ELSEVIER

Signal Processing 82 (2002) 749–757

SIGNAL PROCESSING

www.elsevier.com/locate/sigpro

Continuous wavelet transform with arbitrary scales and O(N) complexity

Arrate Muñoz *, Raphaël Ertlé , Michael Unser

Department of Microtechnology, Biomedical Imaging Group/DMT, Swiss Federal Institute of Technology Lausanne, CH-1015 Lausanne EPEL, Switzerland

Abstract

The continuous wavelet transform (CWT) is a common signal-processing tool for the analysis of nonstationary signals. We propose here a new B-spline-based method that allows the CWT computation at any scale. A nice property of the algorithm is that the computational cost is independent of the scale value. Its complexity is of the same order as that of the fastest published methods, without being restricted to dyadic or integer scales. The method reduces to the filtering of an auxiliary (pre-integrated) signal with an expanded mask that acts as a kind of modified 'à trous' filter. The algorithm is well-suited for a parallel implementation. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Continuous wavelet transform; Arbitrary scale; B-spline

1. Introduction

The continuous wavelet transform (CWT) of a signal f with the wavelet ψ is defined as

$$W_\psi f(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} f(x)\psi\left(\frac{b-x}{a}\right) \mathrm{d}x. \qquad (1)$$

It can be interpreted as the correlation of the input signal with a time-reversed version of ψ rescaled by a factor of a. For a 1-D input signal, the result is a 2-D description of the signal with respect to time b and scale a. The scale a is inversely proportional to the central frequency of the rescaled wavelet $\psi_a(x) = \psi(x/a)$ which is typically a bandpass function; b represents the time location at which we analyze the signal. The larger the scale a, the wider the analyzing function ψ_a, and hence smaller the corresponding analyzed frequency. The output value is maximized when the frequency of the signal matches that of the corresponding dilated wavelet. The main advantage over the Fourier transform (FT) analysis is that the frequency description is localized in time. The advantage over the short-time Fourier transform (STFT) is that the window size varies; low frequencies are analyzed over wide time windows, and high frequencies over narrow time windows, which is more effective than to use a fixed-size analysis. Typical applications of the CWT are the detection and characterization of singularities [3,14], pattern recognition [6], image processing [4,15], fractal analysis [2,12,23], noise reduction [11] and the analysis of biomedical signals [7,10,25].

The main contribution of this paper is the development of a fast algorithm for the computation of the CWT at any real scale a and integer time localization b. Mallat's fast wavelet algorithm [12] uses the multiresolution properties of the wavelet to compute the

* Corresponding author. Tel.: +41-21-693-5142; fax: +41-21-69-3701.

E-mail addresses: arrate.munoz@epfl.ch (A. Muñoz), raphael.ertle@compaq.com (R. Ertlé), michael.unser@epfl.ch (M. Unser).

0165-1684/02/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0165-1684(02)00140-8

CWT at dyadic scales $a = 2^i$ and time shifts $b = 2^i k$, $k \in Z$ [17]; it achieves an overall O(N) complexity. Other techniques compute the wavelet transform at dyadic scales and integer time points with an '*à trous*' approach. Their complexity per scale is O(N), the same as Mallat's algorithm, but with a larger leading constant [3,5,9,16].

Despite their speed, these methods may not be precise enough for some applications, since a dyadic scale progression cannot be finer than an octave sub-band decomposition. To achieve a better scale resolution, other approaches have been proposed, either based on M-band decomposition inside an octave [17,24] or on a generalization of the two-scale relation to general integer N-scale relations [8,22]. However, none of these algorithms can handle arbitrary scales.

Our purpose here is to develop a novel and fast algorithm that works for any real value of a. It takes advantage of a B-spline decomposition of the input signal and of the mother wavelet. The method exploits the fact that B-splines are compactly supported and that the convolution of two B-splines can be expressed analytically [13].

2. Operators and definitions

First, we introduce some operators and definitions that will be helpful to solve our problem.

We express a B-spline of degree n as

$$\beta^n(x) = \Delta^{n+1} * \frac{x_+^n}{n!} * \delta\left(x + \frac{n+1}{2}\right), \tag{2}$$

where $x_+^n = \max(x, 0)^n$ is a one-sided power function; Δ^{n+1} denotes the $(n+1)$-fold iteration of the finite difference operator $\Delta = \delta(x) - \delta(x-1)$. This latter operator also corresponds to a discrete convolution (digital filter) whose z-transform is $\Delta(z) = 1 - z^{-1}$. Likewise, $\Delta^{n+1}(z) = (1 - z^{-1})^{n+1}$.

