BMI/CS 567: Medical Image Analysis **Spring 2020**

Week 10: Wavelet Transforms

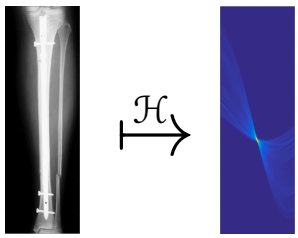
Instructor: Daniel L. Pimentel-Alarcón

© Copyright 2020

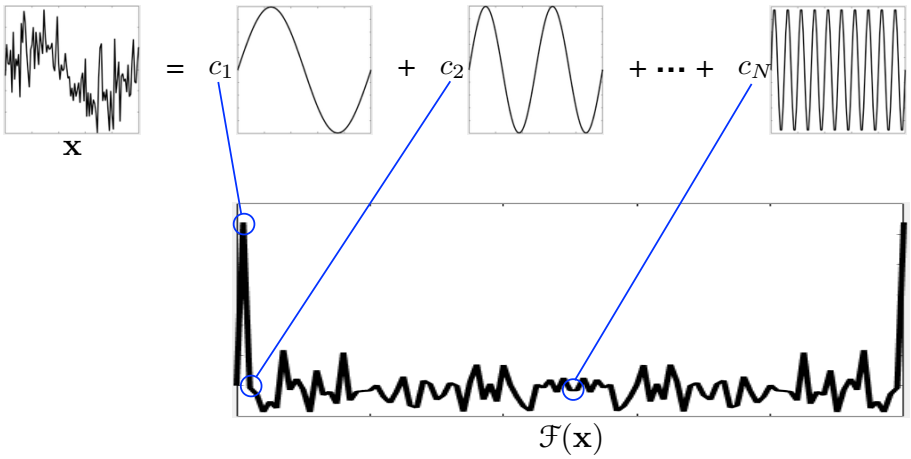
GO GREEN. AVOID PRINTING, OR PRINT 2-SIDED MULTI-PAGE.

10.1 Introduction

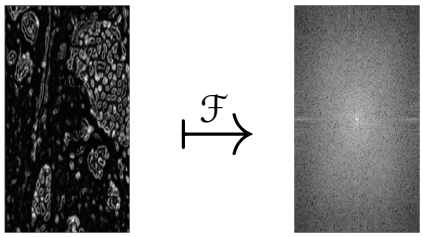
In previous lectures we have studied several image transformations. For example, the Hough transform encodes the probability that each possible line (or other shape) is contained in an image:



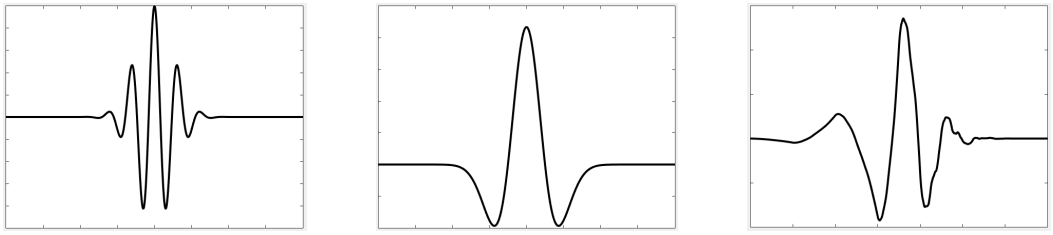
Another example is the Fourier transform, which represents an image in terms of complex exponential (or equivalently, sine and cosine) basis functions:



