Advanced Signal, Image and Surface Processing

Gabriel Peyré
Ceremade, Université Paris-Dauphine
gabriel.peyre@ceremade.dauphine.fr
http://www.ceremade.dauphine.fr/~peyre/numerical-tour/

January 14, 2010

This book details advanced signal, image and surface processing methods. The focus in on the use of sparsity and variational methods to solve difficult problems such as denoising, compression, super-resolution and compressive sensing. The theoritical exposition is kept as simple as possible and the emphasis is put on numerical schemes and practical experimentation using Matlab.

The main reference to get a more detailed exposition of signal and image processing methods is the book of Mallat [20]. It covers all the theoritical details not covered in this book, and should be used as a reference textbook. Another reference for wavelet analysis is the book of Daubechies [10]. Other references for signal and image processing include [33, 30].

The homepage page for the course is

`http://www.ceremade.dauphine.fr/~peyre/numerical-tour/`

You can retrieve from this page additional information such as slides. A set of Matlab numerical tours allows one to experiment and reproduce the results of these notes.

Contents

Chapter 1

Signal and Image Decomposition in Orthogonal Bases

This chapter introduces the concept of orthogonal decomposition. This is the most important tool used in this book to perform signal and image processing. These decompositions can be defined for both continuous or discrete signals.

1.1 Analog Signals

To develop numerical tools and analyze their performances, the mathematical modeling is usually done over a continuous setting. An analog signal is a 1D function $f_0 \in \mathrm{L}^2([0,1])$ where $[0,1]$ denotes the domain of acquisition, which might for instance be time. An analog image is a 2D function $f_0 \in \mathrm{L}^2([0,1]^2)$ where the unit square $[0,1]^2$ is the image domain.

Although these notes are focussed on the processing of sounds and natural images, most of the methods extend to multi-dimensional datasets, which are higher dimensional mappings

$$f_0 : [0,1]^d \to [0,1]^s$$

where d is the dimensionality of the input space ($d = 1$ for sound and $d = 2$ for images) whereas s is the dimensionality of the feature space. For instance, gray scale images corresponds to $(d = 2, s = 1)$, videos to $(d = 3, s = 1)$, color images to $(d = 2, s = 3)$ where one has three channels (R, G, B). One can even consider multi-spectral images where $(d = 2, s \gg 3)$ that is made of a large number of channels for different light wavelengths. Figures 1.1 and 1.2 show examples of such data.

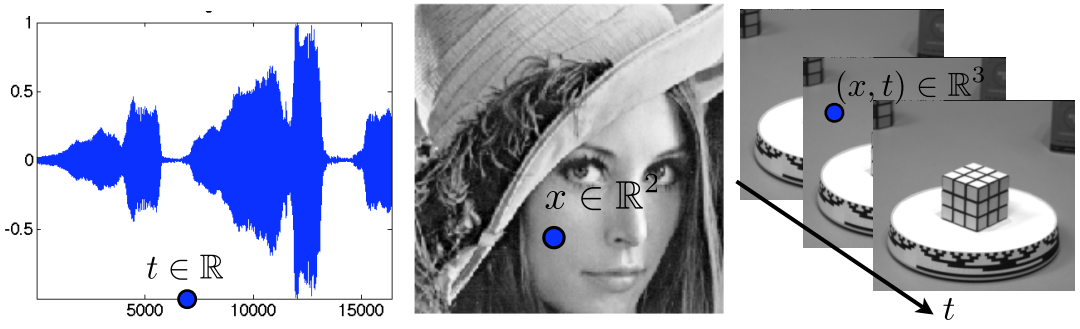


Figure 1.1: *Examples of sounds ($d = 1$), image ($d = 2$) and videos ($d = 3$).*

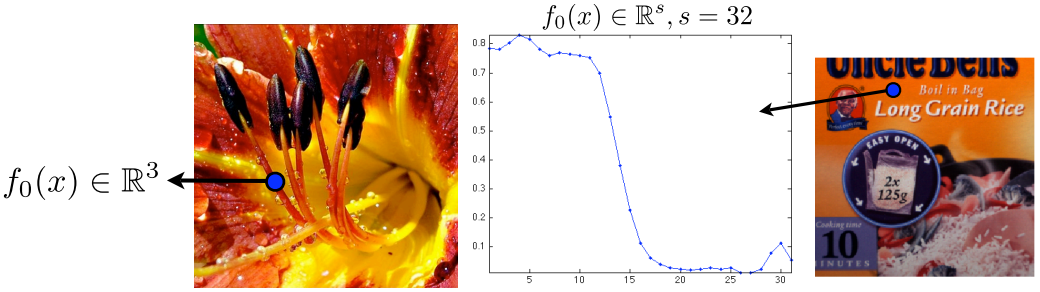


Figure 1.2: *Example of color image $s = 3$ and multispectral image ($s = 32$).*

1.2 Discrete Signals

1.2.1 Acquisition and Sampling

Signal acquisition is a low dimensional projection of the continuous signal performed by some hardward device. This is for instance the case for a microphone that acquires 1D samples or a digital camera that acquires 2D pixel samples. The sampling operation thus corresponds to mapping from the set of continuous functions to a discrete finite dimensional vector with N entries.

$$f_0 \in \mathrm{L}^2([0,1]^d) \mapsto f \in \mathbb{C}^N$$

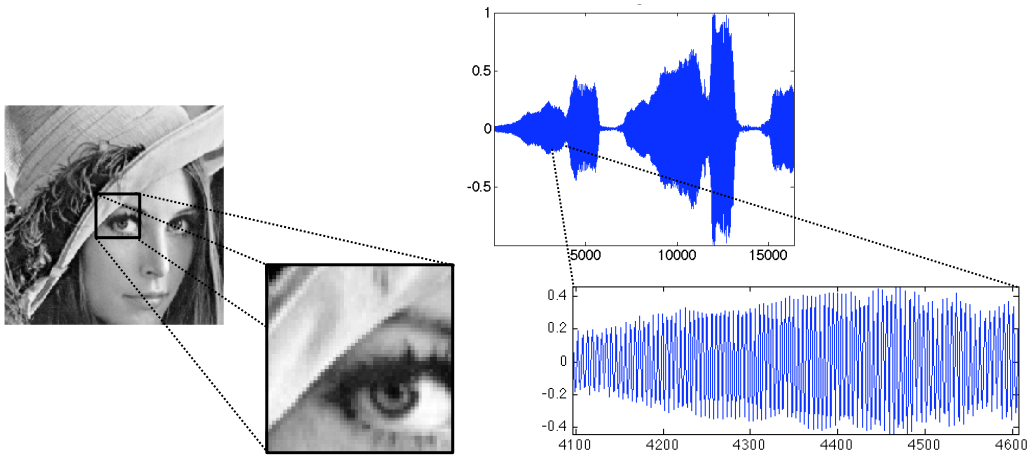


Figure 1.3: *Image and sound discretization.*

Figure 1.3 shows examples of discretized signals.

1.2.2 Linear Translation Invariant Sampler

A translation invariant sampler performs the acquisition as an inner product between the continuous signal and a constant impulse response h translated at the sample location

$$f[n] = \int_{-S/2}^{S/2} f_0(x) h(n/N - x) \mathrm{d}x = f_0 \star h(n/N). \tag{1.1}$$

The precise shape of $h(x)$ depends on the sampling device, and is usually a smooth low pass function that is maximal around $x = 0$. The size S of the sampler determines the precision of the sampling device, and is usually of the order of $1/N$ to avoid blurring (if S is too large) or aliasing (if S is too small).

Section 2.6 details how to reverse the sampling operation in the case where the function is smooth.

1.3 Orthogonal Decompositions

The main ingredient for the processing detailed in this book is the decomposition in a well chosen orthogonal basis. If this decomposition is sparse, meaning that only few coefficients capture most of the signal energy, then many processing problems such as compression, denoising or super-resolution, are efficiently solved.

1.3.1 Continuous Ortho-bases

L^2 measure of similarity. For analog signals $f_0 \in \mathrm{L}^2([0,1]^d)$, we consider the L^2 inner product that defines an Hilbert space structure

$$\langle f_0,\, g_0 \rangle = \int_{[0,1]^d} f_0(x) \bar{g}_0(x) \mathrm{d}x.$$

This inner product defines a distance between signal

$$\|f_0 - g_0\|^2 = \int_{[0,1]^d} |f_0 - g_0|^2 \mathrm{d}x = \langle f_0 - g_0,\, f_0 - g_0 \rangle. \tag{1.2}$$

This L^2 norm is used to assess theoretically and numerically the performance of all processing methods presented in this book. The relevance of this L^2 measure of similarity is questionable, since it does not take into account the visual quality that should be considered as the ultimate judge of performance. Nevertheless, there is currently no better way to access automatically the results of signal and image processing algorithms.

Ortho-bases. An orthonormal basis of $\mathrm{L}^2([0,1]^d)$ is a collection of atoms $\{\psi_m\}_{m \in \mathbb{Z}}$ that spans $\mathrm{L}^2([0,1]^d)$ and satisfies

$$\langle \psi_m,\, \psi_{m'} \rangle = \delta[m - m']. \tag{1.3}$$

where δ is the discrete Dirac, that is defined as $\delta[i] = 0$ if $i \neq 0$ and $\delta[0] = 1$.

One should be careful that $\mathrm{L}^2([0,1]^d)$ is an infinite dimensional Hilbert space, so that $\{\psi_m\}_{m \in \mathbb{Z}}$ is a basis if and only if its linear span is dense in $\mathrm{L}^2([0,1]^d)$. It means that any analog signal $f_0 \in \mathrm{L}^2([0,1]^d)$ can be approximated using the L^2 norm (1.2) with a sequence of approximation signals that are spanned by a finite number of vector from $\{\psi_m\}_{m \in \mathbb{Z}}$.

One can thus formally write the decomposition of a continuous signal in such an ortho-basis as

$$f_0 = \sum_{m \in \mathbb{Z}} \langle f_0,\, \psi_m \rangle \psi_m. \tag{1.4}$$

This decomposition satisfies an energy conservation

$$\|f_0\|^2 = \int_{[0,1]^d} |f_0(x)|^2 \mathrm{d}x = \sum_{m \in \mathbb{Z}} |\langle f_0,\, \psi_m \rangle|^2. \tag{1.5}$$

Relevance of ortho-bases. The computation of the set of inner products $\{\langle f_0,\, \psi_m \rangle\}_m$ offers an alternate representation that might be simpler than the spacial representation $f_0(t)$ is the basis is well chosen to match the structure of the signal.

Processing is then usually performed in three stage

– Computation of the inner products $a[m] = \langle f_0,\, \psi_m \rangle$ for all m.
– Modification of these inner product to obtain new coefficients $\tilde{a}[m]$ (for instance setting to zero small coefficients, quantizing coefficients, ...).
– Reconstruction of a signal from these modified coefficients, that is the output of the algorithm

$$\tilde{f}_0 = \sum_{m \in \mathbb{Z}} \tilde{a}[m] \psi_m.$$

The orthogonality property of the basis is crucial since it allows a simple reconstruction formula (1.4) from a given set of coefficients. It is also important because the energy conservation (1.5) allows to compute the modification error directly over the coefficients

$$\|f_0 - \tilde{f}_0\|^2 = \sum_m |a[m] - \tilde{a}[m]|^2.$$

Fourier ortho-basis. The simplest, and probably most important, example of ortho-basis is the Fourier basis. For 1D signals ($d = 1$), it is defined as

$$\psi_m(t) = e_m(t) = e^{2i\pi mt} = \cos(2\pi mt) + i\sin(2\pi mt). \tag{1.6}$$

The parameter m that indexes the atom is the frequency of its oscillations. One easily checks orthogonality (1.3), and one can prove that any signal in $\mathrm{L}^2([0,1])$ can be written in this basis as (1.4).

Figure 1.4, left, shows examples of the real part of Fourier atoms.

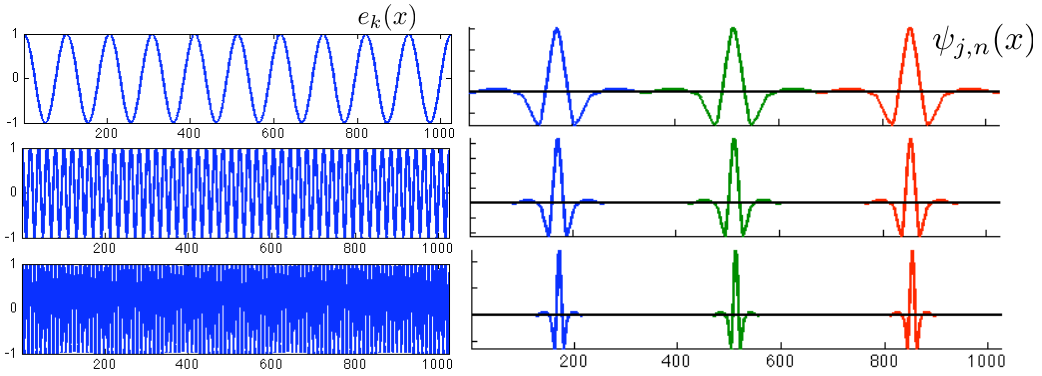


Figure 1.4: *Left: 1D Fourier (real part), right: wavelet bases.*

One should be careful about the fact that Fourier atoms are 1-periodic, so that $e_m(t+1) = e_m(t)$. This implies that processing signal $f(t)$ for $t \in [0,1]$ implicitly corresponds to performing a 1-periodic extension of $f(t)$ for all $t \in \mathbb{R}$. For instance, even if f is continuous on $[0,1]$, it should be treated as discontinuous if $f(0) \neq f(1)$.

Chapter (2) details the use of this Fourier basis to perform linear signal and image processing.

Wavelet ortho-bases. Fourier atoms are global since they are supported on $[0,1]$. This might be an issue to process a signal that contains sharp localized features such as sound transition or edges in images, see

Wavelet bases are composed of atoms that have a wide range of support size and location. They are efficient to process pointwise singularities since they allow to zoom on the singularity and reconstruct it with only a few atoms. More precisely, a 1D wavelet basis is obtained by dyadic translation and dilation of a mother wavelet function ψ

$$\psi_m(t) = 2^{-j/2}\psi(2^{-j}t - n)$$

where $m = (j,n)$, 2^j being the scale and n indexing the position of the atom.

Chapter 3 details how to actually compute a proper wavelet function ψ so that the resulting set of atoms $\{\psi_{j,n}\}_{j,n}$ is an orthogonal basis of $\mathrm{L}^2([0,1])$.

1.3.2 Discrete Ortho-bases

Discrete signals are finite dimensional vector $f \in \mathbb{C}^N$ where N is the number of samples and where each $f[n]$ is the value of the signal at a 1D or 2D location. For a 2D images $f \in \mathbb{C}^N \simeq \mathbb{C}^{N_0 \times N_0}$, $N = N_0 \times N_0$, where N_0 is the number of pixels along each direction.

Discrete signals and images are processed using a discrete inner product that mimics the continuous L^2 inner product

$$\langle f,\, g\rangle = \sum_{n=0}^{N-1} f[n]\bar{g}[n].$$

One thus defines a distance between discretized vectors as

$$\|f-g\|^2 = \sum_{n=0}^{N-1} |f[n]-g[n]|^2.$$

Exactly as in the continuous case, a discrete orthogonal basis $\{\psi_m\}_{0\leqslant m<N}$ of $\mathbb{C}^N$, satisfies

$$\langle \psi_m,\, \psi_{m'}\rangle = \delta[m-m']. \tag{1.7}$$

The decomposition of a signal in such an ortho-basis is written

$$f = \sum_{m=0}^{N-1} \langle f,\, \psi_m\rangle \psi_m.$$

It satisfies a conservation of energy

$$\|f\|^2 = \sum_{n=0}^{N-1} |f[n]|^2 = \sum_{m=0}^{N-1} |\langle f,\, \psi_m\rangle|^2$$

Computing the set of all inner product $\{\langle f,\, \psi_m\rangle\}_{0\leqslant m<N}$ is done in a brute force way in $O(N^2)$ operations. This is not feasible for large datasets where N is of the order of millions. When designing an ortho-basis, one should keep this limitation in mind and enforce some structure in the basis elements so that the decomposition can be computed with fast algorithm. This is the case for the Fourier and wavelet bases, that enjoy respectively $O(N\log(N))$ and $O(N)$ algorithms.

Discrete Fourier basis A discrete orthogonal Fourier basis is obtained by sampling the continuous Fourier atoms defined in (1.6)

$$e_m[n] = \frac{1}{\sqrt{N}} e^{\frac{2i\pi}{N}mn} = \frac{1}{\sqrt{N}} e_m(n/N).$$

One verifies that this formula defines an ortho-basis that satisfies (1.7).

The computation of the Fourier coefficients $\{\langle f,\, e_n\rangle\}_m$ is performed in $O(N\log(N))$ operations with the Fast Fourier Transform (FFT) algorithm, that exploits recursion formula on the coefficients in a divide-and-conqueer manner.

Discrete wavelet basis. Discrete wavelets do not have in general an explicit formula, and the discrete atoms $\psi_{j,n}$ are defined implicitely through a filtering cascade. The important point is that the projection of a signal onto these atoms is computed in $O(N)$ operations with the Fast Wavelet Transform (FWT) algorithm.

1.3.3 2D Extensions of 1D Bases

An image $f \in \mathbb{C}^N$ of $N = N_0 \times N_0$ pixels is sampled on a regular grid. Given a discrete 1D ortho-basis $\{\psi_m\}_{m=0}^{N_0-1}$ of $\mathbb{C}^{N_0}$, one can build a 2D ortho-basis of $\mathbb{C}^N$ using tensor products of 1D function,

$$\psi_{m_1,m_2}[n_1,n_2] = \psi_{m_1}[n_1]\psi_{m_2}[n_2]$$

which defines a separable basis $\{\psi_{m_1,m_2}\}_{m_1,m_2=0}^{N_0-1}$.

The 2D discrete Fourier basis is written

$$e_{m_1,m_2}[n_1,n_2] = \frac{1}{\sqrt{N}} e^{\frac{2i\pi}{N_0}(m_1n_1+m_2n_2)}.$$

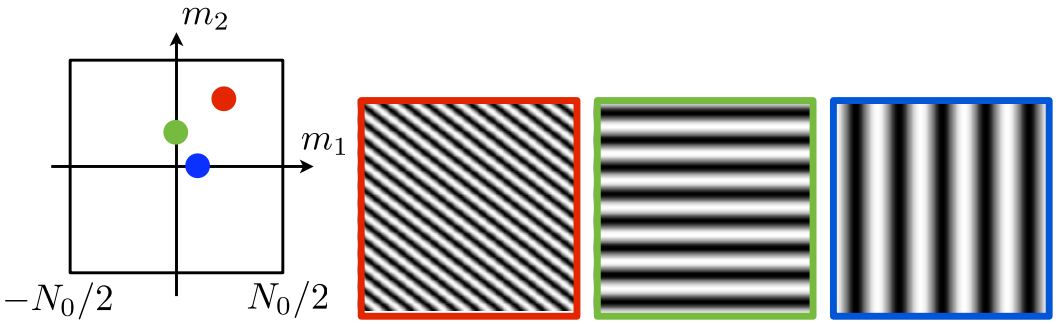


Figure 1.5: *2D Fourier orthogonal bases.*

It corresponds to a discretized wave oscillating in the direction orthogonal to $m = (m_1, m_2)$.

Figure 1.5 shows examples of the real part of 2D Fourier atoms.

Anisotropic wavelets are obtained by tensor products of 1D wavelets

$$\psi_{j_1,j_2,n_1,n_2}[x_1, x_2] = \psi_{j_1,n_1}[x_1]\psi_{j_2,n_2}[x_2].$$

They correspond to atoms stretched along the horizontal and vertical directions.

Isotropic wavelets, are defined in the continuous case as

$$\psi^{\omega}_{j,n}(x) = 2^{-j}\psi^{\omega}(2^{-j}x - n)$$

where $\omega \in \{H, V, D\}$ indicates the horizontal/vertical/diagonal direction of the wavelets. They are often used in image processing, since they have a square support not stretched along the axes. They correspond to a non-separable basis, and require three mother wavelets $\{\psi^H, \psi^V, \psi^D\}$. Figure 1.6 shows examples of such atoms.

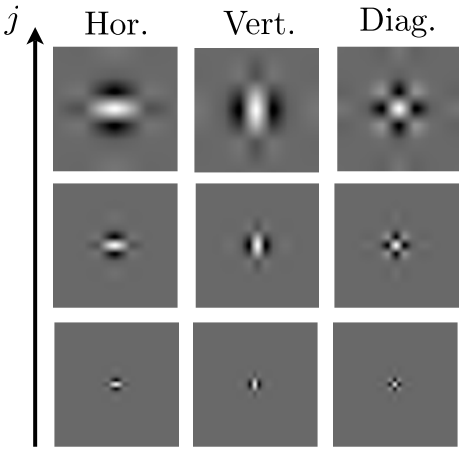


Figure 1.6: *2D isotropic wavelet orthogonal bases.*

Chapter 2

Fourier Processing

Fourier transforms of various kinds are basic tools in almost every signal or image processing method on translation invariant domains. Depending wether the domain is finite or infinite, or wether the signal is continuous or discrete, one obtain four different notions of Fourier transforms

– Infinite continuous domains: typically $\mathbb{R}$.

– Periodic continuous domains: typically $[0,1]$.

– Infinite discrete domains: typically $\mathbb{Z}$.

– Periodic discrete domains: typically $\{0,\dots,N-1\}$.

Bounded domains should be understood as periodic domain, for instance $[0,1]$ is treated as the torus $\mathbb{R}/\mathbb{Z}$ and $\{0,\dots,N-1\}$ is treated as the set $\mathbb{Z}/N\mathbb{Z}$ of integer modulo N. Figure 2.1 shows these four different kinds of domains, and the Fourier transform that inverts the roles of infinite/periodic and continuous/discrete.

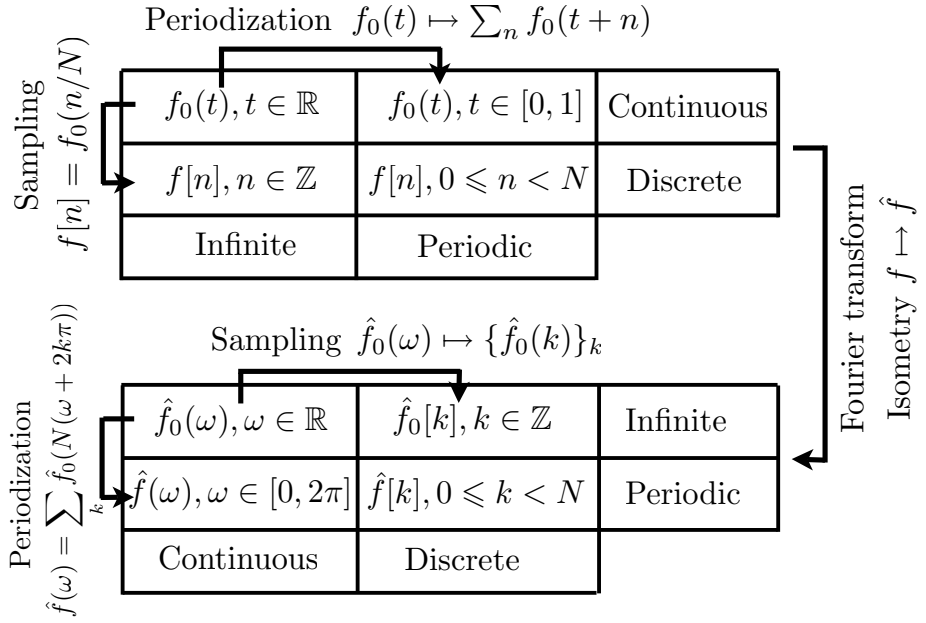


Figure 2.1: *The four different settings for Fourier analysis, and the sampling-periodization relationship.*

Each Fourier transform gives a frequency view point on the signal, that is for instance adapted to compute filtering with respect to translation over the domain. We detail in the following sections these four settings.

These domains are extended to higher dimensions by considering tensor products of 1D domains. For instance, for images, $[0,1]^2$ should be understood as the tensor product of $[0,1]$ with itself, so

that Fourier transforms as easily computed for higher dimensional signals.

2.1 Fourier Transform on the Real Line

An analog signal $f \in \mathrm{L}^2(\mathbb{R})$ is defined on the whole real line $\mathbb{R}$.

2.1.1 Fourier Transform

The Fourier transform of an integrable function $f \in \mathrm{L}^1(\mathbb{R})$ is defined as

$$\hat{f}(\omega) = \int f(t) e^{-i\omega t} \mathrm{d}t. \tag{2.1}$$

This definition extends to finite energy functions $f \in \mathrm{L}^2(\mathbb{R})$ by density.

In some sense, $f(\omega)$ is the inner product between f and the Fourier atom $e_\omega(t) = e^{-i\omega t}$, although one should be careful that e_m is neither in $\mathrm{L}^2(\mathbb{R})$ not in $\mathrm{L}^1(\mathbb{R})$, and that one considers a non-denombrable family of atoms indexed by $\omega \in \mathbb{R}$.

The Fourier mapping $f \mapsto \hat{f}$ is unitary, since one has the Plancherel formula

$$\langle f, g \rangle = \frac{1}{2\pi} \langle \hat{f}, \hat{g} \rangle \quad \text{and} \quad \int |f(t)|^2 \mathrm{d}t = \frac{1}{2\pi} \int |\hat{f}(\omega)|^2 \mathrm{d}\omega,$$

where the inner products are computed on $\mathbb{R}$.

Furthermore, it is easily inverted, if $\hat{f} \in \mathrm{L}^1(\mathbb{R})$, using the Fourier inversion formula

$$f(t) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{i\omega t} \mathrm{d}\omega.$$

This formula shares some analogy with the reconstruction from an orthogonal basis (1.4), with the replacement of $\sum$ by $\int$ and considering a non-denombrable family of atoms.

2.1.2 Fourier Transform and Continuous Filtering

The convolution of two functions $f, g \in \mathrm{L}^1(\mathbb{R})$ is defined as

$$f \star h(t) = \int_{-\infty}^{+\infty} h(u) f(t-u) \mathrm{d}u. \tag{2.2}$$

The Fourier convolution theorem shows that this convolution is simple to compute over the Fourier domain

$$g = f \star g \quad \Longrightarrow \quad \hat{g}(\omega) = \hat{f}(\omega) \hat{h}(\omega)$$

2.2 Fourier Coefficients on the Interval

We now consider analog signals defined on a bounded domain $f \in \mathrm{L}^2([0,1])$, which are equivalently treated as 1-periodic signals such that $f(t+1) = f(t)$.

2.2.1 Fourier Coefficients

A 1D Fourier atom of frequency m is defined as

$$\psi_m(t) = e_m(t) = e^{2i\pi mt} \tag{2.3}$$

where $t \in [0,1]$ with periodic boundary conditions, as already explained in (1.6). These atoms form an orthogonal basis $\{e_m\}_{m \in \mathbb{Z}}$ of $\mathrm{L}^2([0,1])$.

The Fourier coefficients of a function $f \in \mathrm{L}^2([0,1])$ are the inner products with the Fourier atoms

$$\hat{f}[m] = \langle f, e_m \rangle = \int_0^1 f(t) e^{-2i\pi mt} \mathrm{d}t. \tag{2.4}$$

We note that $\hat{f}[m]$ is a complex number, and that if f is real valued, then one has the following hermitian symmetry

$$\hat{f}[-m] = \hat{f}[m]^*.$$

2.2.2 Fourier Coefficients and Continuous Filtering

Filtering. The convolution of two functions $f, g \in \mathrm{L}^1([0,1])$ with periodic boundary conditions is defined as

$$f \star h(t) = \int_0^1 h(u) f(t-u) \mathrm{d}u$$

where $t-u$ is computed modulo 1.

Smoothing is achieved by using a low pass filter h. A simple example is a convolution with a box function

$$f \star \frac{1}{2a} 1_{[-\tau,\tau]}(t) = \frac{1}{2\tau} \int_{-\tau}^{\tau} f(t+u) \mathrm{d}u,$$

which computes a local averaging over an interval of size 2τ. Figure 2.2 shows this filtering process.

