Katholieke Universiteit Leuven

Departement Elektrotechniek

ESAT-SISTA/TR 1997-17

Applications of the Continuous Wavelet Transform in the Processing of Musical Signals[1]

Peter De Gersem Bart De Moor Marc Moonen[2]

March 1997
Published in Proc. of the 13th International Conference on Digital Signal Processing (DSP97), Santorini, Greece, July 2-4, 1997, pp. 563–566

[1]This report is available by anonymous ftp from *ftp.esat.kuleuven.ac.be* in the directory *pub/SISTA/degersem/reports/report97-17.ps.gz*

[2]K.U.Leuven, Dept. of Electrical Engineering (ESAT), Research group SISTA, Kardinaal Mercierlaan 94, 3001 Leuven, Belgium, Tel. 32/16/32 17 09, Fax 32/16/32 19 70, WWW: *http://www.esat.kuleuven.ac.be/sista*. E-mail: *bart.demoor@esat.kuleuven.ac.be*. This work was supported in part by the FWO, the IUAP P4-02, and the GOA-MIPS

Applications of the Continuous Wavelet Transform in the Processing of Musical Signals

Peter De Gersem Bart De Moor Marc Moonen

K.U.Leuven-ESAT/SISTA
Kard. Mercierlaan 94, 3001 Heverlee (Leuven), Belgium
Tel. +32-(0)16-32.18.07 Fax. +32-(0)16-32.19.70
E-mail: Peter.DeGersem@esat.kuleuven.ac.be

Abstract Wavelet Transformations and music have several things in common: both have something to do with time-frequency descriptions and constant relative bandwidth analysis. In this paper, we describe some applications of the Continuous Wavelet Transform in the processing of music. The main applications are *analysis* and *pitch-shifting* (or the equivalent *time-stretching*), for which we give an algorithm.

1. Introduction

The Wavelet Transformat decomposes a signal into a linear combination of basisfunctions (wavelets), which are all derived from one mother wavelet by means of dilation (scale) and translation (time). The mother wavelet (and thus all derived basis functions) is localized in time and in frequency. A one-dimensional signal is transformed in a two-dimensional *time-frequency* representation.

The transform on one particular scale is a bandpass filter, since a wavelet is localized in frequency. On all scales, these filters have the same relative bandwidth, because all wavelets are derived by dilation from the mother wavelet. The time-frequency analysis thus studies low frequencies with more frequency detail but less time-resolution than high frequencies, which get a higher time-resolution.

Music is also a typical time-frequency phenomenon: the notes contain frequency information (pitch), and time information (duration, starting time). The frequency information is logarithmically divided: raising one octave doubles the frequency. It is thus necessary to analyse musical signals with more frequency detail for the low frequencies, and less frequency detail for the high frequencies, a constant relative bandwidth filter...

In the following sections, we go into some more details of the Continuous Wavelet Transform, and describe some applications of this CWT in the processing of music.

2. The Continuous Wavelet Transform

The CWT is defined (see [1]) as

$$\mathcal{F}(a,b) = \int_{-\infty}^{\infty} f(t)\psi^*_{a,b}(t)dt \qquad (1)$$

where

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)$$

are the wavelets derived by translation (b) and dilation (a) from the mother wavelet $\psi(t)$. We mostly use the complex Morlet-wavelet (a modulated Gaussian, see fig. 1)

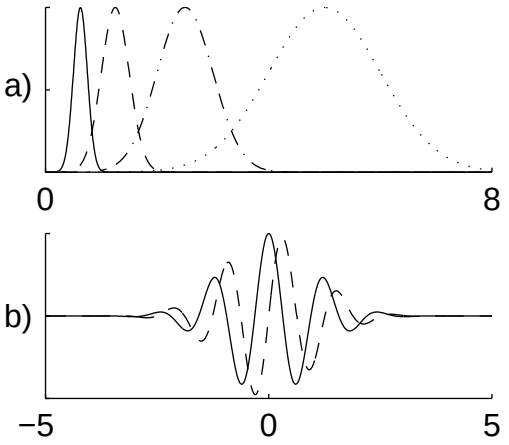

Fig. 1:
a) Morlet wavelets, frequency domain ($\frac{\text{rad}}{\text{s}}$), scales 8, 4, 2, and 1 (left to right),
b) Morlet wavelet, time domain, real and complex part

$$\psi(t) = e^{j\omega_o t}e^{\frac{-t^2}{2}}\,. \qquad (2)$$

The function $f(t)$ can be recovered from the CWT using (again from [1])

$$f(t) = C_\psi^{-1}\int_0^\infty \frac{da}{a^2}\int_{-\infty}^{\infty}\mathcal{F}(a,b)\psi_{a,b}(t)db\,, \qquad (3)$$

where C_ψ is a finite constant, dependent on the chosen wavelet $\psi(t)$. There are some approximate resynthesis formulas available, of which

$$f(t) \simeq K_\psi\int_0^\infty \mathcal{F}(a,t)\frac{da}{a^{3/2}} \qquad (4)$$

is most frequently used [2].