We have the equivalence

$$D^{-(n+1)} f(x) = \frac{x_+^n}{n!} * f(x), \tag{3}$$

where D^{-1} is the continuous integral operator defined as $D^{-1} f(x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t$. We are interested in the analytic formula of the B-spline expanded by a factor a. We will represent the expansion operator as $-\!\!\textcircled{\%a}\!\!-$. We derive the expression by using the exchange rules for the one-sided power functions and for the shift

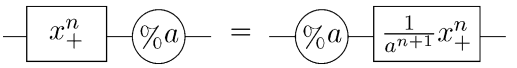

Fig. 1. Exchange rule for x_+^n.

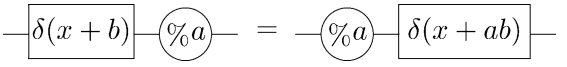

Fig. 2. Exchange rule for a shift by b.

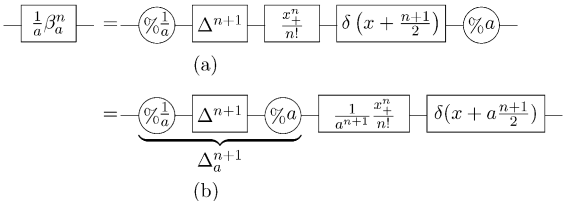

Fig. 3. Diagram that shows how to derive the analytic expression for the B-spline expanded by a factor a. (a) Substitution of β^n by its time-domain explicit expression. (b) Application of the scale change rules for the one-sided power function and the shift (see Figs. 1 and 2).

given in Figs. 1 and 2. The respective proofs can be found in [13]. The result as shown in Fig. 3 is

$$\frac{1}{a}\beta^n\left(\frac{x}{a}\right) = \Delta_a^{n+1} * \frac{1}{a^{n+1}} \frac{x_+^n}{n!} * \delta\left(x + a\frac{n+1}{2}\right), \tag{4}$$

where Δ_a is the rescaled finite-difference operator,

$$\Delta_a^{n+1}(x) = \sum_{k=0}^{n+1} \underbrace{\binom{n+1}{k}(-1)^k}_{q(k)} \delta(x - ak). \tag{5}$$

Using equivalence (3), the rescaled B-spline is rewritten as

$$\frac{1}{a}\beta^n\left(\frac{x}{a}\right) = \Delta_a^{n+1} * \frac{1}{a^{n+1}} D^{-(n+1)} \delta\left(x + a\frac{n+1}{2}\right). \tag{6}$$

From definition (2) of the B-spline and from equivalence (3), we find that the integral of a B-spline of a given degree is a spline with the degree increased by 1. It is given by the following expression (see [13]

for the proof)

$$D^{-(n_1+1)}\beta^{n_2}(x) = \Delta^{-(n_1+1)} * \beta^{(n_1+n_2+1)}\left(x - \frac{n_1+1}{2}\right), \quad (7)$$

where Δ^{-1} is the inverse finite-differences operator defined as $\Delta^{-1}(x) = \sum_{n\geqslant 0}\delta(x-n)$ whose z-transform is $\Delta^{-1}(z) = (1-z^{-1})^{-1}$. It can be defined as the running sum filter, $(\Delta^{-1}s)_k = \sum_{n\leqslant k} s_n$. Note that $y_k = (\Delta^{-1} * s)_k$ can be implemented very efficiently using the recursive equation $y_k = y_{k-1} + s_k$.

Next, we consider the mixed convolution of a continuous signal $v(x)$ with a discrete sequence p_k when the intersample distance is equal to a

$$g(x) = \left(\sum_{k\in\mathbb{Z}} p_k\delta(x-ak)\right) * v(x) = \sum_{k\in\mathbb{Z}} p_k v(x-ak). \quad (8)$$

The graphical interpretation of this formula is given in Fig. 4. The mixed convolution is a weighted sum of shifted replicates of the signal $v(x)$ separated by a distance a.

3. Fast continuous wavelet transform algorithm

3.1. Spline wavelets

Among all existing wavelet bases, B-spline wavelets have the advantage of possessing an explicit formula [1]; most wavelets are defined only implicitly by means of a refinement filter. For example, the well-known Haar wavelet is a weighted sum of two B-splines of degree 0. Other wavelets, such as the first derivative or the second derivative of a Gaussian (Mexican hat wavelet), can be closely approximated by linear combination of B-splines of sufficiently high degrees ($n \geqslant 2$) [19].

