag{7}$$

2.3. Relation with the Fourier transform

From its definition and interpretation, the Mellin transform provides a tight correspondence with the Fourier transform [10]. More precisely, the Mellin transform with parameter $p = -jc$ can be interpreted as a logarithmic-time Fourier transform:

$$M_f(c) = \int_{-\infty}^\infty f(t) \mathrm{e}^{-jc(\ln t)} \mathrm{d}(\ln t). \tag{8}$$

Similarly, we can define the scale transform of a function $f(t)$ using the Fourier transform of a function $g(t)$ [8] with $g(t)$ obtained from $f(t)$ by time-warping ($g(t) = f(\mathrm{e}^t)$):

$$M_f(c) = \int_{-\infty}^\infty g(t) \mathrm{e}^{-jct} \mathrm{d}t. \tag{9}$$

This result can be generalized for any p defined as $p = -jc + \beta$, with $\beta \in \mathbb{R}$, by using $g(t) = f(\mathrm{e}^t)\mathrm{e}^{\beta t}$.

[1] The heading $1/\sqrt{2\pi}$ is for energy normalization purpose.

2.4. Scale periodicity and scale transform interpretation

A parallel can be drawn between the properties of the Fourier and scale transforms. In particular, we can define *scale periodicity* [11] as follows: a function $f(t)$ is said to be scale periodic with period τ if it satisfies $f(t) = \sqrt{\tau} f(t\tau)$, where $\tau = b/a$, with a and b starting and ending points of the scale period. $C_0 = 2\pi/\ln \tau$ is the "fundamental scale" associated with the scale periodic function. By analogy with the Fourier theory, we can define a "scale series" and Parseval theorem. Of particular importance is the "exponential sampling theorem" [11] that, like the Nyquist-Shannon theorem, allows the reconstruction of a scale-band limited signal from its samples. These samples must be distributed exponentially in time according to positions $p_k = \tau_s^k$, with $k \in \mathbb{Z}$, $\tau_s = \mathrm{e}^{\pi/C_m}$, and C_m is the signal maximum scale.

A more general theorem can be formulated by working on β-Mellin (rather than scale) band-limited signals.

2.5. Exponential sampling theorem

Starting from what has been done for the scale transform [11], an extension/generalization of the exponential sampling theorem can be provided for all types of β-Mellin transforms.

Definition 1. The β-Mellin band of a function $f(t)$ is the support of $F(c)$, where $F(c)$ is the β-Mellin transform of $f(t)$.

Definition 2. A function $f(t)$ is β-Mellin band-limited to C_0 when $F(c) = 0$ for all $c \notin (-C_0, C_0)$, where $F(c)$ is the β-Mellin transform of $f(t)$.

Now the exponential sampling theorem for β-Mellin band-limited functions can be stated.

Theorem 1. *A function $f(t) \in L^2(\mathbb{R})$, β-Mellin band-limited[2] to C_0, can be exactly reconstructed from its samples in the time domain if the samples are spaced exponentially along the time axis as in $\{\tau^n\}_{-\infty}^{\infty}$, where $\tau = e^{2\pi/2C_0}$.*

A quick outline of proof can be given. The proof is similar to the one shown in [11] for the scale transform. We need to rebuild the equation chains using the β-Mellin related equations. Let $\psi(t)$ be the following function:

$$\psi(t) = \frac{2}{\sqrt{2\pi}} \frac{\sin(C_0 \ln t)}{\ln t} t^{-\beta}. \tag{10}$$

The β-Mellin transform of $\gamma_\alpha(t) = \alpha^\beta \psi(\alpha t)$ (i.e., γ is a β-scaled version of ψ), where $\alpha = \tau^m$, $\tau = \mathrm{e}^{2\pi/2C_0}$ and $m \in \mathbb{Z}$, is

$$\Gamma(c) = \begin{cases} \mathrm{e}^{jc\ln\alpha}, & |c| \le C_0, \\ 0 & \text{elsewhere.} \end{cases} \tag{11}$$

The β-Mellin transform of $f(t)$, indicated with $M_f^\beta(c)$, is a support-limited function by assumption. Then an expansion of $M_f^\beta(c)$ using Fourier series representation can be performed. The period (in the Fourier sense) of $M_f^\beta(c)$ is supposed to be $T = 2C_0$ (the whole support of $M_f^\beta(c)$, i.e., the bandwidth in β-Mellin domain of $f(t)$),

$$M_f^\beta(c) = \sum_{m=-\infty}^{\infty} a_m \mathrm{e}^{jc\ln\tau^m}, \tag{12}$$

with a_m defined as

$$a_m = \frac{1}{\sqrt{2\pi}} \ln(\tau) \mathrm{e}^{\beta \ln \tau^{-m}} f(\tau^{-m}). \tag{13}$$

