9. Image Processing - Multi-Resolution Image Processing

Institute for Biomedical Imaging, Hamburg University of Technology

- Lecture: Prof. Dr.-Ing. Tobias Knopp
- Exercise: Konrad Scheffler, M.Sc.

Table of Contents

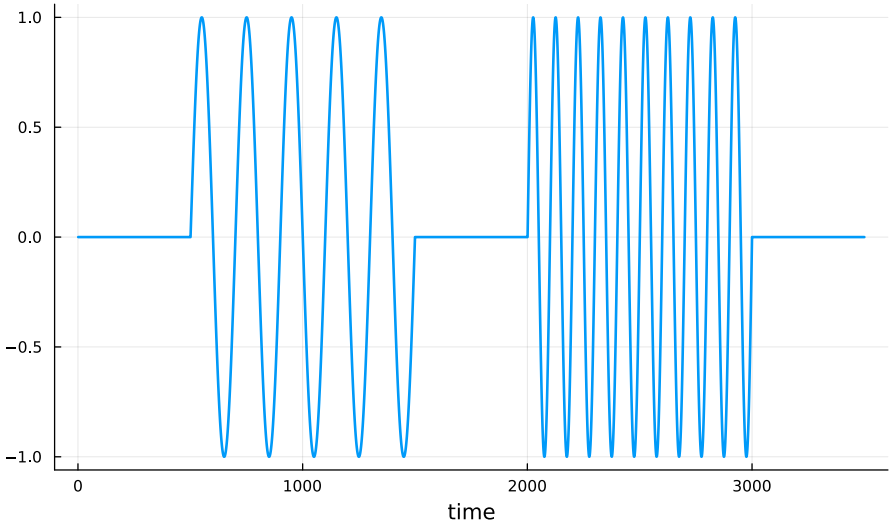
9.1 Motivation

There are different ways to motivate multi-resolution image processing. We start with a very general one coming from signal processing. We then use a slightly different viewpoint discussing image pyramids. Both will end up in a joint framework that is based on the wavelet transform.

> **Note**
>
> Understanding the complete theory of wavelets is a lot more technical than for the Fourier series. We try to make the mathematics as comprehensive as possible.

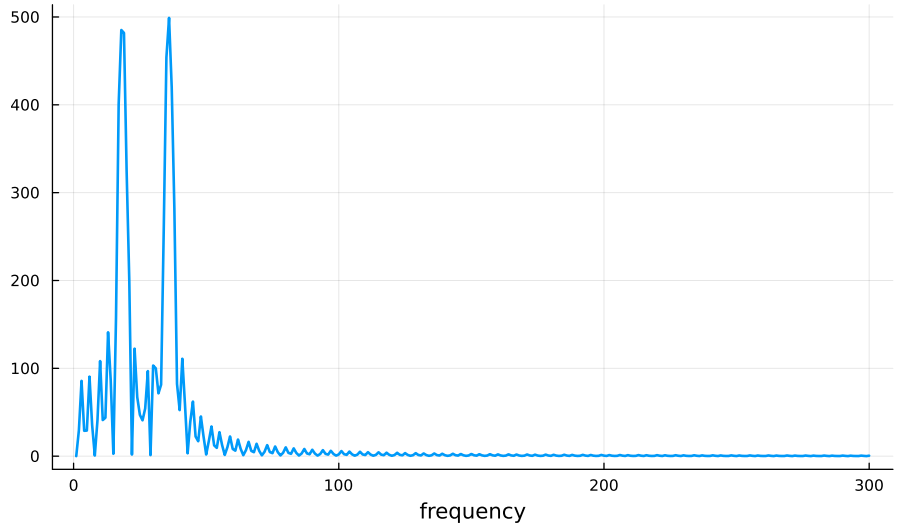
9.1.1 Time / Frequency Analysis

Let us have a look at the following signal:



One might now be interested in the question: What frequency does this signal have?

In order to investigate this we can perform a Fourier transform:



What we see is that the function contains two major frequencies.

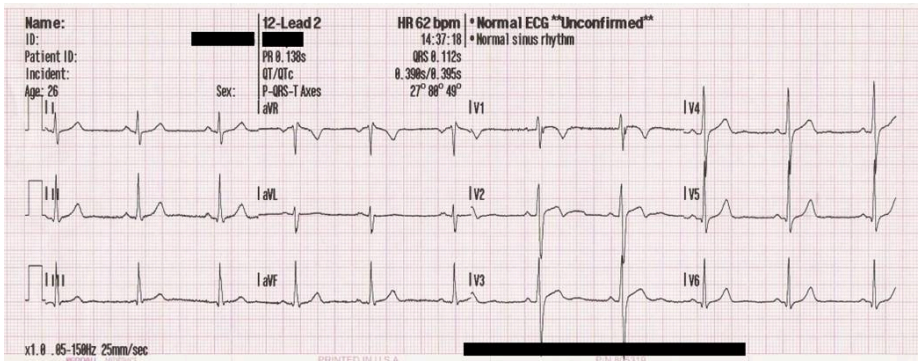
The problem now is the following:

- the time signal does not contain information about the frequency.
- the frequency signal does not contain information about the time, at which the frequency appears.

Why should we care?

Two prominent examples: First is audio signal processing. When playing a song on an instrument, the tone (=base frequency) changes over time. In this case it is very important *when* a certain tone appeared.

When evaluating the heart function, one uses a electrocardiogram, which is basically a time plot that shows the pumping action of the heart:



In this case, again, one is interested in the heart frequency over time. This is also what you look at when using fitness tracker.

9.1.2 Short-Term Fourier Transform

What we want is actually time-resolved frequency information. This can be accomplished by taking small intervals and applying a Fourier transform on the smaller intervals. This is named the short-term Fourier transform. For the continuous case it can be mathematically defined as

$$\mathrm{STFT}(s(t))(\tau, f) = S(\tau, f) = \int_{-\infty}^{\infty} s(t) w(t-\tau) e^{-2\pi i f t}\, dt.$$

Here $w(t)$ is a window function that has two purposes:

1. Cutting out a small snippet of the time signal.
2. Letting the signal go smoothly to zero to avoid spectral leakage.

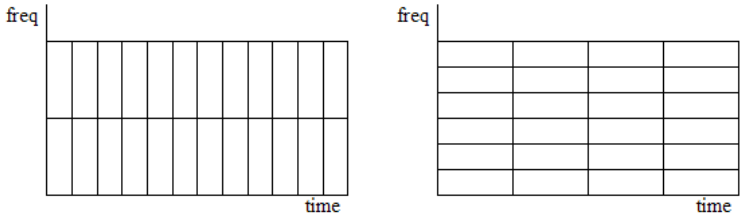
Usually a Hann window is used for that purpose.

The STFT returns not a 1D function but a 2D function with parameters τ and f. Thus, we have a time and frequency resolution now.

9.1.2.1 Time/Frequency Resolution

Although we have defined the STFT in a continuous fashion, it still has a discrete parameter that prevents us to have infinite time and frequency resolution. This is the width of the window w.

The wider w, the more frequencies can calculated but the time resolution decreases. Thus, one can trade of time and frequency resolution as is shown in the following time/frequency diagram:

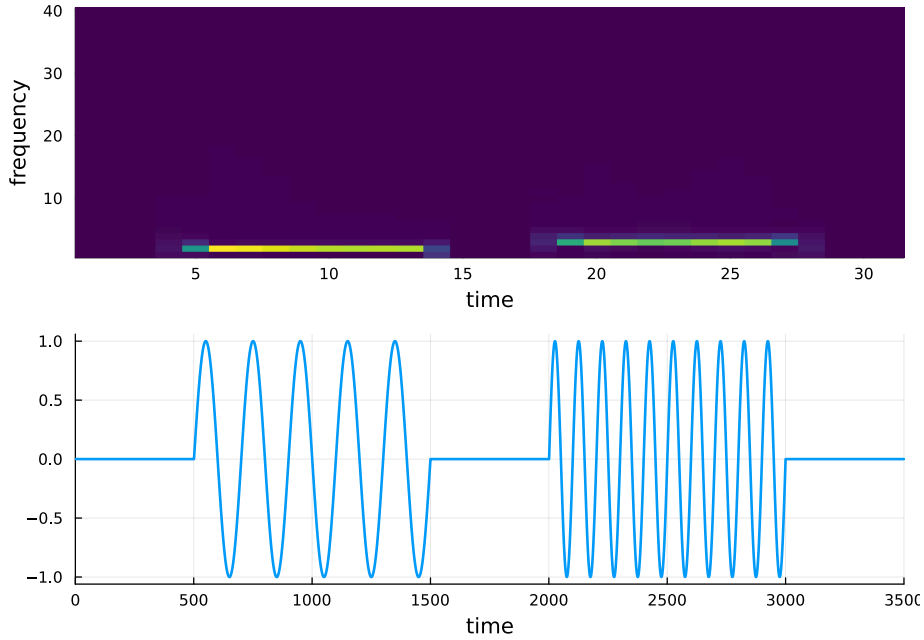


9.1.2.2 Spectrogram

The spectrogram is defined to be the power spectrum of the STFT:

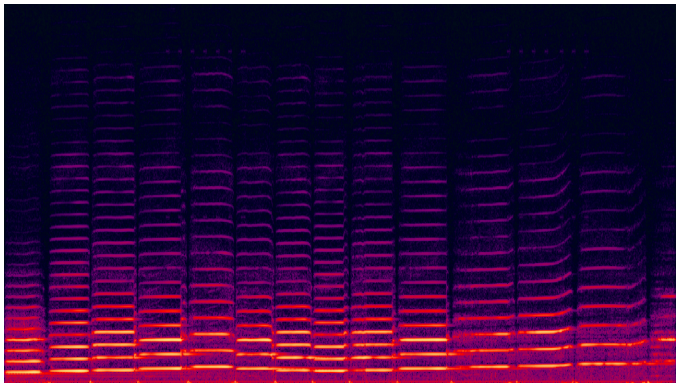
$$\mathrm{spectrogram}(s(t))(\tau, \omega) \equiv |\mathrm{STFT}(s(t))(\tau, \omega)|^2$$

It is used especially in audio signal processing. Here is the spectrogram of our original function:



One can clearly see that both time and frequency are resolved, i.e. we see that the one frequency is twice of the other frequency and where both frequencies occur on the time stripe.