The description of wavelets on to B-spline basis allows for an efficient computation of the convolution products of the CWT that takes advantage of the convolution properties of B-splines. Thus, we choose to express our mother wavelet ψ on a B-spline basis of order n_1. The wavelet at scale a is represented by its B-spline expansion

$$\psi\left(\frac{x}{a}\right) = \sum_{k=-K}^{K} d_k\beta^{n_1}\left(\frac{x}{a} - k\right), \quad (9)$$

where d_k are the B-spline coefficients (see [21] for additional information on how to choose their value).

3.2. CWT computation

3.2.1. Spline wavelet transform

Using the expression for the rescaled B-spline (6), the CWT (1) for the wavelet (9) becomes

$$W_\psi f(a,b) = \frac{1}{\sqrt{a}}\left(f(\cdot) * \sum_{k=-K}^{K} d_k\beta^{n_1}\left(\frac{\cdot}{a} - k\right)\right)(b)$$

$$= \left(\frac{1}{a^{n_1+1/2}}\left(\sum_{k=-K}^{K} d_k\delta(\cdot - ak)\right) * \Delta_a^{n_1+1}(\cdot) * D^{-(n_1+1)} f\left(\cdot + a\left(\frac{n_1+1}{2}\right)\right)\right)(b).$$

We see that the CWT can be calculated from the application of a discrete convolution operator to the (n_1+1)th integral of a shifted version of the analyzed function f.

Replacing $\Delta_a^{n_1+1}$ by its definition (5), the CWT is expressed as a mixed convolution

$$W_\psi f(a,b) = \left(\left(\sum_{k=-K}^{n_1+1+K} p_k\delta(\cdot - al)\right) * v(\cdot)\right)(b), \quad (10)$$

where

$$p_k = \frac{1}{a^{n_1+1/2}}(d * q)_k = \frac{1}{a^{n_1+1/2}}\sum_{l=0}^{n_1+1} d(k-l)q(l), \quad (11)$$

q_l being the finite differences coefficients in (5) and

$$v(x) = D^{-(n_1+1)} f\left(x + a\left(\frac{n_1+1}{2}\right)\right). \quad (12)$$

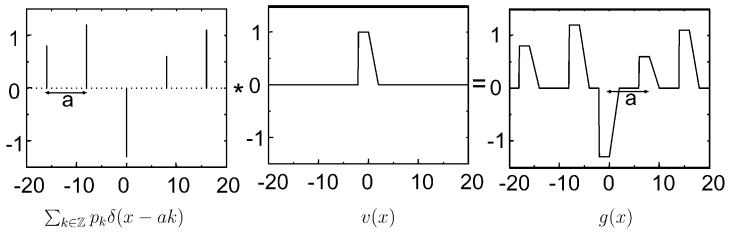

Fig. 4. Graphical interpretation of the mixed convolution between a discrete sequence p_k and a continuous signal $v(x)$, with $a=8$.

3.2.2. Spline input signal

We assume that the continuous input signal $f(x)$ is a spline that interpolates the discrete input samples $f(k)$. Thus, we have $f(x)=\sum_{k\in\mathbb{Z}} c_k\beta^{n_2}(x-k)$ with $c = f * (b^{n_2})^{-1}$, where $(b^{n_2})^{-1}$ is the inverse filter of the B-spline interpolation filter $b^{n_2}=\beta^{n_2}(x)|_{x=k}$, as shown elsewhere [20]. Then, we rewrite $v(x)$ as

$$v(x)=c * D^{-(n_1+1)}\beta^{n_2}\left(x+a\left(\frac{n_1+1}{2}\right)\right). \tag{13}$$

Using the integral formula for a B-spline (7), we get

$$v(x)=g * \beta^{n_1+n_2+1}(x+\tau) \tag{14}$$

where $\tau=(a-1)((n_1+1)/2)$ is a shift factor and g is a discrete sequence obtained by iterative summation of the interpolation coefficients

$$g=\Delta^{-(n_1+1)} * c. \tag{15}$$

We introduce this relation in the CWT expression (10) which yields,

$$W_\psi f(a,b)=\sum_{k=-K}^{K+n_1+1}\sum_{i=0}^{n_1+n_2+1} g_{i+i_0}w_{a,b}(k,i), \tag{16}$$

where $i_0=\lceil b-ak+\tau-(n_1+n_2+2)/2\rceil$ and where

$$w_{a,b}(k,i)=\frac{1}{a^{n_1+1/2}}\,p_k\beta^{n_1+n_2+1}(b-ak-i-i_0+\tau). \tag{17}$$

We have used here the compact-support property of B-splines to reduce the number of terms of the sum over i. In this way, the computation of the CWT reduces to the inner product with the coefficients g_i of a series of precalculated weights $w_{a,b}(k,i)$ (which we can store in a look-up table).