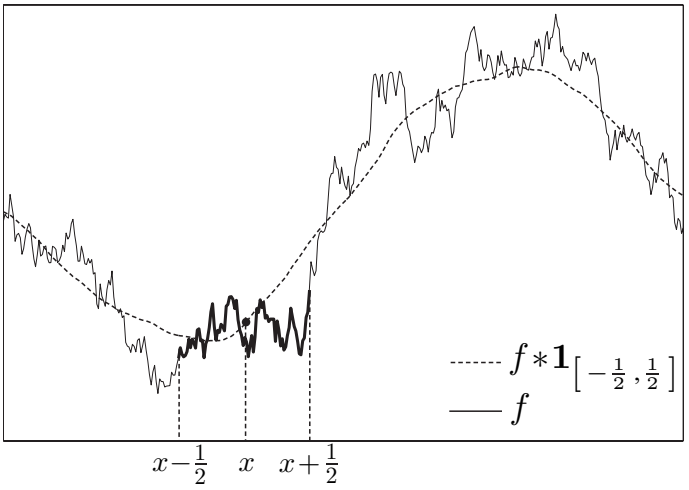


Figure 2.2: *Signal filtering with a box filter (running average).*

Filtering over Fourier domain. The Fourier basis diagonalizes convolution operators $f \mapsto f \star h$, which is equivalent to the Fourier convolution theorem

$$g = f \star g \quad \Longrightarrow \quad \hat{g}[m] = \hat{f}[m]\hat{h}[m] \tag{2.5}$$

The low pass box filtering, makes use of the following filter,

$$h(t) = \frac{1}{2\tau} 1_{[-\tau,\tau]}(t).$$

Its Fourier coefficients are

$$\hat{h}[m] = \int_{-\tau}^{\tau} e^{-2i\pi m t} \mathrm{d}t = \frac{\sin(2\pi m\tau)}{2\pi m\tau}.$$

A smoother filtering function is the Gaussian kernel

$$h_\sigma(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}.$$

Its Fourier coefficients are approximately

$$\hat{h}_\sigma[m] \simeq h_{1/\sigma}(m)$$

so that increasing the filter width σ corresponds to setting close to zero more frequencies during the filtering. Figure 2.3 shows the effect of increasing the width σ to smooth a noisy signal.

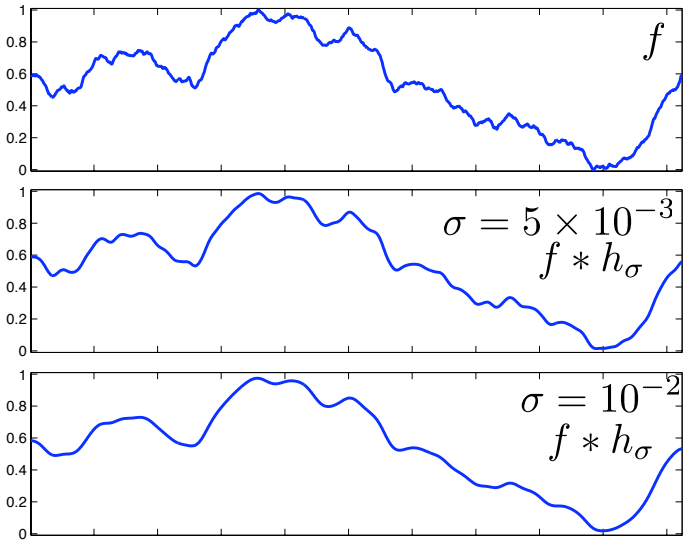


Figure 2.3: *Filtering an irregular signal with a Gaussian filter of increasing filter size σ.*

2.3 Discrete Infinite Fourier Analysis

An infinite discrete signal is defined on an infinite discrete grid as $\{h[n]\}_{n\in\mathbb{Z}}$.

2.3.1 Infinite Discrete Fourier Transform

The Fourier transform a summable $h \in \ell^1(\mathbb{Z})$ is

$$\forall\, \omega \in [0, 2\pi], \quad h(\omega) = \sum_{n\in\mathbb{Z}} h[n] e^{-i\omega n}. \tag{2.6}$$

It can be extended to finite energy signals in $\ell^2(\mathbb{Z})$ by density. This defines a 2π periodic function. This transform is in some sense dual to the Fourier coefficients (2.4) of a periodic function. Indeed, $h[m]$ are coefficients of the function $\hat{h}$ at frequency m since

$$h[m] = \langle \hat{h},\, e_m \rangle \quad \text{where} \quad e_m(t) = \frac{1}{2\pi} e^{imt}$$

where one should be careful about the fact that we use here non-normalized Fourier atoms on $[0, 2\pi]$, which is deferent from the setting defined in (2.3).

2.3.2 Convolution of Infinite Filters

The convolution of two summable signals $f, g \in \ell^1(\mathbb{Z})$ is defined as

$$f \star h[n] = \sum_m f[n] h[m-n].$$

One thus has the following Fourier convolution theorem, that is equivalent to (2.5)

$$g = f \star h \quad \Longrightarrow \quad \hat{g}(\omega) = \hat{f}(\omega)\hat{h}(\omega).$$

2.4 Discrete Finite Fourier Analysis

A sampled discrete signal is a finite dimensional vector $f \in \mathbb{C}^N$.

2.4.1 Discrete Fourier Transform

The discrete orthonormal Fourier basis is obtained by sampling the continuous Fourier basis

$$\left\{ e_m[n] = \frac{1}{\sqrt{N}} \exp\left(\frac{2i\pi}{N} mn \right) \right\}_{0 \leqslant m < N}$$

The discrete Fourier Transform (DFT) computes the N inner products with the discrete Fourier atoms

$$\hat{f}[m] = \sqrt{N} \langle f,\, e_m \rangle = \sum_{n=0}^{N-1} f[n] e^{-\frac{2i\pi}{N} mn}, \tag{2.7}$$

where e_m is traditionaly rescaled by $\sqrt{N}$, and thus $f \mapsto \hat{f}$ is not unitary.

The orthogonality of the atoms implies the following reconstruction formula

$$f[n] = \sum_{m=0}^{N-1} \langle f,\, e_m \rangle e_m = \frac{1}{N} \sum_{m=0}^{N-1} \hat{f}[m] e^{\frac{2i\pi}{N} mn}$$

which corresponds to the inverse discrete Fourier transform.

2.4.2 Fourier and Discrete Filtering

The discrete periodic convolution of two signals f and h is defined as

$$g[n] = f \star h[n] = \sum_{p=0}^{N-1} f[p] h[n - p \bmod N]. \tag{2.8}$$

Similarely to the continuous Fourier transform, the discrete transform diagonalizes translation invariant operators which are filtering $f \mapsto f \star h$. This corresponds to the discrete Fourier convolution theorem

$$g = f \star h \quad \Longrightarrow \quad \hat{g}[m] = \hat{f}[m] \cdot \hat{h}[m]. \tag{2.9}$$

2.4.3 Fast Fourier Transform

The brute force implementation of formula (2.7) requires $O(N^2)$ operations. The Fourier atoms exhibit symmetries which allows one to speed up the computations using the Fast Fourier Transform algorithm (FFT) which only requires $O(N \log(N))$.

Using the convolution formula (2.9), one can compute the convolution over the Fourier domain as follow

$$f \star g = \mathcal{F}^{-1}(\hat{f} \cdot \hat{g}),$$

where $\mathcal{F}^{-1}$ is the inverse Fourier transform and $\cdot$ is the component wise multiplication of vectors. Since $\mathcal{F}$ and $\mathcal{F}^{-1}$ are computed in $O(N \log(N))$ operations, to does $f \star g$. This is an important saving if both f and h are supported over the whole domain $\{0, \ldots, N-1\}$, since implementing (2.8) directly over the spacial domain would require $O(N^2)$ operations.

This complexity reduction had a huge impact in signal processing and numerical analysis, since computing convolutions with long filter is a major bottleneck in intensive computing applications.

2.5 2D Fourier Analysis

The 2D discrete Fourier basis for an image of $N = N_0 \times N_0$ pixels is obtained by tensor product of the 1D basis

$$e_m[n] = \frac{1}{\sqrt{N}} e^{\frac{2i\pi}{N_0} m_1 n_1 + \frac{2i\pi}{N_0} m_2 n_2} = e_{m_1}[n_1] e_{m_2}[n_2]$$

The frequency is $m = (m_1, m_2) \in \{0, \ldots, N_0 - 1\} \times \{0, \ldots, N_0 - 1\}$

The 2D Fourier atoms assume periodic boundary conditions, which corresponds to handling the square $[0,1]^2$ as a torus, identifying left/right and top/bottom boundaries of the image. For images

which do not wrap properly, 2D Fourier analysis creates large vertical and horizontal frequencies and Fourier processing creates artifacts near the image boundaries.

The 2D Fast Fourier Transform computes

$$\hat{f}[m_1, m_2] = \sqrt{N}\langle f, e_{m_1,m_2}\rangle$$

by applying the 1D FFT to each row and then each column of the image, thus requiring $O(N \log(N))$ operations.

Figure 2.4, left, shows an example of such artificial artifacts, and how they are reduced by a proper masking of the image so that it becomes continuous after periodization.

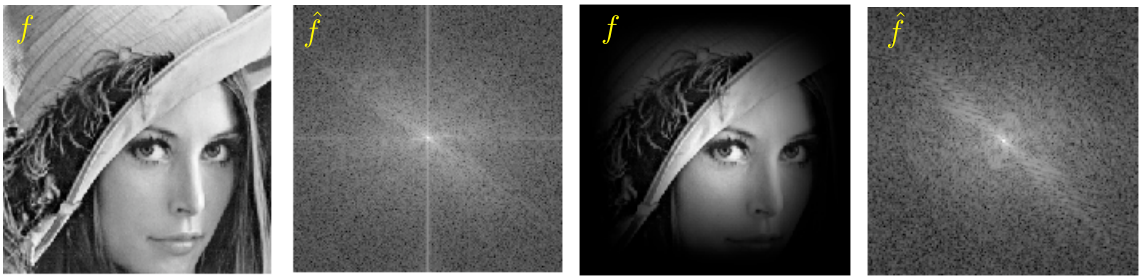


Figure 2.4: *2D Fourier analysis of a image (left), and attenuation of the periodicity artifact using masking (right).*

```
% Compute the Fourier transform.
F = fft2(f);  n = size(f,1);
% Compute marsked Fourier transform.
t = linspace(-pi(),pi(),n);
h = (cos(t)+1)/2; h = h'*h;
F1 = fft2(f.*h);
% Compute Log of Fourier transforms.
L = fftshift(log( abs(F)+1e-1 ));
L1 = fftshift(log( abs(F1)+1e-1 ));
% display
clf; imageplot( {L L1}, {'FFT' 'Masked FFT'} );
```

Matlab code 1: *2D Fourier transform and masked Fourier transforms. Input: image* `f`*, output: Fourier coefficients* `F`*,* `F1`*.*

2.6 Sampling

2.6.1 Pointwise Sampling

If the signal f_0 is smooth and the sampler impulse response h in (1.1) is localized around 0,

$$f[n] = f_0 \star h(n/N) \approx f_0(n/N).$$

An idealized sampling is a pointwise evaluation, which can be formally written using a Dirac distribution

$$f[n] = f_0(n/N) = \langle f_0, \phi_n\rangle \quad \text{where} \quad \phi_n = \delta_{n/N}$$

where f_0 is a assumed to be smooth. In this section, we consider an infinite sampling, so that $\{f[n]\}_{n\in\mathbb{Z}}$ is an infinite sequence.

2.6.2 Continuous/Discrete Fourier Connexion

The idealized sampling operator realize a mapping from functions to infinite sequence

$$f_0 \in \mathrm{L}^2(\mathbb{R}) \longmapsto f \in \mathbb{C}^{\mathbb{Z}}.$$

This mapping is equivalently expressed over the Fourier domain as a periodization using the Poisson summation formula

$$\hat{f}(\omega) = N \sum_{k=-\infty}^{+\infty} \hat{f}_0(N(\omega - 2k\pi)), \tag{2.10}$$

where $\hat{f}(\omega)$ is the 2π-periodic discrete Fourier transform (2.6) and $\hat{f}_0(\omega)$ is the continuous Fourier transform (2.1).

Figure 2.1 shows within a diagram this duality relationship between sampling and periodization.

2.6.3 Shannon Reconstruction

Shannon Theorem, applied in 1D to function in $\mathrm{L}^2(\mathbb{R})$, states that if $\mathrm{Supp}(\hat{f}) \subset [-N\pi, N\pi]$, where $\hat{f}$ is the continuous Fourier transform of f, then f is recovered from the sample $f[i]$. This comes from the fact that under this conditions, the contribution $\hat{f}_0(N(\omega - 2k\pi))$ in the periodization (2.10) does not overlap.

Furthermore, the recovery from the sample is performed with a simple linear interpolation formula

$$f(t) = \sum_i f[i] h(t - i/N) \quad \text{where} \quad h(t) = \frac{\sin(\pi N t)}{\pi N t}.$$

This formula extends to any dimension by tensor product.

The constraint $\mathrm{Supp}(\hat{f}) \subset [-N\pi, N\pi]$ implies that f should be smooth unless N is very large, see 2.5, left, for an image with a small Fourier support. Unfortunately, sounds and natural images are not smooth since they contains sharp transition and edges, see Figure 2.5, right. A precise sampling for such features requires a high number N of samples to avoid aliasing.

Figure 2.5: *A smooth image (left) and a natural image (right).*

Chapter **??** presents the compressive sensing, which is a promising avenue to perform sampling below the Shannon limit for compressible signals.

Chapter 3

Wavelet Processing

Wavelet bases in 1D are obtained by dyadic translations and scalings of a single mother wavelet ψ

$$\left\{\psi_{j,n}(t) = \frac{1}{2^{j/2}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{j\in\mathbb{Z},n\in\mathbb{Z}}. \tag{3.1}$$

A wavelet atom $\psi_{j,n}(t)$ is localized around the point $2^j n$ and has a support size proportional to the scale 2^j.

Since one usually works with signals sampled on $[0,1]$, the scale index j is often restricted to be negative for numerical computations.

3.1 Approximation and Detail Spaces

3.1.1 Approximation Spaces

Constructing a wavelet basis is not as straightforward as defining the Fourier basis. The usual construction first defines approximation spaces, from which the wavelet basis is derived.

Multiresolution analysis. A multiresolution analysis is a set of nested spaces $\{V_j\}_{j\in\mathbb{Z}}$ of $\mathrm{L}^2(\mathbb{R})$

$$\mathrm{L}^2(\mathbb{R}) \subset \ldots \subset V_{j-1} \subset V_j \subset V_{j+1} \subset \ldots \subset \{0\}. \tag{3.2}$$

The left and right inclusions should be understood as

$$\text{Closure}\left(\bigcup_j V_j\right) = \mathrm{L}^2(\mathbb{R}) \quad \text{and} \quad \bigcap_j V_j = \{0\}.$$

These spaces are supposed to be obtained by dyadic dilations, and we thus impose that

$$f(t) \in V_j \iff f(t/2) \in V_{j+1}. \tag{3.3}$$

This corresponds to the intuitive notion of mutiresolution since the function $f(t/2)$ is twice coarser that the function $f(t)$. These spaces should be invariant under dyadic translations

$$f(t) \in V_j \iff \forall\, n \in \mathbb{Z}, f(t+n2^j) \in V_j.$$

The last condition, that is important to derive the construction of a wavelet basis, is that there exists a scaling function ϕ such that $\{\phi(t-n)\}_{n\in\mathbb{Z}}$ is an orthogonal Riesz basis of the space V_0.

We note that if one has at its disposal a translation invariant Riesz basis (not orthogonal) $\{\theta(t-n)\}_n$ of V_0, one can compute a translation invariant orthogonal basis $\{\phi(t-n)\}_{n\in\mathbb{Z}}$ by Fourier orthogonalization

$$\hat{\phi}(\omega) = \frac{\hat{\theta}(\omega)}{\sqrt{\sum_k \hat{\theta}(\omega+2k\pi)}}. \tag{3.4}$$

Scaling function orthobasis. The scaling function ϕ defines an orthogonal basis of all the spaces V_j using dyadic dilations and translations

$$V_j = \mathrm{Span}(\phi_{j,n})_{n\in\mathbb{Z}} \quad \text{where} \quad \phi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\phi\left(\frac{t-2^j n}{2^j, 13}\right) \tag{3.5}$$

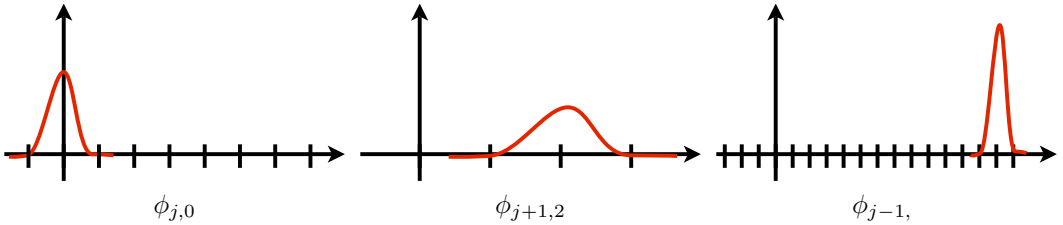


Figure 3.1: *Translation and scaling to generate approximation spaces.*

Figure 3.1 illustrates the translation and scaling effect.

The best approximation at scale 2^j (or resolution 2^{-j}) of a signal f is obtained by linear projection, that is easily defined using the scaling ortho-basis of V_j

$$P_{V_j}(f) = \sum_n \langle f,\, \phi_{j,n}\rangle \phi_{j,n}. \tag{3.6}$$

Boundary conditions. For bounded domains $f \in \mathrm{L}^2([0,1])$, one can use periodic boundary conditions, $[0,1] \simeq \mathbb{R}/\mathbb{Z}$, and use only $j \leqslant 0$. In this case the translation is assumed to be performed modulo 1, and all the theory carries over without modification. This however introduces periodic boundary artifacts.

To remove these artifacts, one uses symmetric boundary conditions. This leads to non-translation invariant bases, and is thus more difficult to implement and boundary wavelets should be modified.

Haar Approximation Spaces The Haar multiresolution is obtained by considering piecewise-constant approximations

$$V_j = \left\{ f \setminus f \text{ constant on } [2^j n, 2^j(n+1)) \right\}. \tag{3.7}$$

The scaling function can be defined as $\theta = 1_{[0,1]}$. Figure 3.2, left, shows this function.

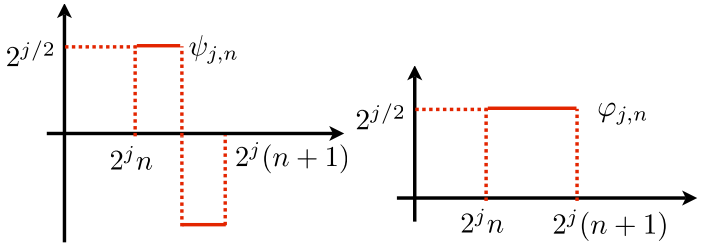


Figure 3.2: *Haar scaling (left) and wavelet (right) functions.*

The projection $P_{V_j}(f)$ defined in (3.6) is the best piecewise constant approximation

$$P_{V_j}(f) = \sum_n \frac{1}{2^j} 1_{[n2^j,(n+1)2^j)} \int_{n2^j}^{(n+1)2^j} f(t)\mathrm{d}t.$$

Figure 3.3 shows examples of such projections for a decreasing resolution.

This multiresolution generalizes to higher order spline approximations, although it is more difficult to compute the scaling function ϕ, which requires to apply the orthogonalization (3.4) to the cardinal spline.

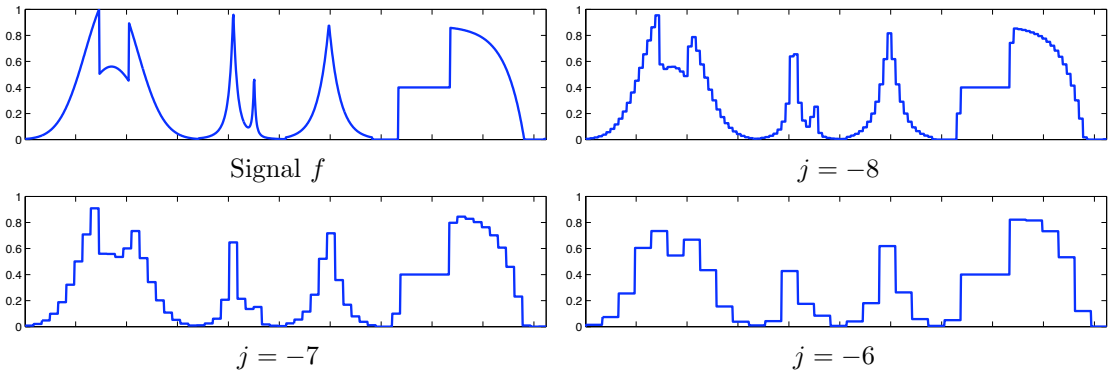


Figure 3.3: *1D Haar multiresolution projection of a function.*

3.1.2 Detail Spaces

To build an orthogonal basis of the whole space $\mathrm{L}^2(\mathbb{R})$, one needs to consider the detail spaces W_j, that are orthogonal complements of the approximation spaces

$$V_{j-1} = V_j \oplus^{\perp} W_j.$$

This leads to the following sequence of embedded spaces

$$\mathrm{L}^2(\mathbb{R}) \longrightarrow \cdots \searrow V_{j-1} \longrightarrow V_j \searrow V_{j+1} \longrightarrow \cdots \longrightarrow \{0\}$$
$$W_{j-1} \qquad W_j \qquad W_{j+1}$$

The decomposition of the whole space into detail spaces is an orthogonal sum

$$\mathrm{L}^2(\mathbb{R}) = \bigoplus_{j=-\infty}^{+\infty} W_j = V_{j_0} \bigoplus_{j \leqslant j_0} W_j.$$

Since the space V_j are spanned by an orthogonal translation invariant basis (3.5), one could hope that this is also the case for the detail spaces, that should be spanned by a wavelet function ψ

$$W_j = \mathrm{Span}(\psi_{j,n})_{n \in \mathbb{Z}} \quad \text{where} \quad \psi_{j,n}(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^j n}{2^j}\right)$$

where $\{\psi_{j,n}\}_{n \in \mathbb{Z}}$ is an orthogonal basis of W_j. In the following we show that is indeed the case and how ψ is defined using filtering.

The projection on details space is

$$P_{W_j}(f) = \sum_n \langle f, \psi_{j,n} \rangle \psi_{j,n} = P_{V_{j-1}}(f) - P_{V_j}(f).$$

Full and truncated wavelet bases. A full wavelet basis of $\mathrm{L}^2(\mathbb{R})$ is defined by considering all the detail spaces W_j

$$\left\{\psi_{j,n} \setminus (j,n) \in \mathbb{Z}^2\right\}.$$

One can also define a truncated basis

$$\{\psi_{j,n} \setminus j \leqslant j_0, n \in \mathbb{Z}\} \cup \{\phi_{j_0,n} \setminus n \in \mathbb{Z}\}.$$

When using periodic boundary condition, the truncation at scale $j_0 \leqslant 0$ defines a wavelet basis of $\mathrm{L}^2(\mathbb{R}/\mathbb{Z})$

$$\left\{\psi_{j,n} \setminus j \leqslant j_0, 0 \leqslant n \leqslant 2^{-j}\right\} \cup \left\{\phi_{j_0,n} \setminus 0 \leqslant n \leqslant 2^{-j_0}\right\}.$$

Haar wavelets. For the Haar multiresolution (3.7), one has

$$W_j = \left\{ f \setminus \forall n \in \mathbb{Z},\ f \text{ constant on } [2^{j+1}n, 2^{j+1}(n+1)) \quad \text{and} \quad \int_{n2^j}^{(n+1)2^j} f = 0 \right\}. \tag{3.8}$$

A possible choice for a mother wavelet function is

$$\psi(t) = \frac{1}{\sqrt{2}} \begin{cases} 1 & \text{for} \quad 0 \leqslant t < 1/2, \\ -1 & \text{for} \quad 1/2 \leqslant t < 1, \\ 0 & \text{otherwise,} \end{cases}$$

as shown on Figure 3.2, right.

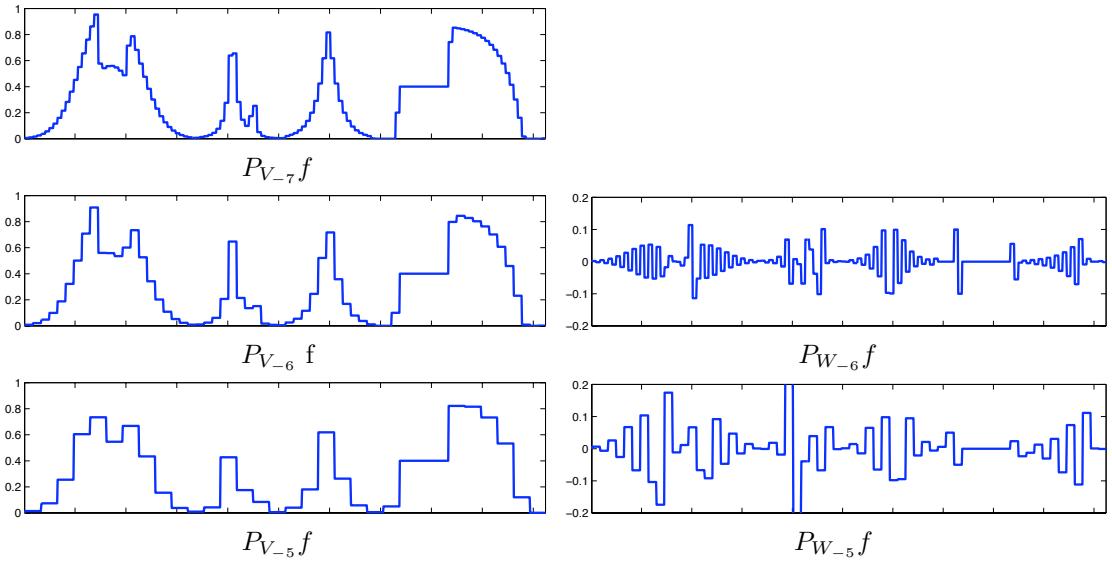


Figure 3.4: *Projection on Haar approximation spaces (left) and detail spaces (right).*

Figure 3.4 shows examples of projections on details spaces, and how they can be derived from projection on approximation spaces.

3.2 1D Wavelet Processing

3.2.1 Wavelet Coefficients

Detail and scaling coefficients. As this is the case for the Fourier transform, a wavelet transform of an analog signal $f_0 \in \mathrm{L}^2([0,1])$ computes the set of coefficients

$$\forall\, 0 \leqslant n < 2^{-j}, \quad d_j[n] = \langle f_0,\, \psi_{j,n} \rangle,$$

using periodic boundary conditions. We note that since there exists a multitude of mother wavelet functions ψ, there also exists many different wavelet transforms.

Figure 3.5 shows examples of wavelet coefficients. For each scale 2^j, there are 2^{-j} coefficients.

The scaling coefficients are defined as

$$\forall\, 0 \leqslant n < 2^{-j}, \quad a_j[n] = \langle f_0,\, \phi_{j,n} \rangle.$$

Sampling consistent with the scaling functions. A discrete signal $f \in \mathbb{C}^N$ corresponds to the sampling of an analog signal $f_0 \in \mathrm{L}^2([0,1])$ on an uniform grid $\{n2^J\}_{0 \leqslant n < N}$ where $2^{-J} = N$.

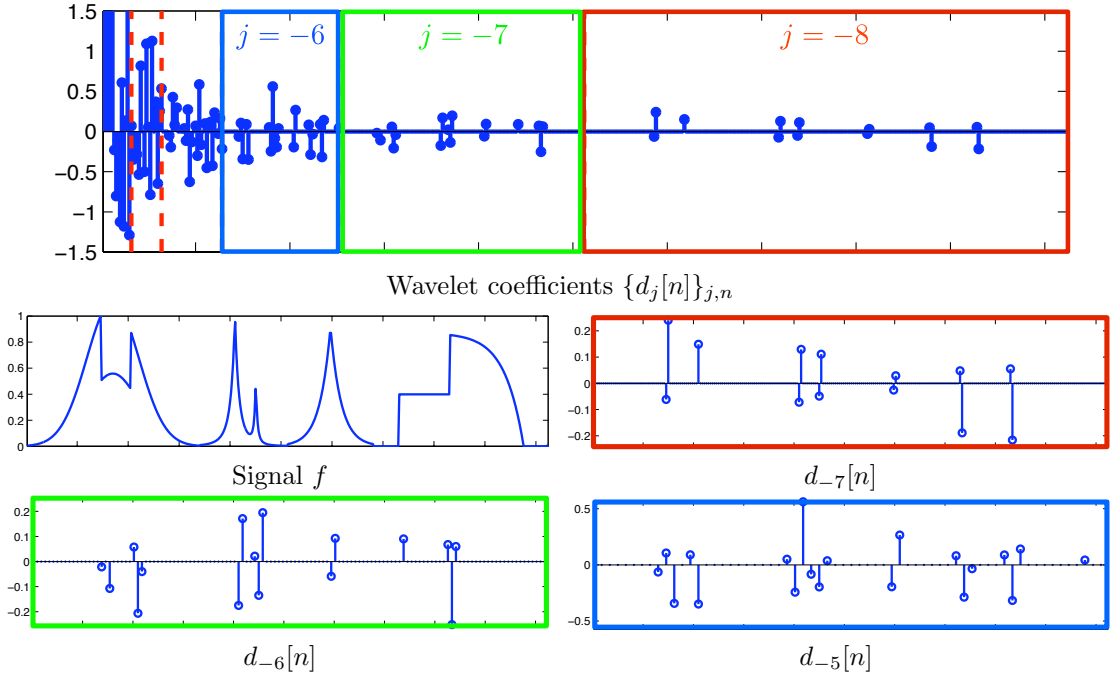


Figure 3.5: *Wavelet coefficients. Top row: all the coefficients. Bottoms rows: zoom on the different scales*

To identify the discrete wavelet transform of f with the continuous transform of f_0, we assume a consistency of the sampling with the scaling function

$$\forall\, 0 \leqslant n < N, \quad f[n] = a_J[n] = \frac{1}{2^{J/2}} \int f_0(t)\phi(t/2^J - n)\mathrm{d}t = \langle f_0,\, \phi_{J,n} \rangle \tag{3.9}$$

Under this hypothesis, the detail coefficients d_j of f_0 are computed from the discrete signal f using a fast algorithm.