Now, starting from the definition of inverse β-Mellin transform, and using (10)–(13), the reconstruction equation for exponential sampling can be obtained:

$$\begin{aligned} f(t) &= \frac{1}{\sqrt{2\pi}} \int_{-C_0}^{C_0} M_f^\beta(c) \mathrm{e}^{(jc-\beta)\ln t} \mathrm{d}c \\ &= \ln\tau \frac{1}{\sqrt{2\pi}} \sum_{m=-\infty}^{\infty} f(\tau^m) \psi(t\tau^{-m}). \end{aligned} \tag{14}$$

Equation (14) allows a perfect (in the Nyquist-Shannon sense) reconstruction of the signal starting from its (exponentially spaced) samples. Furthermore, (14) can be shown to be very close to the Nyquist-Shannon interpolation formula, in fact it can be rewritten as

$$f(t) = \sum_{m=-\infty}^{\infty} f(\tau^m) (t\tau^{-m})^{-\beta} \operatorname{sinc}(C_0 \ln(t\tau^{-m})). \tag{15}$$

The reconstruction function $(t\tau^{-m})^{-\beta} \operatorname{sinc}(C_0 \ln(t\tau^{-m}))$ is composed by a logarithmic-time sine cardinal function modulated by a power function. The summation (15) is made by summing β-dilatocyclic[3] versions of the reconstruction function weighted by each sample taken exponentially in time.

3. THE FAST MELLIN TRANSFORM

Computing a discrete Mellin transform is relatively straightforward. For example, we can do an approximation of the transform integral using the Riemann sum. Unfortunately, doing this would give us algorithms exhibiting quadratic complexity, thus meaning that they are not usable in most

[2] Obviously, the theorem assumes that the β-Mellin transform of $f(t)$ exists.

[3] Similar to the definition given in [5], a β-dilatocyclic signal is a signal composed by expanded/compressed replicas of a base signal, modulated/amplified by a function of β. This concept is in some way similar to the concept of periodic signal.

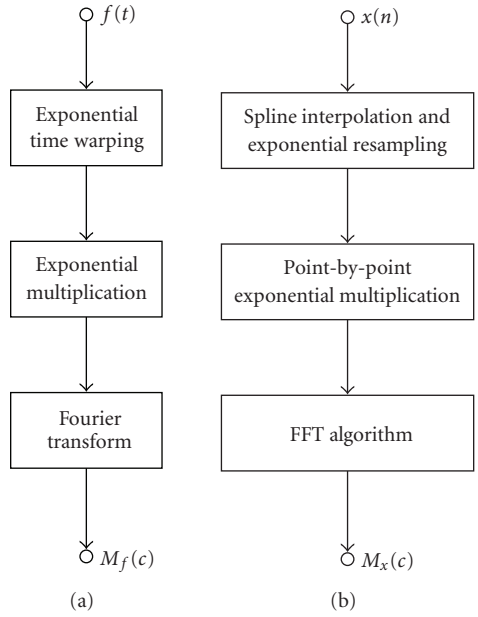

Figure 1: Block diagram of the fast Mellin transform idea (a) and the relative implementation blocks (b).

practical applications. The basic idea of the fast Mellin transform (FMT) algorithm comes from (9), in particular when $\beta = 1/2$ (scale transform). While presented in prior works (i.e., [1, 2, 13]), this idea is here used to build a practical and efficient computer program (in particular a Matlab toolbox). The algorithm approximates

$$M_f(c) = \int_{-\infty}^{\infty} f(\mathrm{e}^t)\mathrm{e}^{\beta t}\mathrm{e}^{-jct}\mathrm{d}t \tag{16}$$

by taking a uniformly sampled function, warping it exponentially, multiplying it by an exponential, and performing a fast Fourier transform (see Figure 1). Naturally, all the problems come from the warping operation. Once digitized, the signal must be resampled from a uniform to an exponential sampling grid. A resampling-based approach has been already studied in vision and image processing. In particular in [3], an implementation of the Fourier-Mellin transform of images has been presented, based on the idea of log warping, which can be dated back to [1, 2]. Conversely, in the implementation of the fractional Mellin transform [14], warping is done logarithmically instead of exponentially.

3.1. Sampled signal

In practical applications, the original analog signal is hardly available, because a uniform-sampling stage is inherent in the acquiring process. So, the raw material is the Shannon-sampled version of a Fourier band-limited signal. The Nyquist-Shannon theorem tells us that in this condition, we can reconstruct the original (analog) signal from the sampled version. This implies that we can resample the original (analog) signal in a different way. In particular, after resampling the signal exponentially (see Section 3.2), two interpretations can be used. The first interpretation is based on an exponentially sampled signal view in which we know that the signal must be considered along a warped time axis. In that view, the signal is a Mellin (more precisely β-Mellin) band-limited signal. In this case, for example, a single-cycle sinusoid can still be plotted with the same shape as the original, but with a higher sample density near the signal start. The other interpretation is the time-warped uniformly sampled signal view. In that view, the warped signal is Fourier band-limited. In this case, for example, the shape of a single-cycle sinusoid will be heavily distorted. The assumptions underlying our implementation are that the signal is (i) time-limited because it is saved in a finite-dimensional storage system; (ii) Fourier band-limited because it results from uniform sampling under Nyquist-Shannon conditions; (iii) β-Mellin band-limited to have a finite number of points in the Mellin transform representation. These conditions are possible only if the original signal is thought as a single period of an infinitely long periodic signal (to have a Fourier band-limited signal) or as a single scale-period of a infinitely long β-dilatocyclic signal.