3. Implementation

In this section, some implementation details of the CWT and its inverse will be dealt with.

3.1. Discretization

To calculate the CWT on a computer, one has to discretize the signal $f(t)$, the wavelet ψ, and the parameters b (time) and a (scale). The discretization

of the signal is often straightforward, since it is almost always given in sampled form (with sampling frequency Fs). The choice of the scales is less trivial: how do we space the different scales (linear, logarithmic), what is the lowest and the highest scale?

Because of the constant relative bandwidth property of the CWT, one normally uses logarithmic scales: the higher the scale, the bigger the bandwidth of the *filter* on that scale, and the sparser the sampling of the scales has to be. We take the scales $a = 2^{i/v}$, with i integer, and v the number of *voices per octave* (the number of scales we consider in between each power of 2, an octave).

The smallest usable scale can be calculated using the Nyquist theorem: the Morlet wavelet with $\omega_0 = 5\frac{\text{rad}}{\text{s}}$ has significant (-54dB) frequency-extent from 0 to $10\frac{\text{rad}}{\text{s}}$, i.e. from 0 to 1.6Hz. The Morlet wavelet at scale a has frequency-extent from 0 to $\frac{1.6}{a}$Hz (see fig. 1a), and this must be smaller or equal than the sampling frequency Fs, so $a_{\min} = \frac{1.6}{Fs}$Hz.

The largest usable scale is dependent on the signal-length: the Morlet wavelet at a particular scale has significant time-extent from $-4a$ to $4a$ (see fig. 1b), so we need $8\,a\,Fs$ samples for representing ψ on scale a. Because the signal $f(t)$ is finite, we have boundary problems for approximating the integral by a convolution. To have at least one valid value, the length of ψ should be less or equal to the length of the signal (N), so the largest usable scale $a_{\max} = \frac{N}{8Fs}$.

The discretization of ψ is no problem either, since we have the formula (2). The sampling of the parameter b is often taken equal to the sampling of the signal, i.e. the coefficients on each scale are calculated on each time step. This should not be always the case, as indicated in one of the next paragraphs.

3.2. Convolution

Once discretized, we can approximate the integrals in formula's 1 and 3 by convolutions, which we can calculate in the time-domain, or in the frequency domain (using FFT's). The simplified reconstruction formula (4) does not even need convolutions, so this is a lot faster than the full reconstruction formula. Both (3 and 4) have an integration over the scales as final step, this is approximated by the trapezium rule.

3.3. Spacing of the wavelets

It does not always make much sense to calculate as much coefficients on high scales as on low scales: on high scales, the wavelets have a big time-extent, and so it seems a good idea to calculate the coefficients on time steps b which are proportional to the scale a. This has also implications for the reconstruction formula, but this is subject of current research.

3.4. Calculation in pieces

If we do the calculation of the CWT on a whole piece of audio in one time, we run into problems if we want to calculate the coefficients on each time step (on very high scales, the convolution needs a long calculation time because the wavelet is very long), and most of the times we are not really interested in the lowest frequencies that can only be obtained on these large scales (using the Morlet wavelet), the center frequency f_c of the *filter* on a particular scale is $\frac{\omega_0}{a}$, e.g. for $\omega_0 = 5\frac{\text{rad}}{\text{s}}$, $Fs = 8$kHz, and $N = 16000$ (i.e. 2s), $f_c = 3.2$Hz (corresponding to an $a_{\max} = \frac{2000}{Fs} = 0.25$. It should be better to do the calculation in blocks, e.g. of $N = 1024$. In this example, $f_c = 50$Hz, corresponding to $a_{\max} = \frac{128}{Fs} = 0.016$. Using the overlap-add method, we can avoid boundary-problems between the blocks. Such block-processing is also necessary if we want to implement the algorithm on a DSP.

3.5. Visualisation and analysis versus processing

The choice of parameter v (number of voices per octave), and the dependence of the time-sampling on the scale, depends on the application: for visualisation and analysis, it is desirable to have the biggest density of coefficients, so we choose v as high as our computer allows (e.g. 8 or 12), and calculate the coefficients on each time-step b. If the aim is to process the coefficients and retransform, we choose the sparsest sampling (of a and b) that still gives accurate results. For our applications, this is also subject of current research.