The following shows a spectrogram of a violine play:

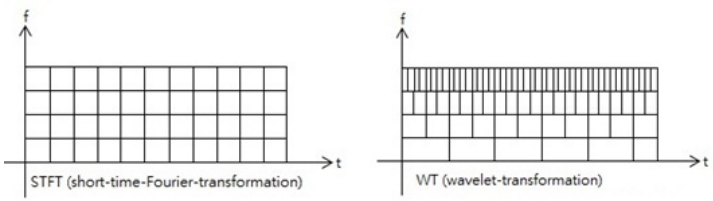


Whats wrong with the STFT?

If we look at the time/frequency diagram of the STFT we can see that we have an equidistant sampling of time and space. This is not very flexible. In particular, we would actually want to have

- high frequency resolution for the low frequencies. For them we do not need high time resolution.
- high time resolution for high frequencies. For them we do not need high frequency resolution.

Thus, what we actually want is a *multi-resolution transform*, i.e. we want to spend less samples for the lower frequencies and more samples on the high frequency parts. The transformation that accomplishes this is named the wavelet transform. The following shows how the wavelet transform samples the time/frequency space.



We come to the proper definition of the wavelet transform later in this lecture.

9.1.2.3 Uncertainty Principle

An important theorem regarding the ability to resolve both time and frequency is the uncertainty principle:

Theorem

A signal cannot be band-limited in time and frequency. In particular it holds for any signal that

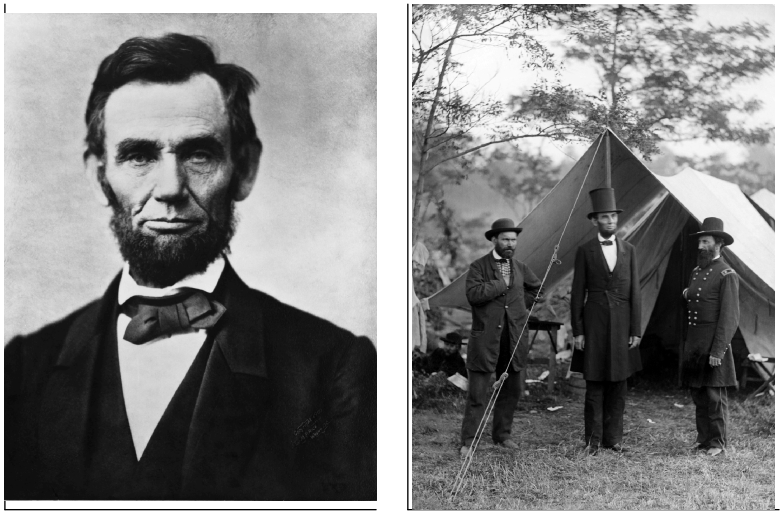
$$\sigma_t \cdot \sigma_f \geq \frac{1}{4\pi}$$

where σ_t and σ_f are the standard deviations of the time and frequency estimates respectively (i.e. the width of the signals).

The link to the STFT is the width of the window function. The smaller we make it in time domain, the wider it gets in frequency domain.

9.1.3 Human Pattern Recognition

We have another convincing argument for performing multi-resolution image processing. Let us look at the following two pictures. What can you see there?



Of course both pictures contain the 16th president of the united states Abraham Lincoln. And it was basically possible to recognize this in a fraction of a second, although Lincoln has a different size in both pictures.

How can we do that? Is our brain extremely powerful and can do STFTs in realtime?

The answer is no, our brain is much smarter and performs multi-resolution pattern matching. So our brain is capable of

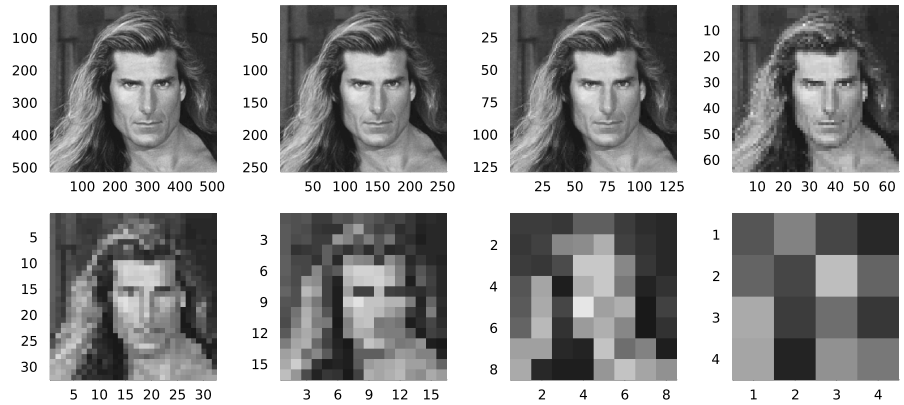
- first looking at the pictures identifying persons
- then focussing on the respective parts and the comparing them.

9.2 Image Pyramids

Before we come to the wavelets we first take a look at so-called image pyramids, which are a predecessor to wavelets.

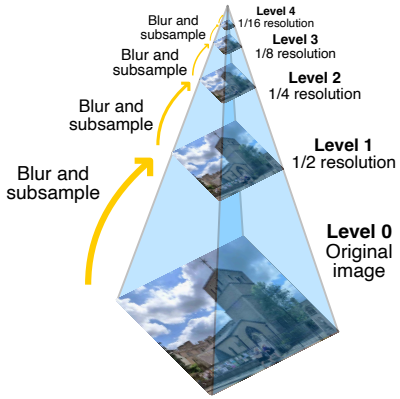
9.2.1 Gaussian Pyramid

We start by looking at the image of Lena in different resolutions:



We call this a Gaussian pyramid.

Why? Because we can stack the images like this:



When switching from one layer to the next we need to decrease the image size. This is done by two operations:

- smoothing filtering
- downsampling (typically a factor of two)

The first step is very crucial to avoid aliasing artifacts. For the moment you can think of a 2x2 Box filter that would do the job.

Mathematically we can define the function

$$\mathbf{downsample} : (I_N \to \Gamma) \to (I_{\frac{N}{2}} \to \Gamma)$$

for the Box filter in the 1D case like this:

$$\mathbf{downsample}(s(i))(j) = g(j) = \sum_{k=1}^{2} s(2j+k) \quad \text{for} \quad j = 1, \ldots, \frac{N}{2}$$

What does the Gaussian Pyramid allow for?

The Gaussian Pyramid allows us to operate on an image on different scales. What basically happens is that we are low-pass filtering in each step, i.e. in contrast to the STFT we are not selecting sharp frequency bands but larger portions of the frequency space of the signal.

The Gaussian Pyramid allows us to perform image processing algorithms (like pattern recognition) on different scales.

Size of Pyramid

One interesting question is how memory wasteful the storage of the pyramid is. Lets calculate this.

1D:

Let the base signal be of size N. Then the pyramid contains the following amount of pixels:

$$\left(N + \frac{N}{2} + \frac{N}{4} + \cdots + 1\right) = \sum_{i=0}^{\log_2(N)} \frac{N}{2^i} = N \sum_{i=0}^{\log_2(N)} \frac{1}{2^i} < N \sum_{i=0}^{\infty} \frac{1}{2^i} = \frac{1}{1-\frac{1}{2}} N = 2N$$

2D:

Let the base signal be of size $N = N_x \times N_y$. Then the pyramid contains the following amount of pixels:

$$\left(N + \frac{N}{4} + \frac{N}{16} + \cdots + 1\right) = \sum_{i=0}^{\log_4(N)} \frac{N}{4^i} = N \sum_{i=0}^{\log_4(N)} \frac{1}{4^i} < N \sum_{i=0}^{\infty} \frac{1}{4^i} = \frac{1}{1-\frac{1}{4}} N = \frac{4}{3} N$$

Thus, in the case of 2D images, the increase in space is just 1/3 more than the original image.

Time Complexity

Another question is how large the time complexity for creating the image pyramid is. Each downsampling requires $\mathcal{O}(\tilde{N})$ operations if $\tilde{N}$ is the size of the considered level. This is because we can use a small fixed-size kernel during the downsampling.

