Tutorial: Introduction to wavelets

Note

This tutorial introduces practically the basic wavelet decomposition and reconstruction algorithms. The objectives are to code some basic programs that will decompose and reconstruct 1D and 2D signals.

1 Introduction

Wavelets are based on a mother wavelet Ψ (s is a scale parameter, τ is the time translation factor $(s,\tau) \in \mathbb{R}_+^* \times \mathbb{R}$):

$$\forall t \in \mathbb{R}, \psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi\left(\frac{t-\tau}{s}\right)$$

The continuous wavelet transform is written as follows, where ψ^* means the complex conjugate of ψ and $<.,.>$ is the scalar product:

$$g(s,\tau) = \int_{-\infty}^{\infty} f(t)\psi_{s,\tau}^*(t)\mathrm{d}s\mathrm{d}\tau = < f, \psi_{s,\tau} >$$

The reconstruction is defined by:

$$f(t) = \frac{1}{C}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{|s|^2} g(s,\tau)\psi_{s,\tau}(t)\mathrm{d}s\ \mathrm{d}\tau$$

with

$$C = \int_{-\infty}^{\infty} \frac{|\hat{\Psi}(\omega)|^2}{|\omega|}\mathrm{d}\omega$$

and $\hat{\Psi}$ is the Fourier Transform of Ψ.

The discrete wavelet transform corresponds to a sampling of the scales. To compute the different scales, one has to introduce a "father" wavelet, that similarly defines a family of functions orthogonal to the family $\psi_{s,\tau}$.

2 Fast discrete wavelet decomposition / reconstruction

A simple algorithm (cascade algorithm, from Mallat) is defined as two convolutions (by a lowpass ld filter for the projection on the ψ family, and a high pass hd filter for the orthogonal projection) followed by a subsampling (see Fig. 1).

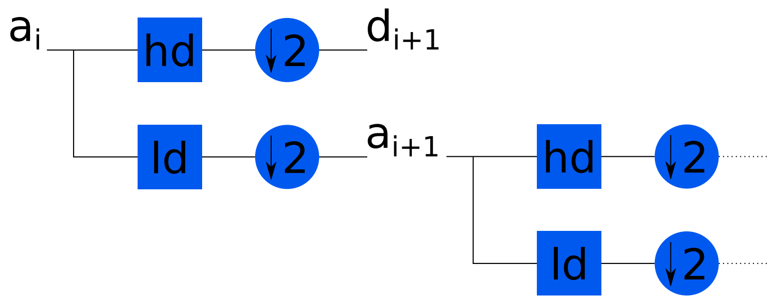

Figure 1: Algorithm for wavelet decomposition. First, a convolution is performed (square node), then, a subsampling. a_i stands for decomposition at scale i, d_i stands for detail at scale i.

2.1 Simple 1D example

Let's consider the signal $[4; 8; 2; 3; 5; 18; 19; 20]$. We will use the Haar wavelets defined by $ld = [1; 1]$ and $hd = [-1; 1]$. Basically, these filters perform a mean and a difference (see Table 1). The result of the decomposition in 3 scales with these wavelets is the concatenation of the details and the final approximation

$$C = \{[-4; -1; -13; -1]; [7; -16]; [-45]; [79]\}$$

Informations

For the sake of simplicity, it is recommended to use the structure `cell` of MATLAB® to store the decomposition of all detail vectors as well as the final approximation vector.

Scale i	Approximation a_i	Details d_i
0 (original signal)	$[4; 8; 2; 3; 5; 18; 19; 20]$	
1	$[12; 5; 23; 39]$	$[-4; -1; -13; -1]$
2	$[17; 62]$	$[7; -16]$
3	$[79]$	$[-45]$

Table 1: Illustration of Haar decomposition in a simple signal.

In the case of these Haar wavelets, code a function that performs the decomposition for a given number of scales. The prototype of this function will be:

```
function C=simpleWaveDec(signal1D, nb_scales)
% wavelet decomposition of <signal1D> into <nb_scales> scales
```

or in python:

```python
def simpleWaveDec(signal, nb_scales):
    """
    wavelet decomposition of <signal> into <nb_scales> scales
    This function uses Haar wavelets for demonstration purposes.
    """
```

2.2 Reconstruction

To reconstruct the original signal (see Fig. 2), we need the definition of two reconstruction filters, hr and lr. For the sake of simplicity, we use $lr = ld/2$ and $hr = -hd/2$. These filter will perform an exact reconstruction of our original signal.