4. Applications on Music

4.1. Analysis

In the domain of computer music (i.e. all aspects of music where computers are involved), one of the basic operations is analysis of music: this is used as a preprocessing step for some sound-synthesis algorithms, for transcription, pattern-recognition, etc. Traditionally, this is done using FFT's (e.g. the phase vocoder approach), which boils down to a constant bandwidth filter-bank. All frequencies are studied with the same frequency-resolution, and this is "overkill" for the high frequencies (e.g., a bandwidth of 10Hz at 8000Hz represents 1/555 of an octave, i.e. an unhearable difference), but insufficient for the low frequencies (a bandwidth of 10Hz at 80Hz represents 1/6 of an octave, i.e. one whole tone). That is why a constant relative bandwidth analysis like the CWT should be more appropriate for this. The drawback of the CWT is the relatively high computational cost (on each scale, a convolution has to be calculated), the complicated theory for discrete computation (of which one can find the details in [1]), and with the used Morlet wavelet, the relative big bandwidth in each frequency-band (on fig. 1a, it is clear that $\frac{\Delta f}{f} \approx 1$, i.e. from an octave below to a fifth above the center frequency). This results in rather unclear time-frequency *pictures*. We are currently investigating more appropriate wavelets, which should have a smaller bandwidth, but as consequence a larger time-extent (impulse-

response), due to the Heisenberg-principle.

4.2. Pitch-shifting

A more appealing application is *pitch-shifting*: this is changing the pitch of a piece of music without changing the length. Remember that one can easily raise the pitch of a piece of audio by playing it faster, but this reduces the length. The equivalent problem is *time-stretching*, where the length of a piece is changed without altering the frequency-content. This can be accomplished by pitch-shifting and resampling.

The solution to this problem in general is not well defined, the processed piece should sound as if the original recording is redone transposed. This is very difficult to automate, because the algorithm should distinguish between steady notes and transient attacks (e.g. for time-stretching: the start of a note of a piano should not be changed in duration, because a piano player does not change the attack of that note, but only holds the key longer).

The algorithm we use here (from [2], [3]) does not account for this features. It decomposes an audio signal in its CWT using the Morlet wavelet, changes the scale-axis, and retransforms the CWT to a signal. Changing the scale-axis is obviously the tricky part: if one wants to raise the pitch of an audio signal by a factor c, and therefore divide all the scales in the CWT of the signal by c, and retransform the result, one does not get the expected result. The point is that one can not just change the coefficients of the CWT. The CWT of a one-dimensional signal is a very redundant representation of that signal (an extra dimension is added), so there exist lots of two-dimensional (time-scale) functions which are no CWT of any signal whatsoever. All two-dimensional functions that are a valid CWT of a particular signal, satisfy the *reproducing kernel Hilbert subspace* (r.k.H.s.) property. One should modify the coefficients of the CWT with care, such as to keep the CWT in the r.k.H.s. Using the complex Morlet wavelet facilitates this: the phase of the coefficients is related to the frequency of the signal analyzed at the scale of the coefficients considered, so if we divide the scales by a a factor c, and change the phases of the coefficients accordingly (unwrapping the phase first, then multiplying by c), we should come a far end in preserving the r.k.H.s. property. No rigourous mathematical analysis has been done yet, but the procedure delivers acceptable results (fig. 2e). The algorithm is summarized:

coefs = `cwt`(*f*,*scales*)
absc = `abs`(*coefs*)
phac = `angle`(*coefs*)
phac_unwrap = `unwrap`(*phac*)
coefs_shifted = *absc*.*`exp`(`i`**phac_unwrap***c*)
scales_shifted = *scales*/*c*
f_shifted = `icwt`(*coefs_shifted*,*scales_shifted*)

Alg. 1: Pitch-shifting using the CWT. `abs` and `angle` calculate the absolute value resp. the phase of a complex number, `unwrap` gets rid of all 2π phase-jumps in the phase.

It works perfectly for pure sines, and gives acceptable results for general audio. There is quite a difference between the reconstruction formula's (3) and (4). At the moment of this writing, there is research focused on how the parameters (see section 3) should be chosen. Also the usage of other wavelets is investigated, but it seems that it is more difficult to preserve the r.k.H.s. property with other wavelets than the Morlet wavelet. Finally, we're also working on a DSP implementation of the algorithm.