In practice, we are typically only interested in the values of b that correspond to the time locations of the original samples, that is, for integer b. Then, we can use the fact that $w_{a,b}(k,i)=w_{a,0}(k,i-b)$ if $b\in\mathbb{Z}$. The algorithm (16) is then equivalent to a discrete convolution. This reduces considerably the number of weights to be precalculated, since only the values $w_{a,0}=w_a$ are required.

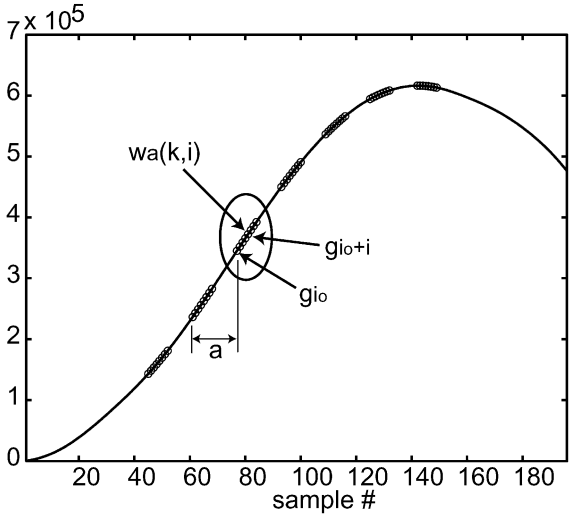

Fig. 5. Spatial structure of the filter w_a: 'clusters' of weights separated by a distance a of each other. Parameters: $n_1=3$, $n_2=3$, $K=1$, $a=16$ and $b=128$.

Note the interesting spatial structure of the filter w_a (see Fig. 5). The CWT computation consists in filtering the coefficients g with $(2K+n_1+2)$ 'clusters' of length (n_1+n_2+2), each cluster being separated from its neighbors by a distance a. This can be seen as a kind of modified '*à trous*' filter.

3.3. Fast implementation

Let us now describe the fast algorithm based on the expansion (16). In the initialization step, the B-spline

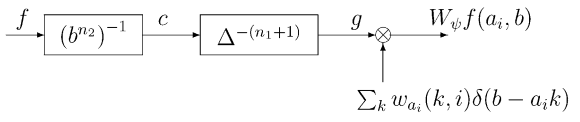

Fig. 6. Schematic representation of the fast wavelet transform: $(b^{n_2})^{-1}$, computation of the interpolation coefficients c; $\Delta^{-(n_1+1)}$, calculation of the (n_1+1)-fold integral of c; $w_{a_i}(k,i)$, look-up table calculation where $k \in [-K, K+n_1+1]$, $i \in [0, n_1+n_2+1]$ and $a_i \in [a_1, a_N]$; N is the number of scales; w_{a_i}, filtering with the mask calculated for each scale; $W_\psi f(a_i,b)$, wavelet transform of f for scale a_i at position b.

expansion coefficients c of the sampled signal f are calculated, and the running-sum operator Δ^{-1} is applied (n_1+1) times; it is computed recursively by iterating (2) (see Section 3.4 for the implementation details). The intermediate result g does not depend on the scale a. For a given scale a, we compute the weights w_a and store them in a 2-D look-up table of dimensions $(2K+n_1+2)(n_1+n_2+2)$. These values are then convolved with the precomputed sequence g. The values w_a and the inter-cluster distance for the filtering depend on a, but the computational complexity is constant and does not depend on a. Moreover, the values w_a do not depend on the signal f. Thus, the computational complexity per point only depends on the values of K, n_1 and n_2. Note that the independence between scales allows for a straightforward parallel implementation (Fig. 6).

3.4. Boundary conditions

To minimize boundary artifacts, we extend our signal $\{s_k\}_{k=0,\ldots,N-1}$ using symmetric mirror boundary conditions defined as $s_{-k} = s_k$, and $s_{N-1-k} = s_{N-1+k}$, for $k = 0, 1, \ldots, N-1$. The boundary conditions defined above are repeated on further extensions to yield a $(2N-2)$ periodic signal. To implement the running sums (inverse finite difference operator) in a consistent manner, we apply it to zero-mean signals, which is perfectly acceptable since the DC-component is filtered out by the wavelet (vanishing moments) anyway. Thus, assuming that s is zero mean, the inverse finite differences operator is given by

$$(\Delta^{-1} * s)_k = \sum_{l=0}^{k \bmod (2N-2)} s_l. \tag{18}$$

For zero-mean signals, the application of the above operator reverses the symmetry of the boundary conditions. That is to say, if the input signal has symmetric boundary conditions, the output will have anti-symmetric boundary conditions defined as $s_k = -s_{-k-1}$, and $s_{N-1+k} = -s_{N-2-k}$, for $k = 0, 1, \ldots, N-1$. An antisymmetric signal is always zero mean. Thus, in our implementation, we alternate between symmetric and anti-symmetric input boundary extensions depending on the parity of the iteration number.