Consequently, the creation of a 2D image pyramid is $\mathcal{O}(N)$, which follows with the same calculation as for the space requirement.

> **Note**
>
> At this point we already see a very important strength of multi-resolution processing algorithms. By dividing the resolution always by a factor of two we can obtain $\mathcal{O}(N)$ algorithms, which are much faster than general linear transformations that require $\mathcal{O}(N^2)$ operations.

9.2.2 Laplacian Pyramid

Now what is of course interesting is the difference between the downsampled image and the original image on a certain pyramid level. To this end we can define a function

$$\mathbf{upsample} : (I_{\frac{N}{2}} \to \Gamma) \to (I_N \to \Gamma)$$

that takes the low resolution image and upsamples it using an appropriate interpolation technique (which does not matter at this point).

The difference between the down- and upsampled image and the original image now can be expressed as

$$h(x,y) = f(x,y) - \mathbf{upsample}(\mathbf{downsample}(f(x,y))$$

Why is that useful?

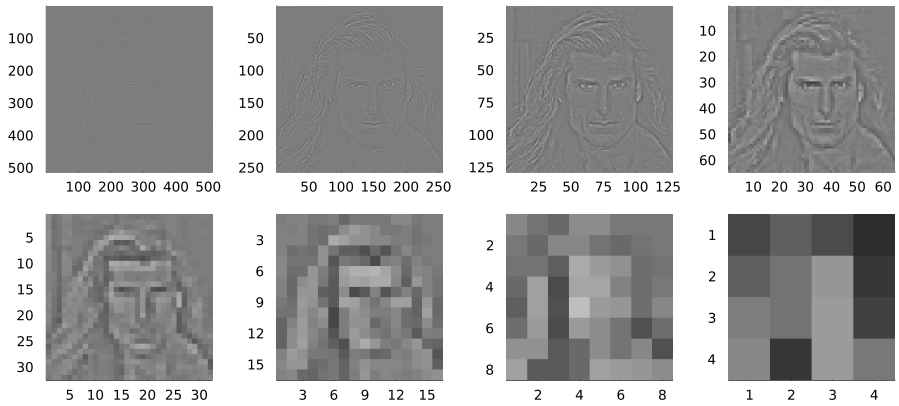
In the Gaussian pyramid we store $f(x,y)$ and $\mathbf{downsample}(f(x,y))$ and $\mathbf{downsample}(\mathbf{downsample}(f(x,y)))\dots$

Now what if we drop $f(x,y)$ and instead store the difference image $h(x,y)$. Can we then restore $f(x,y)$ exactly from $\mathbf{downsample}(f(x,y))$ and $h(x,y)$?

Yes, of course:

$$f(x,y) = h(x,y) + \mathbf{upsample}(\mathbf{downsample}(f(x,y))$$

The idea of the Laplacian pyradmid is now to only store the difference image on each level. Just on the last level we store the downsampled image (usually $\mathbf{2 \times 2}$). This looks like this:



So what we store here is the edge information of the image. In fact, if you go back to the lecture on filtering in spatial domain you will see that we have effectively applied a *high-pass filter*.

Again, why is that useful?

First of all we have the same amount of data being stored as for the Gaussian pyramid. And we are able to perfectly recover each image on every level.

If you compare the Gaussian and the Laplacian pyramid you see that the later is a lot sparser, i.e. it contains in each level mostly zeros (or small values) and only at edges larger values. Consequently we can use this for image compression. To this end we store only the significant values (will be discussed again later).

9.3 Wavelets

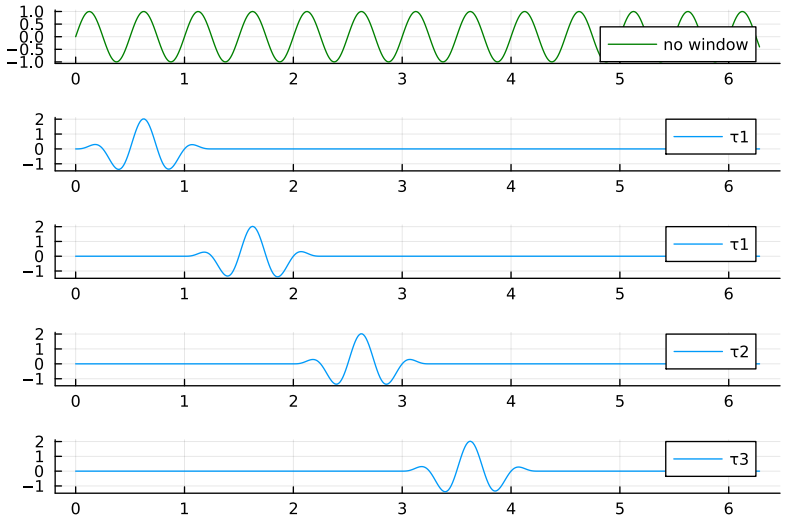
Wavelets have been developed in the 1980s and the 1990s and can be seen as a generalization of the STFT. Two important researchers having pushed wavelets and their theory are Ingrid Daubechies and Stéphane Mallat.

Why the term wavelet?

Wavelets basically means *small wave* and can be motivated even without wavelet theory by just looking at the STFT again:

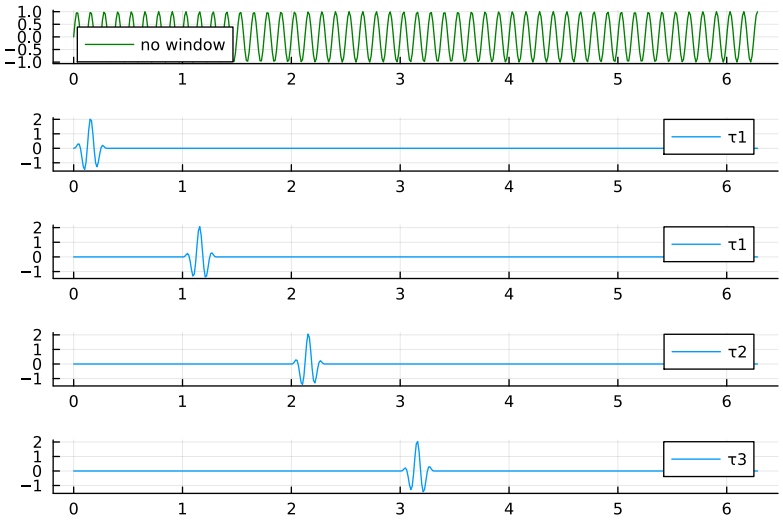
$$\mathrm{STFT}(s(t))(\tau, f) = \int_{-\infty}^{\infty} s(t) w(t-\tau) e^{-2\pi i f t}\, dt$$

One can see that the complex exponential is multiplied with the window. Let us have a look at an exemplary $w(t-\tau)e^{-2\pi i f t}$ for a fixed frequency and four different shifts τ:



So you can see that

- the term wavelet (small wave) makes quite some sense.
- we can already see the shifting, which will be very important for the wavelets.
- the other important operation is the scaling, which is done for the STFT by changing the frequency as is shown in the next figure.



9.3.1 Continuous Wavelet Transform

So we have now all ingredient together to formally define the wavelet transform:

- we have the shifting with τ.
- we have the frequency change, which can also be seen as a *scaling* or *dilation*.
- we have our small wave.

With that the continuous wavelet transform (CWT) can be written as

$$W_\psi(s(t))(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} s(t) \overline{\psi\left(\frac{t-b}{a}\right)} dt\,.$$

If you compare this to the STFT you can see that the only difference is that

- the basis function $w(t-\tau)e^{-2\pi i f t}$ has been replaced by the wavelet ψ. So this is mainly a generalization.
- τ has been renamed to b (to use the common notation).
- instead of the frequency f we consider the scaling a.

9.3.1.1 Mother Wavelet

We name ψ the mother wavelet since it is used to generate an entire family of shifted and dilated wavelets:

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

Thus we can write the continuous wavelet transform as

$$W_\psi(s(t))(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} s(t) \overline{\psi_{a,b}(t)} dt$$

9.3.1.2 Properties

Wavelets are usually taken from the function space $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, i.e. it holds that

$$\int_{-\infty}^{\infty} |\psi(t)|\, dt < \infty \qquad \text{and} \qquad \int_{-\infty}^{\infty} |\psi(t)|^2\, dt < \infty.$$

Additionally the mother wavelet needs to fulfill the *admissibility condition*:

$$C_\psi = \int_{\mathbb{R}} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} \mathrm{d}\omega < \infty$$

where $\hat{\psi}(f)$ is the Fourier transform of $\psi(t)$ and C_ψ is named the admissible constant.

The admissibility condition directly implies that the Fourier transform of ψ vanishes at frequency zero $\hat{\psi}(0) = 0$ so that

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0$$

This explains that

- Wavelets look like waves, i.e. they have the same amount of positive as negative parts.
- Wavelets have a high-pass/band-pass characteristic.

9.3.1.3 Inverse

The inverse of the continuous wavelet transform can be expressed as

$$s(t) = \int_{\mathbb{R}} \int_{\mathbb{R}} W_\psi(s(t))(a,b) \psi_{a,b}(t)\, \mathrm{d}b\, \frac{\mathrm{d}a}{a^2}.$$

> **Note**
>
> The CWT is highly redundant, i.e. it maps a 1D signal into a 2D space. Thus the CWT is mainly used for signal analysis where such a redundancy can be ok/desired if fine-grained information about signal bands is required.
>
> Since the CWT is mainly used for signal analysis, the synthesis (application of the inverse) is not that important.