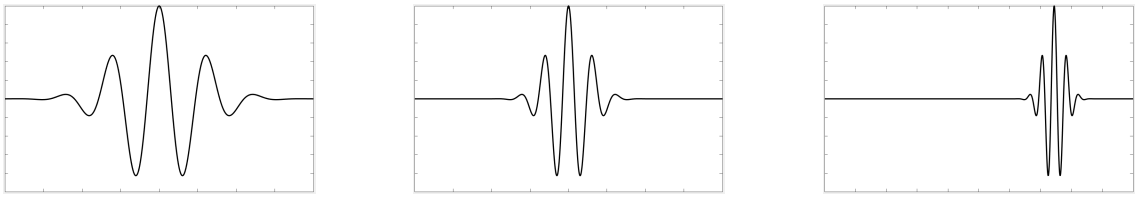
When applied to an image (2-dimensional signal) we get coefficients in two dimensions:



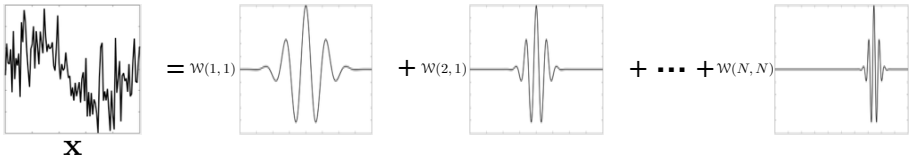
Similar to the Fourier transform, wavelet transforms represent signals (images) in terms of other basis functions. There are several types of wavelet transforms, each with different *mother* functions. Examples include the Morlet, the Mexican Hat, or the Daubechies mother functions:


Morlet Mexican Hat Daubechies

The basis functions of each wavelet type are given by frequency scalings (α) and time-shifts (τ) of the mother function $\psi(t)$:


Mother function $\psi(t)$ Scaling $\psi(t/\alpha)$ Time-shift $\psi(t-\tau)$

The wavelet transform represents a signal in terms of these bases:



10.2 Continuous Wavelet Transform

More formally, if $\psi(t)$ is the wavelet mother function, then the wavelet transform of a function $f(t)$ is given by:

$$\mathcal{W}(\alpha,\tau) \;=\; \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{\alpha}}\psi^*(\frac{t-\tau}{\alpha})dt,$$

Notice that in contrast to the Fourier transform, which only has one parameter (frequency), the wavelet transform has two parameters: a scaling α (which works as proxy of the frequency), and a time-shift parameter τ. The inverse wavelet transform is defined as:

$$f(t) \;=\; \frac{1}{C_\psi}\int_0^\infty\int_{-\infty}^\infty \mathcal{W}(\alpha,\tau)\frac{1}{\sqrt{\alpha}}\psi(\frac{t-\tau}{\alpha})d\tau\frac{d\alpha}{\alpha^2},$$

where

$$C_\psi \;:=\; \int_0^\infty \frac{|\Psi(w)|^2}{w}dw,$$

and $\Psi(w)$ is the Fourier transform of $\psi(t)$. If you want to know more about the wavelet transform, I recommend *A Tutorial of the Wavelet Transform*, by Liu Chun-Lin, available here: `http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial.pdf`

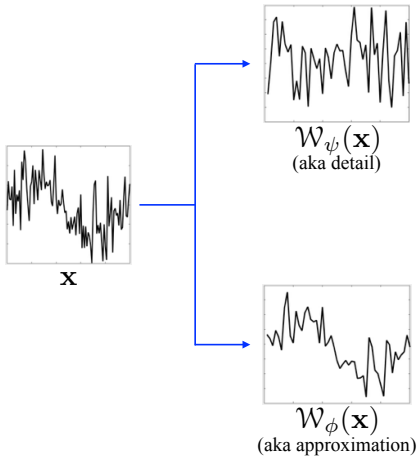
10.3 Discrete Wavelet Transform

The main idea of the discrete wavelet transform is to decompose a function $\mathbf{x} \in \mathbb{R}^N$ in terms of two orthogonal sets of basis functions: $\{\phi_{\alpha_0,\tau}[n]\}_{\mathrm{k}\in\mathbb{Z}}$ and $\{\psi_{\alpha,\tau}[n]\}_{(\alpha,\tau)\in\mathbb{Z}^2,\alpha\geq\alpha_0}$, where

$$\phi_{\alpha_0,\tau}[n] \;:=\; \frac{1}{\sqrt{\alpha_0}}\phi\left[\frac{n-\tau}{\alpha}\right], \qquad \psi_{\alpha,\tau}[n] \;:=\; \frac{1}{\sqrt{\alpha}}\phi\left[\frac{n-\tau}{\alpha}\right].$$