Finally, the boundary conditions—either symmetric or antisymmetric—of the (n_1+1)-fold integrated interpolation coefficients will be handled accordingly by the rest of the algorithm (in our case, convolution with the sequence of precalculated weights).

4. Results

Here, we discuss the implementation of our fast CWT algorithm and compare its execution time with a FFT-based implementation. As an example of application, we show the analysis of a biomedical signal.

4.1. Comparison with FFT-based computation

The FFT has an overall $O(N \log N)$ complexity and is therefore asymptotically slower than our method which has an $O(N)$ complexity. This can be observed from the experimental comparison of the computation times shown in Fig. 7. We see that, for long input signals, our method is indeed faster. The interpolation degree for the input signal was zero. The CWT was computed over four octaves with 12 scales per octave. The wavelet was the second derivative of the quintic (see Fig. 8(b)), quartic and cubic B-spline, respectively [22]. The FFT-based method used a radix-2 algorithm when the signal length was a power of 2 and a mixed-radix method for other signal lengths (MATLAB's FFT algorithm). The region where the algorithm is faster than the FFT-radix 2 method with zero padding is colored in gray. The time required to compute the wavelet in the time domain before its FFT computation was neglected. A parallel implementation for the scale-dependent part of each algorithm would speed up the computations.

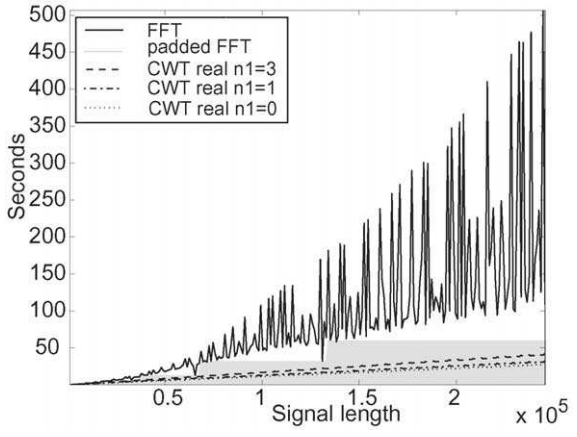

Fig. 7. Comparison of the experimental computation times of the FFT and of our B-spline-based method to calculate the CWT.

4.2. Analysis of a biomedical signal

We have applied our method to the analysis of bowel movements. A magnetically active capsule was swallowed and its gastrointestinal transit was monitored. The measures consisted of its three spatial coordinates and the angles that describe its orientation [18].

Fig. 10(a) corresponds to the x-coordinate of the capsule. The sampling time was 70 ms. We have analyzed it using both the real (Fig. 10(b)) and the complex CWT (Fig. 10(c)) for cubic spline interpolation of the input signal. The y-axis corresponds to a normalized scale in seconds; it is given by $a_0 = a/f_0$, where f_0 is the central frequency of the wavelet. In the real case, the wavelet was the first derivative of a quartic spline expanded by a factor of 2 as shown in Fig. 8(a). A visual inspection highlights the band of period 3 s, which corresponds to the breathing of the patient. We chose to calculate the complex CWT by an extension of the method proposed by Unser [19] for integer scales. The analysis wavelet is the Morlet-like wavelet $\beta^3(x)e^{-4\pi xj}$ (Fig. 9).

Using complex analysis (Fig. 10(c)), we discovered three relevant frequency bands: first, the breathing with a period close to 3 s; then, two more bands with periods around 12 and 20–25 s due to the contractions of the stomach.