9.3.2 Wavelet Series

Since the CWT is so redundant it makes sense to reduce the space by sampling the parameters a and b. Usually this is done in a dyadic fashion ($a = 2^{-m}$ and $b = n2^{-m}$) yielding

$$\psi_{m,n}(t) = 2^{-\frac{m}{2}} \psi\left(2^{-m}t - n\right) \quad \text{where} \quad m,n \in \mathbb{Z}.$$

One can see that the translation is adapted to the scaling such that wider wavelets require less translations.

> **Note**
>
> We from now on assume the wavelets to be real-valued which holds true in most practical use cases.

9.3.2.1 Properties

The functions $\psi_{m,n}(t)$ defined in the previous way are an orthonormal basis of the Hilbert space $L^2(\mathbb{R})$. Thus the functions fulfill:

$$\langle \psi_{m,n}, \psi_{m',n'} \rangle = \int_{-\infty}^{\infty} \psi_{m,n}(t)\psi_{m',n'}(t)\mathrm{d}t = \delta_{m,m'}\delta_{n,n'}$$

where $\delta_{m,n}$ is the Kronecker delta.

With this we can formulate the wavelet series:

$$\begin{aligned} s(t) &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \langle s, \psi_{m,n}\rangle \psi_{m,n}(t) \\ &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} T_{m,n}\psi_{m,n}(t) \end{aligned}$$

with the wavelet coefficients $T_{m,n}$ that can be calculated by

$$T_{m,n} = \langle s, \psi_{m,n}\rangle = \int_{\mathbb{R}} s(t)\psi_{m,n}(t)\mathrm{d}t$$

> **Note**
>
> Compare the wavelet series with the continuous Fourier transform. Both are defined on continuous functions on the real line, which is different than what we had for the Fourier transform. The wavelet series is now not redundant anymore.

9.3.2.2 Scaling Functions

What is missing until now is the connection to the image pyramids that allowed us to express images on different scales. This is accomplished by splitting the sum in the wavelet series into the wavelet part and a part that encodes the low-resolution part of the image.

To understand this we first need to introduced the *scaling functions* ϕ. They are directly related to the wavelet ψ and in fact are derived from ψ (or vice versa).

> **Note**
>
> Before we proceed: Think of ϕ being a low-pass filter (e.g. a box filter) whereas the wavelet ψ is a high- or band-pass filter. We next need some technical details to formally construct the scaling functions.

The scaling functions can be expressed as

$$\phi_{m,n}(t) = 2^{-\frac{m}{2}}\phi\left(2^{-m}t - n\right) \quad \text{where} \quad m,n \in \mathbb{Z}$$

where $\phi(t)$ is the base scaling function, sometimes also called the *father wavelet*. The scaling functions are orthogonal with respect to shifts but not with respect to scalings. The father wavelet is constructed in a way that it fullfills

$$\int_{\mathbb{R}} \phi(t)\mathrm{d}t = 1.$$

Furthermore, the scaling functions are constructed in such a way that they are orthogonal to the wavelets on the same or a higher scale, i.e. we have

$$\begin{aligned} \langle \phi_{m,n}, \phi_{m,n'} \rangle &= \delta_{n,n'} \quad \text{for} \quad m,n,n' \in \mathbb{Z} \\ \langle \phi_{m,n}, \psi_{m',n'} \rangle &= 0 \quad \text{for} \quad m \geq m' \text{ and } m,n,n',n' \in \mathbb{Z} \end{aligned}$$

9.3.2.3 Low-Pass Representation

Using the scaling functions we can perform a low-pass filtering of a signal $s(t)$ by convolving $s(t)$ with $\phi_{m,k}(t)$ yielding the *approximation coefficients*

$$S_{m,k} = \langle s, \phi_{m,k}\rangle = \int_{\mathbb{R}} s(t)\phi_{m,k}(t)\mathrm{d}t.$$

Since $\phi_{m,k}(t)$ are orthogonal on the same scaling level, we can take the discrete signal $S_{m,k}$ and calculate a *continuous approximation* of $s(t)$ by

$$s_m(t) = \sum_{k=-\infty}^{\infty} S_{m,k}\phi_{m,k}(t).$$

> **Note**
>
> Think about what happens if you let go $m \to -\infty$. Then $\phi_{m,k}$ will approach a Dirac distribution and in turn $s_m(t)$ will approach $s(t)$. When switching to the discrete setting, we can thus directly consider $S_{m,n}$ to be our input signal where m is the level defined by our pixel size. We stick with $s(t)$ for a moment but keep in mind that $S_{m,n}$ will be our discrete signal.

9.3.2.4 Scaling and Wavelet Subspaces

Having defined the wavelet and the scaling functions we can look into their respective function spaces. We define

$$\begin{aligned} V_m &= \mathrm{span}(\phi_{m,n} : n \in \mathbb{Z}) \\ W_m &= \mathrm{span}(\psi_{m,n} : n \in \mathbb{Z}) \end{aligned}$$

to be the scaling function and the wavelet function space. From these definitions it is clear that

$$\{0\} \subset \cdots \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \subset L^2(\mathbb{R}),$$

i.e. if we move one scale down from V_m to V_{m-1} we can express the same function plus those on the finer scale. It is also clear that

$$\lim_{m \to -\infty} V_m = L^2(\mathbb{R})$$

since we are basically generating Dirac deltas that can be shifted around aribitrarily fine.

More interesting is now the relation to the wavelet spaces. One can show that W_m is the orthogonal complement of V_m inside the subspace V_{m-1}, i.e.

$$V_m \oplus W_m = V_{m-1},$$

where $\oplus$ is the direct sum of the function spaces. This followed from our assumption that the wavelets are orhtogonal to the scaling functions on the same level.

From this we now can derive:

$$L^2(\mathbb{R}) = \underbrace{\underbrace{\underbrace{V_m \oplus W_m}_{V_{m-1}} \oplus W_{m-1}}_{V_{m-2}} \oplus W_{m-2}}_{V_{m-3}} \oplus \cdots$$

which tells us that we can consider the band-limited function space V_m and put all wavelet spaces $W_{m'}$ with $m' < m$ and then can express the entire $L^2(\mathbb{R})$. If we let $m \to \infty$ we derive

$$L^2(\mathbb{R}) = \bigoplus_{m \in \mathbb{Z}} W_m,$$

which is our original assumption that the wavelets form an orthonormal basis of the Hilber space $L^2(\mathbb{R})$.

9.3.2.5 Multi-Resolution Wavelet Series

What we have shown in the previous section implies that we can either express a signal $s \in L^2(\mathbb{R})$ either as

$$s(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} T_{m,n} \psi_{m,n}(t).$$

or as

$$s(t) = \sum_{n=-\infty}^{\infty} S_{m_0,n} \phi_{m_0,n}(t) + \sum_{m=-\infty}^{m_0} \sum_{n=-\infty}^{\infty} T_{m,n} \psi_{m,n}(t).$$

The first part is the low resolution part $s_{m_0}(t)$ and the second part are the details at different levels. By defining

$$d_m(t) = \sum_{n=-\infty}^{\infty} T_{m,k} \psi_{m,n}(t)$$

we obtain

$$s(t) = s_{m_0}(t) + \sum_{m=-\infty}^{m_0} d_m(t)$$

From that we can directly derive

$$s_{m-1}(t) = s_m(t) + d_m(t)$$

which tells us that if we add the signal detail at an arbitrary scale (index m) to the approximation at that scale we get the signal approximation at an increased resolution (i.e. at a smaller scale, index $m-1$). This is called a multi-resolution representation.

> **Note**
>
> We already have everything in place for performing a multi-resolution decomposition, which calculates all difference images $d_m(t)$ until the pixel resolution and additionally needs to store one low-resolution image $s_{m'}(t)$.
>
> One could of course calculate $s_{m'}(t)$ and all the $d_m(t)$ directly by calculating the coefficients $T_{m,n}$ and $S_{m',n'}$ via the inner products. However, this is not efficient and we therefore derive in the next two subsections a more efficient ways to do this.

9.3.2.6 Two-Scale Relation

To derive the fast wavelet transform we need the two scale relations that can be derived as follows. Lets have a look at the inner products of the wavelet and the scaling functions with the scaling functions one level lower:

$$c_k = \langle \phi_{0,0}, \phi_{-1,k} \rangle = \int_{\mathbb{R}} \phi_{0,0}(t) \phi_{-1,k}(t)\, \mathrm{d}t = \sqrt{2} \int_{\mathbb{R}} \phi(t) \phi(2t-k)\, \mathrm{d}t$$
$$b_k = \langle \psi_{0,0}, \phi_{-1,k} \rangle = \int_{\mathbb{R}} \psi_{0,0}(t) \phi_{-1,k}(t)\, \mathrm{d}t = \sqrt{2} \int_{\mathbb{R}} \psi(t) \phi(2t-k)\, \mathrm{d}t$$

c_k and b_k are named the scaling and wavelet coefficients, respectively. Since $\phi, \psi \in V_{-1}$ we have

$$\phi(t) = \sum_{k=-\infty}^{\infty} \langle \phi_{0,0}, \phi_{-1,k} \rangle \phi_{-1,k}(t) = \sum_{k=-\infty}^{\infty} \sqrt{2}\, c_k\, \phi(2t-k)$$
$$\psi(t) = \sum_{k=-\infty}^{\infty} \langle \psi_{0,0}, \phi_{-1,k} \rangle \phi_{-1,k}(t) = \sum_{k=-\infty}^{\infty} \sqrt{2}\, b_k\, \phi(2t-k)$$

These are called the two-scale relations. They imply that we can break down $\phi(t)$ into several parts that can be expressed as a scaled and shifted versions of $\phi(t)$ again. The same holds true for the wavelets $\psi(t)$.