As before, each family of wavelet transform (Morlet, Daubechies, etc.) has a pair of mother functions (ϕ, psi). The coefficients w.r.t. ϕ are called the *approximation* coefficients, and the coefficients w.r.t. ψ are called *detail* coefficients. Intuitively, the discrete wavelet transform splits $\mathbf{x}$ into approximation coefficients that give a low-definition approximation of $\mathbf{x}$, and detail coefficients that give the high-definition details. These coefficients are given by:

$$\begin{aligned}\mathcal{W}_\phi[\alpha_0,\tau] \;&=\; \frac{1}{\sqrt{N}}\sum_n \mathbf{x}[n]\phi_{\alpha_0,\tau}[n],\\ \mathcal{W}_\psi[\alpha,\tau] \;&=\; \frac{1}{\sqrt{N}}\sum_n \mathbf{x}[n]\psi_{\alpha_0,\tau}[n], \qquad \alpha>\alpha_0.\end{aligned}$$

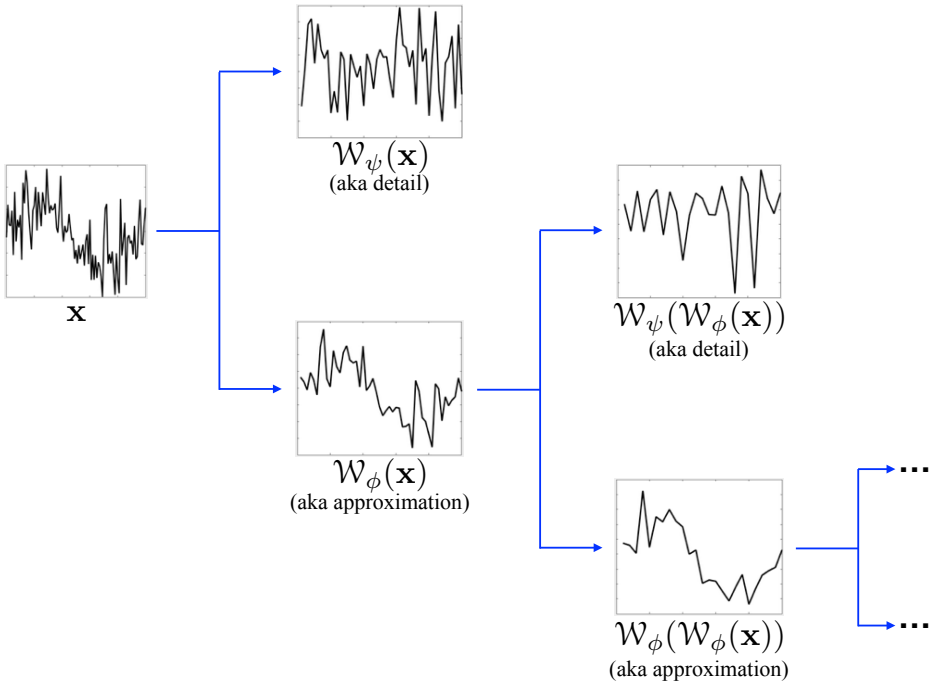


This wavelet transform can be easily computed in Matlab using the `dwt` function, which receives as parameters the function $\mathbf{x}$ whose transform we want to compute, the wavelet family (e.g., `'db2'` for the Daubechies mother function), and returns the approximation and detail coefficients:

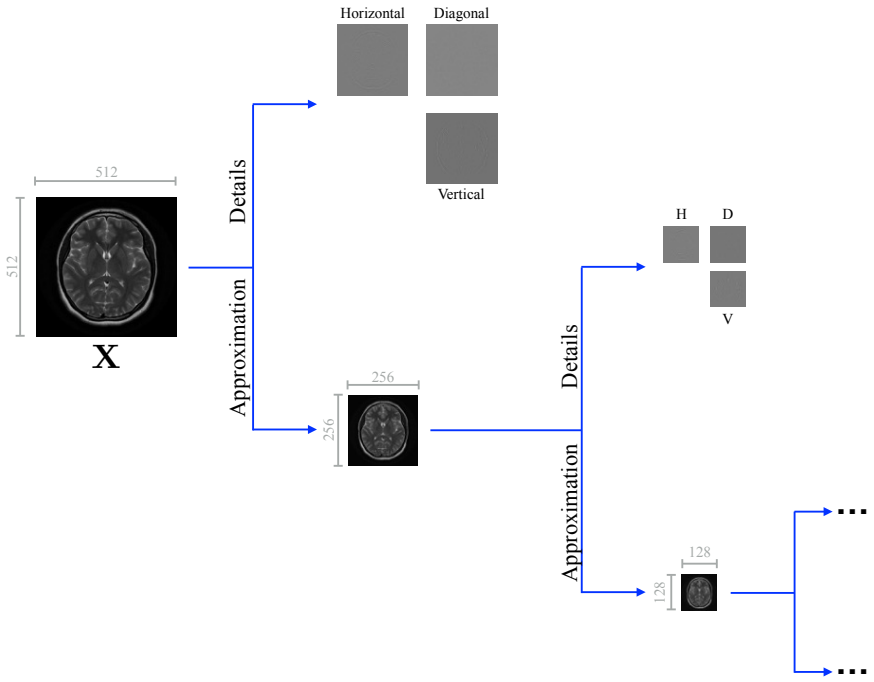
```
[CA,CD] = dwt(x,'db2');
```

10.4 Wavelet Transform in Practice

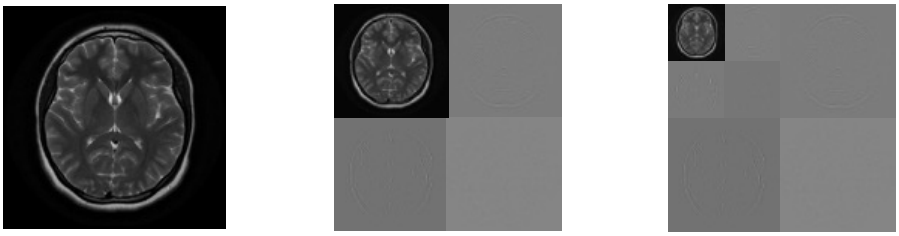
The wavelet transform is a powerful compression tool. The main reason is that for a lot of signals, wavelet detail coefficients tend to be sparse. A typical approach is to recursively compute the wavelet transform on the approximation component of $\mathbf{x}$. That is, split $\mathbf{x}$ into its detail and approximation components $\mathcal{W}_\psi(\mathbf{x})$ and $\mathcal{W}_\phi(\mathbf{x})$, then recursively split $\mathcal{W}_\phi(\mathbf{x})$ into new approximation and detail components $\mathcal{W}_\psi(\mathcal{W}_\phi(\mathbf{x}))$ and $\mathcal{W}_\phi(\mathcal{W}_\phi(\mathbf{x}))$, and so on.