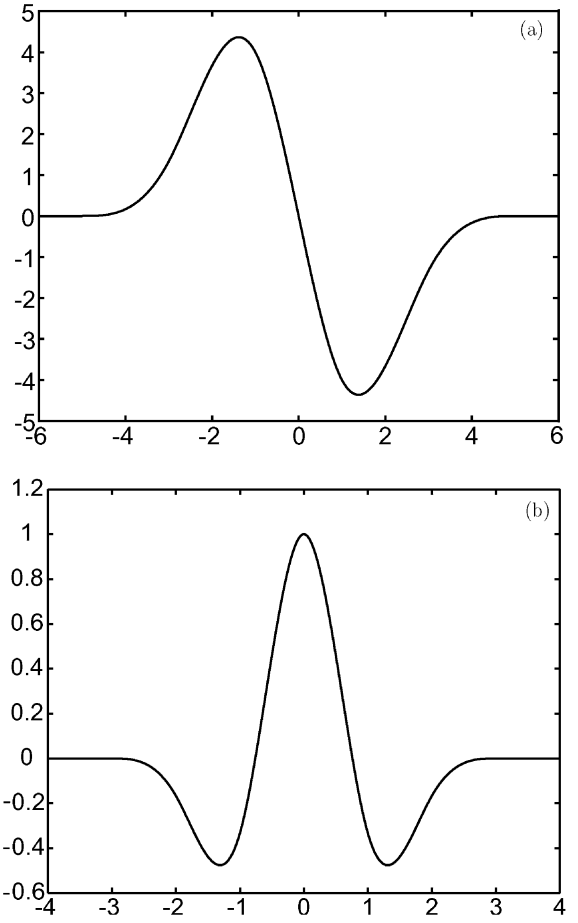

Fig. 8. (a) First derivative of the quartic B-spline enlarged by a factor 2. (b) Second derivative of the quintic spline wavelet.

5. Discussion: integer scale method

In a previous paper [22], Unser et al. describe a fast algorithm for the CWT computation at integer scales using B-splines as basis functions. Their method can be shown to be equivalent to ours when a is an integer. This follows from the identity

$$\Delta^{-(n_1+1)}(z)\Delta_a^{n_1+1}(z) = \left(\frac{1-z^a}{1-z}\right)^{n_1+1} = \left(\sum_{k=0}^{a-1} z^k\right)^{n_1+1} = (U_a^0(z))^{n_1+1}.$$

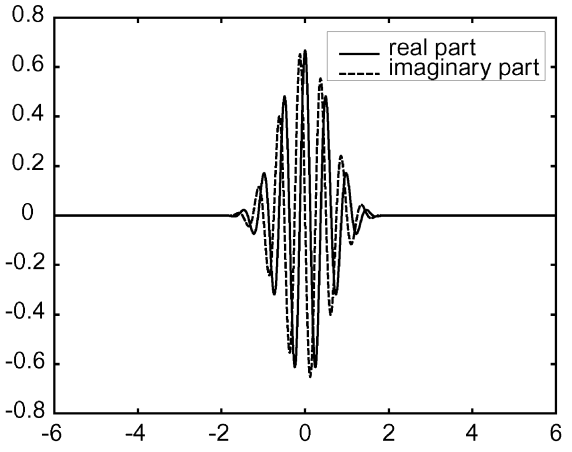

Fig. 9. Gabor-like wavelet $\beta^3(x)e^{-4\pi xj}$.

Next we summarize the two ways of computing the CWT at integer scales. We start with the method of Unser et al. for which we write

$$W_\psi s(a,b) = \underbrace{f * (b^{n_2})^{-1} * b^{n_1+n_2+1}}_{\text{initialization}} * \underbrace{(u_a^0)^{n_1+1} * [d]_{\uparrow a}}_{\text{scale-dependent}},$$

where the upsampling operator $[\cdot]_{\uparrow a}$ is defined as

$$[d]_{\uparrow a}(k) = \begin{cases} d(\frac{k}{a}) & \text{if } a \text{ divides } k, \\ 0 & \text{elsewhere} \end{cases}$$

and the z-transform of the sequence u_a^0 is given by $U_a^0(z) = z^{(a-1)/2} \sum_{k=0}^{a-1} z^{-k}$. We then summarize our method by

$$W_\psi s(a,b) = \underbrace{f * (b^{n_2})^{-1} * \Delta^{-(n_1++1)}}_{\text{initialization}} * \underbrace{b^{n_1+n_2+1} * \Delta_a^{n_1+1} * [d]_{\uparrow a}}_{\text{scale-dependent}}.$$

In the first approach, u_a^0 can be computed recursively by a moving-sum method that requires only one addition and one subtraction per point. For the first method (not taking into account the initialization) we need $2(n_1+1)$ additions for the computation of $(u_a^0)^{n_1+1}$ and $(2K+1)$ multiplications and $2K$ additions per point for the convolution with the upsampled version of the sequence d, $[d]_{\uparrow a}$. Then, for one point, one needs $(2(n_1+1)+2K)$ additions and $(2K+1)$ multiplications in total.