> **Note**
>
> In practice both series have only few non-zero coefficients, i.e. one designs the wavelet in that way.

Finally, if we now consider our dyadic wavelets $\psi_{m,n}$ and the dyadic scaling functions $\psi_{m,n}$ we can exploit the two-scale relations to end up at

$$\psi_{m+1,n}(t) = \sum_{k=-\infty}^{\infty} b_k \, \phi_{m,2n+k}(t)$$
$$\phi_{m+1,n}(t) = \sum_{k=-\infty}^{\infty} c_k \, \phi_{m,2n+k}(t)$$

We will need these relations later.

9.3.2.7 Connection Between Coefficients

The coefficients c_k and b_k are directly connected. In practice this means that one defines the scaling function ϕ by defining c_k and from that directly derives b_k, which defines the wavelet ψ. This is because we required the wavelets to be the orthogonal complement to the scaling functions.

The relation is given by

$$b_k = (-1)^k c_{1-k} \qquad k \in \mathbb{Z}.$$

To show that these coefficents lead to orthogonal wavelets we calculate

$$\begin{aligned}
\langle \phi_{0,0}, \psi_{0,n} \rangle &= \int_{\mathbb{R}} \phi_{0,0}(t) \psi_{0,n}(t) \mathrm{d}t \\
&= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} c_k b_l \underbrace{\int_{\mathbb{R}} \phi_{-1,k}(t) \phi_{-1,2n+l}(t) \mathrm{d}t}_{\delta_{k,2n+l}} \\
&= \sum_{k=-\infty}^{\infty} c_k b_{k-2n} \\
&= \sum_{k=-\infty}^{\infty} c_k (-1)^{k-2n} c_{1-k+2n} \\
&= \sum_{k=-\infty}^{\infty} c_k (-1)^{k} c_{1-k+2n} \\
&= \frac{1}{2} \left(\sum_{k=-\infty}^{\infty} c_k (-1)^{k} c_{1-k+2n} + \sum_{k=-\infty}^{\infty} c_k (-1)^{k} c_{1-k+2n} \right) \\
&= \frac{1}{2} \left(\sum_{k=-\infty}^{\infty} c_k (-1)^{k} c_{1-k+2n} + \sum_{\kappa=-\infty}^{\infty} c_{1-\kappa+2n} (-1)^{1-\kappa+2n} c_{\kappa} \right) \\
&= \frac{1}{2} \left(\sum_{k=-\infty}^{\infty} c_k (-1)^{k} c_{1-k+2n} + \sum_{\kappa=-\infty}^{\infty} (-1) c_{1-\kappa+2n} (-1)^{\kappa} c_{\kappa} \right) \\
&= \frac{1}{2} \sum_{k=-\infty}^{\infty} c_k (-1)^{k} c_{1-k+2n} - c_{1-k+2n} (-1)^{k} c_k = 0
\end{aligned}$$

In addition we can also derive some constraints on the coefficients c_k:

$$\begin{aligned}
1 &= \int_{\mathbb{R}} \phi(t) \, \mathrm{d}t \\
&= \int_{\mathbb{R}} \sum_{k=-\infty}^{\infty} \sqrt{2} c_k \phi(2t-k) \, \mathrm{d}t \\
&= \sum_{k=-\infty}^{\infty} \sqrt{2} c_k \underbrace{\int_{\mathbb{R}} \phi(2t-k) \, \mathrm{d}t}_{\frac{1}{2}}
\end{aligned}$$

Thus we have

$$\sum_{k=-\infty}^{\infty} c_k = \sqrt{2}.$$

Furthermore we have

$$\begin{aligned}
\langle \phi_{0,0}, \phi_{0,n} \rangle &= \int_{\mathbb{R}} \phi_{0,0}(t) \phi_{0,n}(t) \mathrm{d}t \\
&= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} c_k c_l \underbrace{\int_{\mathbb{R}} \phi_{-1,k}(t) \phi_{-1,2n+l}(t) \mathrm{d}t}_{\delta_{k,2n+l}} \\
&= \sum_{k=-\infty}^{\infty} c_k c_{k-2n}
\end{aligned}$$

Since $\langle \phi_{0,0}, \phi_{0,n} \rangle = \delta_{0,n}$ we have

$$\sum_{k=-\infty}^{\infty} c_k c_{k-2n} = \delta_{0,n}$$

9.3.2.8 Example: The Haar Wavelet

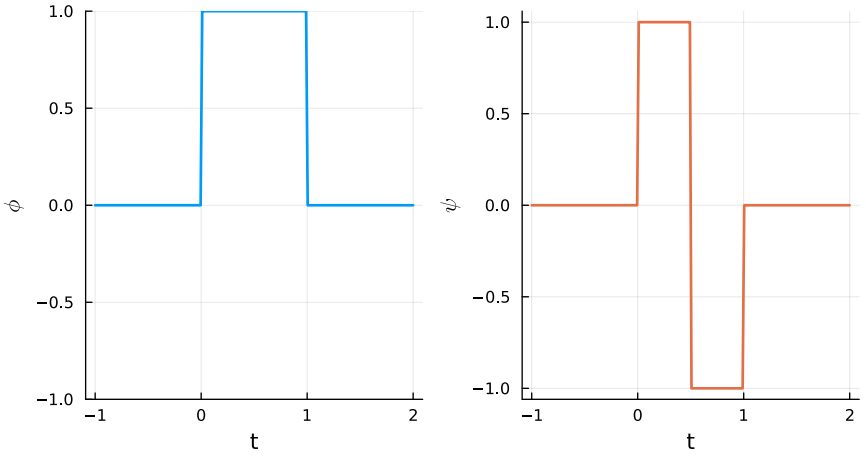
Lets have a look at the Haar wavelet, which is the most basic wavelet one can think of. It is defined as

$$\psi(t) = \begin{cases} 1 & \text{if} \quad 0 \leq t < \frac{1}{2} \\ -1 & \text{if} \quad \frac{1}{2} \leq t < 1 \\ 0 & \text{else} \end{cases}$$

and has the scaling function

$$\phi(t) = \begin{cases} 1 & \text{if} \quad 0 \leq t < 1 \\ 0 & \text{else} \end{cases}$$

Both functions are shown next:



The only non-zero coefficients are $c_0 = \frac{1}{\sqrt{2}}$ and $c_1 = \frac{1}{\sqrt{2}}$ ($b_0 = \frac{1}{\sqrt{2}}$, $b_1 = -\frac{1}{\sqrt{2}}$). I.e. the two-scale relations can be expressed as:

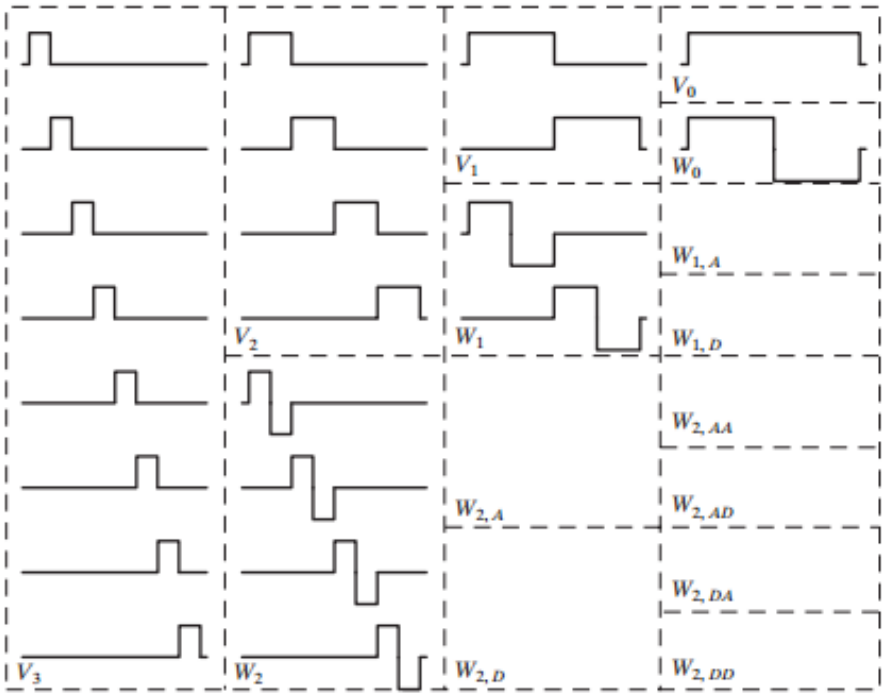
$$\phi(t) = \phi(2t) + \phi(2t-1) = \sqrt{2}(c_0\phi(2t) + c_1\phi(2t-1))$$

and

$$\psi(t) = \phi(2t) - \phi(2t-1) = \sqrt{2}(b_0\phi(2t) + b_1\phi(2t-1)).$$

One can see that ϕ can be constructed by adding two shifted and scaled versions of ϕ. To construct ψ we take one positive scaled ϕ and one negative shifted and scaled ϕ.