3.2. Exponential resampling

Several problems arise when making an exponential resampling starting from an unknown uniformly sampled signal: how many samples are needed, how they should be distributed over time, how the signal start time alters this information, how can we reconstruct the signal, and so forth. While being aware of prior answers [11, 13], we address these questions in this section.

First of all, we must fulfill the Nyquist-Shannon sampling condition, so the distance between two adjacent samples in the exponential resampling cannot exceed the distance of the original uniform sampling step. This means that the sampling period T_s is the upper limit for the distance between the last two contiguous samples in exponential resampling. The second constraint is that the resampling process must cover the entire signal, from its starting point to its ending point. The two constraints force us to have more samples in the exponential resampled signal. The original signal starting point t_0 is very important: in fact, the more t_0 is close to zero, the more samples are needed in the exponential resampling process. Thus, if we let $t_0 = 0$, we need an infinite number of exponential samples. So, for using this algorithm we need a starting point strictly greater than zero. We can write the exponential sampling like a sequence:

$$\{(\tau_s)^k\}_{k=-\infty}^{\infty}, \tag{17}$$

where k is the sample index and τ_s can be called the exponential base step. So, using the first[4] constraint ($\tau_s^{k_e} - \tau_s^{k_e-1} = T_s$), we can find

$$\tau_s = \frac{t_0 + nT_s}{t_0 + (n-1)T_s}, \tag{18}$$

[4] In theory, we should write $\tau_s^{k_e} - \tau_s^{k_e-1} \leq T_s$, but if we want to use as few samples as possible, we can use $\tau_s^{k_e} - \tau_s^{k_e-1} = T_s$. $\tau_s^{k_e}$ is the last sample and k_e is the last sample index (sample at the ending temporal point t_e).

where n is the number of samples of the uniformly sampled signal. Now, using the second constraint ($\tau_s^{k_0} = t_0 \wedge \tau_s^{k_e} = t_0 + nT_s = t_e$), the number of needed exponential samples is

$$eN = \frac{\ln\left((t_0 + nT_s)/t_0\right)}{\ln\left((t_0 + nT_s)/(t_0 + (n-1)T_s)\right)} + 1. \tag{19}$$

If we use T_s as the starting point (i.e., $t_0 = T_s$), we can obtain the lighter approximate expression

$$eN = \frac{\ln(n+1)}{\ln((n+1)/n)}. \tag{20}$$

Furthermore, if we use a high number of samples (in practice, e.g., a number greater than 16 is sufficient) we can approximate (20) in a very simple form [13] using a known important limit ($\lim_{x\to\infty}(1 + 1/x)^x = \mathrm{e}$):

$$eN = n \ln n. \tag{21}$$

Now we have the exponential sampling step and the number of samples needed, so we can proceed with resampling (see Figure 2).

3.3. Sinc, cardinal spline, and spline interpolation

Resampling (in a nonuniform way) an already sampled signal is not trivial. In theory, the Nyquist-Shannon sampling theorem tells that a signal, under well-known conditions, can be reconstructed from its samples using a sinc interpolation. Unfortunately, fast sinc interpolation on an exponential grid is cumbersome, even using lookup table [15]. In [16] (but also [17, 18] are important for more stable algorithms), an idea for reducing the theoretically infinite computation of a sinc interpolation to a finite summation has been presented, but the computation still requires a quadratic algorithm. A fast interpolation technique that can approximate sinc interpolation is *cardinal spline* interpolation [19]. This interpolation is a modified version of the cubic Hermite spline interpolation. The Hermite spline is a third-degree spline with each polynomial of the spline in Hermite form. The Hermite form consists of two control points and two control tangents for each polynomial. On each subinterval, the interpolating polynomial depends on the starting point p_i and an ending point p_{i+1}, with starting and ending tangents m_i and m_{i+1}, respectively. A cardinal spline is a cubic Hermite spline whose tangents are defined by the points and a tension parameter c. The tension allows the computation of the tangents. A general-purpose tension value can be 0.5 and the cardinal spline using this value is called Catmull-Rom spline. In the FMT algorithm, various values of tension have been tested along with other types of spline interpolation, in particular, natural cubic spline interpolation [19]. A natural cubic spline is a spline constructed of piecewise third-order polynomials which pass through a set of m control points. The second derivative of each polynomial is set to zero at the endpoints, and this provides a boundary condition that completes the system of $m - 1$ equations. This interpolation is simpler than cardinal spline, yet offering the same goodness of approximation.