At each splitting step, the signal size is reduced in half. For images, at each level we split into horizontal, vertical, and diagonal *details*, and one global *approximation*:

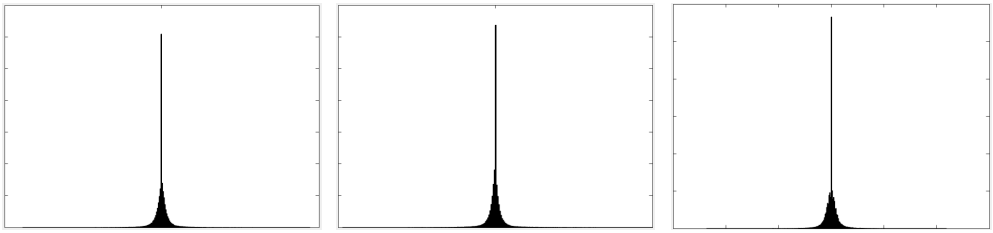


Wavelet transforms of images are often depicted stacking all these components:



Original Image | Level-1 $\mathcal{W}$ transform | Level-2 $\mathcal{W}$ transform

Notice that all the *detail* coefficients are very sparse. For example, here are the histograms of the level-1 wavelet details of the previous image:



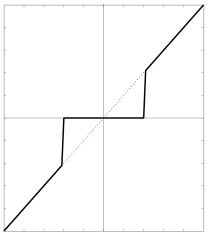
Horizontal Details | Vertical Details | Diagonal Details

Consequently, one can *ignore* the near-zero coefficients, and store the wavelet transform very efficiently, resulting in powerful compression.

The process of *ignoring* the near-zero coefficients is equivalent to thresholding, typically done in two ways

Hard-thresholding

$$x_\lambda \;:=\; \begin{cases} x & \text{if } |x| > \lambda \\ 0 & \textit{otherwise.} \end{cases}$$



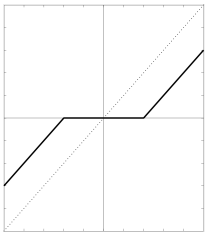
Notice that the hard threshold operator can also be written as

$$\mathbf{x}_\lambda \;=\; \mathbf{x} \odot \mathbb{1}_{\{|\mathbf{x}|>\lambda\}},$$

where $\mathbb{1}$ is the indicator function, and $\cdot$ denotes the Hadamard (point-wise) product.

Soft-Thresholding

$$x_\lambda \;:=\; \begin{cases} x-\lambda & \text{if } x > \lambda \\ x+\lambda & \text{if } x < -\lambda \\ 0 & \textit{otherwise.} \end{cases}$$



Notice that the soft threshold operator can also be written as:

$$\mathbf{x}_\lambda = \mathbf{x} \odot \max\left(1 - \frac{\lambda}{|\mathbf{x}|}, 0\right).$$

Empirically people have observed that the soft-threshold tends to produce smoother results, while the hard-threshold tends to keep more contrast.

Universal Threshold

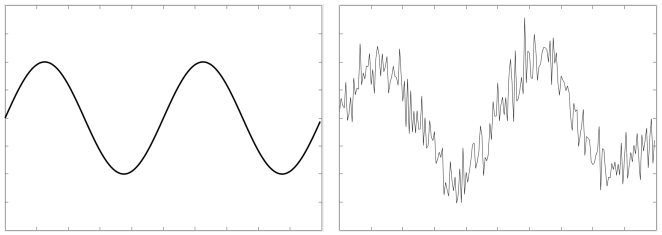
The trick is usually how to chose the threshold λ. Luckily, there are some rules of thumb, like the so-called *universal threshold*:

$$\lambda_\star \;=\; \frac{2\log N \cdot \text{median}(|\mathcal{W}_\phi(\mathbf{x})|)}{0.675},$$

where N is the size of $\mathbf{x}$, and $\mathcal{W}_\phi(\mathbf{x})$ is the set of level-1 detail coefficients of $\mathbf{x}$.