This hypothesis is questionable, since the sensor impulse response h in (1.1) is given by the hardware, and is likely to differs from ϕ. It is possible to account for this imperfect match by modifying the values of $f[n]$ prior to computing the wavelet coefficients, but in practice, the approximation $f[n] \approx \langle f_0,\, \phi_{J,n} \rangle$ is sufficient.

Discrete signals and discrete wavelets. The wavelet coefficients depend linearly on both the continuous and the discrete signals, and can thus be written as

$$\forall\, J < j \leqslant 0,\ \forall\, 0 \leqslant n < 2^{-j}, \quad d_j[n] = \langle f_0,\, \psi_{j,n} \rangle = \langle f,\, \bar{\psi}_{j,n} \rangle \tag{3.10}$$

where $\psi_{j,n}$ are the continuous wavelet atoms (3.1) and $\bar{\psi}_{j,n} \in \mathbb{C}^N$ are discrete wavelet vectors defined implicitly using this relation. This defines a discrete wavelet basis of $\mathbb{C}^N$

$$\{\bar{\psi}_{j,n}\}_{J<j\leqslant 0,\ 0\leqslant n<2^{-j}} \cup \{\bar{\phi}_{0,0}\} \tag{3.11}$$

where by convention $\bar{\phi}_{0,0} = 1/\sqrt{N}$ is the constant vector.

For large N, these discrete atoms resemble their continuous counterparts, but since they are defined on a discrete grid, they cannot be generated by dilation of a single mother wavelet. They however satisfy a translation relationship on the discrete grid

$$\forall\, 0 \leqslant k < N, \quad \bar{\psi}_{j,n}[k] = \bar{\psi}_{j,0}[k - 2^j n].$$

3.2.2 Forward Wavelet Transform

For now we assume that the wavelet function ψ and scaling function ϕ are given, and we derive a fast iterative algorihtm. This section shows how to apply this algorithm without knowing in closed form these functions.

Low-pass coefficients. The embedding and refinement relations (3.3) and (3.2) imply that $\phi(t/2) \in V_0$. One can thus expand $\phi(t/2)$ into the orthogonal basis of scaling functions to obtain

$$\frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} h[n]\phi(t-n)$$

where the coefficients of the filter $h \in \mathbb{C}^{\mathbb{Z}}$ are defined as

$$h[n] = \frac{1}{\sqrt{2}}\langle \phi(t/2),\, \phi(t-n)\rangle.$$

High-pass coefficients. One can also decompose $\psi(t/2)$ into the scaling function orthogonal basis to obtain

$$\frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} g[n]\phi(t-n)$$

where the coefficients of the filter $g \in \mathbb{C}^{\mathbb{Z}}$ are defined as

$$g[n] = \frac{1}{\sqrt{2}}\langle \psi(t/2),\, \phi(t-n)\rangle.$$

Refinement relationship. Using the change of variable $t \to \frac{t-2^j p}{2^{j-1}}$, one obtains

$$\frac{1}{\sqrt{2}}\phi\left(\frac{t-2^j p}{2^j}\right) = \sum_{n\in\mathbb{Z}} h[n]\phi\left(\frac{t}{2^{j-1}} - (n+2p)\right).$$

Using the second change of variable $n \to n-2p$, one obtains

$$\phi_{j,p} = \sum_{n\in\mathbb{Z}} h[n-2p]\phi_{j-1,n}. \tag{3.12}$$

One also has the following refinement relation for the wavelets

$$\frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} g[n]\phi(t-n).$$

Similar change of variables lead to

$$\psi_{j,p} = \sum_{n\in\mathbb{Z}} g[n-2p]\phi_{j-1,n}. \tag{3.13}$$

Subsampled filtering step. One step of the wavelet transform algorithm computes (a_j, d_j) from a_{j-1}.

Using (3.12) and (3.13), one has the following relationship for the recursive computation of the coarse scale approximation coefficients

$$a_j[p] = \sum_{n\in\mathbb{Z}} h[n-2p]a_{j-1}[n] = a_{j-1} \star \tilde{h}[2p], \tag{3.14}$$

$$d_j[p] = \sum_{n\in\mathbb{Z}} g[n-2p]a_{j-1}[n] = a_{j-1} \star \tilde{g}[2p], \tag{3.15}$$

where $\tilde{x}[n] = x[-n]$. These relations correspond to low and high pass filterings followed by a sub-sampling

$$a_{j+1} = (a_j \star \tilde{h}) \downarrow 2,$$
$$d_{j+1} = (a_j \star \tilde{g}) \downarrow 2,$$

where the downsampling operator is defined as

$$(a \downarrow 2)[n] = a[2n].$$

Figure 3.6 shows two steps of application of these refinement relationships. The code 2 implements the decomposition steps (3.14) and (3.15)

```
% Low/High pass filtering followed by sub-sampling.
a = subsampling( cconv(f,h) );
d = subsampling( cconv(f,g) );
% Up-sampling followed by filtering.
f1 =  cconv(upsampling(a),reverse(h)) + cconv(upsampling(d),reverse(g));
% Check that we really recover the same signal.
disp(strcat((['Error |f-f1|/|f| = ' num2str(norm(f-f1)/norm(f))])));
```

Matlab code 2: *Filtering followed by sub-sampling: forward and backward. Input: signal* `f`*, outputs coarse and detail coefficients* `a`*,* `d`*.*

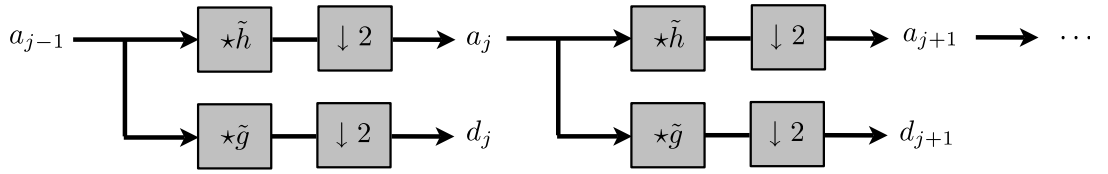


Figure 3.6: *Forward filter bank decomposition.*

Fast wavelet transform algorithm. The fast wavelet transform (FWT) applies iteratively the steps (3.14) and (3.15), starting from $j = J$ where $a_J = f$ is known. The FWT operates as follow:
– Input: signal $f \in \mathbb{C}^N$.
– Initialization: $a_J = f$.
– For $j = J, \ldots, j_0 - 1$.

$$a_{j+1} = (a_j \star \tilde{h}) \downarrow 2$$
$$d_{j+1} = (a_j \star \tilde{g}) \downarrow 2$$

– Output: the coefficients $\{d_j\}_{j_0 \leqslant j < J} \cup \{a_{j_0}\}$.

Figure 3.7 shows the process of extracting iteratively the wavelet coefficients. Figure 3.8 shows an example of computation, where at each iteration, the coefficients of a_j and d_j are added to the left of the output vector. The code 3 implements this forward transform.

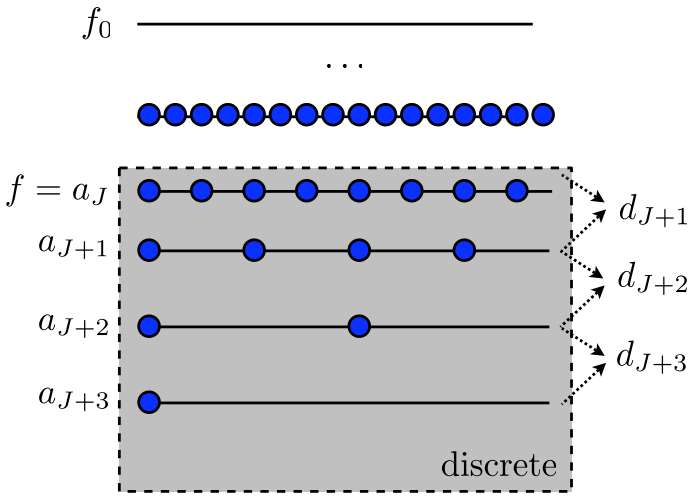


Figure 3.7: *Pyramid computation of the coefficients.*

The computational complexity of the FWT applied to a vector of N entries is

$$\sum_{j=-\log_2(N)}^{0} 2^j N(|h| + |g|) = O(N \times (|h| + |g|))$$

operations. It thus has a linear complexity with respect to N, which is faster than the FFT algorithm that has $O(N \log(N))$ complexity. Furthermore, its complexity also increases with the size of the filters.

```
Jmax = log2(n)-1; Jmin = 0; fw = f;
for j=Jmax:-1:Jmin
    Coarse = subsampling(cconv(fw(1:2^(j+1)),h));
    Detail = subsampling(cconv(fw(1:2^(j+1)),g));
    fw(1:2^(j+1)) = cat(1, Coarse, Detail );
end
```

Matlab code 3: *FWT algorithm, the input is* `f` *and the output is* `fw` *that stores all wavelet coefficients.*

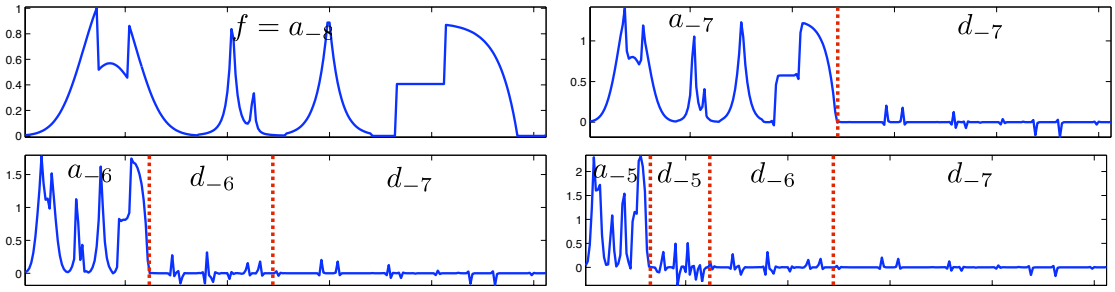


Figure 3.8: *Wavelet decomposition algorithm.*

Haar Refinement For the Haar wavelets, one has

$$\phi_{j,n} = \frac{1}{\sqrt{2}}(\phi_{j-1,2n} + \phi_{j-1,2n+1}),$$

$$\psi_{j,n} = \frac{1}{\sqrt{2}}(\phi_{j-1,2n} - \phi_{j-1,2n+1}).$$

This corresponds to the filters

$$h = [\ldots,\ 0,\ h[0] = \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 0, \ldots],$$

$$g = [\ldots,\ 0,\ h[0] = \frac{1}{\sqrt{2}},\ -\frac{1}{\sqrt{2}},\ 0, \ldots].$$

The Haar wavelet transform algorithm thus processes by iterating averaging and differences:
– **Input:** signal $f \in \mathbb{C}^N$.
– **Initialization:** $a_J = f$.
– **For** $j = J, \ldots, j_0 - 1$.

$$a_{j+1}[n] = \frac{1}{\sqrt{2}}(a_{j-1}[2n] + a_{j-1}[2n+1]),$$
$$d_{j+1}[n] = \frac{1}{\sqrt{2}}(a_{j-1}[2n] - a_{j-1}[2n+1]).$$

– **Output:** the coefficients $\{d_j\}_{j_0 \leqslant j < J} \cup \{a_{j_0}\}$.

3.2.3 Inverse Wavelet Transform

A forward elementary step

$$a_{j-1} \in \mathbb{R}^{2^{1-j}} \longmapsto (a_j, d_j) \in \mathbb{R}^{2^{-j}} \times \mathbb{R}^{2^{-j}}$$

is an orthogonal mapping

$$\|a_{j-1}\|^2 = \|a_j\|^2 + \|d_j\|^2.$$

The backward elementary step

$$(a_j, d_j) \in \mathbb{R}^{2^{-j}} \times \mathbb{R}^{2^{-j}} \longmapsto a_{j-1} \in \mathbb{R}^{2^{1-j}}$$

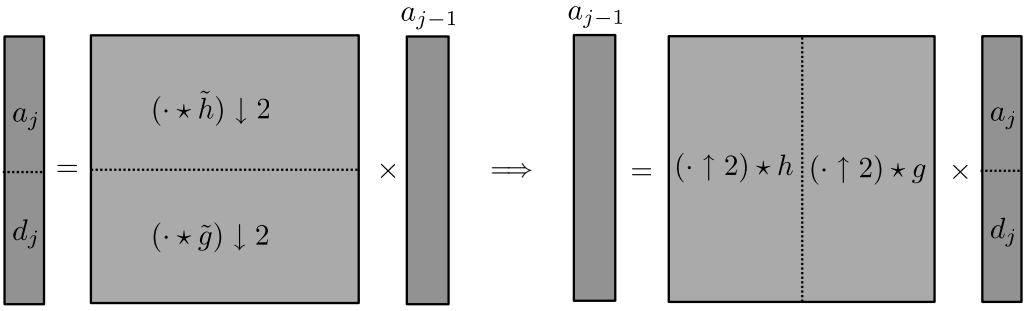


Figure 3.9: *Wavelet inversion in matrix format.*

is thus the transposed of the forward mapping. This is shown using matrix notations in Figure 3.9.

The transpose of sub-sampling is the up-sampling operator, defined by

$$(a \uparrow 2)[n] = \begin{cases} a[k] & \text{if} \quad n = 2k, \\ 0 & \text{if} \quad n = 2k+1. \end{cases}$$

The transpose of filtering by $\tilde{h}$ is filtering by the reverse filter h. One thus has

$$a_{j-1} = (a_j \uparrow 2) \star h + (d_j \uparrow 2) \star g.$$

The inverse Fast wavelet transform iteratively applies this elementary step

– Input: $\{d_j\}_{j_0 \leqslant j < J} \cup \{a_{j_0}\}$.

– For $j = j_0, \ldots, J+1$.

$$a_{j-1} = (a_j \uparrow 2) \star h + (d_j \uparrow 2) \star g.$$

– Output: $f = a_J$.

This process is shown using a block diagram in Figure 3.10, which is the inverse of the block diagram 3.6. The code 4 implements this inverse transform.

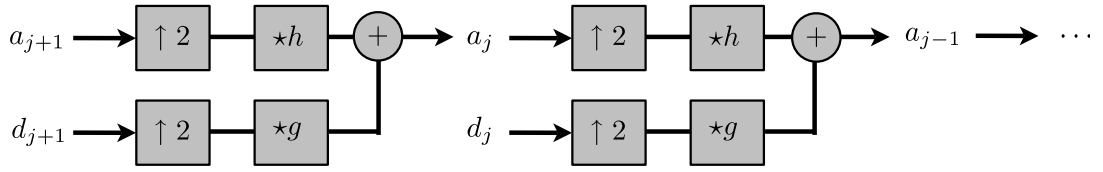


Figure 3.10: *Backward filterbank recomposition algorithm.*

```
f1 = fw;
for j=Jmin:Jmax
    Coarse = f1(1:2^j);
    Detail = f1(2^j+1:2^(j+1));
    Coarse = cconv(upsampling(Coarse,1),reverse(h),1);
    Detail = cconv(upsampling(Detail,1),reverse(g),1);
    f1(1:2^(j+1)) = Coarse + Detail;
end
```

Matlab code 4: *Inverse FWT algorithm, the input is* `fw` *that stores all wavelet coefficients and the output is* `f1`.

3.3 2D Wavelet Processing

There is two ways to extends a 1D wavelet basis into a 2D basis. The simplest way, detailed in Section 3.3.2, computes tensor products of wavelet functions. A more complicated way, detailed in Section 3.3.3, makes use of three different 2D mother wavelet functions, which enables wavelet atoms with a square support.

3.3.1 2D Multiresolutions

Separable multiresolutions. A 2D separable multiresolution analysis of $\mathrm{L}^2(\mathbb{R}^2)$ is obtained from a 1D muliresolution $\{V_j\}_j$ of $\mathrm{L}^2(\mathbb{R})$ as follow

$$V_j \otimes V_j = \{f(x_1)g(x_2) \setminus f \in V_j,\ g \in V_j\}\,.$$

For each $j \in \mathbb{Z}$, this tensor product approximation space is generated by tensor product of scaling functions

$$V_j \otimes V_j = \mathrm{Span}\{\phi^C_{j,n}\}_{n\in\mathbb{Z}^2}.$$

where

$$\phi^C_{j,n}(x) = \frac{1}{2^j}\phi^C\left(\frac{x-2^j n}{2^j}\right) \quad \text{and} \quad \phi^C(x) = \phi(x_1)\phi(x_2).$$

This construction extends to multiresolutions of $\mathrm{L}^2([0,1]^2)$ by restricting the indices to

$$j \leqslant 0, \quad \text{and} \quad 0 \leqslant n_1, n_2 < 2^{-j}.$$

2D consistent discretization. An analog image $f_0 \in \mathrm{L}^2([0,1]^2)$ is sampled on a discrete grid $\{(n_1,n_2)2^J\}_{n=0}^{N_0-1}$ of $N = N_0 \times N_0$ pixels, where $N_0 = 2^{-J}$.

Similarly to the 1D setting (3.9), we assume a consistency between the sampling scheme and the scaling function, such that for an image $f \in \mathbb{C}^N$ of N pixels

$$\forall\, 0 \leqslant n_1, n_2 < N_0, \quad f[n] = a_J[n] = \langle f_0,\, \phi^C_{J,n}\rangle. \tag{3.16}$$

Wavelet coefficients of f_0 can then be computed from the discrete signal $f \in \mathbb{C}^N$.

Haar 2D multiresolution. For the Haar multiresolution, one obtains 2D piecewise-constant Haar approximation. A function of $V_j \otimes V_j$ is constant on squares of size $2^j \times 2^j$. Figure 3.11 shows an example of projection of an image onto these 2D Haar approximation spaces.

Figure 3.11: *2D Haar approximation.*

3.3.2 Anisotropic 2D Wavelets

Anisotropic basis. A separable (anisotropic) wavelet basis is obtained from a mother wavelet function ψ as follow

$$\psi_{j_1,j_2,n_1,n_2}(x) = \psi_{j_1,n_1}(x_1)\psi_{j_2,n_2}(x_2).$$

It corresponds to an orthogonal basis of $\mathrm{L}^2(\mathbb{R}^2)$ or $\mathrm{L}^2([0,1]^2)$ with periodic boundary conditions.

Anisotropic wavelet coefficients. Anisotropic wavelet coefficients of $f_0 \in \mathrm{L}^2([0,1]^2)$ are computed from $f \in \mathbb{C}^N$ as

$$\forall\, J < j_2, j_2 \leqslant 0,\ \forall\, 0 \leqslant n_1 < 2^{-j_1},\ \forall\, 0 \leqslant n_2 < 2^{-j_2}, \quad \langle \psi_{j_1,j_2,n_1,n_2},\, f_0\rangle = \langle \bar{\psi}_{j_1,j_2,n_1,n_2},\, f\rangle \tag{3.17}$$

where $\bar{\psi}_{j_1,j_2,n_1,n_2}$ generates a 2D discrete anisotropic wavelet basis of $\mathbb{C}^N$, that is also a tensorial basis

$$\bar{\psi}_{j_1,j_2,n_1,n_2}[x] = \bar{\psi}_{j_1,n_1}[x_1]\bar{\psi}_{j_2,n_2}[x_2],$$

where $\bar{\psi}_{j_1,n_1}$ is a 1D discrete wavelet vector, defined in (3.10).

Anisotropic wavelet transform. The anisotropic wavelet transform algorithm computes the set of coefficients $\langle \psi_{j_1,j_2,n_1,n_2}, f_0 \rangle$ defined in (3.17).

One has

$$\langle \bar{\psi}_{j_1,j_2,n_1,n_2}, f \rangle = \langle \bar{\psi}_{j_2,n_2}, f_{j_1,n_1} \rangle \quad \text{where} \quad f_{j_1,n_1}[x_2] = \langle \bar{\psi}_{j_1,n_1}, f[\cdot, x_2] \rangle^{[1D]}$$

where $\langle \cdot, \cdot \rangle^{[1D]}$ is a 1D inner product. The value of $f_{j_1,n_1}[x_2]$ for all j_1, n_1 is computed by applying the fast 1D wavelet transform the column of f indexed by x_2. The value of $\langle \bar{\psi}_{j_1,j_2,n_1,n_2}, f \rangle$ for all j_2, n_2 is computed by applying the fast 1D wavelet transform to f_{j_1,n_2}.

The fast anisotropic wavelet transform thus applies the fast 1D wavelet transform to each row and then each column of the discrete image f (the role of rows and columns is interchangeable). Its complexity is $O(N)$ operations where N is number of pixels. Figure 3.12 shows this forward transform. The inverse transform processes similarly by applying the 1D inverse transform to row and then columns. Figure 3.13, left, shows a set of wavelet coefficients.

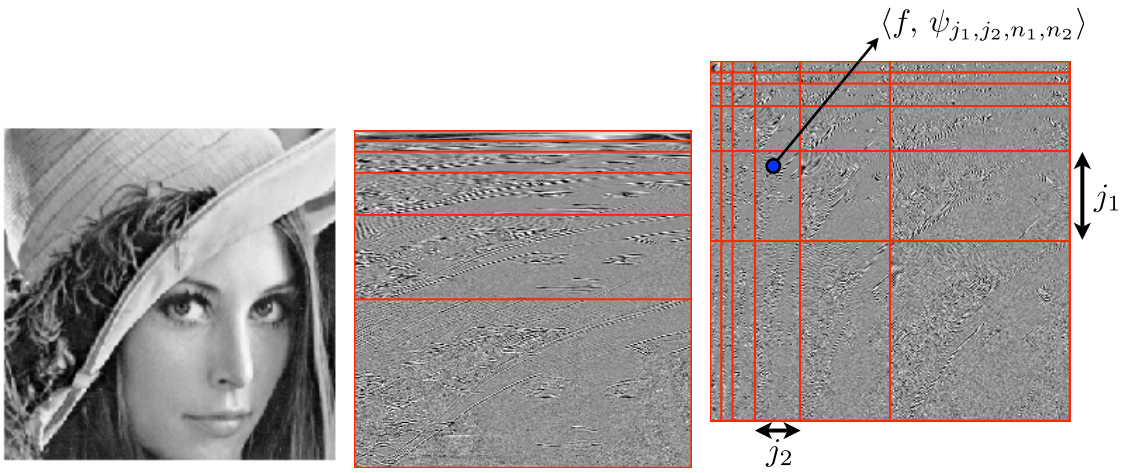


Figure 3.12: *Steps of the anisotropic wavelet transform.*

The wavelet function ψ_{j_1,j_2,n_1,n_2} as a support of size proportional to $2^{j_1} \times 2^{j_2}$, and can thus be highly stretched along the horizontal and vertical axes. Approximation using such an anisotropic wavelet basis leads to axis-aligned artifacts that are visually noticeable. An isotropic wavelet decomposition, explained in Section 3.3.3, is thus preferred in practice.

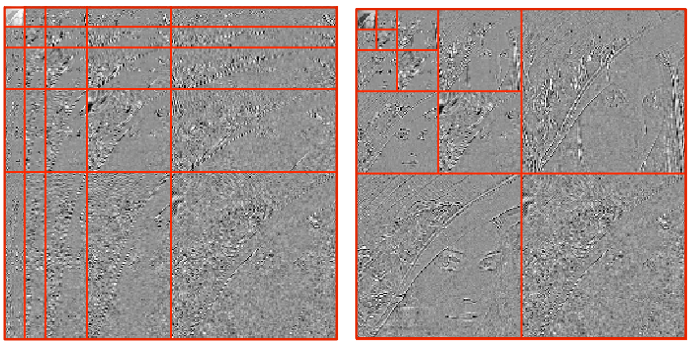


Figure 3.13: *Anisotropic (left) versus isotropic (right) wavelet coefficients.*

3.3.3 Isotropic 2D Wavelets

2D detail spaces. Introducing the wavelet orthogonal complements W_j leads to the following decomposition

$$\begin{aligned} V_{j-1} \otimes V_{j-1} &= (V_j \oplus^{\perp} W_j) \otimes (V_j \oplus^{\perp} W_j) \\ &= (V_j \otimes V_j) \oplus (V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j). \end{aligned}$$

In this decomposition, $V_j \otimes V_j$ is the coarse scale approximation, while one has the following horizontal, vertical, and diagonal detail spaces

$$W_j^H = V_j \otimes W_j, \quad W_j^V = W_j \otimes V_j, \quad \text{and} \quad W_j^D = W_j \otimes W_j.$$

Introducing the 2D wavelet detail space $W_j^{(2)}$, one obtain the following decomposition

$$V_{j-1} \otimes V_{j-1} = (V_j \otimes V_j) \oplus W_j^{(2)} \quad \text{where} \quad W_j^{(2)} = W_j^H \oplus W_j^V \oplus W_j^D.$$

This leads to the following diagram of embedded spaces

$$\mathrm{L}^2(\mathbb{R}^2) \longrightarrow \cdots \searrow W_{j-1}^{(2)} \quad V_{j-1} \otimes V_{j-1} \longrightarrow V_j \otimes V_j \searrow W_j^{(2)} \quad V_{j+1} \otimes V_{j+1} \searrow W_{j+1}^{(2)} \quad \cdots \longrightarrow \{0\}$$

Each of the three wavelet spaces is spanned with a wavelet

$$\forall\, \omega \in \{V, H, D\}, \quad W_j^{\omega} = \mathrm{Span}\{\psi_{j,n_1,n_2}^{\omega}\}_{n_1,n_2}$$

where

$$\forall\, \omega \in \{V, H, D\}, \quad \psi_{j,n_1,n_2}^{\omega}(x) = \frac{1}{2^j}\psi^{\omega}\left(\frac{x_1 - 2^j n_1}{2^j}, \frac{x_2 - 2^j n_2}{2^j}\right)$$

and where the three mother wavelets are

$$\psi^H(x) = \psi(x_1)\phi(x_2), \quad \psi^V(x) = \phi(x_1)\psi(x_2), \quad \text{and} \quad \psi^D(x) = \psi(x_1)\psi(x_2).$$

Figure 3.14 displays an examples of these wavelets.

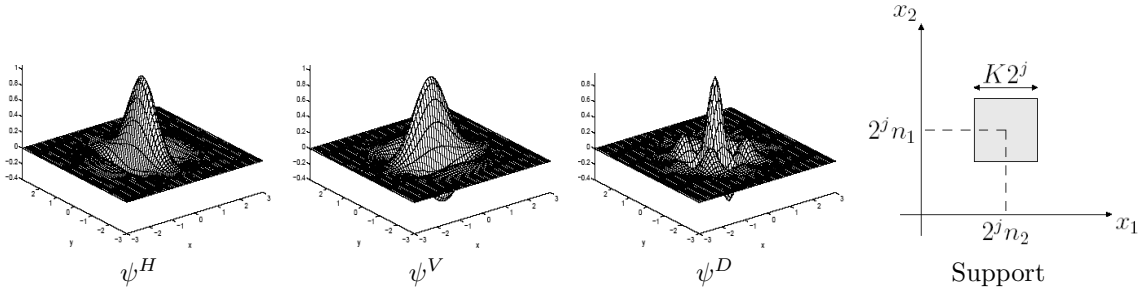


Figure 3.14: *2D wavelets and their approximative support (right).*

Discrete 2D wavelet coefficients. We suppose that the analog image $f_0 \in \mathrm{L}^2([0,1]^2)$ is sampled consistently according to (3.16). Discrete wavelet coefficients are defined as

$$\forall\, \omega \in \{V, H, D\},\ \forall\, J < j \leqslant 0,\ \forall\, 0 \leqslant n_1, n_2 < 2^{-j}, \quad d_j^{\omega}[n] = \langle f_0,\, \psi_{j,n}^{\omega} \rangle = \langle f,\, \bar{\psi}_{j,n}^{\omega} \rangle.$$

Approximation coefficients are defined as

$$a_j[n] = \langle f_0,\, \phi_{j,n}^{C} \rangle.$$

This defines a discrete orthogonal wavelet basis of $\mathbb{C}^N$

$$\left\{ \bar{\psi}_{j,n}^{\omega} \setminus J < j \leqslant 0,\ 0 \leqslant n_1, n_2 < 2^{-j},\ \omega \in \{H, V, D\} \right\} \cup \{\bar{\phi}_{0,0}\},$$

where $\bar{\phi}_{0,0} = 1/\sqrt{N}$ is the constant vector. Figure 3.15 shows examples of wavelet coefficients, that are packed in an image of N pixels. Figure 3.16 shows other examples of wavelet decompositions.