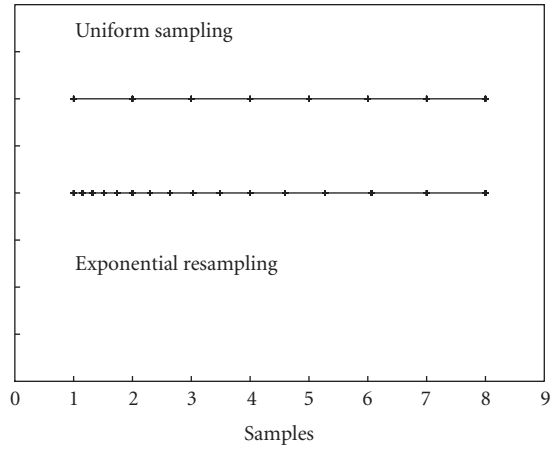

Figure 2: Uniform sampling and (critical) exponential resampling.

The use of cardinal spline interpolation is, from a theoretical point of view, a good choice. In fact, the cardinal spline, generated by cardinal B-spline [19], has a behavior similar to the sinc function. Like the sinc function, each cardinal spline vanishes at all integers except the origin, and the value at 0 is 1. Furthermore, at limit, the cardinal spline converges to the sinc function.

Eventually, it is an experimental analysis of errors that guides the choice of the interpolation method, as presented in Section 4.2, for different oversampling factors.

4. IMPLEMENTATION AND EXPERIMENTS

Using the ideas presented in Section 3, a fast Mellin transform has been developed. The algorithm takes a signal uniformly sampled and performs an exponential resampling. The signal is considered to be sampled at Nyquist frequency and, to obtain a good tradeoff between accuracy and speed, the number of new (exponential) samples used is $2eN$. Here, the starting point of the signal is considered to be T_s (the uniform sampling step), but in Section 4.2 a solution for computing the Mellin transform with a different starting point is given. The algorithm can use a natural spline interpolation or cardinal spline interpolation. Either solutions has a linear computational cost (natural spline interpolation is embedded in Matlab): more precisely, the asymptotic complexity is $\mathcal{O}(N)$, where N is the number of exponential samples. After resampling, an exponential point-by-point multiplication is performed (the $e^{\beta t}$ component of (16)) with a computational cost of $\mathcal{O}(N)$. Then a fast Fourier transform is computed. The FFT has a subquadratic computational cost, more precisely $\mathcal{O}(N \ln N)$ (see Figure 1). At last, an energy normalization is performed, again a linear operation ($\mathcal{O}(N)$). So the whole asymptotic complexity depends only on the FFT and is $\mathcal{O}(N \ln N)$. Written in terms of n (the initial number of uniform samples), the asymptotic complexity is $\mathcal{O}(n \ln^2 n)$.

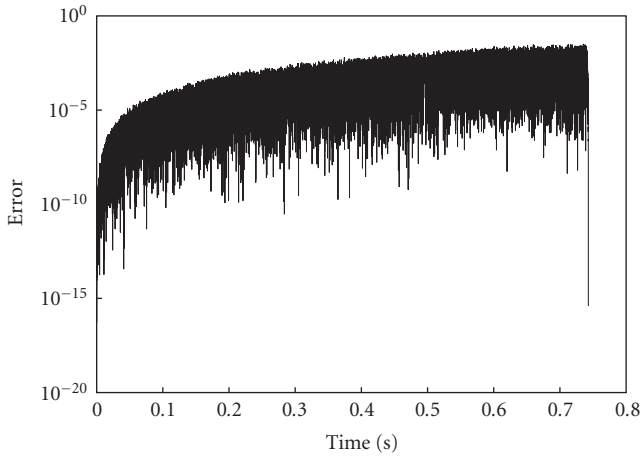

Figure 3: Reconstruction error for a white noise using twice as many samples as strictly needed ($2eN$).

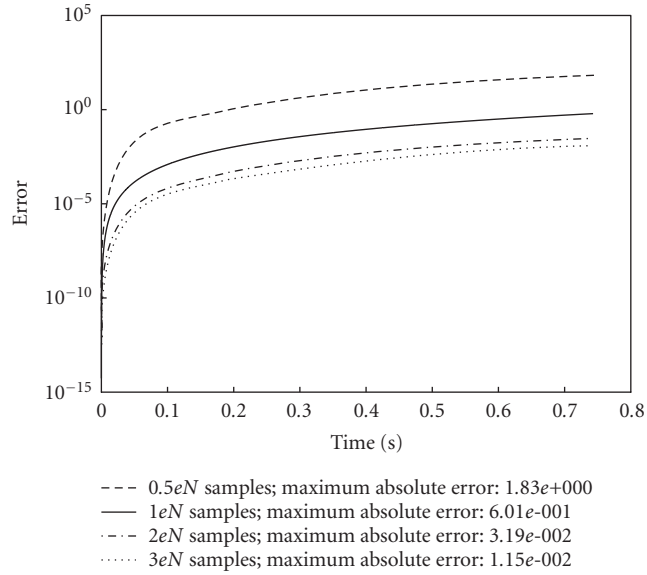

Figure 4: Error trend (in time) for a white noise. When taking twice as many samples as required ($2eN$), the maximum error of these signals goes towards 10^{-2}. Each curve is derived from piecewise-linear regression of the actual error curve, as the one shown in Figure 3.