The following shows how the Haar basis functions and the corresponding scaling functions look like on three different scales:



9.3.2.9 Fast Wavelet Transform

The goal is now to get from the signal $S_{m,n}$ to the coefficients $T_{m+1,n}$ and $S_{m+1,n}$ and vice versa. If we can do that, it is possible to decompose $S_{m,n}$ step by step into finer and finer details.

Level Up

To go from the signal to the coefficients, we want to calculate the coefficients $S_{m+1,n}$ and $T_{m+1,n}$ from $S_{m,n}$. We have

$$\begin{aligned}
S_{m+1,n} &= \int_{-\infty}^{\infty} s(t)\phi_{m+1,n}(t)\,\mathrm{d}t \\
&= \int_{-\infty}^{\infty} s(t)\left(\sum_{k=-\infty}^{\infty} c_k\,\phi_{m,2n+k}(t)\right)\mathrm{d}t \\
&= \sum_{k=-\infty}^{\infty} c_k\left(\int_{-\infty}^{\infty} s(t)\,\phi_{m,2n+k}(t)\,\mathrm{d}t\right) \\
&= \sum_{k=-\infty}^{\infty} c_k S_{m,2n+k} \\
&= \sum_{k=-\infty}^{\infty} c_{k-2n} S_{m,k}.
\end{aligned}$$

In the last step we have applied an index shift. In the same way we can also derive

$$T_{m+1,n} = \sum_{k=-\infty}^{\infty} b_{k-2n} S_{m,k}.$$

Level Down

Next task is to get from $S_{m,n}$ and $T_{m,n}$ to $S_{m-1,n}$. We already know that $s_{m-1}(t) = s_m(t) + d_m(t)$, which can be enrolled to

$$\begin{aligned}
s_{m-1}(t) &= \sum_{n=-\infty}^{\infty} S_{m,n}\phi_{m,n}(t) + \sum_{n=-\infty}^{\infty} T_{m,n}\psi_{m,n}(t) \\
&= \sum_{n=-\infty}^{\infty} S_{m,n} \sum_{k=-\infty}^{\infty} c_k\, \phi_{m-1,2n+k}(t) + \sum_{n=-\infty}^{\infty} T_{m,n} \sum_{k=-\infty}^{\infty} b_k\, \phi_{m-1,2n+k}(t) \\
&= \sum_{n=-\infty}^{\infty} S_{m,n} \sum_{k=-\infty}^{\infty} c_{k-2n}\, \phi_{m-1,k}(t) + \sum_{n=-\infty}^{\infty} T_{m,n} \sum_{k=-\infty}^{\infty} b_{k-2n}\, \phi_{m-1,k}(t) \\
&= \sum_{k=-\infty}^{\infty} \phi_{m-1,k}(t) \left(\sum_{n=-\infty}^{\infty} c_{k-2n} S_{m,n} + \sum_{n=-\infty}^{\infty} b_{k-2n} T_{m,n} \right)
\end{aligned}$$

On the other hand we can also express $s_{m-1}(t)$ by just the basis functions $\phi_{m-1,k}(t)$:

$$s_{m-1}(t) = \sum_{k=-\infty}^{\infty} S_{m-1,k}\phi_{m-1,k}(t).$$

When equating both expressions we can see that $S_{m-1,k}$ needs to be the same as the term in brackets. After relabeling k as n we obtain:

$$S_{m-1,n} = \sum_{k=-\infty}^{\infty} c_{n-2k} S_{m,k} + \sum_{k=-\infty}^{\infty} b_{n-2k} T_{m,k}$$

This now allows us to calculate $S_{m-1,n}$ from $S_{m,n}$ and $T_{m,n}$.

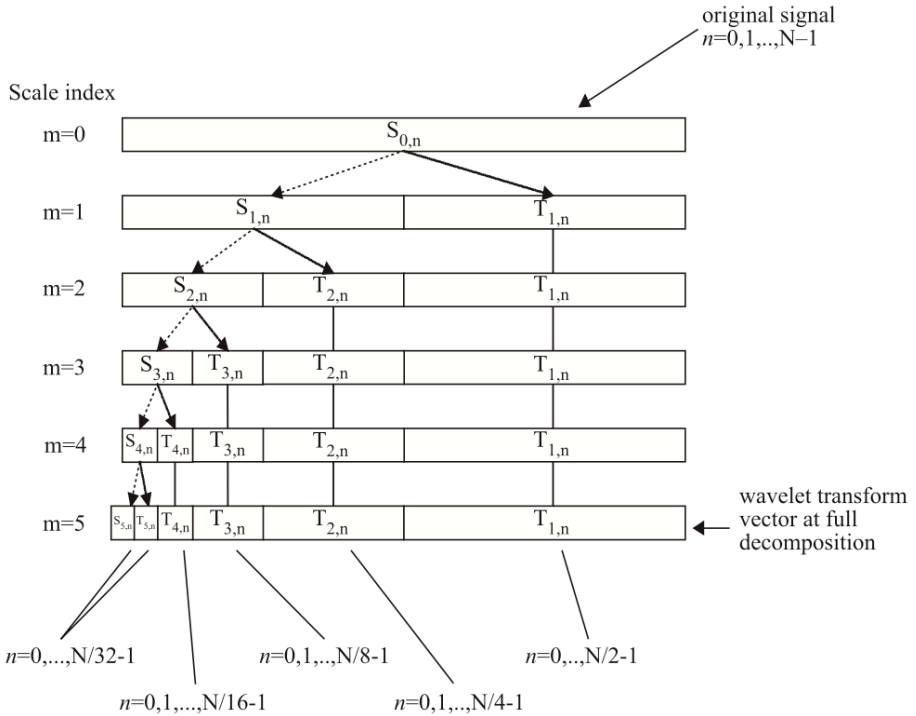
Note

An interesting fact is that we can calculate the coefficients without direct knowledge of the wavelet ψ and the scaling function ϕ. In practice, for the discrete wavelet transform, one thus just needs the coefficients c_k and b_k.

9.3.3 Discrete Wavelet Transform

In the last section we were discussing the wavelet series but we had already the discrete setting in mind. What is important to note is that we stepped over the coefficients c_k and b_k with a spacing of two indices, which means that the number of non-zero coefficients halves if we go one step up and consider that we started with a finite length input signal.

This brings us then to the discrete wavelet transform (DWT). Since we are bisecting the signal in each step and generate from $S_{m,n}$ the signals $S_{m+1,n}$ and $T_{m+1,n}$ we can update the original signal *inplace* which also means that the DWT has zero redundancy. This is graphically shown in the next picture:



Note

We left out the formal definition of the DWT since this lecture contains enough mathematical details in the previous sections to understand the basic concept.

Note

One does not necessary perform the full DFT but it is also possible to stop at one level. This is even necessary when considering signals that are not powers of two. See what happens if we apply the DWT to an odd signal.

```
▸ [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0]
```

```
dwt([1.0,2,3,4,5,6,7,8,9], wavelet(WT.db1))
```

```
0
```

```
maxtransformlevels([1.0,2,3,4,5,6,7,8,9])
```

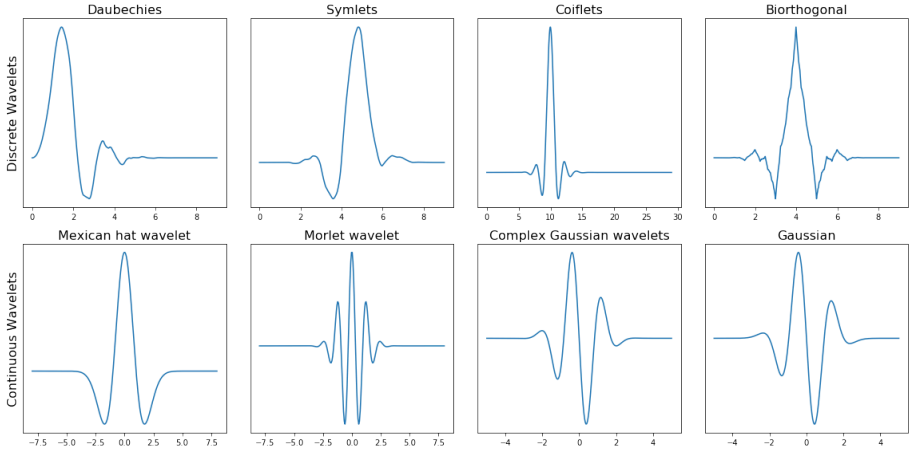
9.3.3.1 Time Complexity

The time complexity for performing the DWT is $\mathcal{O}(N)$, which is very fast when taking into account that the DWT matrix has $\mathcal{O}(N^2)$ entries. This is even faster than the FFT which had a time complexity of $\mathcal{O}(N \log N)$.