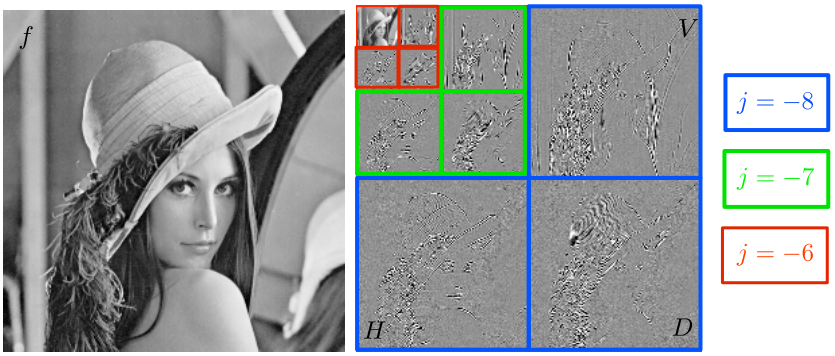


Figure 3.15: *2D wavelet coefficients.*

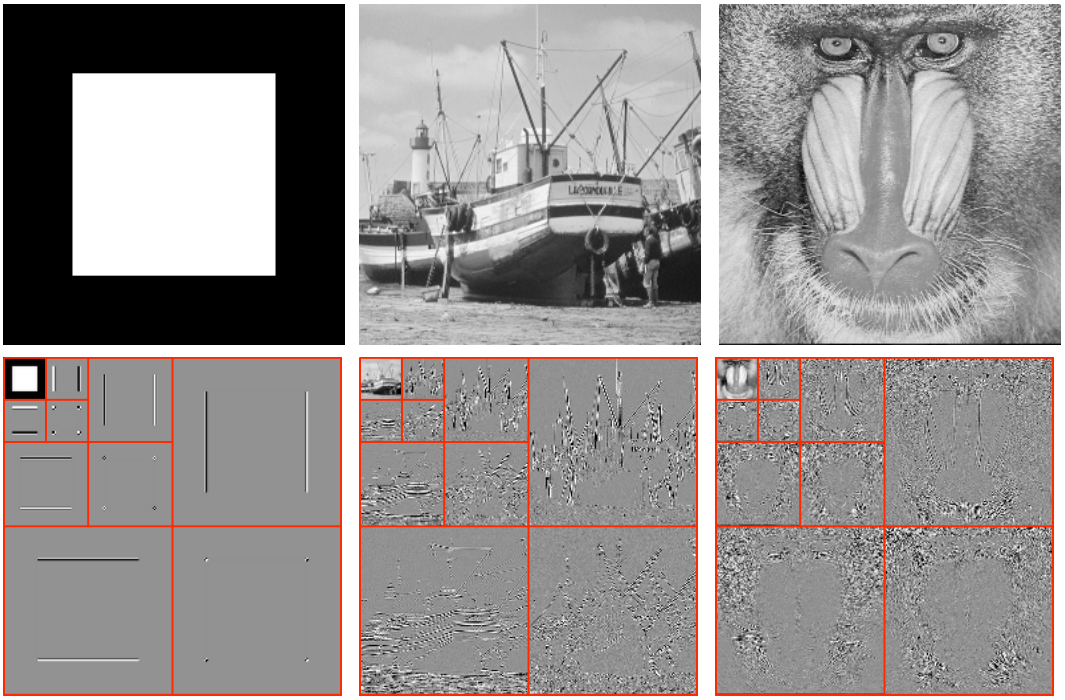


Figure 3.16: *Examples of images (top row) and the corresponding wavelet coefficients (bottom row)* .

Forward 2D wavelet transform basic step. A basic step of the computation of the 2D wavelet transform computes detail coefficients and a low pass residual from the fine scale coefficients

$$a_{j-1} \longmapsto (a_j, d_j^H, d_j^V, d_j^D).$$

Similarly to the 1D setting, this mapping is orthogonal, and is computed using the 1D filtering and sub-sampling formula (3.15) and (3.14).

One first applies 1D horizontal filtering and sub-sampling

$$\tilde{a}_j = (a_{j-1} \star^H \tilde{h}) \downarrow^H 2$$
$$\tilde{d}_j = (a_{j-1} \star^H \tilde{h}) \downarrow^H 2,$$

where $\star^H$ is the horizontal convolution, that applies the 1D convolution to each column of a matrix

$$a \star^H b[n_1, n_2] = \sum_{m_1=0}^{P-1} a[n_1 - m_1, n_2] b[m_1]$$

where $a \in \mathbb{C}^{P \times P}$ and $b \in \mathbb{C}^P$ are matrix and vectors. The notation $\downarrow^H 2$ accounts for sub-sampling in the horizontal direction

$$(a \downarrow^H 2)[n_1, n_2] = a[2n_1, n_2].$$

One then applies 1D vertical filtering and sub-sampling to $\tilde{a}_j$ and $\tilde{d}_j$ to obtain

$$\begin{array}{ll} a_j &= (\tilde{a}_j \star^V \tilde{h}) \downarrow^V 2, \\ d_j^V &= (\tilde{a}_j \star^V \tilde{g}) \downarrow^V 2, \end{array} \qquad \begin{array}{ll} d_j^H &= (\tilde{d}_j \star^V \tilde{h}) \downarrow^V 2, \\ d_j^D &= (\tilde{d}_j \star^V \tilde{g}) \downarrow^V 2, \end{array}$$

where the vertical operators are defined similarly to horizontal operators but operating on rows.

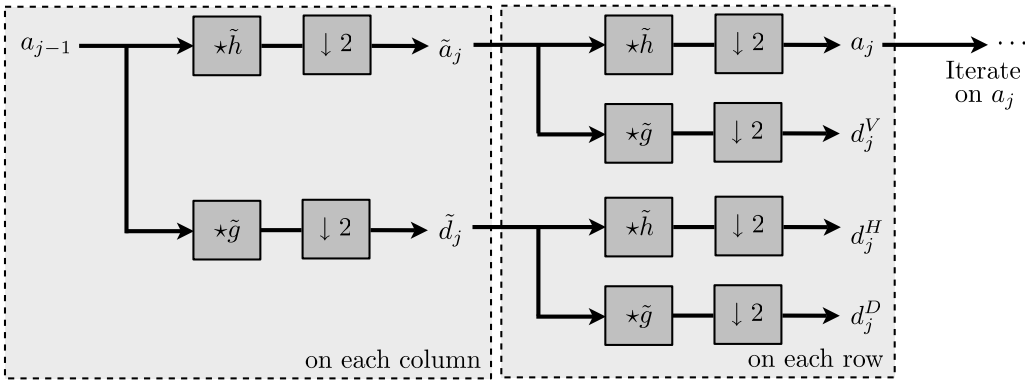


Figure 3.17: *Forward 2D filterbank step.*

Coefficients a_j Transform on rows Transform on columns

Figure 3.18: *One step of the 2D wavelet transform algorithm.*

These two forward steps are shown in block diagram in Figure 3.17. These steps can be applied in place, so that the coefficients are stored in an image of N pixels, as shown in Figure 3.18. This gives the traditional display of wavelet coefficients used in Figure 3.16.

Fast 2D wavelet transform. The 2D FWT algorithm iterates these steps through the scales:
– Input: signal $f \in \mathbb{C}^N$.
– Initialization: $a_J = f$.
– For $j = J, \ldots, j_0 - 1$.

$$\begin{array}{ll} \tilde{a}_j &= (a_{j-1} \star^H \tilde{h}) \downarrow^H 2, \\ \tilde{d}_j &= (a_{j-1} \star^H \tilde{h}) \downarrow^H 2, \\ a_j &= (\tilde{a}_j \star^V \tilde{h}) \downarrow^V 2, \end{array} \qquad \begin{array}{ll} d_j^V &= (\tilde{a}_j \star^V \tilde{g}) \downarrow^V 2, \\ d_j^H &= (\tilde{d}_j \star^V \tilde{h}) \downarrow^V 2, \\ d_j^D &= (\tilde{d}_j \star^V \tilde{g}) \downarrow^V 2. \end{array}$$

– Output: the coefficients $\{d_j^\omega\}_{j_0 \leqslant j < J, \omega} \cup \{a_{j_0}\}$.

The code 5 implements this 2D FWT algorithm.

```
Jmax = log2(n)-1; Jmin = 0;
MW = M;
for j=Jmax:-1:Jmin
    A = MW(1:2^(j+1),1:2^(j+1));
    for d=1:2
        Coarse = subsampling(cconv(A,h,d),d);
        Detail = subsampling(cconv(A,g,d),d);
        A = cat3(d, Coarse, Detail );
    end
    MW(1:2^(j+1),1:2^(j+1)) = A;
end
```

Matlab code 5: *2D FWT algorithm, the input is* `M` *and the output is* `MW` *that stores all wavelet coefficients.*

Fast 2D inverse wavelet transform. The inverse transform undo the horizontal and vertical filtering steps. The first step computes

$$\tilde{a}_j = (a_j \star^V h) \uparrow^V 2 + (d_j^V \star^V g) \uparrow^V 2,$$
$$\tilde{d}_j = (d_j^H \star^V h) \uparrow^V 2 + (d_j^D \star^V g) \uparrow^V 2,$$

where the vertical up-sampling is

$$(a \uparrow^V 2)[n_1, n_2] = \begin{cases} a[k, n_2] & \text{if} \quad n_1 = 2k, \\ 0 & \text{if} \quad n = 2k+1. \end{cases}$$

The second inverse step computes

$$a_{j-1} = (\tilde{a}_j \star^H h) \uparrow^H 2 + (\tilde{d}_j \star^H g) \uparrow^H 2.$$

Figure 3.19 shows in block diagram this inverse filter banks, that is the inverse of the diagram 3.17.

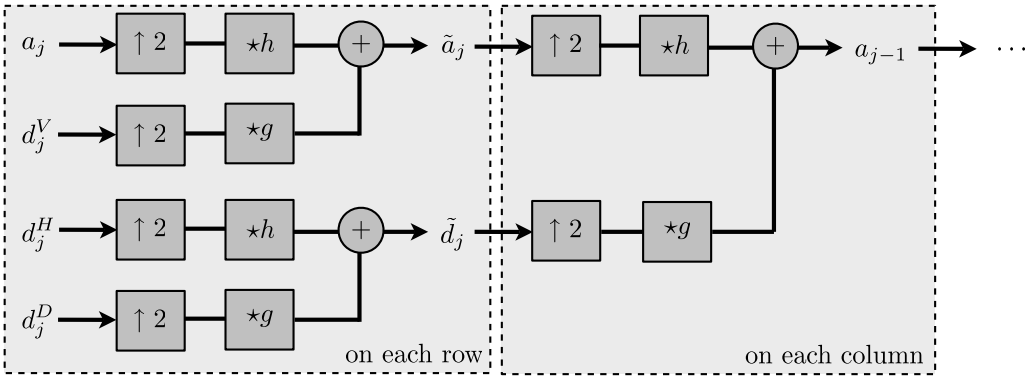


Figure 3.19: *Backward 2D filterbank step.*

The inverse Fast wavelet transform iteratively applies these elementary steps

– Input: $\{d_j^\omega\}_{j_0 \leqslant j < J, \omega} \cup \{a_{j_0}\}$.

– For $j = j_0, \ldots, J+1$.

$$\tilde{a}_j = (a_j \star^V h) \uparrow^V 2 + (d_j^V \star^V g) \uparrow^V 2,$$
$$\tilde{d}_j = (d_j^H \star^V h) \uparrow^V 2 + (d_j^D \star^V g) \uparrow^V 2,$$
$$a_{j-1} = (\tilde{a}_j \star^H h) \uparrow^V 2 + (\tilde{d}_j \star^H g) \uparrow^V 2.$$

– Output: $f = a_J$.

3.4 Wavelet Design

To be able to compute the wavelet coefficients using the FWT algorithm, it remains to know how to compute the scaling and wavelet functions. The FWT only makes use of the filters h and

```
Jmax = log2(n)-1; Jmin = 0;
MW = M;
for j=Jmax:-1:Jmin
    A = MW(1:2^(j+1),1:2^(j+1));
    for d=1:2
        Coarse = subsampling(cconv(A,h,d),d);
        Detail = subsampling(cconv(A,g,d),d);
        A = cat3(d, Coarse, Detail );
    end
    MW(1:2^(j+1),1:2^(j+1)) = A;
end
```

Matlab code 6: *2D inverse FWT algorithm, the input is* `MW` *that stores all wavelet coefficients and the output is* `M1`*.*

g, so instead of explicitly knowing the functions ϕ and ψ, one can only know these filters. Indeed, most of the known wavelets do not have explicit formula, and are implicitly defined through the cascade of the FWT algorithm.

This section shows what are the constraints h and g should satisfy, and gives practical examples. Furthermore, it shows that the knowledge of h determines g under the constraint of having quadrature filters, which is the most usual choice for wavelet analysis.

3.4.1 Low-pass Filter Constraints

Condition (C_1). The refinement equation reads

$$\frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} h[n]\phi(t-n).$$

Over the Fourier domain, this equation reads

$$\hat{\phi}(2\omega) = \frac{1}{\sqrt{2}}\hat{h}(\omega)\hat{\phi}(\omega)$$

where $h(\omega)$ is the 2π-periodic Fourier transform of infinite filters defined in (2.6), whereas $\hat{\phi}(\omega)$ is the Fourier transform of function. One can show that $|\hat{\phi}(0)| = 1$, so that this relation implies the first condition (C_1)

$$(\mathrm{C}_1) \qquad \hat{h}[0] = \sqrt{2}.$$

Condition (C_2). The orthogonality of $\phi(\cdot - n)\}_n$ is rewritten using a continuous convolution (2.2) as

$$\forall\, n \in \mathbb{Z}, \quad \phi \star \bar{\phi}(n) = 0$$

where $\bar{\phi}(x) = \phi(-x)$, and thus over the Fourier domain,

$$\sum_k |\hat{\phi}(\omega + 2k\pi)|^2 = 1.$$

This leads to condition (C_2)

$$(\mathrm{C}_2) \qquad |\hat{h}(\omega)|^2 + |\hat{h}(\omega+\pi)|^2 = 2.$$

One can then prove that

$$\{\phi_{j,n}\}_n \text{ ortho-basis of } V_j \implies (\mathrm{C}_1) + (\mathrm{C}_2).$$

Condition (C_3). The converse is not true, and one needs to control the behavior of $\hat{h}$ near 0. For instance, if $\hat{h}$ is C^1 around 0 and if condition (C_3)

$$(\mathrm{C}_3) \qquad \inf_{\omega\in[-\pi/2,\pi/2]} |\hat{h}(\omega)| > 0$$

holds then

$$(\mathrm{C}_1) + (\mathrm{C}_2) + (\mathrm{C}_3) \implies \{\phi_{j,n}\}_n \text{ ortho-basis of } V_j.$$

3.4.2 High-pass Filter Constraints

Condition ($\mathbf{C_4}$). The refinement equation for the wavelet reads

$$\frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} g[n]\phi(t-n)$$

and thus over the Fourier domain

$$\hat{\psi}(2\omega) = \frac{1}{\sqrt{2}}\hat{g}(\omega)\hat{\phi}(\omega). \tag{3.18}$$

The orthogonality of $\{\psi(\cdot - n)\}_n$ is re-written

$$\forall\, n \in \mathbb{Z}, \quad \psi \star \bar{\psi}(n) = 0$$

and thus over the Fourier domain

$$\sum_k |\hat{\psi}(\omega + 2k\pi)|^2 = 1.$$

Using the Fourier domain refinement equation (3.18), this is equivalent to condition (C_4)

$$(\mathrm{C}_4) \qquad |\hat{g}(\omega)|^2 + |\hat{g}(\omega+\pi)|^2 = 2.$$

Figure **??** shows the Fourier transform of two filters that satisfy this complementary condition.

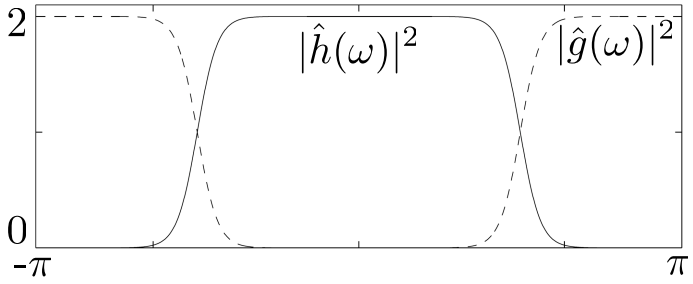


Figure 3.20: *Complementarity between a low pass and a high pass wavelet filters h and g that satisfy condition (C_4).*

Condition ($\mathbf{C_5}$). The orthogonality between $\{\psi(\cdot - n)\}_n$ and $\{\phi(\cdot - n)\}_n$ is written as

$$\forall\, n \in \mathbb{Z}, \quad \psi \star \bar{\phi}(n) = 0$$

and hence over the Fourier domain

$$\sum_k \hat{\psi}(\omega + 2k\pi)\hat{\phi}^*(\omega + 2k\pi) = 0.$$

Using the Fourier domain refinement equation (3.18), this is equivalent to condition (C_5)

$$(\mathrm{C}_5) \qquad \hat{g}(\omega)\hat{h}(\omega)^* + \hat{g}(\omega+\pi)\hat{h}(\omega+\pi)^* = 0.$$

One can then prove that under conditions $\mathbf{(C_1)}$ + $\mathbf{(C_2)}$ + $\mathbf{(C_3)}$,

$$\{\psi_{j,n}\}_n \text{ ortho-basis of } W_j \iff (\mathrm{C}_4) + (\mathrm{C}_5).$$

Quadrature mirror filters. Quadrature mirror filters (QMF) impose the value of the high pass filter as follow

$$g(\omega) = e^{-i\omega}\hat{h}(\omega+\pi)^* \iff g[n] = (-1)^{1-n}h[1-n]. \tag{3.19}$$

This choice of filter ensures that (C_4) and (C_5) are satisfied.

This choice is the natural choice to build wavelet filters, and is implicitly assumed in most constructions.

3.4.3 Wavelet Design Constraints

There exists only one Fourier transform, but there is a large choice of different mother wavelet functions ψ. They are characterized by

– Size of the support.
– Number of oscillations (the so called number p of vanishing moments).
– Symmetry (only possible for non-orthogonal bases).
– Smoothness (number of derivatives).

We now detail how these constraints are integrated together with conditions (C_1)-(C_5).

Vanishing moments. A wavelet ψ has p vanishing moments if

$$\forall\, k \leqslant p-1, \quad \int_0^1 \psi(x)\, x^k \mathrm{d}x = 0. \tag{3.20}$$

This ensures that $\langle f,\, \psi_{j,n}\rangle$ is small if f is C^α, $\alpha < p$ on $\mathrm{Supp}(\psi_{j,n})$.

This condition can be equivalently expressed over Fourier as

$$\forall\, k \leqslant p-1, \quad \frac{\mathrm{d}^k\hat{h}}{\mathrm{d}\omega^k}(\pi) = \frac{\mathrm{d}^k\hat{g}}{\mathrm{d}\omega^k}(0) = 0.$$

Support. Figure 3.21 shows the wavelet coefficients of a piecewise smooth signal. Coefficients of large magnitude are clustered near the singularities, because the wavelet ψ has enough vanishing moments.

To avoid that many wavelets create large coefficients near singularities, one should choose ψ with a small support. This requirement is however contradictory with the vanishing moment property (3.20). Indeed, one can prove that for an orthogonal wavelet basis with p vanishing moments

$$|\mathrm{Supp}(\psi)| \geqslant 2p-1,$$

where $\sup(a)$ is the largest closed interval outside of which the function f is zero.

Smoothness. In compression or denoising applications, an approximate signals is recovered from a partial set I_M of coefficients,

$$f_M = \sum_{(j,n)\in I_M} \langle f,\, \psi_{j,n}\rangle \psi_{j,n}.$$

This approximation f_M has the same smoothness as ψ.

To avoid visually unpleasant artifacts, one should thus choose a smooth wavelet function ψ. This is only for cosmetic reasons, since increasing smoothness does not leads to a better approximation. However, for most wavelet family, increasing the number of vanishing moments also increases the smoothness of the wavelets. This is for instance the case of the Daubechies family exposed in the next section.

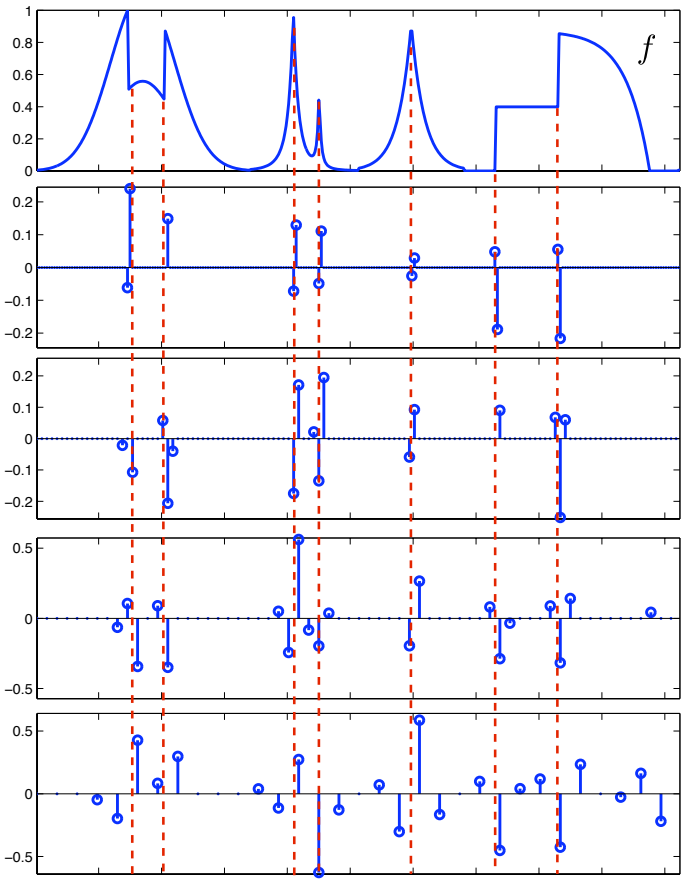


Figure 3.21: *Location of large wavelet coefficients.*

3.4.4 Daubechies Wavelets

To build a wavelet ψ with a fixed number p of vanishing moments, one designs the filter h, and use the quadrature mirror filter relation (3.19) to compute g. One thus look for h such that

$$|\hat{h}(\omega)|^2 + |\hat{h}(\omega+\pi)|^2 = 2, \quad \hat{h}(0) = \sqrt{2}, \quad \text{and} \quad \forall\, k < p,\ \frac{\mathrm{d}^k \hat{h}}{\mathrm{d}\omega^k}(\pi) = 0.$$

This corresponds to algebraic relationships between the coefficients of h, and it turns out that they can be solved explicitly using the Euclidean division algorithm for polynomials.

This leads to Daubechies wavelets with p vanishing moments, which are orthogonal wavelets with a minimum support length of $2p-1$.

For $p = 1$, it leads to the Haar wavelet, with

$$h = [h[0] = 0.7071; 0.7071].$$

For $p = 2$, one obtains the celebrated Daubechies 4 filter

$$h = [0.4830; h[0] = 0.8365; 0.2241; -0.1294],$$

and for $p = 3$,

$$h = [0; 0.3327; 0.8069; h[0] = 0.4599; -0.1350; -0.0854; 0.0352].$$

Wavelet display. Figure 3.22 shows examples of Daubechies mother wavelet functions with an increasing number of vanishing moments. These displays are obtained by computing in fact a discrete wavelet $\bar{\psi}_{j,n}$ defined in (3.11) for a very large number of samples N. This discrete wavelet is computed by applying the inverse wavelet transform to the coefficients $d_{j'}[n'] = \delta[j-j']\delta[n-n']$.

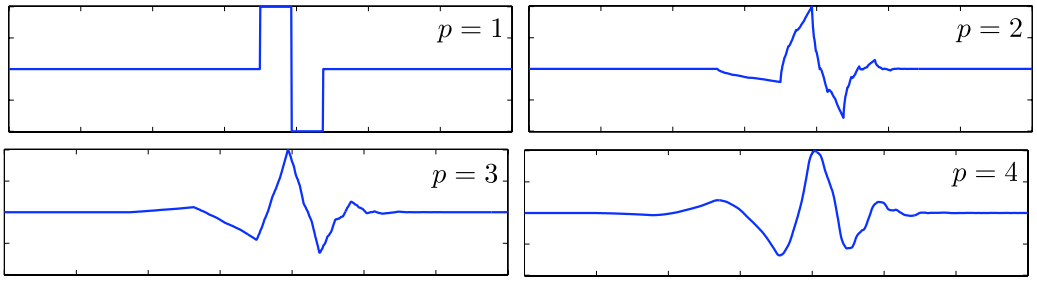


Figure 3.22: *Examples of Daubechies mother wavelets ψ with an increasing number p of vanishing moments.*

Chapter 4

Approximation and Compression

This chapter studies the theory of signal and image approximation, and gives an application to lossy compression. This theoritical analysis is performed for continuous functions $f \in \mathrm{L}^2([0,1]^d)$ for $d = 1, 2$. This analysis is important to studies the performance of compression, denoising, and super-resolution applications.

4.1 Approximation

4.1.1 Approximation in an Ortho-basis

We consider an orthogonal basis $\mathcal{B} = \{\psi_m\}_m$ of $\mathrm{L}^2([0,1]^d)$, with for instance $d = 1$ (signals) or $d = 2$ (images). We recall that the decomposition of a signal in an orthonormal basis

$$f = \sum_{m \in \mathbb{Z}} \langle f, \psi_m \rangle \psi_m$$

gives back the original signal and thus produces no error. Processing algorithms modify the coefficients $\langle f, \psi_m \rangle$ and introduce some error.

The simplest processing computes an approximation by considering only a sub-set $I_M \subset \mathbb{Z}$ of M coefficients and performing the reconstruction from this subset

$$f_M = \sum_{m \in I_M} \langle f, \psi_m \rangle \psi_m, \quad \text{where} \quad M = |I_M|.$$

The reconstructed signal f_M is the orthogonal projection of f onto the space

$$V_M = \mathrm{Span}\left\{\psi_m \setminus m \in I_M\right\}.$$

Since V_M might depend on f, this projection $f \mapsto f_M$ might be non-linear.

Since the basis is orthogonal, the approximation error is

$$\|f - f_M\|^2 = \sum_{m \notin I_M} |\langle f, \psi_m \rangle|^2.$$

The important question is now to choose the set I_M, which might depend on the signal f itself.

4.1.2 Linear Approximation

Linear approximation is obtained by fixing once for all I_M, and thus using the same set of coefficients for all f. The mapping $f \mapsto f_M$ is a linear orthogonal projection on V_M, and it satisfies

$$(f + g)_M = f_M + g_M$$

For the Fourier basis, one usually selects the low-frequency atoms

$$I_M = \{-M/2 + 1, \ldots, M/2\}.$$

For a 1D wavelet basis, one usually selects the coarse wavelets

$$I_M = \{m = (j,m) \setminus j \geqslant j_0\}$$

where j_0 is selected such that $|I_M| = M$.

Original f Linear f_M^ℓ Non-linear f_M^n

Figure 4.1: *Linear versus non-linear wavelet approximation.*

Figure 4.1, center, shows an example of such a linear approximation with wavelets. Linear approximation tends to performs poorly near singularities, because they introduce some blurring.

4.1.3 Non-linear Approximation

A non-linear approximation is obtained by choosing I_M depending on f. In particular, one would like to choose I_M to minimize the approximation error $\|f - f_M\|$. Since the basis is orthogonal, this is achieved by selecting the M largest coefficients in magnitude

$$I_M = \{M \text{ largest coefficients } |\langle f, \psi_m\rangle|\}.$$

This can be equivalently obtained using a thresholding

$$I_M = \{m \setminus |\langle f, \psi_m\rangle| > T\}$$

where T depends on the number of coefficients M,

$$M = \#\{m \setminus |\langle f, \psi_m\rangle| > T\}.$$

Computation of the threshold. There is a bijective 1:1 mapping between T and M obtained by ordering the coefficient magnitudes $|\langle f, \psi_m\rangle|$ by decaying order,

$$T = d_M \quad \text{where} \quad \{d_m\}_{m=0}^{N-1} = \{|\langle f, \psi_m\rangle|\}_0^{N-1} \quad \text{and} \quad d_m \geqslant d_{m+1}. \tag{4.1}$$

Figure 4.2 shows this mapping between M and T.