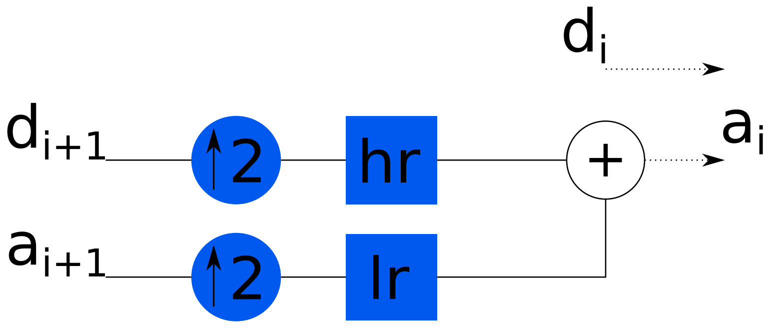

Figure 2: Reconstruction algorithm. The oversampling is done by inserting zeros.

Using this algorithm, you can now go on to the next exercise.

Code a function that performs the reconstruction of the signal. The prototype of this function will be:

```matlab
function signal=simpleWaveRec(C)
% wavelet simple reconstruction function of a 1D signal
% C: Wavelet coefficients in cell of arrays
% signal: reconstructed signal
```

with the previous definition of C, or in python:

```python
def simpleWaveRec(C):
    """
    wavelet simple reconstruction function of a 1D signal
    C: Wavelet coefficients
    """
```

3 2D wavelet decomposition

Let A be the matrix of an image. We consider that A is of size $2^n \times 2^n$, $n \in \mathbb{N}$. We consider, as for the 1D transform, the filters ld and hd.

The wavelet decomposition is as follows:

- Apply ld and hd on rows of A. Results are denoted $ld_r A$ and $hd_r A$, of size $2^n \times 2^{n-1}$, with r standing for row.
- Then, apply ld and hd again, to get the four new matrices: $ld_c ld_r A$, $ld_c hd_r A$, $hd_c ld_r A$ and $hd_c hd_r A$, of sizes $2^{n-1} \times 2^{n-1}$, with c standing for column. The matrix $ld_c ld_r A$ is the approximation, and the other matrices are the details.

- Code a function to perform the wavelet decomposition and for a given number of scales (lower than n). Test it on images of size $2^n \times 2^n$.
- Code the reconstruction function.

4 Built-in functions

Let us explore some built-in functions.

4.1 Continuous wavelet decomposition

One has to make the difference between the continuous wavelet transform and the discrete transform.

Informations

The MATLAB® function that performs continuous wavelet transform is cwt. A GUI dedicated to wavelets is available in MATLAB®: type the command wavemenu and try some 1D and 2D signals.

```matlab
load noissin; % load a signal1D

% perform continuous wavelet transform at scales specified by
% 1:48
c = cwt(noissin, 1:48, 'db4', 'plot');
```

Informations

In scipy.signal you can find function to perform wavelets decomposition, and among them cwt for the continuous wavelet transform. There is also the module pywt that may be more developped (see also cwt).

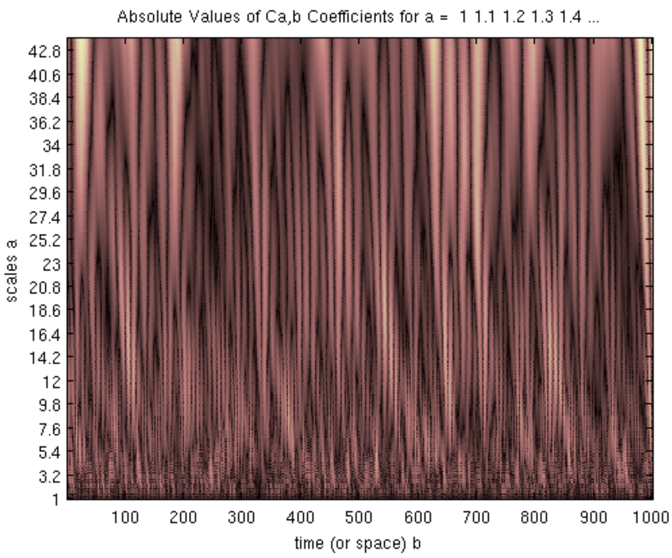

Figure 3: Continuous wavelet transform.

4.2 Discrete decomposition

1. Generate the signal $sin(2\pi * f_1 * t) + sin(2\pi * f_2 * t)$, with $f_1 = 3$Hz and $f_2 = 50$Hz.
2. Apply the wavelet transformation for a given wavelet (like 'db4').
3. Display the 5th approximation of the wavelet decomposition.
4. Display the 4th detail coefficients of the wavelet decomposition.

Each of the MATLAB® function is available for 1D, 2D or even 3D signals.
You can test the following functions wavedec, wavedec2 or wavedec3:

```
[C,L] = wavedec(signal, scale, 'wavelet_name')
[C,L] = wavedec(signal, scale, ld, hd)
```

Look at pywt.downcoef for the decomposition at a given level and pywt.wavedec for the entire decomposition.