4.1. Assumptions and approximations

The algorithm works using the assumptions and approximations presented in the previous sections and are summarized here.

First, there are errors due to quantization and finite-precision arithmetics. Then we can mention all the approximations bound to the algorithmic realization. Namely, spline interpolation introduces errors; (21) is a limit approximation; signals are supposed to start at $t_0 = T_s$, where T_s is the sampling period of the uniformly sampled signal; no information on Mellin bandwidth is typically available beforehand.[5]

4.2. Errors and reversibility

The algorithm is clearly based on subblocks: the interpolation block, the FFT block, and the multiplication and normalization blocks. In the case of complexity analysis, all the focus was on the FFT and on the relation between the number of uniform samples and the number of exponential samples. The error analysis, instead, is all focused on the interpolation block. Other computational errors are negligible. As it was explained in Section 3.3, the exponential distribution of samples and the need of a fast interpolation algorithm force us to choose an approximation for the sinc interpolation and this introduces errors.[6] Alternative distributions of interpolation nodes have been tried to reduce error, like Chebyshev or Leja nodes, but although the interpolation error becomes smaller, the displacement of the samples on the exponential grid is dramatically less accurate and this introduces an even larger error in the computation of the transform.

So, the preferred solution for error reduction is oversampling. Using more samples than those strictly required by the sampling theory allows the implemented transform to be more precise. This oversampling can be tuned with respect to the user needs. A good choice is to use twice as many samples as those required by theory. In this way, the maximum interpolation error goes towards 10^{-2} on amplitude-normalized signals. The "worst-case scenario" (shown in Figure 4) is when using signals that, in the final part of them, have frequency components near the Nyquist limit. Violet noise and white noise are simple examples that maximize the error, but it is sufficient that only the final part of the signal has high frequency components. In fact, at the end of the resampling grid, the samples are spaced as in the original uniformly sampled signal. So, in that region of the signal the exponential sampling is very close to the uniform and then you do not have the benefit of the oversampling and errors can be bigger. In the final part of the signal, the interpolator is an approximation of the theoretical sinc interpolator. The closer the frequencies are to the Nyquist limit, the more the difference between spline and sinc interpolators is noticeable (see Figures 6 and 7). In Figure 3, the reconstruction error is shown, while in Figure 4 the curves show the trends of the reconstruction errors as obtained from piecewise linear regression on log-scale plots. From a computational point of view, the oversampling introduces a multiplication by a constant to the number of exponential sampling points, so the asymptotic complexity remains $\mathcal{O}(n \ln^2 n)$.

In Figure 5, a short-time SNR plot has been drawn. The plot shows how the SNR varies over time for a white noise

[5] Indeed, the unknown Mellin bandwidth can be approximately computed after exponential warping.

[6] Actually, true sinc interpolation would also introduce errors, due to the intrinsic problem of the noninfiniteness of the computer-computed sinc function.

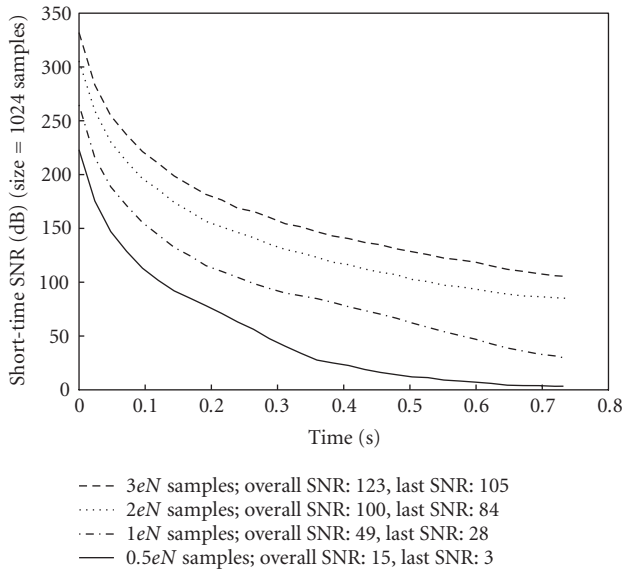

Figure 5: SNR over time for four different oversampling factors. Test signal: white noise, 16 bits, 44100 Hz, 65536 samples.