Note that the decay of the ordered coefficients is linked to the non-linear approximation decay, since

$$\|f - f_M\|^2 = \sum_{m>M} d_m^2$$

and

$$d_M^2 \leqslant \frac{2}{M} \sum_{m=M/2+1}^{M} d_m^2 \leqslant \frac{2}{M} \sum_{m>M/2}^{M} d_m^2 = \frac{2}{M}\|f - f_{M/2}\|^2.$$

This proves that

$$d_m = O(m^{-\frac{\alpha+1}{2}}) \quad \Longleftrightarrow \quad \|f - f_M\|^2 = O(M^{-\alpha}). \tag{4.2}$$

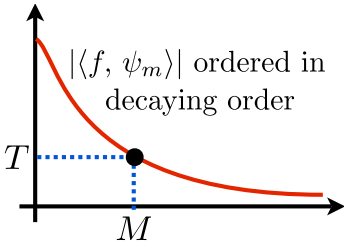


Figure 4.2: *Decay of the ordered coefficients and determination of the threshold for non-linear approxiation.*

Hard thresholding. The non-linear approximation is re-written as

$$f_M = \sum_{|\langle f, \psi_m \rangle| > T} \langle f, \psi_m \rangle \psi_m = \sum_m S_T(\langle f, \psi_m \rangle) \psi_m, \tag{4.3}$$

where

$$S_T(x) = \begin{cases} x & \text{if} \quad |x| > T \\ 0 & \text{if} \quad |x| \leqslant T \end{cases} \tag{4.4}$$

is the hard thresholding, that is displayed in Figure 4.3.

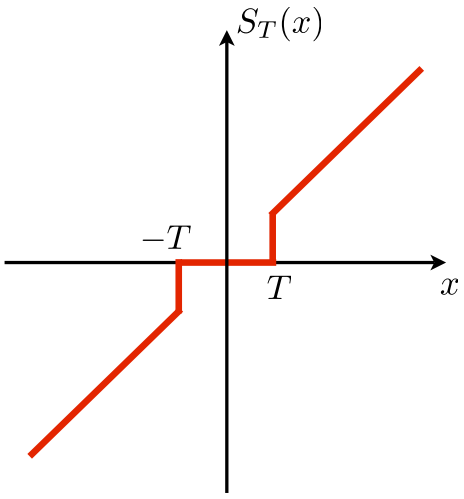


Figure 4.3: *Hard thresholding.*

4.2 Signal and Image Modeling

A signal model is a constraint $f \in \Theta$, where $\Theta \subset \mathrm{L}^2([0,1]^d)$ is a set of signals one is interested in. Figure 4.4 shows different class of models for images, that we describe in the following paragraph.

4.2.1 Uniformly Smooth Signals and Images

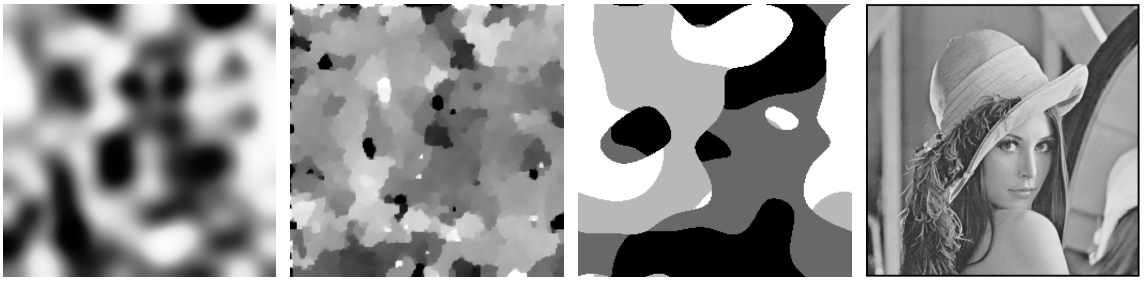
Signals with derivatives. The simplest model is made of uniformly smooth signals, that have bounded derivatives

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]^d) \setminus \|f\|_{\mathrm{C}^\alpha} \leqslant C \right\}, \tag{4.5}$$

where $C > 0$ is a fixed constant, and where in 1D

$$\|f\|_{\mathrm{C}^\alpha} = \max_{k \leqslant \alpha} \left\| \frac{\mathrm{d}^k f}{\mathrm{d}t^k} \right\|_\infty .$$

This extends to higher dimensional signals by considering partial derivatives along each direction.



Regular Bounded variations Cartoon Natural

Figure 4.4: *Examples of different kinds of image models.*

Sobolev smooth signals and images. A smooth C^α signals in (4.5) has derivatives with bounded energy, so that

$$\frac{\mathrm{d}^\alpha f}{\mathrm{d}t^\alpha}(t) = f^{(\alpha)}(t) \in \mathrm{L}^2([0,1]).$$

Using the fact that

$$\hat{f}^{(\alpha)}[m] = (2i\pi m)^\alpha \hat{f}[m]$$

where $\hat{f}$ is the Fourier coefficient defined in (2.4), one defines a so-called Sobolev functional

$$\|f\|^2_{\mathrm{Sob}(\alpha)} = \sum_{m\in\mathbb{Z}} |2\pi m|^{2\alpha} |\langle f,\, e_m\rangle|^2, \tag{4.6}$$

that satisfies $\|f\|_{\mathrm{Sob}(\alpha)} = \|f^{(\alpha)}\|$ for smooth functions. This Sobolev functional is extended to signals that have derivatives in the sense of distribution in $\mathrm{L}^2([0,1])$.

This definition extends to distributions and signals $f \in \mathrm{L}^2([0,1]^d)$ of arbitrary dimension d as

$$\|f\|^2_{\mathrm{Sob}(\alpha)} = \sum_{m\in\mathbb{Z}^d} (2\pi\|m\|)^{2\alpha} |\langle f,\, e_m\rangle|^2, \tag{4.7}$$

The C^α-Sobolev model

$$\Theta = \left\{ f \in \ell^2([0,1]^d) \setminus \max_{k\leqslant\alpha} \|f\|^2_{\mathrm{Sob}(k)} \leqslant C \right\} \tag{4.8}$$

generalizes the C^α smooth image model (4.5).

Figure 4.5 shows images with an increasing Sobolev norm for $\alpha = 2$.

Figure 4.5: *Images with increasing Sobolev norm.*

4.2.2 Piecewise Regular Signals and Images

Piecewise smooth signals. Piecewise smooth signals in 1D are functions $f \in \mathrm{L}^2([0,1])$ that are C^α smooth outside a set of less than K pointwise discontinuities

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]) \setminus \exists\, (t_i)_{i=0}^{K-1},\ \|f_{(t_i,t_{i+1})}\|_{\mathrm{C}^\alpha} \leqslant C \right\} \tag{4.9}$$

where $f_{(t_i,t_{i+1})}$ is the restriction of f to the open interval (t_i, t_{i+1}).

Piecewise smooth images. Piecewise smooth images are 2D functions $f \in \mathrm{L}^2([0,1]^2)$ that are C^α regular outside a set of less than K curves that have a finite perimeter

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]^2) \setminus \exists \Gamma = (\gamma_i)_{i=0}^{K-1},\ \|f\|_{\mathrm{C}^\alpha(\Gamma^c)} \leqslant C_1 \quad \text{and} \quad |\gamma_i| \leqslant C_2 \right\} \tag{4.10}$$

where $|\gamma_i|$ is the length of the curve γ_i and where $\|f\|_{\mathrm{C}^\alpha(\Gamma^c)}$ is the maximum norm of the derivatives of f outside the set of curves Γ.

Segmentation methods such as the one proposed by Mumford and Shah [21] implicitly assume such a piecewise smooth image modem.

4.2.3 Bounded Variation Signals and Images

Signals with edges are obtained by considering functions with bounded variations

$$\Theta = \left\{ f \in \mathrm{L}^2(\mathbb{R}^d) \setminus \|f\|_\infty \leqslant C_1 \quad \text{and} \quad \|f\|_{\mathrm{TV}} \leqslant C_2 \right\}. \tag{4.11}$$

For $d = 1$ and $d = 2$, this model generalizes the model of piecewise smooth signals (4.9) and images (4.10).

The total variation of a smooth function is

$$\int \|\nabla f(x)\| \mathrm{d}x$$

where

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_i} \right)_{i=0}^{d-1} \in \mathbb{R}^d$$

is the gradient vector at x. The total variation is extended to discontinuous images, that might for instance exhibit jumps across singular curves (the edges of the image). In particular, the total variation of a piecewise smooth image is the sum of the lengths of its level sets

$$\|f\|_{\mathrm{TV}} = \int_{-\infty}^{\infty} |\mathcal{L}_t(f)| \mathrm{d}t < +\infty \quad \text{where} \quad \mathcal{L}_t(f) = \{x \setminus f(x) = t\}, \tag{4.12}$$

and where $|\mathcal{L}_t(f)|$ is the length of $\mathcal{L}_t(f)$. For a set $\Omega \subset \mathbb{R}^2$ with finite perimeter $|\partial\Omega|$, then

$$\|1_\Omega\|_{\mathrm{TV}} = |\partial\Omega|.$$

The model of bounded variation was introduced in image processing by Rudin, Osher and Fatemi [25].

4.2.4 Cartoon Images

The bounded variation image model (4.11) does not constrain the smoothness of the level set curves $\mathcal{L}_t(f)$. Geometrical images have smooth contour curves, which should be taken into account to improve the result of processing methods.

The model of C^α cartoon images is composed of 2D functions that are C^α regular outside a set of less that K regular edge curves γ_i

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]^2) \setminus \exists \Gamma = (\gamma_i)_{i=0}^{K-1},\ \|f\|_{\mathrm{C}^\alpha(\Gamma^c)} \leqslant C_1 \quad \text{and} \quad \|\gamma_i\|_{\mathrm{C}^\alpha} \leqslant C_2 \right\} \tag{4.13}$$

Figure 4.6: *Cartoon image with increasing total variation.*

where each γ_i is a arc-length parameterization of the curve $\gamma_i : [0, A] \mapsto [0, 1]^2$. Figure 4.6 shows cartoon images with increasing total variation $\|f\|_{\mathrm{TV}}$.

Typical images might also be slightly blurred by optical diffraction, so that one might consider a blurred cartoon image model

$$\tilde{\Theta} = \left\{ \tilde{f} = f \star h \in \mathrm{L}^2([0,1]^2) \setminus f \in \Theta \quad \text{and} \quad h \in \mathcal{H} \right\} \tag{4.14}$$

where Θ is the model of sharp (unblurred) images (4.13) and $\mathcal{H}$ is a set of constraints on the blurring kernel, for instance $h \geqslant 0$ should be smooth, localized in space and frequency. This unknown blurring makes difficult brute force approaches that detects the edges location Γ and then process the regular parts in $[0, 1]^2 \backslash \Gamma$.

Figure 4.7 shows examples of images in Θ and $\tilde{\Theta}$.

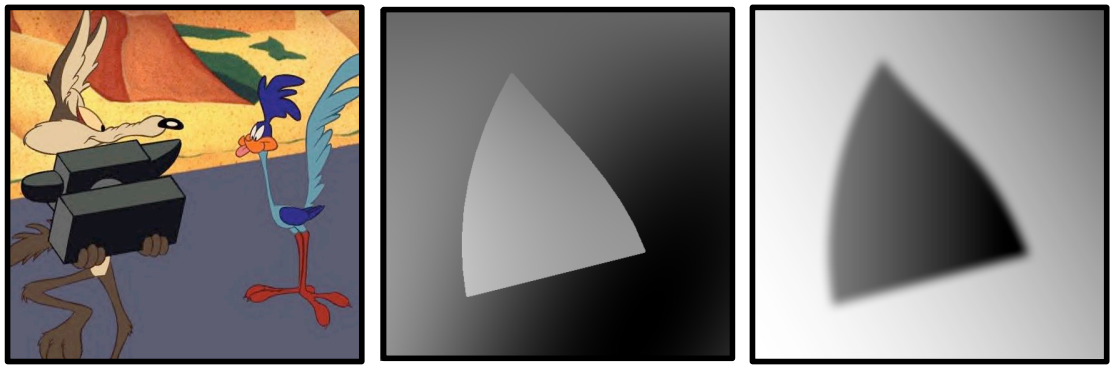


Figure 4.7: *Examples of cartoon images: sharp discontinuties (left and center) and smooth discontinuities (right).*

4.3 Efficient approximation

4.3.1 Decay of Approximation Error

To perform an efficient processing of signals or images in Θ, the goal is to design an orthogonal basis such that the non-linear approximation error $\|f - f_M\|$ decays as fast as possible to 0 when M increases.

Polynomial error decay. This error decay measured using a power law

$$\forall\, f \in \Theta,\ \forall\, M, \quad \|f - f_M\|^2 \leqslant C_f M^{-\alpha} \tag{4.15}$$

where α is independent of f and should be as large as possible. The parameter α depends on the basis and on Θ. It is a class-complexity parameter that describes the overall complexity of signals in Θ with respect to the orthogonal basis one considers for approximation. The parameter C_f depends on f and describes the complexity of the signal f within its class Θ.

Relevance for compression, denoising and inverse problems. Monitoring the decay of approximation error is not only interesting from a mathematical point of view. Section 4.4 shows that the compression error is close to the non-linear approximation error. Bases that are efficient for approximation are thus also efficient for compression.

Chapter 5 shows that a similar conclusion holds for non-linear denoising with thresholding. Efficient denoisers are obtained by performing a non-linear approximation of the noisy image in a well chosen basis. The average denoising error with respect to a random noise is closely related to the approximation error.

Chapter **??** shows that ill-posed inverse problems such as super-resolution of missing information can be solved by taking advantage of the compressibility of the signal or the image in a well chosen basis. A basis that is efficient for approximation of the high resolution signal is needed to recover efficiently missing information. The performance of these schemes is difficult to analyze, and the basis atoms must also be far enough from the kernel of the operator that removes information.

4.3.2 Comparison of Signals

For a fixed basis (for instance wavelets), the decay of $\|f - f_M\|$ allows one to compare the complexity of different images. Figure 4.9 shows that natural images with complicated geometric structures and textures are more difficult to approximate using wavelets.

Since the approximation error often decays in a power-low fashion (4.15), the curves are displayed in a log-log plot, so that

$$\log(\|f - f_M\|^2) = \text{cst} - \alpha \log(M)$$

and hence one should expect an affine curve with slope $-\alpha$. Due to discretization issue, this is only the case for value of $M \ll N$, since the error quickly drops to zero for $M \approx N$.

Smooth Cartoon Natural #1 Natural #2

Figure 4.8: *Several different test images.*

4.3.3 Comparison of Bases

For a given image f, one can compare different ortho-bases using the decay of $\|f - f_M\|$. Figure 4.11 shows the efficiency of several bases to approximate a fixed natural image with contours and textures. The Fourier basis described in Section 2.5 is highly innefficient because of periodic boundary artifact and the global support of its atoms that fail to capture contours. The cosine basis uses symmetric boundary conditions and thus removes the boundary artifacts, but it still not able to resolve efficiently localized features. The local DCT basis corresponds to the union of local cosine bases defined on small square patches. It is more efficient since its atoms have a local support. However, it gives bad approximation for a small number M of kept coefficients,

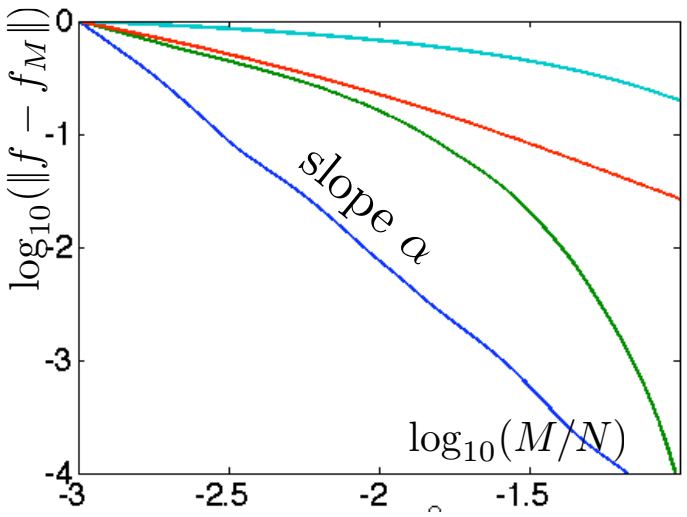


Figure 4.9: *Comparison of approximation error decay in wavelets for different images shown in Figure 4.8.*

because of blocking artifacts. The isotropic wavelet basis detailed in Section 3.3.3 gives the best approximation results because its is both composed of localized atoms and does not have a block structure but rather a multiresolution structure.

Fourier	Cosine	Local DCT	Wavelets
SNR=17.1dB	SNR=17.5dB	SNR=18.4dB	SNR=19.3dB

Figure 4.10: *Comparison of approximation errors for different bases using the same number $M = N/50$ of coefficients.*

4.4 Transform Coding

4.4.1 Coding

State of the art compression schemes correspond to transform coders, that code quantized coefficients in an ortho-basis. They first computes the coefficients of the decomposition of the signal into an well-chosen basis (for instance wavelets)

$$a[m] = \langle f, \psi_m \rangle \in \mathbb{R}.$$

Quantization corresponds to rounding the coefficients to an integer using a step size $T > 0$

$$q[m] = Q_T(a[m]) \in \mathbb{Z} \quad \text{where} \quad Q_T(x) = \text{sign}(x) \left\lfloor \frac{|x|}{T} \right\rfloor.$$

We note that this quantizer has a twice larger zero bin, so that coefficients in $[-T, T]$ are set to zero.

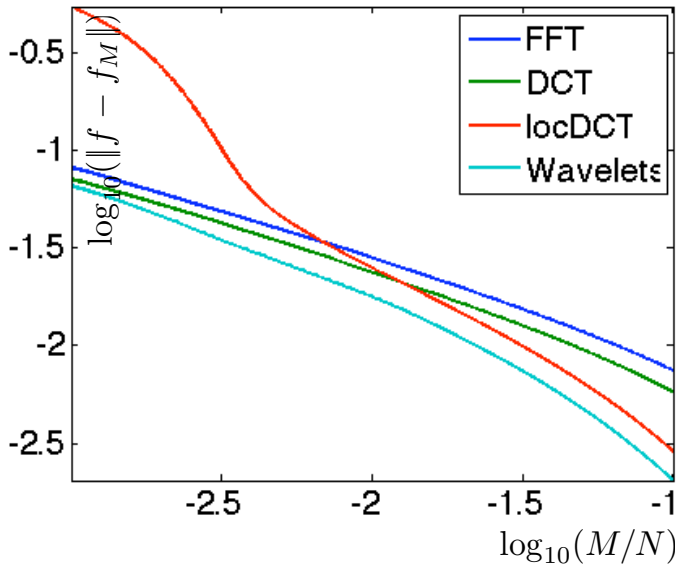


Figure 4.11: *Comparison of approximation error decay for different bases.*

This quantizer nonlinearity should be compared to the hard thresholding nonlinearity (4.4) used to perform non-linear approximation. The quantizer not only set to zero small coefficients that are smaller than T in magnitude, it also modifies larger coefficients by rounding, see Figure 4.12.

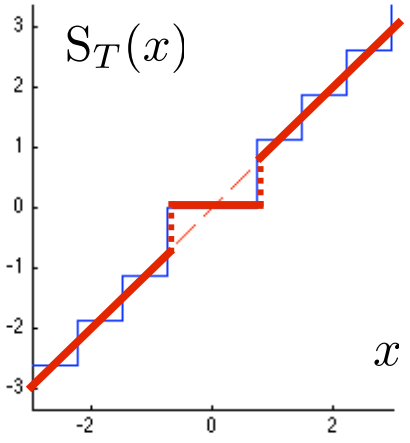


Figure 4.12: *Thresholding and quantization non-linearity mappings.*

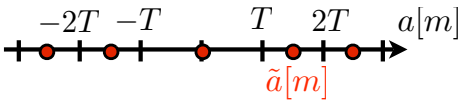
The resulting integer values $q[n]$ are stored into a binary file of length R, which corresponds to a number of bits. Sections 4.4.3 and 4.4.4 detail two different approach to perform this transformation from integer to bits. The goal is to reduce as much as possible the number R of bits.

4.4.2 De-coding

The decoder retrieves the quantized coefficients $q[m]$ from the binary file, and dequantizes the coefficients using

$$\tilde{a}[m] = \text{sign}(q[m]) \left(|q[m] + \frac{1}{2} \right) T. \tag{4.16}$$

This corresponds to retrieving the value from quantization at the center of quantization bins:



The compressed-decompressed image using R bits is then reconstructed as

$$f_R = \sum_{m \in I_T} \tilde{a}[m] \psi_m = \sum_{m \in I_T} Q_T(\langle f, \psi_m \rangle) \psi_m,$$

thus producing a decompression error $\|f - f_R\|$.

This decompression reconstruction (4.4.2) should be compared with the non-linear approximation formula (4.3). One necessarily has $\|f - f_M\| \leqslant \|f - f_R\|$, but in practice these two errors have comparable magnitudes. Indeed, the de-quantization formula (4.16) implies that for $|a[m]| > T$,

$$|a[m] - \tilde{a}[m]| \leqslant \frac{T}{2}.$$

One thus has

$$\|f - f_R\|^2 = \sum_m (a[m] - \tilde{a}[m])^2 \leqslant \sum_{|a[m]|<T} |a[m]|^2 + \sum_{|a[m]| \geqslant T} \left(\frac{T}{2}\right)^2 \tag{4.17}$$

$$\leqslant \|f - f_M\|^2 + MT^2/4 \tag{4.18}$$

where

$$M = \#\{m \setminus \tilde{a}[m] \neq 0\}.$$

4.4.3 Support Coding

To measure how large is the additional error term $MT^2/4$ in (4.17), one needs to choose a method to store the quantized coefficients $q[m]$ into a file.

For aggressive compression scenario, where R and M are small with respect to the size N of the image, the support

$$I_M = \{m \setminus \tilde{a}[m] \neq 0\}$$

is highly sparse. It thus make sense to code first this support and then to code the actual value $q[m] \neq 0$ for $m \in I_M$.

Signals constraints. To derive a connexion between the non-linear approximation error $\|f - f_M\|^2$ and the coding error $\|f - f_R\|^2$, we make several assumptions on the signal f. We first suppose that the coefficients $\langle f, \psi_m \rangle$ are highly sparse, by imposing a fast decay of the ordered coefficients d_m defined in (4.1)

$$d_m \sim m^{-\frac{\alpha+1}{2}}, \tag{4.19}$$

where $u_m \sim v_m$ means that there exists two constant $A, B > 0$ such that $Au_m \leqslant v_m \leqslant Bu_m$. We note that condition (4.19) implies in particular that $\|f - f_M\| \sim M^{-\alpha}$, so that the signals are well approximated using the basis $\{\psi_m\}_m$ if α is large.

Furthermore, a practical compression algorithm is only capable of dealing with discrete signals of size N. We thus considers that the algorithm has access to N inner products $\{\langle f, \psi_m \rangle\}_{0 \leqslant m < N}$ that are computed using a decomposition algorithm from a discretized signal or image of size N. For instance, Section 3.2 details a discrete wavelet transform, and introduces a compatibility condition (3.9) on the sampling operator for this inner product computation to be possible from discrete data.

For the compression from the discrete signals to be the same as a compression of a continuous signal, we impose that N is large enough so that

$$\forall\, m \geqslant N, \quad |\langle f, \psi_m \rangle| < T$$

so that the coefficients not quantized to 0 are contained within the set $\{\langle f, \psi_m \rangle\}_{0 \leqslant m < N}$ of the N computed coefficients.

The other hypothesis beyond (4.19) is that the sampling precision N is not too large, and in particular, that there is a polynomial grows of N with respect to the number M of coefficients to code

$$N \sim M^{\beta} \tag{4.20}$$

for some $\beta > 0$. For the wavelet and Fourier bases, and for all the classes of signals and images detailed in Section 4.2, one can show that this is indeed the case, if one orders the basis elements $\{\psi_m\}_m$ from low frequencies (or coarse scales) to high frequencies (or fine scales). This is because in these bases, the coefficients $\langle f, \psi_m \rangle$ have a polynomial decay with m if f as some smoothness. See Sections 4.5.2 and 4.6.1 for a proof of this decay for Sobolev and piecewise regular signals.

Support coding. Since the size of the support is $|I_M| = M$, one can code the entries of I_M as being a set of size M within a larger set of N coefficients using a number of bits

$$R_{\text{ind}} \leqslant \log_2 \binom{N}{M} = O(M \log_2 \left(\frac{N}{M}\right)) = O(M \log_2(M)) \tag{4.21}$$

where $\binom{N}{M}$ is the number of possible choices for M elements in a set of N elements, and where we have used hypothesis (4.19) to derive the last equality.

Values coding. The quantized values satisfy $q[m] \in \{-A, \ldots, A\}$, with

$$A \leqslant \frac{1}{T} \max_m |\langle f, \psi_m \rangle| = O(T^{-1}),$$

so one can code them using a number of bits

$$R_{\text{val}} = O(M |\log_2(T)|) = O(M \log_2(M)) \tag{4.22}$$

where we have used hypothesis (4.19) that implies $|\log_2(T)| \sim \log_2(M)$.

Total number of bits. Putting (4.21) and (4.22) together, the total number of bits for this support coding approach is thus

$$R = R_{\text{ind}} + R_{\text{val}} = O(M \log_2(M)).$$

Inverting this relationship proves that

$$M = O(R \log_2(R)). \tag{4.23}$$

Rate/distortion error decay. Condition (4.19) implies that $MT^2 \sim M^{-\alpha}$, so that putting (4.17) and (4.23) together proves

$$\|f - f_R\|^2 = O(\log^{\alpha}(R) R^{-\alpha}).$$

This shows the importance of the study of non-linear approximation, and in particular the design of bases that are efficient for approximation of a given signal model Θ.

4.4.4 Entropic Coding

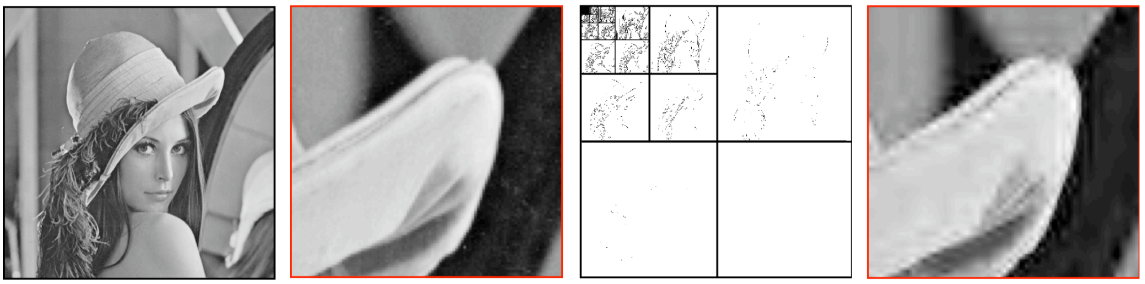
To further reduce the file size R (in bits), one can use an entropic coder to transform the integer values $q[m]$ into bits. Such coding scheme makes use of the statistical redundancy of the quantized values, that have a large number of zero entries and very few large entries. The theory of coding was formalized by Shannon [28].

Probabilistic modeling. The quantized coefficients $q[m] \in \{-A, \ldots, A\}$ are assumed to take values in an alphabet of $Q = 2A + 1$ elements. A coding scheme performs the transformation

$$\{q[m]\}_m \longmapsto \{0, 1, 1, \ldots, 0, 1\} \in \{0, 1\}^R.$$

To reduce the average value of R, one makes use of a statistical model, which assumes that the $q[m]$ are drawn independently at random from a known probability distribution

$$\mathbb{P}(q[m] = i) = p_i \in [0, 1].$$



Original f | Zoom | Wavelet support | Compressed f_R

Figure 4.13: *Image compression using wavelet support codding.*

Huffman code. A Huffman code is a code with variable length, since it perform a mapping from symbols to binary strings

$$q[m] = i \in \{-A, \ldots, A\} \longmapsto c_i \in \{0,1\}^{|c_i|}$$

where $|c_i|$ is the length of the binary code word c_i, that should be larger if p_i is small. A Huffman tree algorithm is able to build a code such that

$$|c_i| \leqslant \lceil \log_2(p_i) \rceil$$

so that

$$R \leqslant (\mathcal{E}(p) + 1)N$$

where $\mathcal{E}$ is the entropy of the distribution, defined as

$$\mathcal{E}(p) = -\sum_i p_i \log_2(p_i).$$

Figure 4.14 shows different probability distribution. The entropy is small for highly sparse distribution. Wavelet coefficients of natural images tend to have a low entropy because many coefficients are small.

The Huffman scheme codes symbols independently, leading to a sub-optimal code if some of the p_i are large, which is usually the case for wavelet coefficients. One usually prefers arithmetic coding schemes, that codes groups of symbols, and are able to get close to the entropy bound $R \approx \mathcal{E}(p)N$ for large N.

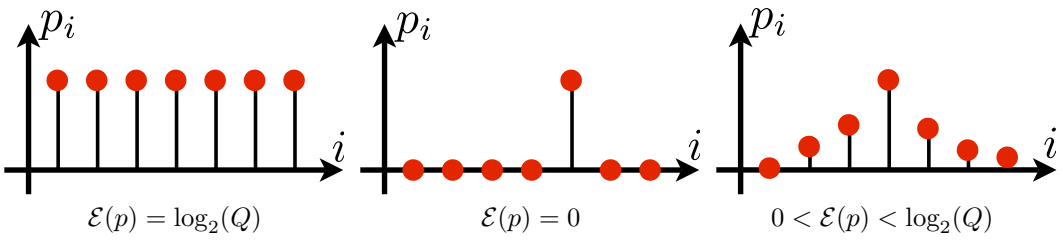


Figure 4.14: *Three different probability distributions.*

4.4.5 JPEG-2000

JPEG-2000 is the latest still image compression standard. It corresponds to a wavelet transform coder that performs a clever adaptive entropy coding that makes use of the statistical redundancy of wavelet coefficients of natural images. The wavelet transform is not orthogonal, it is a symmetric 7/9 biorthogonal, with symmetric boundary condition and a lifting implementation. This transform is however close to orthogonality, so that the previous discussion about orthogonal approximation and coding is still relevant.