The reason that the DWT is even faster is that the FFT, which also breaks the signal in halves, operates on both signal halves. The DWT, on the other hand, just operates on one of the halves and keeps the second halve untouched when going one level up.

9.3.4 Wavelet Types

Until now we kept the wavelet function and its scaling function generic, which means that various wavelets can be used in practice. Here is a small overview:



You can also visit the wavelet browser.

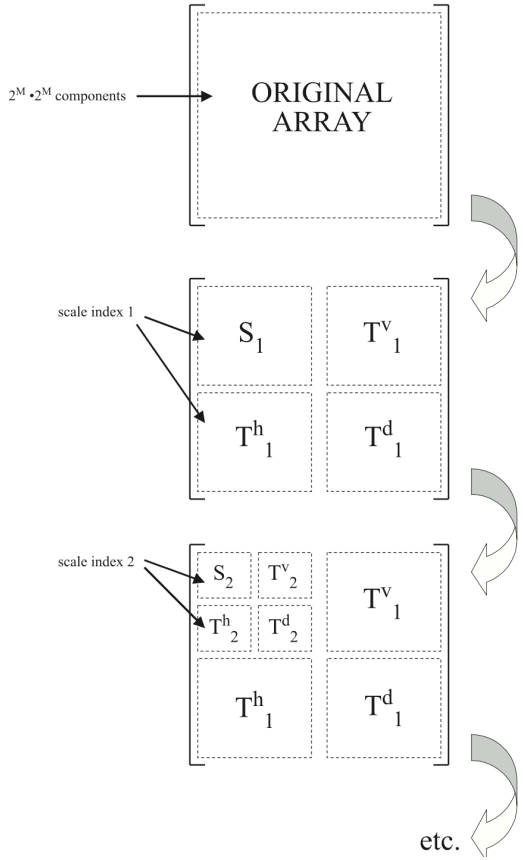
When choosing a different wavelet the resulting coefficients $T_{m,n}$ and $S_{m,n}$ will also be different. This allows to tailor the wavelet to a specific signal class. In practice the differences are, however, not too large and usually it is a good choice to select one of the Daubechies wavelets.

9.3.5 Multi-Dimensional Wavelet Transform

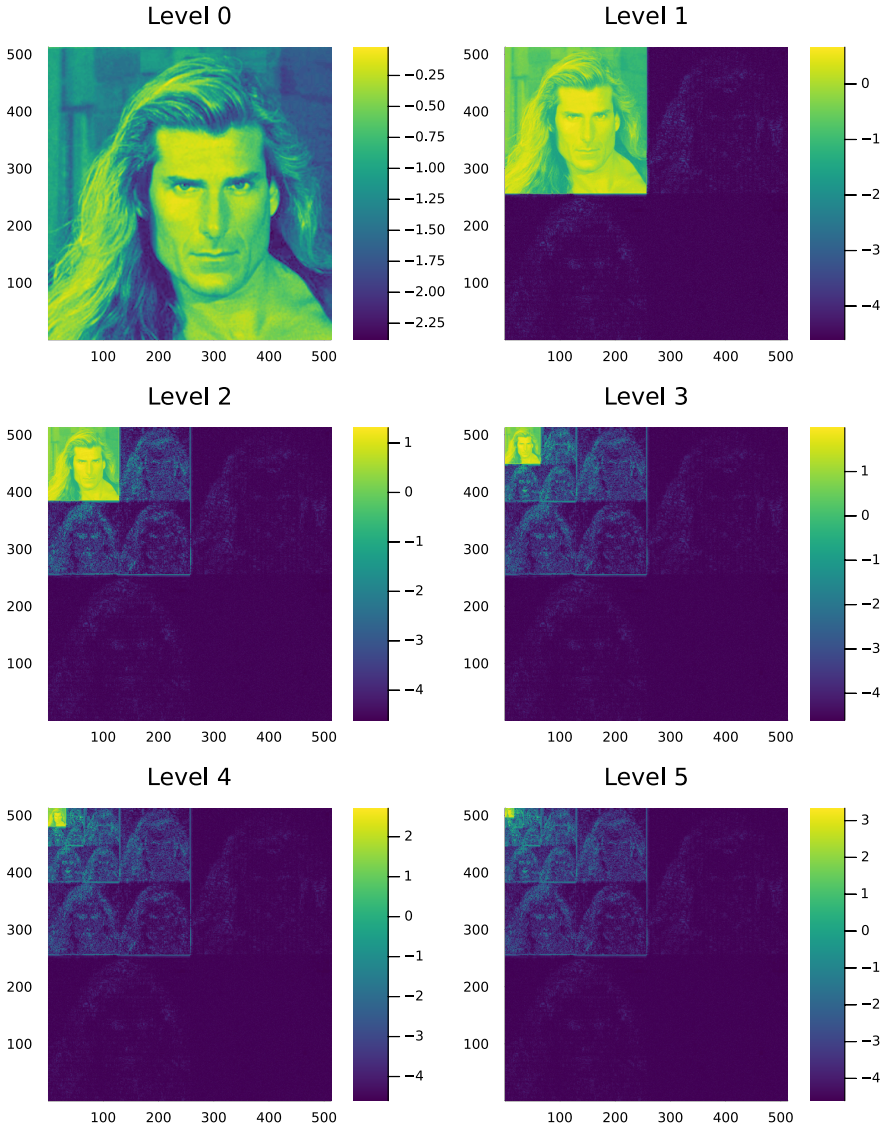
We of course want to apply the DWT to images and therefore need to extend everything to multiple dimensions. This is done using a tensor product approach. In 2D we then have the following wavelet and wavelet scaling functions:

- 2D scaling function: $\phi(t_1, t_2) = \phi(t_1)\phi(t_2)$
- 2D horizontal wavelet: $\psi^{\mathrm{h}}(t_1, t_2) = \phi(t_1)\psi(t_2)$
- 2D vertical wavelet: $\psi^{\mathrm{v}}(t_1, t_2) = \psi(t_1)\phi(t_2)$
- 2D diagonal wavelet: $\psi^{\mathrm{d}}(t_1, t_2) = \psi(t_1)\psi(t_2)$

We thus need to break a 2D image into four parts and then successively apply one level of the transformation. This is illustrated in the following picture:



Lets now go through the levels for Fabio:

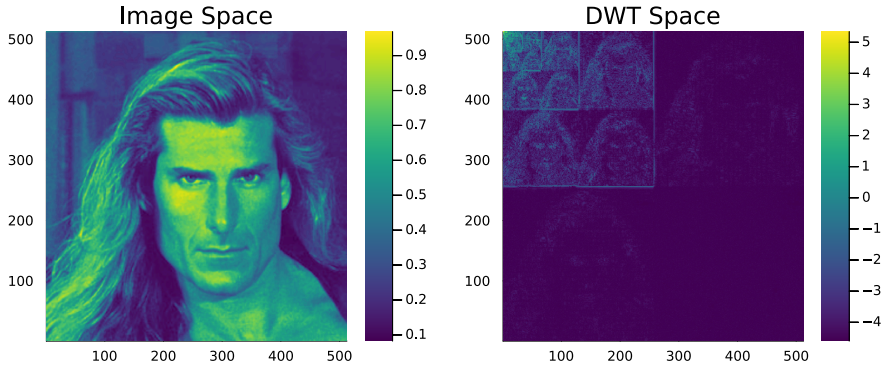


9.4 Applications

The wavelet transform has several applications in signal and image processing and is overlapping with what can be done with usual Fourier space filtering. Here we sketch some of the applications.

9.4.1 Image Compression

The first application that we consider is image compression. Let us have a look at Lena and its discrete wavelet transform:



Observation

- The DWT coefficients are mostly zero. Only at edges higher coefficients can be observed.
- The DWT is, for natural images, thus a sparsifying transform.

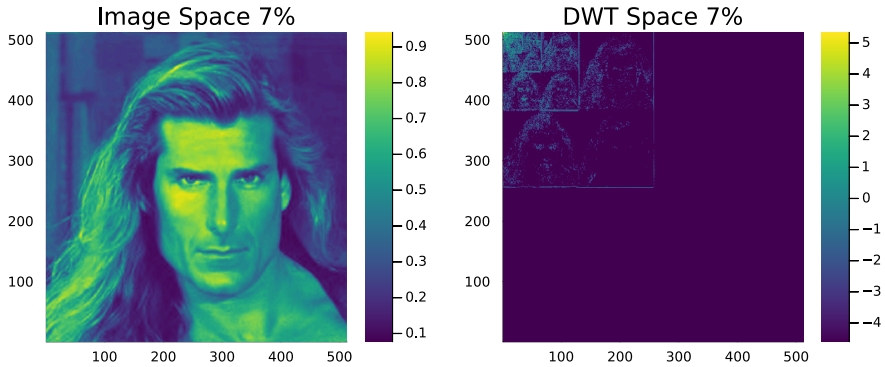
Based on this observation one can derive an algorithm where only a fraction of the information is stored. To this end one

- calculates the DWT.
- applies a threshold setting a large fraction of coefficients to zero.
- store the resulting sparse matrix using an efficient data format.