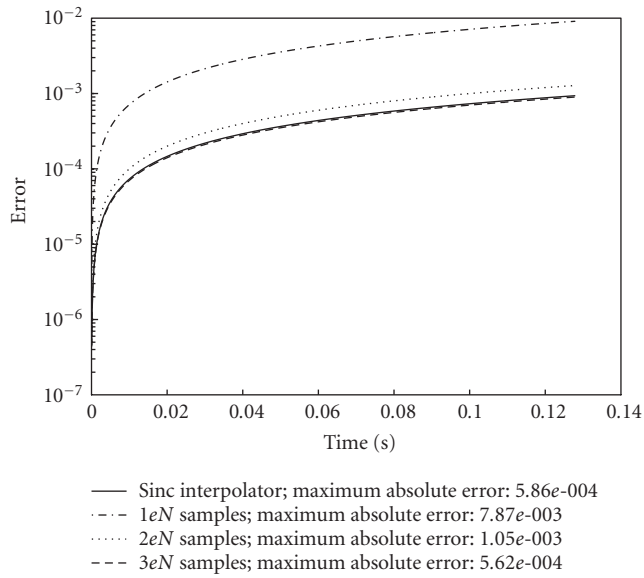

Figure 6: Error trend over time for three different oversampling factors and a sinc interpolator. Test signal: sparrow chirp, 16 bits, 16000 Hz, 2048 samples.

Table 1: Elapsed times in seconds. Times for complete operation (including loading file from disk).

Number of samples	Oversampling			
	$3eN$	$2eN$	$1eN$	$0.5eN$
2^{18}	306.594	56.703	45.297	8.297
2^{16}	9.953	6.735	3.531	2.328
2^{14}	2.218	1.312	0.656	0.359
2^{12}	0.782	0.297	0.140	0.110
2^{10}	0.515	0.094	0.047	0.046
2^{08}	0.453	0.047	0.031	0.031

Table 2: Elapsed times in seconds. Times for FMT algorithm only.

Number of samples	Oversampling			
	$3eN$	$2eN$	$1eN$	$0.5eN$
2^{18}	167.735	39.625	17.157	6.890
2^{16}	7.172	4.985	2.640	1.640
2^{14}	1.890	1.078	0.547	0.390
2^{12}	0.484	0.203	0.093	0.062
2^{10}	0.250	0.047	0.032	0.016
2^{08}	0.219	0.015	0.016	0.016

signal. In this example, the overall SNR for three-times oversampling ($3eN$) is 123 dB, for $2eN$ is 100 dB, for $1eN$ is 49 dB, and for $0.5eN$ is 15 dB.

Tables 1 and 2 give a short snapshot of the algorithm performance. Data of the first table are recorded elapsed times for entire operations (from loading a wave file to the end of the transform computation, including separation of phase and magnitude, etc.), while data from the second table are recorded elapsed times only for the FMT algorithm without any other operation. The first and the last rows of each table are affected by machine limitations. In fact for $n = 2^{18}$, the machine begins to paginate memory to disk so the performance is heavily affected. For $n = 2^8$, secondary operations unbound to the algorithm result to be heavier than the algorithm itself. The machine that has been used for the tests was a notebook with 3 Ghz Pentium 4 processor, with 512 Mb of RAM running Matlab 7R14 for WindowsXP.

Theoretically, the most accurate interpolation is sinc interpolation. So, we compared cardinal spline interpolation and sinc interpolation. Results can be viewed in Figures 6 and 7, computed using real recording (sparrow chirp) containing frequencies near Nyquist limit (signal sampled at 16 kHz). The experiments showed that in every case, a non-efficient interpolation (i.e., $\mathcal{O}(n^2)$ complexity) is too slow and not practical. For example, analyzing 8192 samples required 202 minutes. Conversely, the factor-3 oversampling with spline interpolation is almost as accurate as sinc interpolation.

The FMT is reversible (if the Mellin transform of a function exists, then the inverse of the transformed function also exists) and an IFMT algorithm has been implemented. The IFMT is entirely based on the FMT. The only caveat is in the computation of the inverse of the equation $N = n \ln n$, which can be performed with a bisection method. The interpolation scheme is the same as the one used in the FMT, and the process of interpolation is simply reversed. Alternatively, backward interpolation can proceed by warping linear time to logarithmic time. Again, the error is totally due to interpolation. The pairs transform and antitransform allow us to work in the Mellin domain and then go back to the original time domain [9].