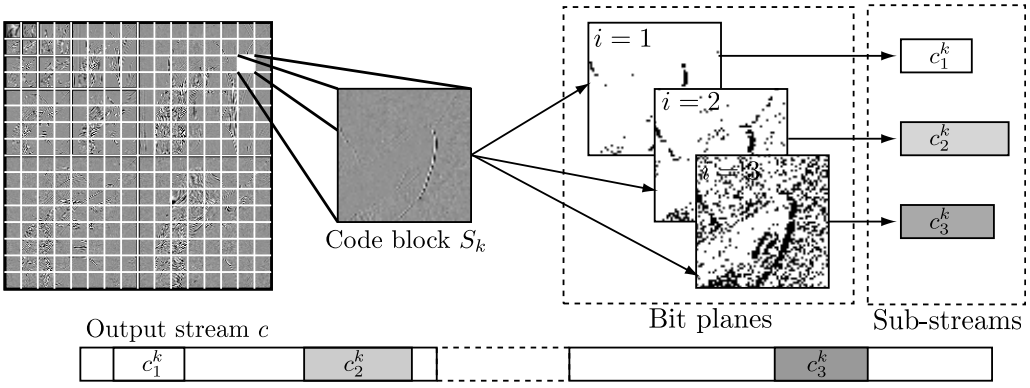


Figure 4.15: *JPEG-2000 coding architecture.*

Figure 4.15 shows an overview of JPEG-2000 architecture. Figure 4.16 shows a comparison between JPEG and JPEG-2000 compressors. JPEG is based on a local DCT transform, and suffers from blocking artifacts at low bit rates, which is not the case of JPEG-2000. This new standard also comes with several important features, such as regions of interest, which allows to refine the coding in some specific parts of the image.

Figure 4.16: *Comparison of JPEG (left) and JPEG-2000 (right) coding.*

Dyadic quantization. The wavelet coefficients are quantized with a varying quantization step $T_i = 2^{-i}T_0$. This allows one to progressively increase the precision of the coded coefficients. For each i, a bit plane coding pass produces new bits to refine the value of the coefficients when i increases.

Steam packing. The bits obtained using the bit plane pass with quantizer $T_i = 2^{-i}T_0$ are entropy coded using a contextual coder. This coder processes square blocks S_k of coefficients. This local coding enhances the parallelization of the method. This local block coding produces a bit stream c_i^k, and these streams are optimally packed into the final coded file to reduce the distortion $\|f - f_R\|$ for almost every possible number R of bits. This stream packing ensures scalability of the output bit stream. It means that one can receive only the R first bits of a large coded file and get a low resolution decoded image f_R that has an almost minimal distortion $\|f - f_R\|$.

Bit plane coding pass. For each threshold T_i, for each scale and orientation $(j, \omega \in \{V, H, D\})$, for each coefficient location $n \in S_k$, JPEG-2000 coder encodes several bit reflecting the value of the wavelet coefficient $d_j^\omega[n]$. In the following we drop the dependancy on (j, ω) for simplicity.

- If $d_j^\omega[n] < T_{i-1}$, the coefficient was not significant at bit-plane $i-1$. It thus encodes a significance bit $b_i^1[n]$ to tell wether $d_j^\omega[n] \geqslant T_i$ or not.
- If $b_i^1[n] = 1$, meaning that the coefficient has became significant, it codes it sign as a bit $b_i^2[n]$.
- For every position n that was previously significant, meaning $d_j^\omega[n] \geqslant T_{i-1}$, it codes a value refinement bit $b_i^3[n]$ to tell wether $d_j^\omega[n] \geqslant T_i$ or not.

Contextual coder. The final bits streams c_i^k are computed from the produced bits $\{b_i^s[n]\}_{s=1}^3$ for $n \in S_k$ using a contextual coder. The contextual coding makes use of spacial redundancies in wavelet coefficients, especially near edges and geometric singularities that create clusters of large coefficients. The coefficients $n \in S_k$ are traversed in zig-zag order as shown on Figure 4.17.

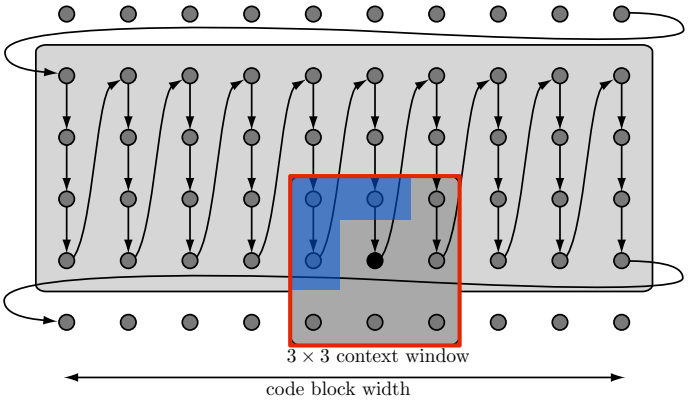


Figure 4.17: *Coding order and context for JPEG-2000 coding.*

For each coefficient location $n \in S_k$, the context value $v_i^s[n]$ of the bit $b_i^s[n]$ to code at position x is an integer computed over a 3×3 window

$$w_n = \{(n_1 + \varepsilon_1, n_2 + \varepsilon_2)\}_{\varepsilon_i = \pm 1}.$$

This local context $v_i^s[n]$ integrates in a complicated way the previous bit plane values $\{b_{i-1}^s[\tilde{n}]\}_{\tilde{n} \in w_n}$, and neighboring bits at plane $\{b_i^s[\tilde{n}]\}_{\tilde{n} \in w_n, \tilde{n} \text{ coded}}$ that have already been coded.

The bit value $b_i^s[n]$ is then coded with an arithmetic coding by making use of the conditional probability distribution $\mathbb{P}(b_i^s[n]|v_i^s[n])$. The choice made for the computation $v_i^s[n]$ allows to reduce significantly the entropy of this conditional probability condition with respect to the original distribution $\mathbb{P}(b_i^s[n])$, thus reducing the overall number of bits.

4.5 Fourier Approximation of Smooth Functions

The smooth signal and image model (4.5) assumed that the analog function have bounded α continuous derivatives. A function f with a large α has more smoothness, and is thus simpler to approximate. Figure 4.5 shows images with increasing smoothness.

4.5.1 1D Fourier Approximation

A 1D signal $f \in \mathrm{L}^2([0,1])$ is associated to a 1-periodic function $f(t+1) = f(t)$ defined for $t \in \mathbb{R}$.

Low pass approximation. We consider a linear Fourier approximation, that only retains low frequencies

$$f_M^{\text{lin}} = \sum_{m=-M/2}^{M/2} \langle f, e_m \rangle e_m$$

where we use the 1D Fourier atoms

$$\forall\, n \in \mathbb{Z}, \quad e_m(t) = e^{2i\pi m t}.$$

We note that f_M actually requires $M+1$ Fourier atoms.

Figure 4.18 shows examples of such linear approximation for an increasing value of M. Since the original function f is singular (no derivative), this produces a large error and one observe ringing artifacts near singularities. Code 7 implement this low pass linear approximation and code 8 implement the non-linear approximation.

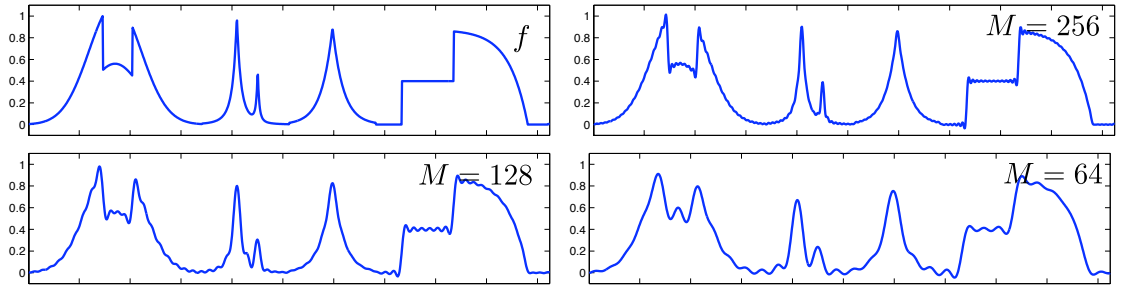


Figure 4.18: *Fourier approximation of a signal.*

```
F = fftshift(fft2(f));
n = size(f,1);
F1 = zeros(n);
sel = (n/2-m:n/2+m)+1;
F1(sel,sel) = F(sel,sel);
f1 = real( ifft2(fftshift(F1)) );
```

Matlab code 7: *2D Fourier linear approximation. Input: image* `f`*, parameter* `m`*, output: approximation* `f1`*. The number of coefficients is* $M = (2\mathtt{m}+1)^2$.

```
F = fft2(f);
a = sort(abs(F(:))); a = a(end-1:1);
T = a(M+1);
F = F .* (abs(F)>T);
f1 = real( ifft2(F) );
```

Matlab code 8: *2D Fourier linear approximation. Input: image* `f` *and number of coefficients* `M`*, output: approximation* `f1`.

Low pass approximation and filtering. This low pass approximation corresponds to a filtering, since

$$f_M = \sum_{m=-M/2}^{M/2} \langle f, e_m \rangle e_m = f \star h_M \quad \text{where} \quad \hat{h}[k] = 1_{[-M/2,M/2]}[k].$$

Decay of Fourier coefficients. Using integration by part, one shows that for a C^α function f in the smooth signal model (4.5), one has

$$|\langle f,\, e_m\rangle| \leqslant |2\pi m|^{-\alpha}\|f^{(\alpha)}\|.$$

This implies a fast decay of the linear approximation error

$$\|f - f_M^{\mathrm{lin}}\| = \sum_{|m|>M/2} |\langle f,\, e_m\rangle|^2 \leqslant \|f^{(\alpha)}\| \sum_{|m|>M/2} |2\pi m|^{-2\alpha} = O(M^{-2\alpha+1}).$$

We show next that a more precise estimation caries over for the larger class of Sobolev signals.

4.5.2 Sobolev Signal Approximation

For signal f in the Sobolev model (4.8), one has

$$\|f^{(\alpha)}\|^2 = \sum_m |2\pi m|^{2\alpha}|\langle f,\, e_m\rangle|^2 \geqslant \sum_{|m|>M/2} |2\pi m|^{2\alpha}|\langle f,\, e_m\rangle|^2 \tag{4.24}$$

$$\geqslant (\pi M)^{2\alpha} \sum_{m>M/2} |\langle f,\, e_m\rangle|^2 \geqslant (\pi M)^{2\alpha}\|f - f_M^{\mathrm{lin}}\|^2. \tag{4.25}$$

This shows that the linear approximation error of a 1D Sobolev smooth signal satisfies

$$\|f - f_M^{\mathrm{lin}}\|^2 \leqslant C\|f^{(\alpha)}\|^2 M^{-2\alpha}. \tag{4.26}$$

One can also shows that this asymptotic error decay is optimal, and that the non-linear approximation error in Fourier also decays like $O(M^{-2\alpha})$.

For a signal in the piecewise smooth model (4.9), such as the one shows in Figure 4.18, one only has a slow decay of the linear and non-linear approximation error

$$\|f - f_M\|^2 \leqslant C_f M^{-1} \tag{4.27}$$

and Fourier atoms are not anymore optimal for approximation.

4.5.3 Sobolev Images

This analysis caries over to images and higher dimensional datasets by considering a Sobolev functional (4.7) for $d > 1$.

The linear and non-linear approximation of an α-regular Sobolev image then satisfy

$$\|f - f_M\|^2 \leqslant C\|f^\alpha\|^2 M^{-\alpha}.$$

For d-dimensional data $f : [0,1]^d \to \mathbb{R}$, one would have an error decay of $O(M^{-2\alpha/d})$.

For an image in the piecewise smooth model (4.10), the linear and non-linear error decays are slow,

$$\|f - f_M\|^2 \leqslant C_f M^{-1/2}, \tag{4.28}$$

and Fourier atoms are not anymore optimal for approximation.

4.6 Wavelet Approximation of Piecewise Smooth Functions

Wavelet approximation improve significantly over Fourier approximation to capture singularities. This is due to the localized support of wavelets.

Figure 4.19: *Linear (top row) and non-linear (bottom row) Fourier approximation.*

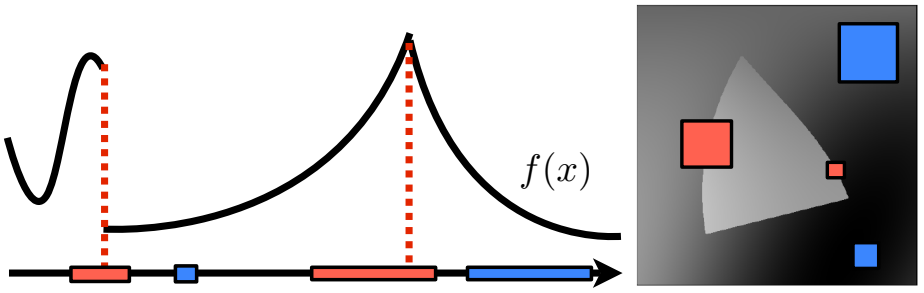


Figure 4.20: *Singular par of signals (left) and image (right).*

4.6.1 Decay of Wavelet Coefficients

To efficiently approximate regular parts of signals and images, one uses wavelet with a large enough number p of vanishing moments

$$\forall\, k < p, \quad \int \psi(x) x^k \mathrm{d}x = 0.$$

This number p should be larger than the regularity α of the signal outside singularities (for instance jumps or kinks).

To quantify the approximation error decay for piecewise smooth signals (4.9) and images (4.10), one needs to treat differently wavelets that are in regular and singular areas. Figure 4.20 shows for a signal and an image the localization of singular and regular parts.

If f is C^α on $\mathrm{supp}(\psi_{j,n})$, with $p \geqslant \alpha$, then one can perform a Taylor expansion of f around the point $2^j n$

$$f(x) = P(x - 2^j n) + R(x - 2^j n) = P(2^j t) + R(2^j t)$$

where $\deg(P) < \alpha$ and

$$|R(x)| \leqslant C_f \|x\|^\alpha.$$

One then bounds the wavelet coefficient

$$\langle f,\, \psi_{j,n} \rangle = \frac{1}{2^{j\frac{d}{2}}} \int f(x) \psi\left(\frac{x - 2^j n}{2^j}\right) \mathrm{d}x = 2^{j\frac{d}{2}} \int R(2^j t) \psi(t) \mathrm{d}t$$

where we have performed the change of variable $t = \frac{x - 2^j n}{2^j}$.

This shows that if $\mathrm{supp}(\psi_{j,n})$ does not contain a singularity of the signal ($d = 1$) or image ($d = 2$) f, one has

$$|\langle f, \psi_{j,n}\rangle| \leqslant C_f \|\psi\|_1 2^{j(\alpha+d/2)}. \tag{4.29}$$

For wavelets that are too close to a singularity, one can only use the fact that f is bounded, so that

$$|\langle f, \psi_{j,n}\rangle| \leqslant \|f\|_\infty \|\psi\|_1 2^{j\frac{d}{2}}.$$

4.6.2 1D Piecewise Smooth Approximation

For 1D signal in the piecewise regular model (4.9), large wavelets coefficients $\langle f, \psi_{j,n}\rangle$ are clustered around the singularities of the signal. We call $\mathcal{S} \subset [0,1]$ the finite set of singular points.

Coefficient segmentation. The singular support at scale 2^j is the set of coefficients corresponding to wavelets that are crossing a singularity

$$\mathcal{C}_j = \{n \setminus \mathrm{supp}(\psi_{j,n}) \cap \mathcal{S} \neq \emptyset\} \tag{4.30}$$

It has a constant size because of the dyadic translation of wavelets

$$|\mathcal{C}_j| \leqslant K|\mathcal{S}| = \text{constant}.$$

Using (4.29) for $d = 1$, the decay of regular coefficients is bounded as

$$\forall\, n \in \mathcal{C}_j^c, \quad |\langle f, \psi_{j,n}\rangle| \leqslant C 2^{j(\alpha+1/2)}.$$

Using (4.6.1) for $d = 1$, the decay of singular coefficients is bounded as

$$\forall\, n \in \mathcal{C}_j, \quad |\langle f, \psi_{j,n}\rangle| \leqslant C 2^{j/2}.$$

Once a fixed threshold T is fixed to compute the non-linear approximation, one defines cut-off scales for regular and singular coefficients that depend on T

$$2^{j_1} = (T/C)^{\frac{1}{\alpha+1/2}} \quad \text{and} \quad 2^{j_2} = (T/C)^2.$$

Figure 4.21 shows a schematic segmentation of the set of wavelet coefficients into regular and singular parts, and also using the cut-off scales.

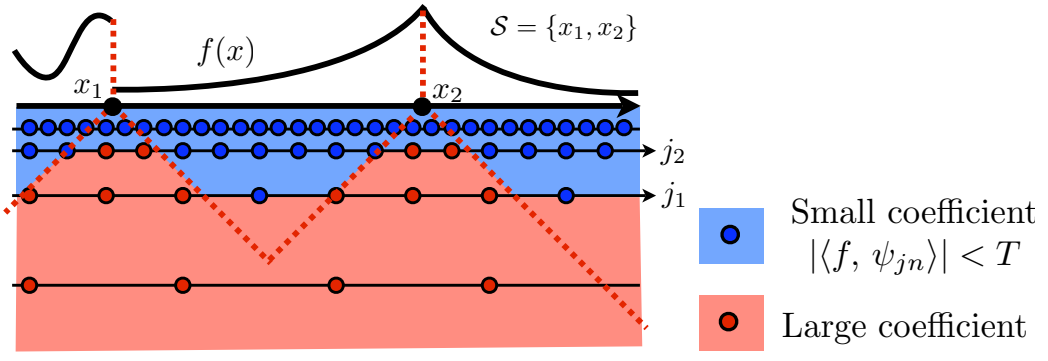


Figure 4.21: *Segmentation of the wavelet coefficients into regular and singular parts.*

Counting the error. These cut-off scales allow us to define a hand-crafted approximation signal

$$\tilde{f}_M = \sum_{j \geqslant j_2} \sum_{n \in \mathcal{C}_j} \langle f, \psi_{j,n} \rangle \psi_{j,n} + \sum_{j \geqslant j_1} \sum_{n \in \mathcal{C}_j^c} \langle f, \psi_{j,n} \rangle \psi_{j,n}. \tag{4.31}$$

The approximation error generated by this M-term approximation $\tilde{f}_M$ is larger than the best M-term approximation error, and hence

$$\|f - f_M\|^2 \leqslant \|f - \tilde{f}_M\|^2 \leqslant \sum_{j < j_2, n \in \mathcal{C}_j} |\langle f, \psi_{j,n} \rangle|^2 + \sum_{j < j_1, n \in \mathcal{C}_j^c} |\langle f, \psi_{j,n} \rangle|^2 \tag{4.32}$$

$$\leqslant \sum_{j < j_2} (K|\mathcal{S}|) \times C^2 2^j + \sum_{j < j_1} 2^{-j} \times C^2 2^{j(2\alpha+1)} \tag{4.33}$$

$$= O(2^{j_2} + 2^{2\alpha j_1}) = O(T^2 + T^{\frac{2\alpha}{\alpha+1/2}}) = O(T^{\frac{2\alpha}{\alpha+1/2}}). \tag{4.34}$$

Counting the number of measurements. The number of coefficients needed to build the approximating signal $\tilde{f}_M$ is

$$M \leqslant \sum_{j \geqslant j_2} |\mathcal{C}_j| + \sum_{j \geqslant j_1} |\mathcal{C}_j^c| \leqslant \sum_{j \geqslant j_2} K|\mathcal{S}| + \sum_{j \geqslant j_1} 2^{-j} \tag{4.35}$$

$$= O(|\log(T)| + T^{\frac{-1}{\alpha+1/2}}) = O(T^{\frac{-1}{\alpha+1/2}}). \tag{4.36}$$

Putting everything together. Putting equations (4.32) and (4.35) together, one obtains that if f is in the piecewise smooth signal model (4.9), the non-linear approximation error in wavelet obeys

$$\|f - f_M\|^2 = O(M^{-2\alpha}). \tag{4.37}$$

This improves significantly over the $O(M^{-1})$ decay of Fourier approximation (4.27). Furthermore, this decay is the same as the error decay of uniformly smooth signal (4.26). In 1D, wavelet approximations do not "see" the singularities. The error decay (4.37) can be shown to be asymptotically optimal.

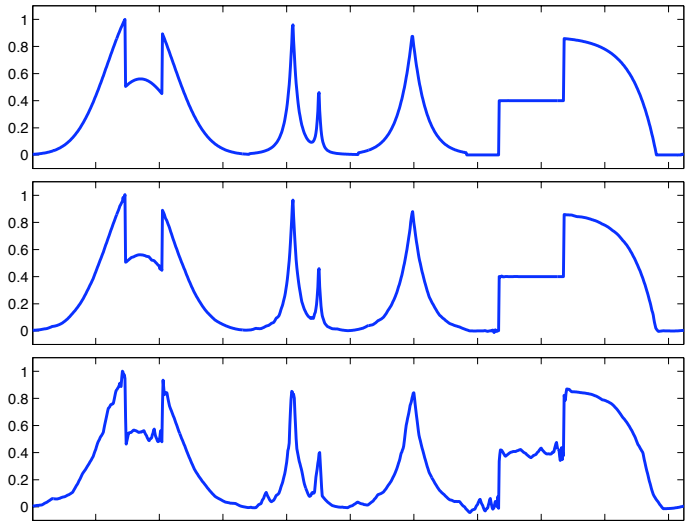


Figure 4.22: *1D wavelet approximation.*

Figure 4.22 shows examples of wavelet approximation of singular signals.

4.6.3 2D Piecewise Smooth Approximation

For an image in the piecewise smooth model (4.10), we define the singular support $\mathcal{C}_j$ as in (4.30). The major difference with the 1D setting, is that for 2D images, the size of the singular support grows when the scale 2^j goes to zero

$$|\mathcal{C}_j^\omega| \leqslant 2^{-j} K |\mathcal{S}|,$$

where $|\mathcal{S}|$ is the perimeter of the singular curves $\mathcal{S}$, and $\omega \in \{V, H, D\}$ is the wavelet orientation. Using (4.29) for $d = 2$, the decay of regular coefficients is bounded as

$$\forall\, n \in (\mathcal{C}_j^\omega)^c, \quad |\langle f,\, \psi_{j,n}^\omega \rangle| \leqslant C 2^{j(\alpha+1)}.$$

Using (4.6.1) for $d = 2$, the decay of singular coefficients is bounded as

$$\forall\, n \in \mathcal{C}_j^\omega, \quad |\langle f,\, \psi_{j,n}^\omega \rangle| \leqslant C 2^j.$$

After fixing T, the cut-off scales are defined as

$$2^{j_1} = (T/C)^{\frac{1}{\alpha+1}} \quad \text{and} \quad 2^{j_2} = T/C.$$

We define similarly to (4.31) a hand-made approximation. Similarly to (4.32), we bound the approximation error as

$$\|f - f_M\|^2 \leqslant \|f - \tilde{f}_M\|^2 = O(2^{j_2} + 2^{2\alpha j_1}) = O(T + T^{\frac{2\alpha}{\alpha+1}}) = O(T)$$

and the number of coefficients as

$$M = O(T^{-1} + T^{\frac{-1}{\alpha+1}}) = O(T^{-1}).$$

This leads to a decay of the non-linear wavelet approximation error

$$\|f - f_M\|^2 = O(M^{-1}). \tag{4.38}$$

This improves significantly over the $O(M^{-1/2})$ decay of Fourier approximation (4.28). This result is however deceiving, since it does not take advantage of the C^α regularity of the image outside the edge curves.

This error decay is still valid for the more general model of images with bounded variations (4.11). One can shows that wavelets are asymptotically optimal to approximate images with bounded variations.

Figure 4.23: *2D wavelet approximation.*

Figure 4.23 shows wavelet approximations of a bounded variation image. Code 9 implements the linear wavelet approximation and code 10 implements the non-linear approximation.

```
fw = perform_wavelet_transf(f,j0,+1);
n = size(f,1); fw1 = zeros(n);
fw1(1:n/4,1:n/4) = fw(1:m,1:m);
f1 = perform_wavelet_transf(fw1,j0,-1);
```

Matlab code 9: *2D wavelets linear approximation. Input: image* `f`*, parameter* `m`*, coarse scale* `j0`*, output: approximation* `f1`*. The number of coefficients is* $M = \texttt{m}^2$.

```
fw = perform_wavelet_transf(f,j0,+1);
a = sort(abs(fw(:))); a = a(end:-1:1);
T = a(M+1);
fw1 = fw .* (abs(fw)>T);
f1 = perform_wavelet_transf(fw1,j0,-1);
```

Matlab code 10: *2D wavelets linear approximation. Input: image* `f` *and number of coefficients* `M`*, coarse scale* `j0`*, output: approximation* `f1`.

4.7 Cartoon Images Approximation

The square support of wavelet makes them inefficient to approximate geometric images (4.13), whose edges are more regular than the level set of bounded variation images (4.11), which are only assumed to be of finite length.

4.7.1 Wavelet Approximation of Cartoon Images

Result (4.38) shows that wavelet approximation of images in the cartoon models (4.13) decays at least like $O(M^{-1})$. One can show that simple cartoon images like $f = 1_\Omega$ where Ω is a disk reach this low decay speed. This is because the square support of wavelets forbid them to take advantage of the regularity of edge curves. The approximation error for the smoothed cartoon model (4.14) is also slow if the width of the blurring kernel is small with respect to the number M of coefficients.

Figure 4.24 shows that many large coefficients are located near edge curves, and retaining only a small number leads to a bad approximation with visually unpleasant artifacts.

Figure 4.24: *Wavelet approximation of a cartoon image.*

4.7.2 Finite Element Approximation

To improve over the wavelet approximation, one can design an adapted triangulation that is highly anisotropic near edges. Figure 4.25 shows an example of such a triangulation.

A triangulation is obtain by sampling M points over the image domain $[0,1]^2$ and then connecting them using triangles. One then defines a piecewise linear interpolation $\tilde{f}_M$ over these triangles.

As shown in Figure 4.26, an efficient approximation of a C^2-cartoon image (4.13) for $\alpha = 2$ is obtained by seeding $\approx M/2$ approximately equilateral triangles of width $\approx M^{-1/2}$ in the areas

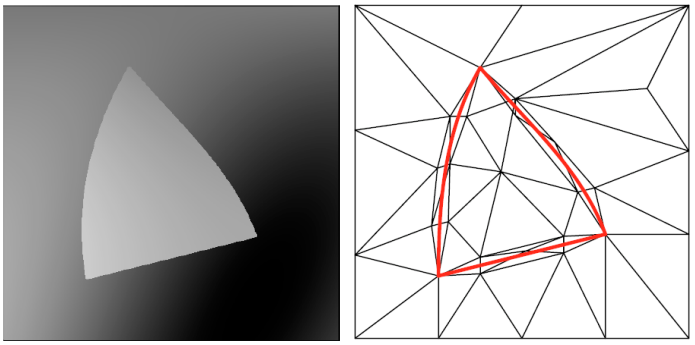


Figure 4.25: *Left: cartoon image, right: adaptive triangulation.*

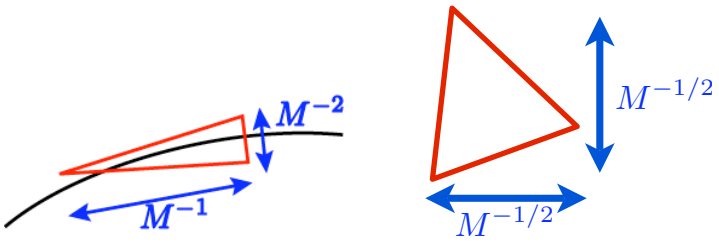


Figure 4.26: *Aspect ratio of triangle away from edges (left) and near an edge (right).*

where the image is regular. Near the edges, using the C^2 regularity of the singular curve, one can seed $\approx M/2$ anisotropic triangles of length M^{-1} and width $\approx M^{-1/2}$. One can show that such an adaptive triangulation leads to an approximation error

$$\|f - f_M\|^2 = O(M^{-2}), \tag{4.39}$$

which improves over the wavelet approximation error decay (4.38).

This scheme is however difficult to implement in practice, since the edge curves are not known and difficult to find. This is in particular the case for smooth cartoon images when the smoothing kernel h is unknown.