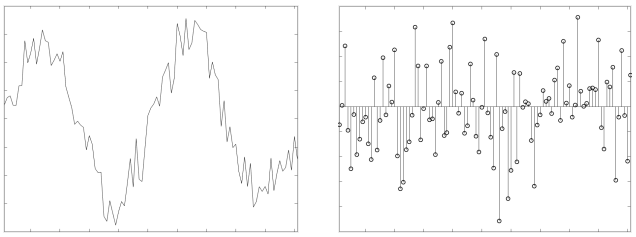
10.5 Denoising

Besides compression, thresholding small wavelet coefficients can also work as a noise reduction tool. For example, consider the following noiseless and noisy signal:



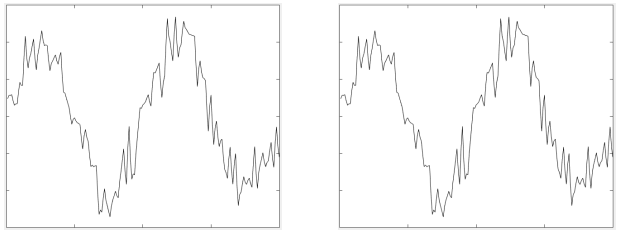
Noiseless Signal Noisy Signal

To denoise the later, we can compute its wavelet decomposition:



Approximation Detail

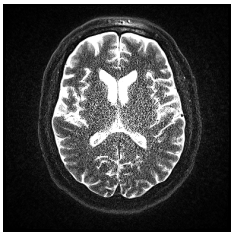
We can threshold the detail coefficients using the universal threshold $\lambda_\star$, and invert the wavelet transform to obtain the following denoised signals:



Hard-Thresholding Soft-Thresholding

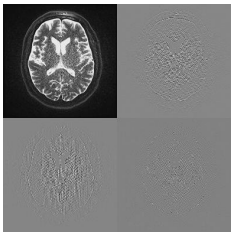
Notice that while not ideal, these signals are less noisy than the initial signal. This denoising procedure can be further improved with higher-level wavelet transforms.

For images we can do something similar. Consider the following image:



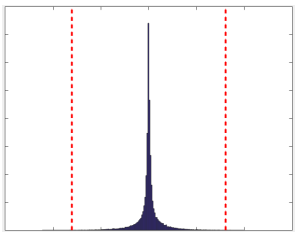
$\mathbf{X}$

We can compute its wavelet transform:

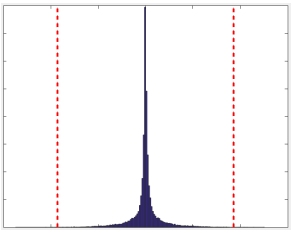


$\mathcal{W}(\mathbf{X})$

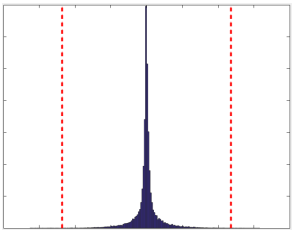
Notice that the details are very sparse, and we can threshold them (universal threshold $\lambda_\star$ indicated in red)



Horizontal Details

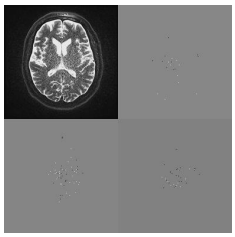


Vertical Details

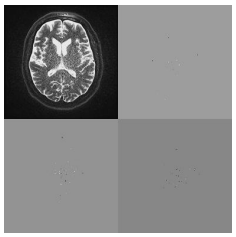


Diagonal Details

Here are the thresholded details:

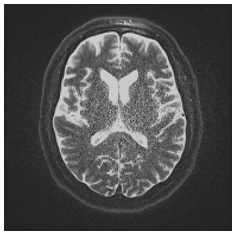


Hard-Thresholding

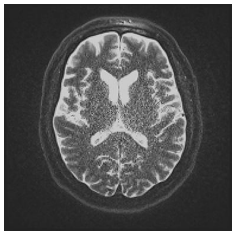


Soft-Thresholding

After inverting with the thresholded coefficients, we obtain the following denoised results:



Hard-Thresholding



Soft-Thresholding

Notice that while not ideal, these images are less noisy than the initial image. This denoising procedure can be further improved with higher-level wavelet transforms.