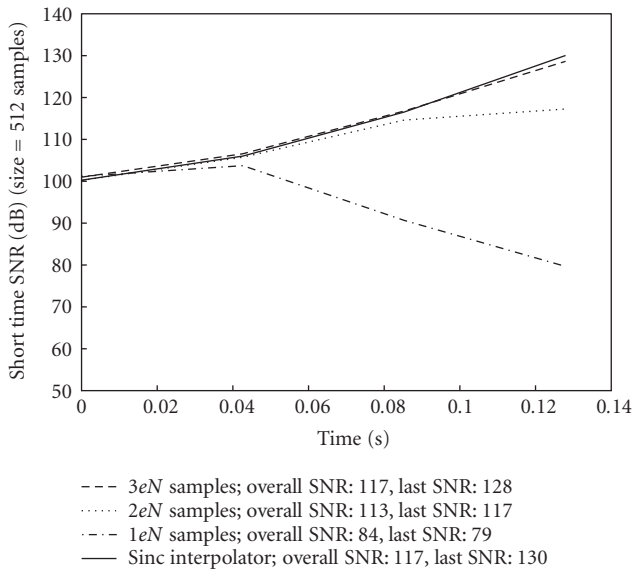

Figure 7: SNR over time for three different oversampling factors and a sinc interpolator. Test signal: sparrow chirp, 16 bits, 16000 Hz, 2048 samples.

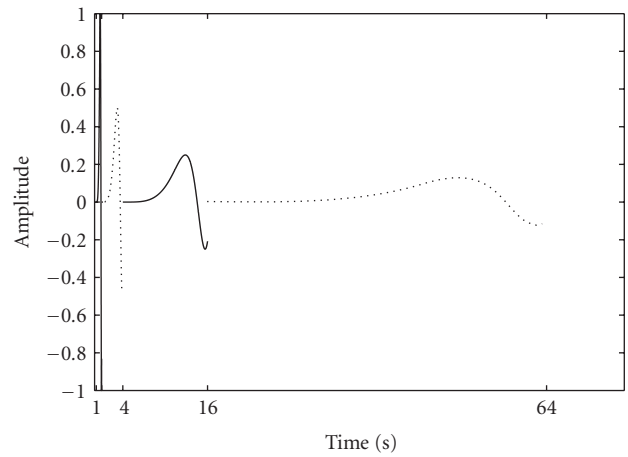

Figure 8: Scale shifts tuned with the scale period of the original signal (the original signal starts at 1 second and ends at 4 seconds). One scale-compressed version with the same scale period has been reproduced from 0.25 second to 1 second, and two scale-expanded versions with the same scale period are reproduced from 4 seconds to 16 seconds, and from 16 seconds to 64 seconds.

4.3. Scale shifting and hybrid transform

The FMT works under the assumption that the signal starts from T_s, where T_s is the uniform sampling interval. The impact of this hypothesis can be important, especially when the transform is used for scale analysis. In fact, the starting point of the signal changes the associated Mellin distribution, because the Mellin is not shift-invariant. If the objective is to analyze just the Mellin magnitude, a simple *scale shifting* can be done. This means that the signal in the original domain must be shifted and scaled according to its scale period. The scale period, in the case of an unknown finite-length signal, is the ratio between the ending instant and the starting instant. When shifting the signal to a new starting point (T_s for our purposes), the ratio must be still the same, so the signal must be scaled, that is, compressed or expanded preserving the total energy of the signal. However, this solution presents problems if phase analysis is needed or if the original signal starts near zero, as in the limiting zero case, the scale-periodicity is not computable. To avoid these problems, the scale shift must be done with a granularity computed according to the scale period (see Figure 8), thus implying that zero padding will be necessary to compensate the differences between the obtained point and the wanted starting point. Moreover, if the starting point is far from T_s, the required sampled frequency becomes too high, thus becoming unpractical.

If the signal starts exactly at zero, a hybrid approach can be pursued: the part of signal from 0 to T_s can be transformed directly. For example, we can consider the signal constant in the one-sample interval between 0 and T_s, and proceed by explicit area computation. This initial contribution can be summed with the FMT of the remaining part of the signal starting from T_s. In conclusion, the algorithm can be extended to afford the choice of the starting point, possibly setting it to multiples of T_s. Nevertheless, if the transform is used only for scale normalization or for filtering or recognition applications, the starting point looses importance.

4.4. Availability of the code

A matlab implementation of the FMT and some processing examples are freely available at http://profs.sci.univr.it/~desena/FMT.

5. CONCLUSIONS

This paper proposed a fast algorithm for the discrete-scale (and β-Mellin) transform. The idea is based on the well-known relation between the Mellin and Fourier transforms, and has been developed to be practical and accurate. As opposed to other implementations, this work tries to solve the problem entirely in the time domain by choosing an efficient, yet accurate, exponential resampling process. The proposed algorithm has been analyzed in terms of computational complexity and precision. In particular, the fast algorithm has been compared with a nonapproximated interpolation solution.

ACKNOWLEDGMENTS

The authors would like to thank Stefano De Marchi for his help with interpolation methods, and Carlo Drioli for the discussions on nonuniform sampling theory.