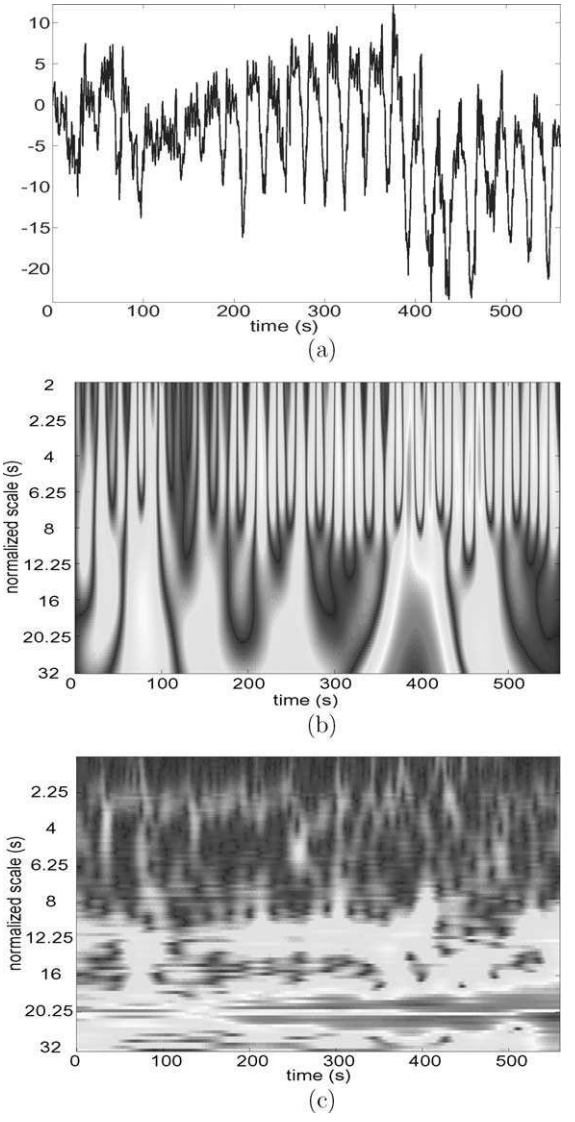

Fig. 10. Trajectory of a magnet within the digestive track (x component). (b) Real CWT using the wavelet in Fig. 8(a). (c) Complex CWT using the Morlet-like wavelet shown in Fig. 9.

In the second method (not taking into account the initialization step either) we need $(2K+n_1+2)(n_1+n_2+2)$ multiplications and $(2K+n_1+1)(n_1+n_2+1)$ additions per point, corresponding to the filtering of g with the weights w_a.

Thus, the fast implementation of Unser et al. outperforms ours for integer scales and is therefore preferable in this particular setting. Our method, on

the other hand, is more versatile and can be used for arbitrary scale progressions.

6. Conclusions

We have presented a novel B-spline-based CWT algorithm that is able to compute the CWT at *any* real scale, making it possible to use arbitrary scale progressions. The computational complexity per computed coefficient is O(1), as is the case with the most efficient wavelet algorithms for dyadic or integer scales. The overall operation count only depends on the wavelet shape and on the degrees of the B-spline basis on which the wavelet and the input signal are described, but is independent of the value of the scale. Thanks to the good approximation properties of B-splines, virtually any wavelet can be used (either via their B-spline interpolation or projection). Moreover, the algorithm lends itself well to a parallel implementation as it is not iterative across scales.

The price to pay for the generality of this algorithm is that the leading constant in the O(N) complexity can be large (typically 56 for a cubic spline Mexican-hat wavelet and cubic interpolation of the input signal). Thus, the method really starts paying off when the size of the signal is large (say $N \geqslant 1000$ samples). For smaller sizes, it may be more efficient to use a simpler FFT-based method. Note, however, that the specialized version of the algorithm for integer scales beats the FFT in all cases.

A demonstration of our method is available on the web at http://bigwww.epfl.ch/demo/cwtspline.

Acknowledgements

This work was funded by the Swiss National Science Foundation, Grant #2100-053540. We also thank the reviewers for their insightful comments.

References

[1] A. Aldroubi, M. Unser (Eds.), Wavelets in Medicine and Biology, CRC Press, Boca Raton, FL, USA, 1996.

[2] J.P. Antoine, D. Barache, R.M. Cesar, L. da Costa, Multiscale shape analysis using the continuous wavelet transform, in: Proceedings of the International Conference on Image Processing, Vol. 1, September 1996, pp. 291–294.