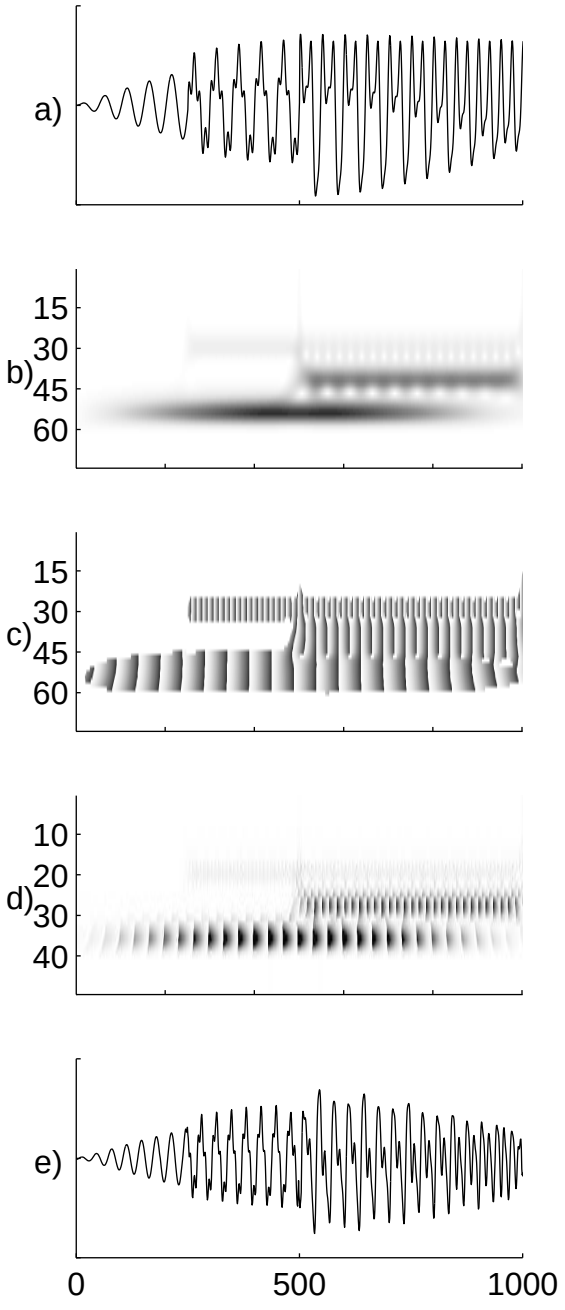

Fig. 2:
a) The original signal $f(t)$, consisting of a sum of three sines, each with a different frequency, phase, and initial shift,
b) The absolute value of the CWT of $f(t)$ (the y-axis contains the scales),
c) The phase of the CWT of $f(t)$,
d) The phase superimposed on the absolute value of the pitch-shifted CWT (factor 1.5),
e) The resulting reconstructed signal.

4.3. Others

The CWT can be used for more sound transformations than described above. One could e.g. retransform the absolute values of the coefficients of the CWT of one signal, with the phases of the coefficients of the CWT of another signal, knowing very well that the resulting combination is not *valid*, but apparently the retransformed sound is (at least in the ears of the right persons). Other applications of wavelets in music are described in [4].

5. Conclusions

The CWT is an interesting tool for analysing, processing, and synthesizing music. In this paper, we described some possible applications. The biggest drawback of the CWT at this moment is the computational cost. There exist more efficient schemes (for the CWT, and definitely for the Discrete Wavelet Transform), but on first sight they lose some of the interesting properties we used. This also is subject of current research.

References

[1] I. Daubechies, *Ten Lectures on Wavelets*, vol. 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1992.

[2] R. Kronland-Martinet, "The wavelet transform for analysis, synthesis and processing of speech and music sounds," *Computer Music Journal*, vol. 12, pp. 11–20, Winter 1988.

[3] D. Arfib, "Analysis, transformation, and resynthesis of musical sounds with the help of a time-frequency representation," in *Representations of Musical Signals* (G. De Poli, ed.), pp. 87–118, MIT Press, 1991.

[4] P. De Gersem, M. Moonen, and B. De Moor, "Applications of wavelets in audio and music," in *Wavelet Analysis: a new tool in signal and image processing, KVIV Study-day*, (Antwerp, Belgium), Dec. 11, 1996.

Acknowledgements

This research work was carried out at the ESAT laboratory of the Katholieke Universiteit Leuven, in the frame of the Concerted Research Action MIPS ('Model-based Information Processing Systems') of the Flemish Government, and the Interuniversity Attraction Pole IUAP P4-02 ('Modeling, Identification, Simulation and Control of Complex Systems') of the Belgian Government. The first author is Research Assistant of the Fund for Scientific Research - Flanders (Belgium)(F.W.O.). The second and third author are Research Associates of the F.W.O. The scientific responsibility is assumed by its authors.