There is currrently no known algorithm that can automatically produces the error decay (4.39). One thus has to use heuristics and greedy algorithm to find the location of the sampling points and computes the triangles. Figure (4.27) shows an example of compression using such a greedy seeding algorithm, that works well in practice.

Adpated triangulation $\tilde{f}_M$ JPEG-2000

Figure 4.27: *Comparison of adaptive triangulation and JPEG-2000, with the same number of bits.*

4.7.3 Curvelets Approximation

Instead of using an adaptive approximation scheme such as finite elements, one can replace the wavelet basis by a set of oriented anisotropic atoms. The curvelet frame was proposed by Candès and Donoho for this purpose [3].

Curvelets. The curvelet construction starts from a curvelet function c that is oriented along the horizontal direction, and perform stretching

$$c_{2^j}(x_1, x_2) \approx 2^{-3j/4} c(2^{-j/2} x_1, 2^{-j} x_2),$$

translation and rotation

$$c^{\theta}_{2^j,u}(x_1, x_2) = c_{2^j}(R_\theta(x - u))$$

where R_θ is the rotation of angle θ.

The atoms $c^{\theta}_{2^j,u}$ is located near u, with an orientation θ, and has an aspect ratio "width $\approx$ length^2", which is the same aspect used to build an adaptive finite element approximation. This aspect ratio is essential to capture the anisotropic regularity near edges for images in the cartoon model (4.13) for $\alpha = 2$.

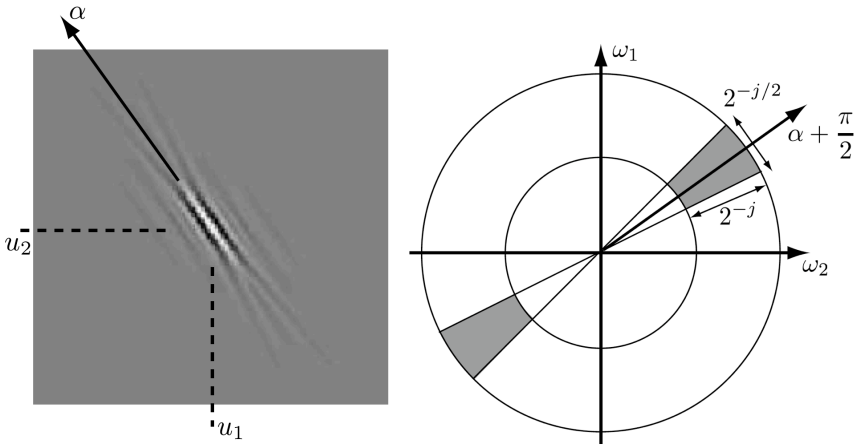


Figure 4.28: *Left: a curvelet c_m, right: its Fourier transform localization.*

Figure 4.29 shows the spacial and frequency localization of curvelets.

Parameter discretization. To build an image representation, one need to sample the u and θ parameter. To maintain a stable representation, the sub-sampling of the angles depends on the scale

$$\forall\, 0 \leqslant k < 2^{-\lceil j/2 \rceil + 2}, \quad \theta^{(j)}_k = k\pi 2^{\lceil j/2 \rceil - 1}$$

and the spacial grid depends on the scale and on the angles

$$\forall\, m = (m_1, m_2) \in \mathbb{Z}^2, \quad u^{(j,\theta)}_m = R_\theta(2^{j/2} m_1, 2^j m_2).$$

Figure 4.29 shows this sampling pattern.

Curvelet tight frame. This sampling leads to a stable redundant family

$$C_{j,m,k}(x) = c^{\theta}_{2^j,u}(x) \quad \text{where} \quad \theta = \theta^{(j)}_k \quad \text{and} \quad u = u^{(j,\theta)}_m,$$

that obeys a conservation of energy

$$\|f\|^2 = \sum_{j \in \mathbb{Z}} \sum_{k=0}^{2^{-\lceil j/2 \rceil + 2}} \sum_{m \in \mathbb{Z}^2} |\langle f,\, C_{j,m,k} \rangle|^2$$

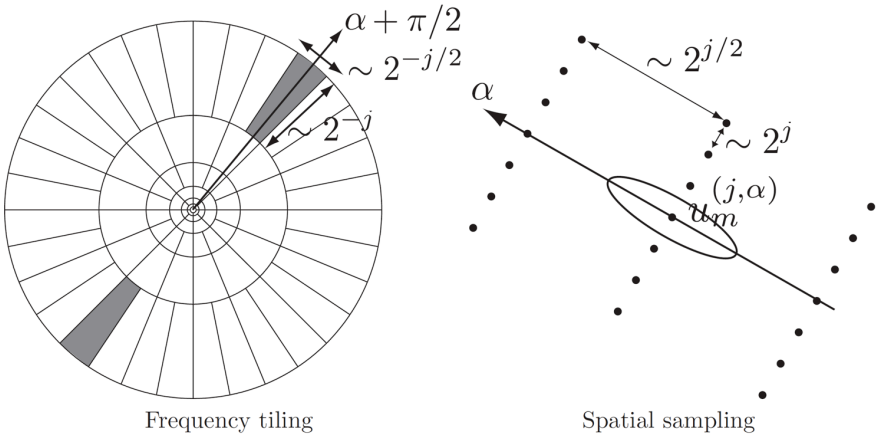


Figure 4.29: *Sampling pattern for the curvelet positions.*

and a reconstruction formula

$$f = \sum_{j \in \mathbb{Z}} \sum_{k=0}^{2^{-\lceil j/2 \rceil + 2}} \sum_{m \in \mathbb{Z}^2} \langle f, C_{j,m,k} \rangle C_{j,m,k}$$

that extends the properties of orthogonal basis (tight frame), although the representation is redundant (the atoms are not orthogonal).

A numerical implementation of this tight frame also defines a discrete tight frame for image of N pixels, that is made of $\approx 5N$ atoms [4].

Curvelet approximation. A non-linear M-term approximation in curvelets is defined as

$$f_M = \sum_{|\langle f, C_{j,m,k} \rangle| > T} \langle f, C_{j,m,k} \rangle C_{j,m,k}$$

where T is a threshold that depends on M. One should note that f_M is not necessarily the best M-term curvelet approximation since the curvelet frame is not orthogonal.

For position $u_m^{(j,\theta)}$ that are far away from an edges, the vanishing moments of the curvelets create a small coefficient $\langle f, C_{j,m,k} \rangle$. If $u_m^{(j,\theta)}$ is close to an edge curve whose tangent has direction $\tilde{\theta}$, then the coefficient $\langle f, C_{j,m,k} \rangle$ decays very fast to zero when $|\theta - \tilde{\theta}|$ increases. Figure 4.30 shows the principle of this curvelet approximation, and compares it with directional wavelets that have a square support.

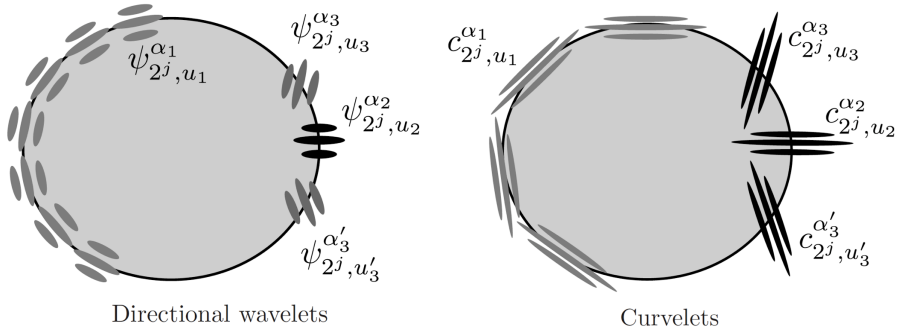


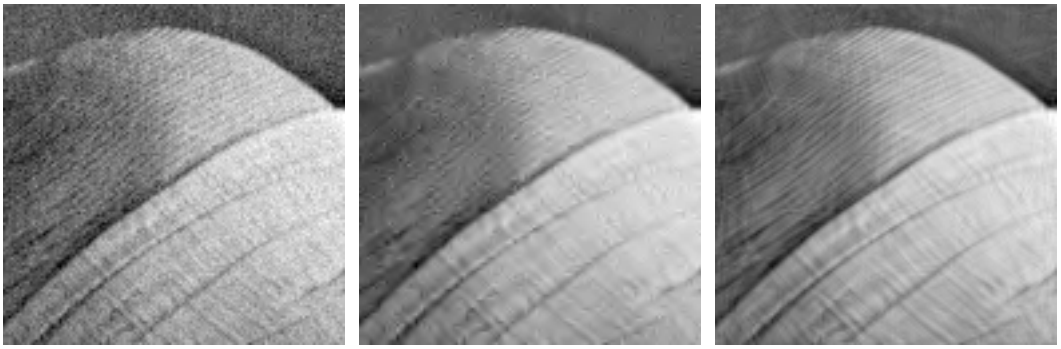
Figure 4.30: *Comparison of the principle of wavelets (left) and curvelet (right) approximations of a cartoon image.*

Using these two properties together with the sparse sampling of the curvelet in space and orientation leads to the following approximation error decay

$$\|f - f_M\|^2 = O(\log^3(M) M^{-2})$$

for image in the cartoon model (4.13) for $\alpha = 2$. This is close to the decay of adaptive triangulations (4.39), but this time one computes f_M with a fast $O(N \log(N))$ algorithm for an image of N pixels.

In practice, the redundancy of the curvelet frame makes it not suitable for image compression. Its efficiency is however useful for denoising purpose, where it can improve over wavelet to denoise geometric images and textures, see Figure 4.31. The result is obtained by using a thresholding denoiser as detailed in Section 5.3.1.



Noisy f Wavelet $\tilde{f}$ Curvelets $\tilde{f}$

Figure 4.31: *Comparison of wavelets (translation invariant) and curvelet denoising.*

Chapter 5

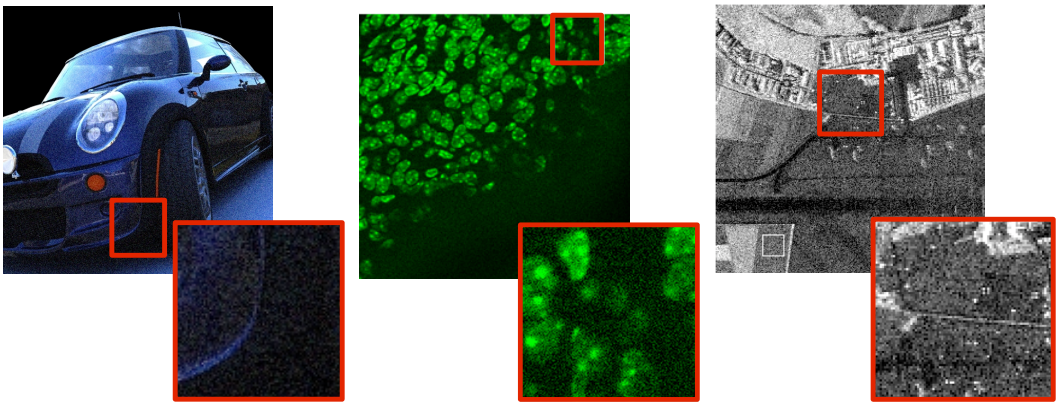
Denoising with Thresholding

Together with compression, denoising is the most important processing application, that is pervasive in almost any signal or image processing pipeline. Indeed, data acquisition always comes with some kind of noise, so modeling this noise and removing it efficiently is crucial.

5.1 Noise Modeling

5.1.1 Noise in Images

Image acquisition devices always produce some noise. Figure 5.1 shows images produced by different hardware, where the regularity of the underlying signal and the statistics of the noise is very different.



Digital camera Confocal imaging SAR imaging

Figure 5.1: *Example of noise in different imaging device.*

One should thus model both the acquisition process and the statistics of the noise to fit the imaging process. Then one should also model the regularity and geometry of the clean signal to choose a basis adapted to its representation. This chapter describes how thresholding methods can be used to perform denoising in some specific situations where the noise statistics are close to being Gaussian and the mixing operator is a sum or can be approximated by a sum.

In the following, we consider only finite dimensional signal $f \in \mathbb{C}^N$.

5.1.2 Image Formation

Figure 5.2 shows an idealized view of the image formation process, that mixes a clean image f_0 with a noise w to obtain noisy observations $f = f_0 \oplus w$, where $\oplus$ might for instance be a sum or a multiplication.

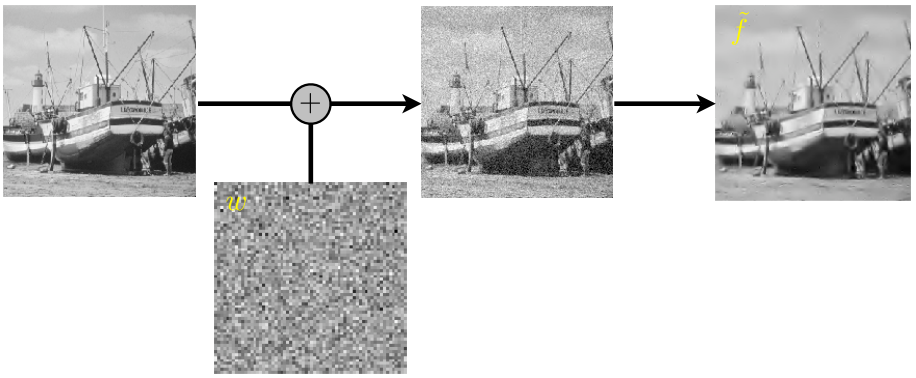


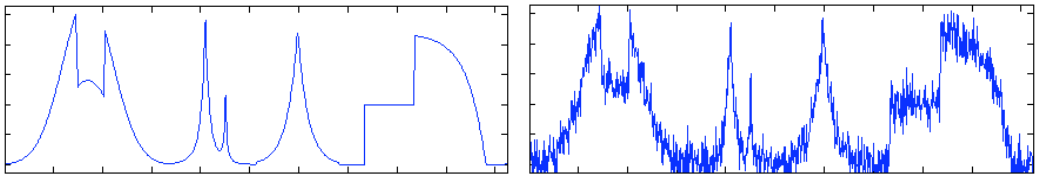
Figure 5.2: *Image formation with noise modeling and denoising pipepline.*

Statistical modeling considers w as a random vector with known distribution, while numerical computation are usually done on a single realization of this random vector, still denoted as w.

Additive Noise. The simplest model for such image formation consists in assuming that it is an additive perturbation of a clean signal f_0

$$f = f_0 + w$$

where w is the noise residual. Statistical noise modeling assume that w is a random vector, and in practice one only observes a realization of this vector. This modeling thus implies that the image f to be processed is also a random vector. Figure 5.3 and 5.4 show examples of noise addition to a clean signal and a clean image.



f_0 f

Figure 5.3: *1D additive noise example.*

f_0 f

Figure 5.4: *2D additive noise example.*

The simplest noise model assumes that each entry $w[n]$ of the noise is a Gaussian random variable of variance σ^2, and that the $w[n]$ are independent. This is the white noise model.

Depending on the image acquisition device, one should consider different noise distributions, such as for instance uniform noise $w[n] \in [-a, a]$ or Impulse noise

$$p(w[n] = x) \propto e^{-|x/\sigma|^\alpha} \quad \text{where} \quad \alpha < 2$$

In many situations, the noise perturbation is not additive, and for instance its intensity might depend on the intensity of the signal. This is the case with Poisson and multiplicative noises considered in Section 5.4.

5.1.3 Denoiser

A denoiser (also called estimator) is an estimation $\tilde{f}$ of f_0 computed from the observation f alone. It is thus also a random vector that depends on the noise w. Since f is a random vector of mean f_0, the numerical denoising process corresponds to the estimation of the mean of a random vector from a single realization. Figure 5.5 shows an example of denoising.

The quality of a denoiser is measured using the average mean square risk $\mathbb{E}_w(\|f_0 - \tilde{f}\|^2)$. where $\mathbb{E}_w$ is the esperance (averaging) with respect to the noise w. Since f_0 is unknown, this corresponds to a theoretical measure of performance, that is bounded using a mathematical analysis. In the numerical experiments, one observes a single realization $f = f_0 + w$, and the performance is estimated from this single denoising using the SNR

$$\text{SNR}(\tilde{f}, f0) = -20 \log_{10}(\|\tilde{f} - f_0\|/\|f_0\|),$$

which is expressed in dB. This measure of performance requires the knowledge of the clean signal f_0, and should thus only be considered as an experimentation tool, that might not be available in a real life denoising scenario where clean data are not available. Furthermore, the use of an L^2 measure of performance is questionable, and one should also observe the result to judge of the visual quality of the denoising.

f_0 f $\tilde{f}$

Figure 5.5: *Left: clean image, center: noisy image, right: denoised image.*

5.2 Linear Denoising using Filtering

5.2.1 Translation Invariant Estimators

A linear estimator $\mathcal{E}(f) = \tilde{f}$ of f_0 depends linearly on f, so that $\mathcal{E}(f + g) = \mathcal{E}(f) + \mathcal{E}(g)$. A translation invariant estimator commutes with translation, so that $\mathcal{E}(f_\tau) = \mathcal{E}(f)_\tau$, where $f_\tau(t) = f(t - \tau)$. Such a denoiser can always be written as a filtering

$$\tilde{f} = f \star h$$

where $h \in \mathbb{R}^N$ is a (low pass) filter, that should satisfy at least

$$\sum_n h[n] = \hat{h}[0] = 1.$$

Figure 5.6 shows an example of denoising using a low pass filter.

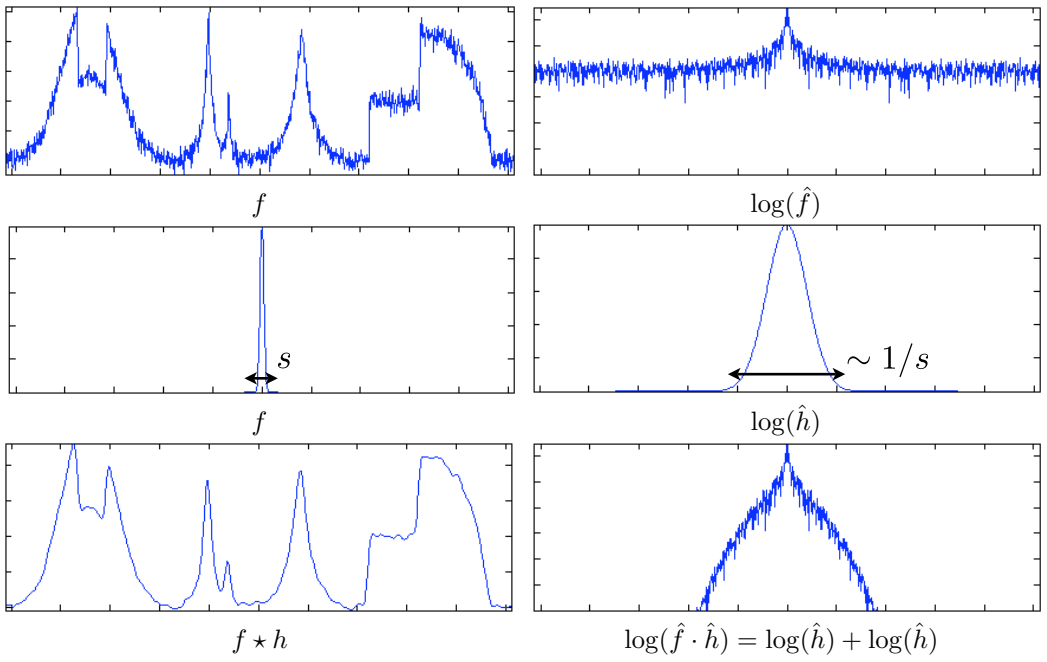


Figure 5.6: *Denoising by filtering over the spacial (left) and Fourier (right) domains.*

The filtering strength is usually controlled the width s of h. A typical example is the Gaussian filter

$$\forall\ -N/2 < i \leqslant N/2, \quad h_s[i] = \frac{1}{Z_s} \exp\left(-\frac{i^2}{2s^2}\right) \tag{5.1}$$

where Z_s ensures that $\sum_i h_s[i] = 1$ (low pass). Figure 5.6 shows the effect of Gaussian filtering over the spacial and Fourier domains.

Figure 5.7 shows the effect of low pass filtering on a signal and an image with an increasing filter width s. Linear filtering introduces a blur and are thus only efficient to denoise smooth signals and image. For signals and images with discontinuities, this blur deteriorates the signal. Removing a large amount of noise necessitates to also smooth significantly edges and singularities.

5.2.2 Optimal Filter Selection

The selection of an optimal filter is a difficult task. Its choice depends both on the regularity of the (unknown) data f_0 and the noise level σ. A simpler option is to optimize the filter width s among a parametric family of filters, such as for instance the Gaussian filters defined in (5.1).

The denoising error can be decomposed as

$$\|\tilde{f} - f_0\| \leqslant \|h_s \star f_0 - f_0\| + \|h_s \star w\|$$

The filter width s should be optimized to perform a tradeoff between removing enough noise ($\|h_s \star w\|$ decreases with s) and not smoothing too much the singularities (($\|h_s \star f_0 - f_0\|$ increases with s).

Figure (5.8) shows the oracle SNR performance, defined in (**??**).

Figure 5.9 and 5.10 show the results of denoising using the optimal filter width $s^\star$ that minimizes the SNR for a given noisy observation.

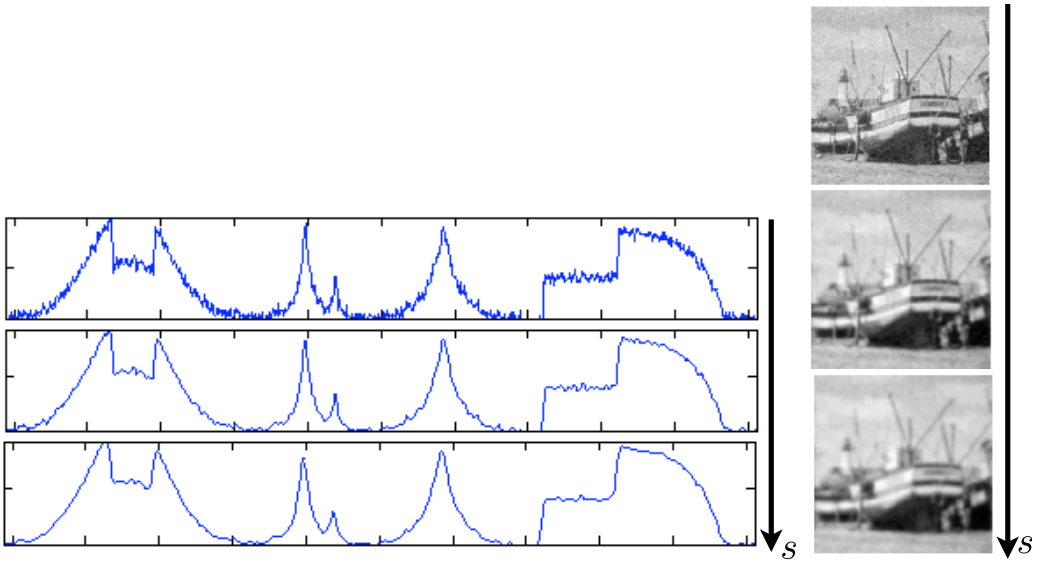


Figure 5.7: *Denoising using a filter of increasing width s.*

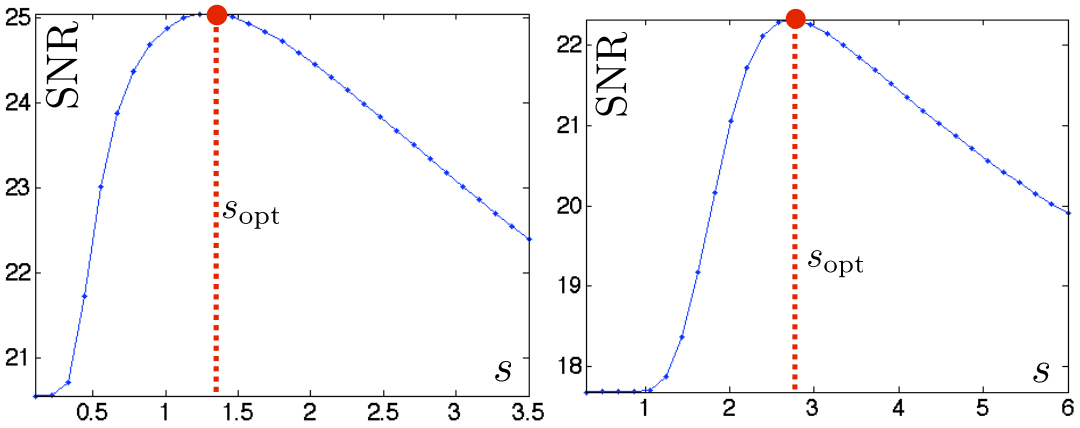


Figure 5.8: *Curves of SNR as a function of the filtering width in 1D (left) and 2D (right).*

These optimal filtering appear quite noisy, and the optimal SNR choice is usually quite conservative. Increasing the filter width introduces a strong blurring that deteriorates the SNR, although it might look visually more pleasant.

5.2.3 Wiener Filter

If one has a random model both for the noise $w \sim W$ and for for the signal $f_0 \sim F$, one can derives an optimal filters in average over both the noise and the signal realizations. One further assumes that w and f_0 are independent realization. The optimal h thus minimizes

$$\mathrm{E}_{W,F}(\|h \star (F+W) - F\|^2)$$

If both F is wide-sense stationary, and W is a Gaussian white noise of variance σ^2, then the optimal filer is known as the Wiener filter

$$\hat{h}(\omega) = \frac{|\hat{F}(\omega)|^2}{|\hat{F}(\omega)|^2 + \sigma^2}$$

where $|\hat{F}|^2$ is the power spectrum of F,

$$\hat{F}(\omega) = \hat{C}(\omega) \quad \text{where} \quad C[n] = \mathbb{E}(\langle F, F[\cdot + n] \rangle),$$

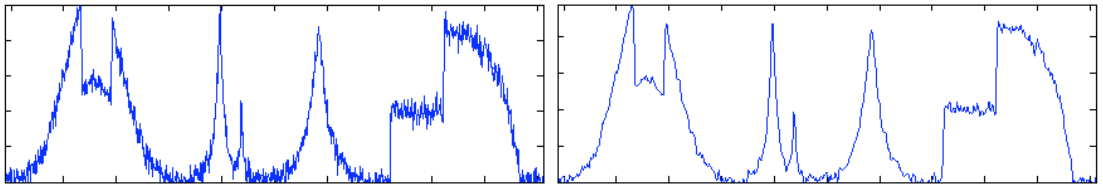


Figure 5.9: *Noisy image (left) and denoising (right) using the optimal filter width.*

Figure 5.10: *Noisy image (left) and denoising (right) using the optimal filter width.*

the Fourier transform of an infinite vector is defined in (2.6).

In practice, one rarely has such a random model for the signal, and interesting signals are often not stationary. Most signals exhibit discontinuities, and are thus poorly restored with filtering.

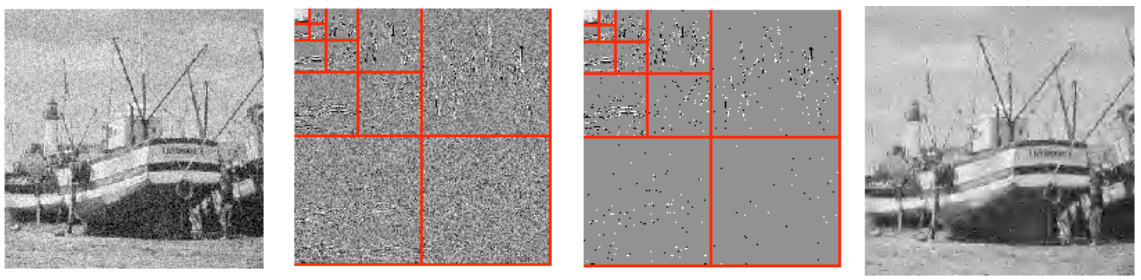
5.3 Non-linear Denoising using Thresholding

5.3.1 Hard Thresholding

We consider an orthogonal basis $\{\psi_m\}_m$ of $\mathbb{C}^N$, for instance a discrete wavelet basis. The noisy coefficients satisfy

$$\langle f, \psi_m \rangle = \langle f_0, \psi_m \rangle + \langle w, \psi_m \rangle. \tag{5.2}$$

Since a Gaussian white noise is invariant under an orthogonal transformation, $\langle w, \psi_m \rangle$ is also a Gaussian white noise of variance σ^2. If the basis $\{\psi_m\}_m$ is efficient to represent f_0, then most of the coefficients $\langle f_0, \psi_m \rangle$ are close to zero, and one observes a large set of small noisy coefficients, as shown on Figure 5.11. This idea of using thresholding estimator for denoising was first systematically explored by Donoho and Jonhstone [13].



f $\langle f, \psi^{\omega}_{j,n} \rangle$ $S_T(\langle f, \psi^{\omega}_{j,n} \rangle)$ $\tilde{f}$

Figure 5.11: *Denoising using thresholding of wavelet coefficients.*

A thresholding estimator removes these small amplitude coefficients using a non-linear hard

thresholding

$$\tilde{f} = \sum_{|\langle f, \psi_m \rangle| > T} \langle f, \psi_m \rangle \psi_m = \sum_m S_T(\langle f, \psi_m \rangle) \psi_m.$$

where S_T is defined in (4.4). This corresponds to the computation of the best M-term approximation $\tilde{f} = f_M$ of the noisy function f. Figure 5.11 shows that if T is well chose, this non-linear estimator is able to remove most of the noise while maintaining sharp features, which was not the case with linear filtering estimatiors.