[3] K. Berkner, R.O. Wells, A fast approximation to the continuous wavelet transform with applications, in: Record of the 31st Asilomar Conference on Signals, Systems and Computers, Vol. 2, 1997, pp. 135–139.

[4] K.R. Castleman, Digital Image Processing, Prentice-Hall, Englewood Cliffs, NJ, USA, 1996.

[5] Y.T. Chan, K.C. Ho, Filter design for CWT computation using the Shensa algorithm, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 3, 1999, pp. 1321–1324.

[6] L. da, F. Costa, R.M. Cesar Jr., Shape Analysis and Classification: Theory and Practice, CRC Press, Orlando, USA, 2001.

[7] H. Goely, R.D. Jones, P.J. Bones, Continuous wavelet transform for the detection and classification of epileptiform activity in the EEG, in: Proceedings of the First Joint BMES/EMBS Conference Serving Humanity, Advancing Technology, October 1999.

[8] K.C. Ho, Fast CWT computation at integer scales by the generalized MRA structure, IEEE Trans. Signal Process. 46 (2) (February 1998) 501–506.

[9] K.C. Ho, Y.T. Chan, Optimum filter design for the '*à trous*' algorithm, in: Proceedings of the IEEE-SP International Symposium on Time-frequency and Time-scale Analysis, 1998, pp. 125–127.

[10] A.R. Ismail, S.S. Asfour, Continuous wavelet transform application to EMG signals during human gait, in: Record of the 32nd Asilomar Conference on Signals, Systems and Computers, Vol. 1, 1998, pp. 325–329.

[11] J.M. Lewis, C.S. Burrus, Approximate continuous wavelet transform with an application to noise reduction, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 1998, Vol. 3, pp. 1533–1536.

[12] S. Mallat, A Wavelet Tour of Signal Processing, Academic Press, San Diego, CA, USA, 1998.

[13] A. Muñoz, T. Blu, M. Unser, Least-squares image resizing using finite differences, IEEE Trans. Image Process. 10 (9) (September 2001) 1365–1378.

[14] A. Munteanu, J. Cornelis, P. De Muynck, A. Bezerianos, P. Cristea, Accurate detection of coronary arteries with the continuous wavelet transform, Comput. Cardiol. 24 (1997) 601–604.

[15] L. Prasad, S.S. Iyengar, S.S. Ayengar, Wavelet Analysis with Applications to Image Processing, CRC Press, Orlando, USA, 1997.

[16] O. Rioul, Fast algorithms for the continuous wavelet transform, in: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 3, 1991, pp. 2213–2216.

[17] O. Rioul, P. Duhamel, Fast algorithm for discrete and continuous wavelet transforms, IEEE Trans. Information Theory 38 (2) (March 1992) 569–595.

[18] V. Schlageter, P.A. Besse, R.S. Popovic, P. Kucera, Tracking system with five degrees of freedom using a 2-D arrray of Hall sensors and a permanent magnet, Sensors and Actuators 2951 (2001) 1–6.

[19] M. Unser, Fast Gabor-like windowed Fourier and continuous wavelet transforms, IEEE Signal Process. Lett. 1 (5) (May 1994) 76–79.
[20] M. Unser, Splines: a perfect fit for signal and image processing, IEEE Trans. Signal Process. 16 (November 1999) 22–38.
[21] M. Unser, A. Aldroubi, M. Eden, A family of polynomial spline wavelet transforms, Signal Process. Mag. 30 (1993) 141–162.
[22] M. Unser, A. Aldroubi, S.J. Schiff, Fast implementation of the continuous wavelet transform with integer scales, IEEE Trans. Signal Process. 42 (12) (December 1994) 3519–3523.
[23] M. Vrhel, C. Lee, M. Unser, Fractal dimension estimation using the fast continuous wavelet transform, in: Proceedings of the Wavelet Applications in Signal and Image Processing III, San Diego, USA, July 12–14, 1995, SPIE, Vol. 2569, pp. 478–488.
[24] M.J. Vrhel, C.L. Lee, M. Unser, Rapid computation of the continuous wavelet transform by oblique projections, IEEE Trans. Signal Process. 45 (4) (April 1997) 891–900.
[25] H. Yoshida, B. Keserci, D.D. Casalino, O. Ozturk, A. Coskun, A. Savranlar, Segmentation of liver tumors in ultrasound images based on scale-space analysis of the continuous wavelet transform, in: IEEE Ultrasonics Symposium, 1998, pp. 1713–1716.