REFERENCES

[1] D. Casasent and D. Psaltis, "Position, rotation, and scale invariant optical correlation," *Applied Optics*, vol. 15, no. 7, pp. 1795–1799, 1976.

[2] D. Casasent and D. Psaltis, "New optical transforms for pattern recognition," *Proceedings of the IEEE*, vol. 65, no. 1, pp. 77–84, 1977.

[3] S. Derrode and F. Ghorbel, "Robust and efficient Fourier-Mellin transform approximations for gray-level image reconstruction and complete invariant description," *Computer Vision and Image Understanding*, vol. 83, no. 1, pp. 57–78, 2001.

[4] S. Derrode and F. Ghorbel, "Shape analysis and symmetry detection in gray-level objects using the analytical Fourier-Mellin representation," *Signal Processing*, vol. 84, no. 1, pp. 25–39, 2004.

[5] J. Bertand, P. Bertrand, and J. P. Ovarlez, "The Mellin transform," in *The Transforms and Applications Handbook*, A. D. Poularikas, Ed., The Electrical Engineering Handbook, pp. 11-1–11-68, CRC Press LLC, Boca Raton, Fla, USA, 1995.

[6] J. Bertrand, P. Bertrand, and J. P. Ovarlez, "Discrete Mellin transform for signal analysis," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '90)*, vol. 3, pp. 1603–1606, Albuquerque, NM, USA, April 1990.

[7] J. P. Ovarlez, J. Bertrand, and P. Bertrand, "Computation of affine time-frequency distributions using the fast Mellin transform," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '92)*, vol. 5, pp. 117–120, San Francisco, Calif, USA, March 1992.

[8] L. Cohen, "The scale representation," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3275–3292, 1993.

[9] A. De Sena and D. Rocchesso, "A fast Mellin transform with applications in DAFx," in *Proceedings of the 7th International Conference on Digital Audio Effects (DAFx '04)*, pp. 65–69, Napoli, Italy, October 2004.

[10] T. Irino and R. D. Patterson, "Segregating information about the size and shape of the vocal tract using a time-domain auditory model: the stabilised wavelet-Mellin transform," *Speech Communication*, vol. 36, no. 3-4, pp. 181–203, 2002.

[11] H. Sundaram, S. D. Joshi, and R. K. P. Bhatt, "Scale periodicity and its sampling theorem," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1862–1865, 1997.

[12] F. Gerardi, "Application of Mellin and Hankel transforms to networks with time-varying parameters," *IRE Transactions on Circuit Theory*, vol. 6, no. 2, pp. 197–208, 1959.

[13] E. J. Zalubas and W. J. Williams, "Discrete scale transform for signal analysis," in *Proceedings of the 20th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '95)*, vol. 3, pp. 1557–1560, Detroit, Mich, USA, May 1995.

[14] E. Biner and O. Akay, "Digital computation of the fractional Mellin transform," in *Proceedings of the 13th European Signal Processing Conference (EUSIPCO '05)*, Antalya, Turkey, September 2005.

[15] J. O. Smith, *Digital Audio Resampling Home Page*, January 2002.

[16] T. Schanze, "Sinc interpolation of discrete periodic signals," *IEEE Transactions on Signal Processing*, vol. 43, no. 6, pp. 1502–1503, 1995.

[17] F. Candocia and J. C. Principe, "Comments on "sine interpolation of discrete periodic signals"," *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 2044–2047, 1998.

[18] S. R. Dooley and A. K. Nandi, "Notes on the interpolation of discrete periodic signals using sinc function related approaches," *IEEE Transactions on Signal Processing*, vol. 48, no. 4, pp. 1201–1203, 2000.

[19] M. Unser, "Splines: a perfect fit for signal and image processing," *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, 1999.

Antonio De Sena received the Laurea degree in computer science in 2004 from the University of Verona, Department of Computer Science, where he is now a Ph.D. student. He worked at the University of Verona under a research contract between May 2004 and December 2004. In 2007, he has been visiting the Hunter College, City University of New York, for several months of studies. His works are related to sound processing and analysis. In particular, he is interested in the Mellin transform and the scale transform applied to digital audio filtering and effects, speech recognition, and time-frequency analysis.

Davide Rocchesso received the Laurea degree in electrical engineering and the Ph.D. degree from the University of Padova, Italy, in 1992 and 1996, respectively. In 1994 and 1995, he was a Visiting Scholar at the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University. Since 1991, he has been collaborating with the Center of Computational Sonology (CCS), University of Padova, as a Researcher and Live-Electronic Designer. Between 1998 and 2006, he has been with the University of Verona, Italy, as an Assistant and Associate Professor. At the Computer Science Department of the University of Verona, he coordinated the EU Project Sounding Object. He is now Associate Professor at the Department of Art and Industrial Design of the IUAV University of Venice. He launched the EU COST Action Sonic Interaction Design (SID). His main interests are in audio signal processing, physical modeling, and interaction design.