In order to load compressed data one can then

- load the sparse matrix.
- apply an inverse DWT.

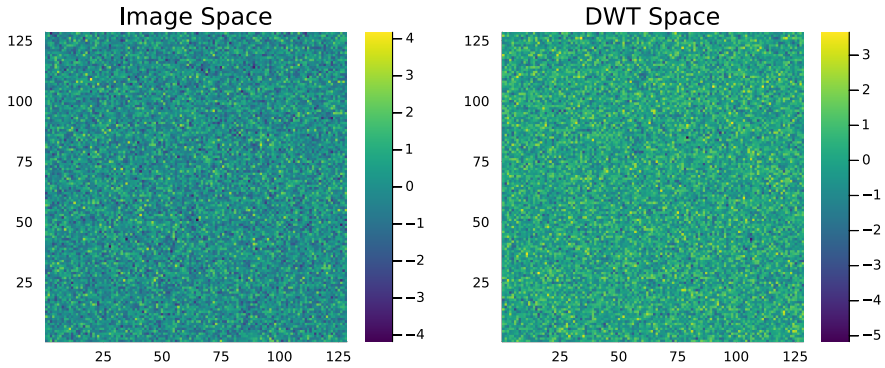
The following shows a data reduction by a factor of 10, which is a very good value for 2D image compression. One can hardly see any difference to the original image.



Wavelet based image compression has been implemented in the JPEG_2000 picture standard that achieves higher compression rates than JPEG.

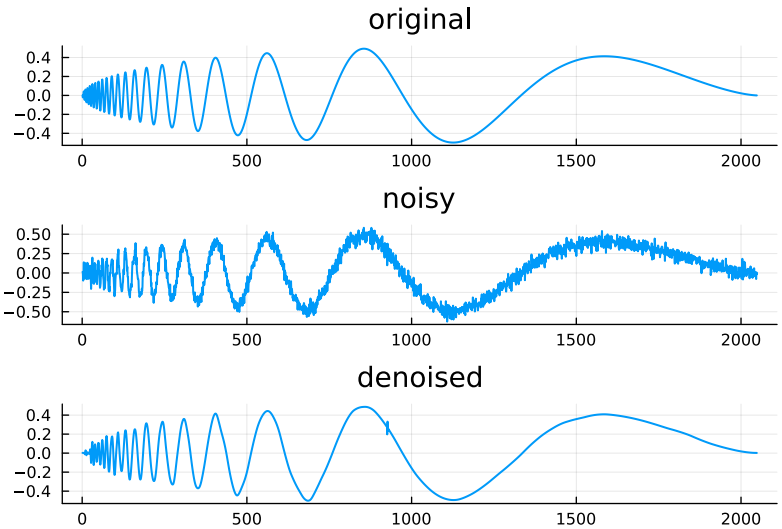
9.4.2 Noise Reduction

Another important application of the wavelet transform is noise reduction. Similar to the Fourier transform, the noise distributes equally into all coefficients:

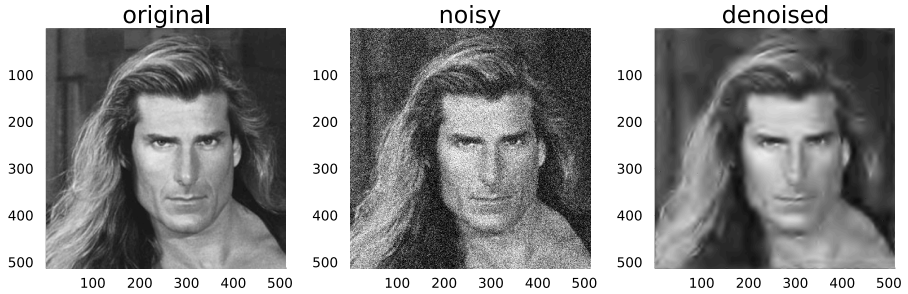


The signal usually goes down going from one to the next level. Thus, we can threshold the noise more effectively in wavelet space than in image space.

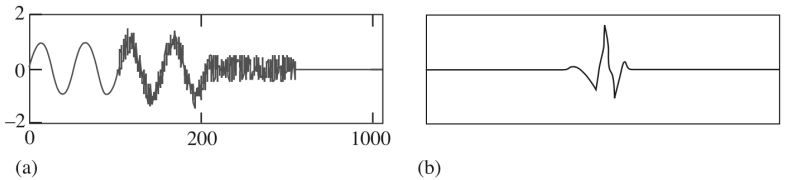
The following shows a denoise example:

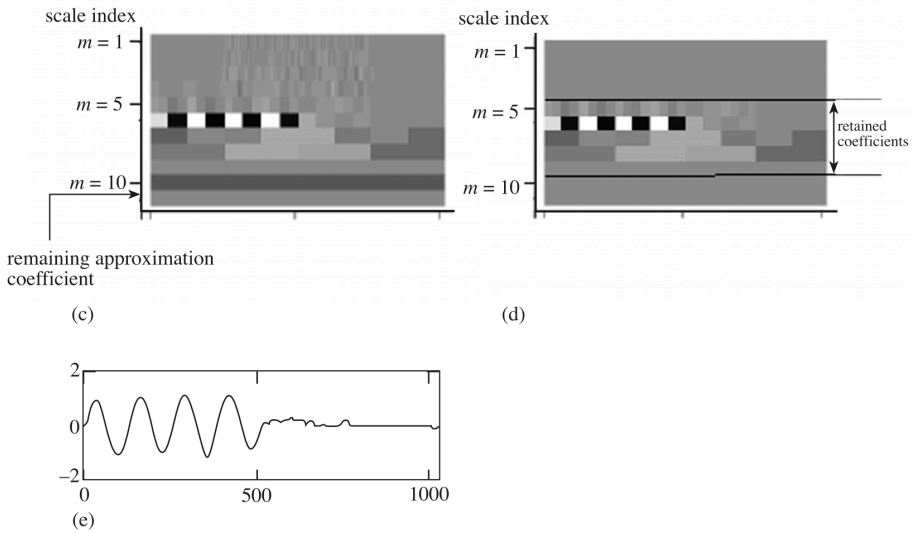


Here is another example, where the noise is applied to an image.



Wavelets also allow for removing local noise, what would not be possible with Fourier-based methods:





9.4.2 Further Applications

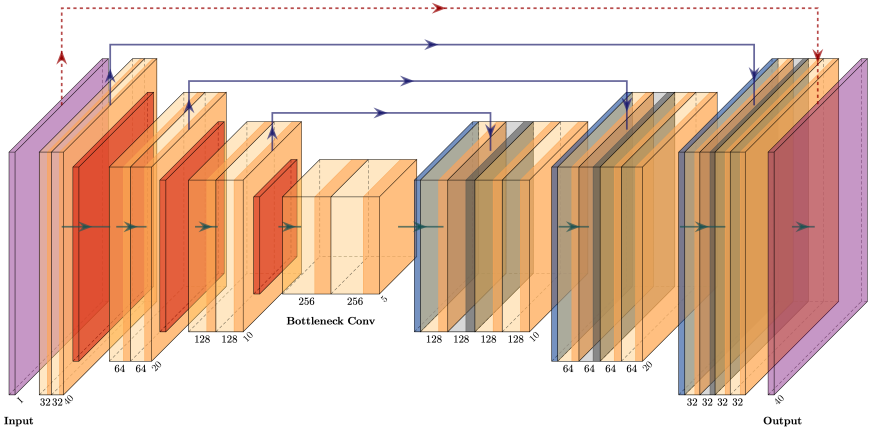
The wavelet transform has various additional applications. For example:

- It can be used for *feature extraction*. For instance fingerprint recognition techniques can be based on the wavelet transform.
- In time data analysis (speach recognition, ECG processing), the wavelet transform can be used to match patterns in the wavelet domain.

- In addition the DWT is often used as a *sparsifying transform*. This means that a dense signal is sparsified and can be represented by fewer coefficients. This principle is often used in the field of compressed sensing, which has the goal of reconstructing high resolution images from only few measurements (i.e. less samples than the number of image pixels).

9.5 Deep Neural Networks

The field of machine learning, which is nowadays commonly using deep neural networks is heavily influenced by multi-resolution analysis. Let us have a look at a UNet:



What you can see here is that the input image is forwarded layer-by-layer and each layer performs a convolution and/or up- and downsampling operations. In this way a deep neural network can learn very complex transformations and in-fact perform a (custom) multi-resolution analysis.

But note that deep neural networks are much more than just a multi-resolution analysis. The kernels (which can be compared to wavelets) are not static within a neural network but they are fitted to the data within the learning process. This means a neural network is much more flexible (pro) but needs data (con) for training. But despite these differences it is still remarkable that both research areas share a common concept.

9.6 Wrapup

In this lecture we introduced multi-resolution image processing as a powerful tool for various common image processing tasks.

- We found that human pattern recognition works in a multi-resolution fashion.
- We used the wavelet transform as the formal tool to specify multi-resolution image processing. Image pyramids were used as a simple motivating example.
- Wavelet transforms can be applied to continuous and discrete signals. For 2D/3D signals a tensor approach is applied.
- Wavelets have various important applications and can be for example used for
  - image compression
  - noise reduction
  - sparse sampling