Code 11 shows how to implement this thresholding for a 1D or 2D wavelet transform.

```
% Forward wavelet transform
fw = perform_wavelet_transf(f,j0,+1);
% Hard thresholding.
fw = fw .* (abs(fw)>T);
% Backward wavelet transform
f1 = perform_wavelet_transf(fw,j0,-1);
```

Matlab code 11: *Denoising by hard thresholding. Input: noisy image* `f`*, coarse scale* `j0`*, threshold* `T`*, output: denoised image* `f1`*.*

5.3.2 Soft Thresholding

We recall that the hard thresholding operator is defined as

$$S_T(x) = S_T^0(x) = \begin{cases} x & \text{if } |x| > T, \\ 0 & \text{if } |x| \leqslant T. \end{cases} \tag{5.3}$$

This thresholding performs a binary decision that might introduces artifacts. A less aggressive nonlinearity is the soft thresholding

$$S_T^1(x) = \max(1 - T/|x|, 0)x. \tag{5.4}$$

Figure 5.12 shows the 1D curves of these 1D non-linear mapping.

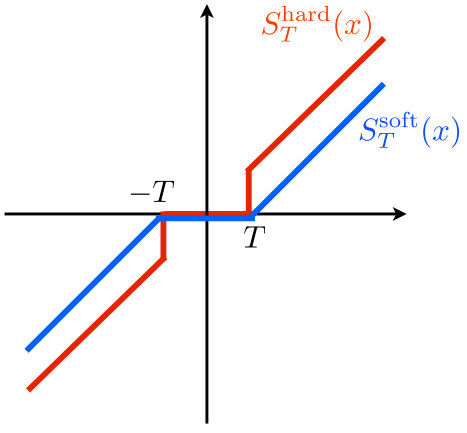


Figure 5.12: *Hard and soft thresholding functions.*

For $q = 0$ and $q = 1$, these thresholding defines two different estimators

$$\tilde{f}^q = \sum_m S_T^q(\langle f, \psi_m \rangle) \psi_m$$

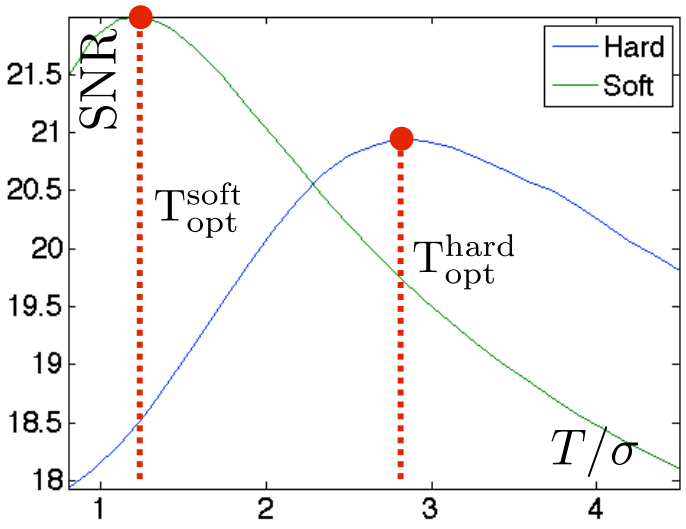


Figure 5.13: *Curves of SNR with respect to T/σ for hard and soft thresholding.*

Coarse scale management. The soft thresholded S^1_T introduces a bias since it diminishes the value of large coefficients. For wavelet transforms, it tends to introduces unwanted low-frequencies artifacts by modifying coarse scale coefficients. If the coarse scale is 2^{j_0}, one thus prefers not to threshold the coarse approximation coefficients and use, for instance in 1D,

$$\tilde{f}^1 = \sum_{0 \leqslant n < 2^{-j_0}} \langle f, \phi_{j_0,n} \rangle \phi_{j_0,n} + \sum_{j=j_0}^{0} \sum_{0 \leqslant n < 2^{-j}} S^1_T(\langle f, \psi_{j_0,n} \rangle) \psi_{j_0,n}.$$

Code 12 implements this soft thresholding with coarse scale management.

```
% Forward wavelet transform
fw = perform_wavelet_transf(f,j0,+1);
% Soft thresholding.
fw1 = max(1-T./(abs(fw)+1e-20), 0).*fw;
% Add back the coarse scale coefficients
fw1(1:2^j0) = fw(1:2^j0);
% Backward wavelet transform
f1 = perform_wavelet_transf(fw,j0,-1);
```

Matlab code 12: *Denoising by soft thresholding. Input: noisy image* `f`*, coarse scale* `j0`*, threshold* `T`*, output: denoised image* `f1`*.*

Empirical choice of the threshold. Figure 5.13 shows the evolution of the SNR with respect to the threshold T for these two estimators, for a natural image f_0. For the hard thresholding, the best result is obtained around $T \approx 3\sigma$, while for the soft thresholding, the optimal choice is around $T \approx 3\sigma/2$. These results also shows that numerically, for thresholding in orthogonal bases, soft thresholding is slightly superior than hard thresholding on natural signals and images.

Although these are experimental conclusions, these results are robust across various natural signals and images, and should be considered as good default parameters.

5.3.3 Minimax Optimality of Thresholding

Sparse coefficients estimation. To analyze the performance of the estimator, and gives an estimate for the value of T, we first assumes that the coefficients

$$a_0[m] = \langle f_0, \psi_m \rangle \in \mathbb{R}^N$$

are sparse, meaning that most of the $a_0[m]$ are zero, so that its ℓ^0 norm

$$\|a_0\|_0 = \#\{m \setminus a_0[m] \neq 0\}$$

20.9dB 21.8dB

Figure 5.14: *Comparison of hard (left) and soft (right) thresholding.*

is small. As shown in (5.2), noisy coefficients

$$\langle f, \psi_m \rangle = a[m] = a_0[m] + z[m]$$

are perturbed with an additive Gaussian white noise of variance σ^2. Figure 5.15 shows an example of such a noisy sparse signal.

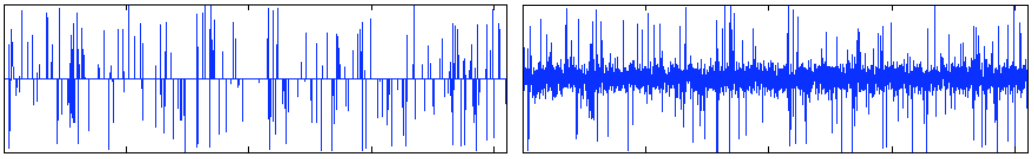


Figure 5.15: *Left: sparse signal a, right: noisy signal.*

Universal threshold value. If

$$\min_{m:a_0[m]\neq 0} |a_0[m]|$$

is large enough, then $\|f_0 - \tilde{f}\| = \|a_0 - S_T(a)\|$ is minimum for

$$T \approx \tau_N = \max_{0 \leqslant m < N} |z[m]|.$$

τ_N is a random variable that depends on N. One can show that its mean is $\sigma\sqrt{2\log(N)}$, and that as N increases, its variance tends to zero and τ_N is highly concentrated close to its mean. Figure 5.16 shows that this is indeed the case numerically.

Asymptotic minimax optimality. Donoho and Jonhstone [13] have shown that the universal threshold $T = \sigma\sqrt{2\log(N)}$ is a good theoritical choice for the denoising of signals that are well approximated in $\{\psi_m\}_m$. In particular, they show that if the non-linear approximation error in this basis decays like

$$\|f - f_M\|^2 = O(M^{-\alpha}),$$

then the denoising average error with hard and soft thresholding decays like

$$\mathbb{E}_w(\|f_0 - \tilde{f}\|^2) = O(\sigma^{\frac{2\alpha}{\alpha+1}}),$$

and that this decay rate with σ is in some sense optimal.

This universal threshold choice is however very conservative since it is guaranteed to remove almost all the noise. In practice, as shown in Figure 5.14, better results are obtained on natural signals and images by using $T \approx 3\sigma$ and $T \approx 3\sigma/2$ for hard and soft thresholdings.

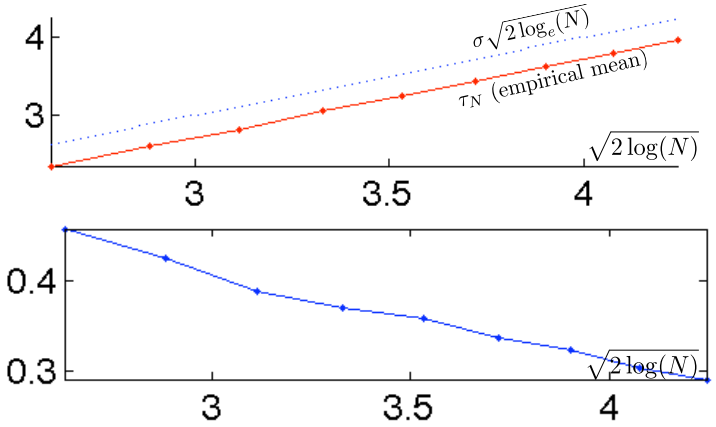


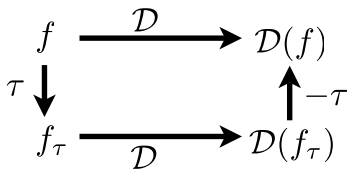
Figure 5.16: *Empirical estimation of the mean of Z_n (top) and standard deviation of Z_n (bottom)*

5.3.4 Translation Invariant Thresholding Estimators

Translation invariance. Let $f \mapsto \tilde{f} = \mathcal{D}(f)$ by a denoising method, and $f_\tau(x) = f(x-\tau)$ be a translated signal or image for $\tau \in \mathbb{R}^d$, ($d = 1$ or $d = 2$). The denoising is said to be translation invariant at precision Δ if

$$\forall\, \tau \in \Delta, \quad \mathcal{D}(f) = \mathcal{D}(f_\tau)_{-\tau}$$

where Δ is a lattice of $\mathbb{R}^d$. The denser Δ is, the more translation invariant the method is. This corresponds to the fact that $\mathcal{D}$ computes with the translation operator.



Imposing translation invariance for a fine enough set Δ is a natural constraint, since intuitively the denoising results should not depend on the location of features in the signal or image. Otherwise, some locations might be favored by the denoising process, which might result in visually unpleasant denoising artifacts.

For denoising by thresholding

$$\mathcal{D}(f) = \sum_m S_T(\langle f, \psi_m \rangle)\psi_m.$$

then translation invariance is equivalent to asking that the basis $\{\psi_m\}_m$ is translation invariant at precision Δ,

$$\forall\, m,\ \forall\, \tau \in \Delta,\ \exists m,\ \exists \lambda \in \mathbb{C}, \qquad (\psi_{m'})_\tau = \lambda \psi_m$$

where $|\lambda| = 1$.

The Fourier basis is fully translation invariant for $\Delta = \mathbb{R}^d$ over $[0,1]^d$ with periodic boundary conditions and the discrete Fourier basis is translation invariant for all interger translations $\Delta = \{0, \ldots, N_0 - 1\}^d$ where $N = N_0$ is the number of points in 1D, and $N = N_0 \times N_0$ is the number of pixels in 2D.

Unfortunately, an orthogonal wavelet basis

$$\{\psi_m = \psi_{j,n}\}_{j,n}$$

is not translation invariant both in the continuous setting or in the discrete setting. For instance, in 1D,

$$(\psi_{j',n'})_\tau \notin \{\psi_{j,n}\} \quad \text{for} \quad \tau = 2^j/2.$$

Cycle spinning. A simple way to turn a denoiser Δ into a translation invariant denoiser is to average the result of translated images

$$\mathcal{D}_{\text{inv}}(f) = \frac{1}{|\Delta|} \sum_{\tau \in \Delta} \mathcal{D}(f_\tau)_{-\tau}. \tag{5.5}$$

One easily check that

$$\forall\, \tau \in \Delta, \quad \mathcal{D}_{\text{inv}}(f) = \mathcal{D}_{\text{inv}}(f_\tau)_{-\tau}$$

To obtain a translation invariance up to the pixel precision for a data of N samples, one should use a set of $|\Delta| = N$ translation vectors. To obtain a pixel precision invariance for wavelets, this will result in $O(N^2)$ operations.

Figure 5.17 shows the result of applying cycle spinning to an orthogonal hard thresholding denoising using wavelets, where we have used the following translation of the continuous wavelet basis $\Delta = \{0, 1/N, 2/N, 3/N\}^2$, which corresponds to discrete translation by $\{0, 1, 2, 3\}^2$ on the discretized image. The complexity of the denoising scheme is thus 16 wavelet transforms. The translation invariance brings a very large SNR improvement, and significantly reduces the oscillating artifacts of orthogonal thresholding. This is because this artifacts pop-out at random locations when τ changes, so that the averaging process reduces significantly these artifacts.

Figure 5.18 shows that translation invariant hard thresholding does a slightly better job than translation invariant soft thresholding. The situation is thus reversed with respect to thresholding in an orthogonal wavelet basis. Code 13 implement this cycle spinning for a 2D wavelet thresholding.

21.8dB 23.4dB

Figure 5.17: *Comparison of wavelet orthogonal soft thresholding (left) and translation invariant wavelet hard thresholding (right).*

```
[dY,dX] = meshgrid(0:s-1,0:s-1);
f1 = f*0;
for i=1:m^2
    fs = circshift(f,[dX(i) dY(i)]);
    fw = perform_wavelet_transf(fs,j0,1);
    fw = fw .* (abs(fw)>T);
    fs = perform_wavelet_transf(fw,j0,-1);
    fs = circshift(fs,-[dX(i) dY(i)]);
    f1 = (i-1)/i*f1 + 1/i*fs;
end
```

Matlab code 13: *Denoising by cycle spinning. Input: noisy image* `f`*, coarse scale* `j0`*, threshold* `T`*, number of spins in each direction* `s`*, output: denoised image* `f1`*.*

Translation invariant wavelet frame. An equivalent way to define a translation invariant denoiser is to replace the orthogonal basis $\mathcal{B} = \{\psi_m\}$ by a redundant family of translated vectors

$$\mathcal{B}_{\text{inv}} = \{(\psi_m)_\tau\}_{m, \tau \in \Delta}. \tag{5.6}$$

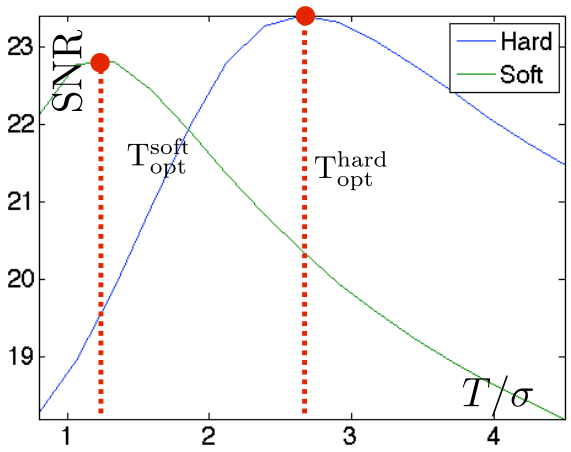


Figure 5.18: *Curve of SNR with respect to T/σ for translation invariant thresholding.*

One should be careful about the fact that $\mathcal{B}_{\text{inv}}$ is not any more an orthogonal basis, but it still enjoy a conservation of energy formula

$$\|f\|^2 = \frac{1}{|\Delta|} \sum_{m,\tau \in \Delta} |\langle f, (\psi_m)_\tau \rangle|^2 \quad \text{and} \quad f = \frac{1}{|\Delta|} \sum_{m,\tau \in \Delta} \langle f, (\psi_m)_\tau \rangle (\psi_m)_\tau.$$

This kind of redundant family are called tight frames.

One can then define a translation invariant thresholding denoising

$$\mathcal{D}_{\text{inv}}(f) = \frac{1}{|\Delta|} \sum_{m,\tau \in \Delta} S_T(\langle f, (\psi_m)_\tau \rangle)(\psi_m)_\tau. \tag{5.7}$$

This denoising is the same as the cycle spinning denoising defined in (5.5).

The frame $\mathcal{B}_{\text{inv}}$ might contain up to $|\Delta||\mathcal{B}|$ basis element. For a discrete basis of signal with N samples, and a translation lattice of $|\Delta| = N$ vectors, it corresponds to up to N^2 elements in $\mathcal{B}_{\text{inv}}$. Hopefully, for a hierarchical basis such as a discrete orthogonal wavelet basis, one might have

$$(\psi_m)_\tau = (\psi_{m'})_{\tau'} \quad \text{for} \quad m \neq m' \quad \text{and} \quad \tau \neq \tau',$$

so that the number of elements in $\mathcal{B}_{\text{inv}}$ might be much smaller than N^2. For instance, for an orthogonal wavelet basis, one has

$$(\psi_{j,n})_{k2^j} = \psi_{j,n+k},$$

so that the number of basis elements is $|\mathcal{B}_{\text{inv}}| = N \log_2(N)$ for a 2D basis, and $3N \log_2(N)$ for a 2D basis. The fast translation invariant wavelet transform, also called "a trou" wavelet transform, computes all the inner products $\langle f, (\psi_m)_\tau \rangle$ in $O(N \log_2(N))$ operations. Implementing formula (5.7) is thus much faster than applying the cycle spinning (5.5) equivalent formulation.

Translation invariant wavelet coefficients are usually grouped by scales in $\log_2(N)$ (for $d = 1$) or by scales and orientations $3 \log_2(N)$ (for $d = 2$) sets of coefficients. For instance, for a 2D translation invariant transform, one consider

$$\forall\, n \in \{0, \dots, 2^j N_0 - 1\}^2,\ \forall\, k \in \{0, \dots, 2^{-j}\}^2, \quad d_j^\omega[2^{-j}n + k] = \langle f, (\psi_{j,n})_{k2^j} \rangle$$

where $\omega \in \{V, H, D\}$ is the orientation. Each set d_j^ω has N coefficients and is a band-pass filtered version of the original image f, as shown on Figure 5.19.

Figure 5.20 shows how these set of coefficients are hard thresholded by the translation invariant estimator.

5.3.5 Exotic Thresholdings

It is possible to devise many thresholding nonlinearities that interpolate between the hard and soft thresholder. We present here two examples, but many more exist in the literature. Depending on the statistical distribution of the wavelet coefficients of the coefficients of f in the basis, these thresholders might produce slightly better results.

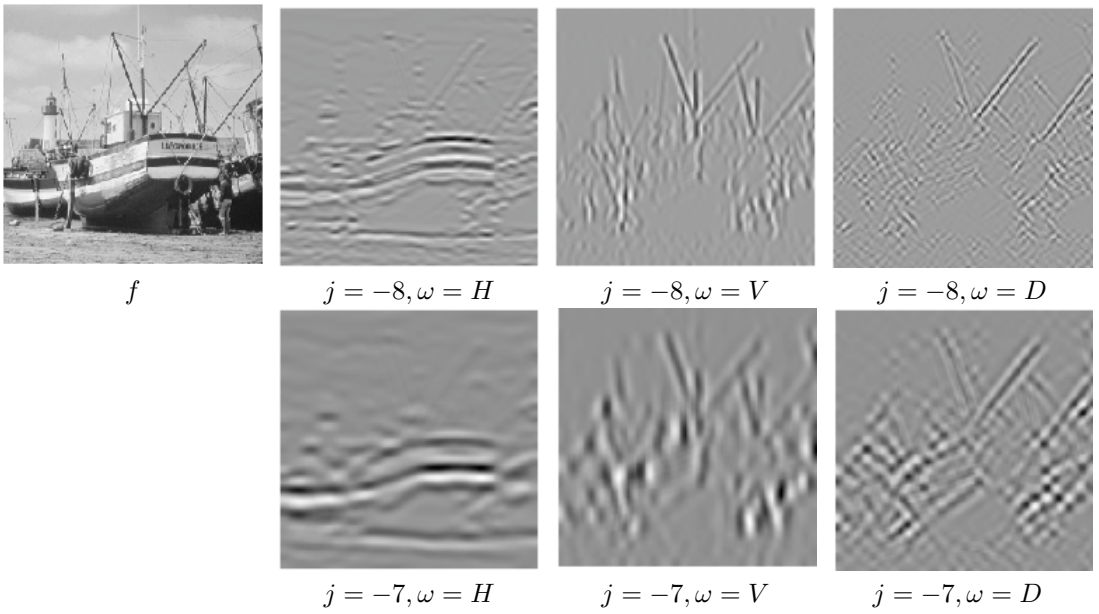


Figure 5.19: *Translation invariant wavelet coefficients.*

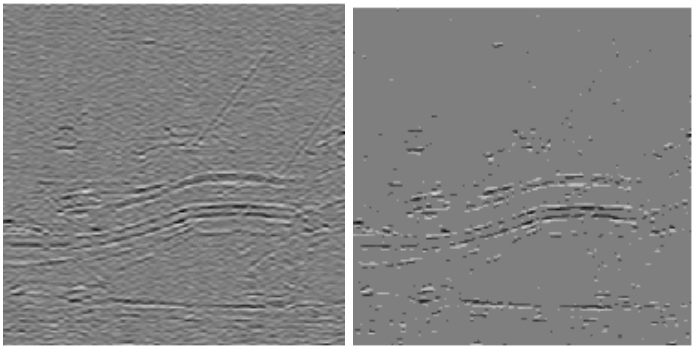


Figure 5.20: *Left: translation invariant wavelet coefficients, for $j = -8, \omega = H$, right: tresholded coefficients.*

Semi-soft thresholding. One can define a family of intermediate thresholder that depends on a parameter $\mu > 1$

$$S_T^\theta(x) = g_{\frac{1}{1-\theta}}(x) \quad \text{where} \quad g_\mu(x) = \begin{cases} 0 & \text{if} \quad |x| < T \\ x & \text{if} \quad |x| > \mu T \\ \text{sign}(x)\frac{|x|-T}{\mu-1} & \text{otherwise.} \end{cases}$$

One thus recovers the hard thresholding as S_T^0 and the soft thresholding as S_T^1. Figure 5.21 display an example of such a non-linearity.

Figure 5.22 shows that a well chosen value of μ might actually improves over both hard and soft thresholders. The improvement is however hardly noticeable visually.

Stein thresholding. The Stein thresholding is defined using a quadratic attenuation of large coefficients

$$S_T^{\text{Stein}}(x) = \max\left(1 - \frac{T^2}{|x|^2}, 0\right) x.$$

This should be compared with the linear attenuation of the soft thresholding

$$S_T^1(x) = \max\left(1 - \frac{T}{|x|}, 0\right) x.$$

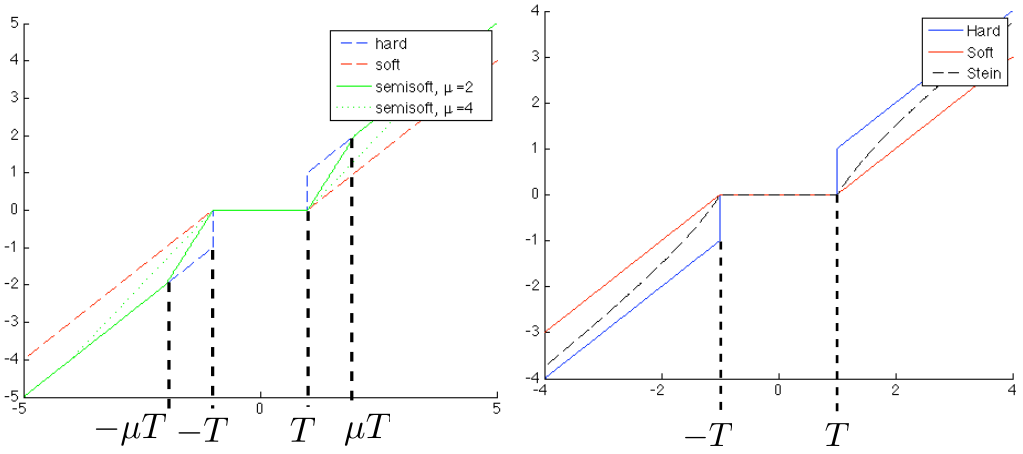


Figure 5.21: *Left: semi-soft thresholder, right: Stein thresholder.*

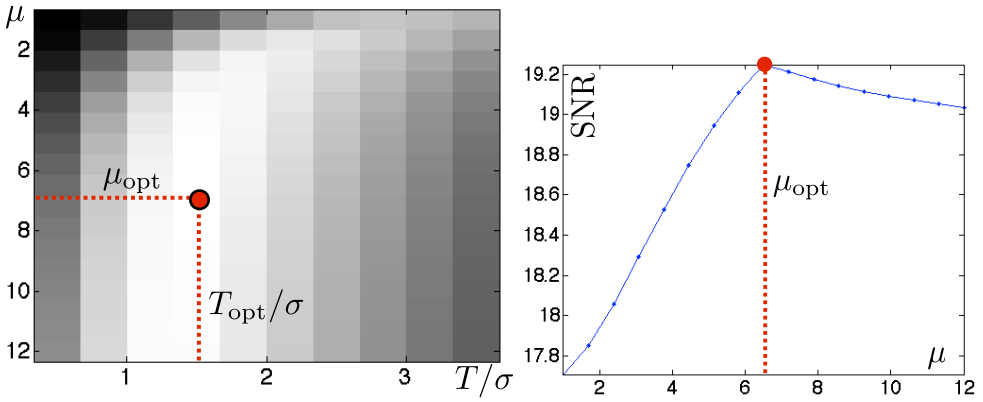


Figure 5.22: *Left: image of SNR with respect to the parameters μ and T/σ, right: curve of SNR with respect to μ using the best T/σ for each μ.*

The advantage of the Stein thresholder with respect to the soft thresholding is that

$$|S_T^{\text{Stein}}(x) - x| \to 0 \quad \text{whereas} \quad |S_T^1(x) - x| \to T,$$

where $x \to \pm\infty$. This means that Stein thresholding does not suffer from the bias of soft thresholding.

For translation invariant thresholding, Stein and hard thresholding perform similarly on natural images.

5.3.6 Block Thresholding

The non-linear thresholding method presented in the previous section are diagonal estimators, since they operate a coefficient-by-coefficient attenuation

$$\tilde{f} = \sum_m A_T^q(\langle f, \psi_m \rangle)\langle f, \psi_m \rangle \psi_m$$

where

$$A_T^q(x) = \begin{cases} \max(1 - x^2/T^2, 0) & \text{for} \quad q = \text{Stein} \\ \max(1 - |x|/T, 0) & \text{for} \quad q = 1 \text{ (soft)} \\ 1_{|x|>T} & \text{for} \quad q = 0 \text{ (hard)} \end{cases}$$

Block thresholding takes advantage of the statistical dependancy of wavelet coefficients, by computing the attenuation factor on block of coefficients. This is especially efficient for natural images,

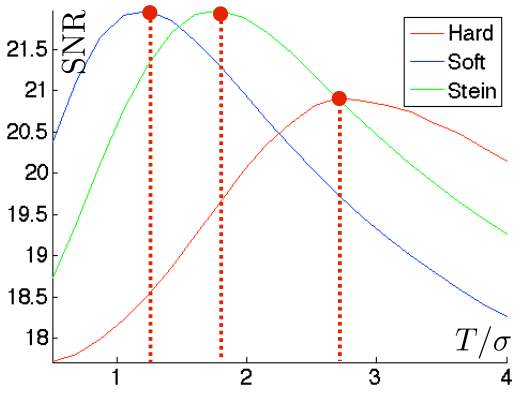


Figure 5.23: *SNR curves with respect to T/σ for Stein threhsolding.*

where edges and geometric features create clusters of high magnitude coefficients. Block decisions also help to remove artifacts due to isolated noisy large coefficients in regular areas.

The set of coefficients is divided into disjoint blocks, and for instance for 2D wavelet coefficients

$$\{(j,n,\omega)\}_{j,n,\omega} = \bigcup_k B_k,$$

where each B_k is a square of $s \times s$ coefficients, where the block size s is a parameter of the method. Figure 5.24 shows an example of such a block.

The block energy is defined as

$$B_k = \frac{1}{s^2} \sum_{m \in B_k} |\langle f, \psi_m \rangle|^2,$$

and the block thresholding

$$\tilde{f} = \sum_m S_T^{\text{block},q}(\langle f, \psi_m \rangle)\psi_m$$

makes use of the same attenuation for all coefficients within a block

$$\forall\, m \in B_k, \quad S_T^{\text{block},q}(\langle f, \psi_m \rangle) = A_T^q(E_k)\langle f, \psi_m \rangle.$$

for $q \in \{0, 1, \text{stein}\}$. Figure 5.24 shows the effect of this block attenuation, and the corresponding denoising result.

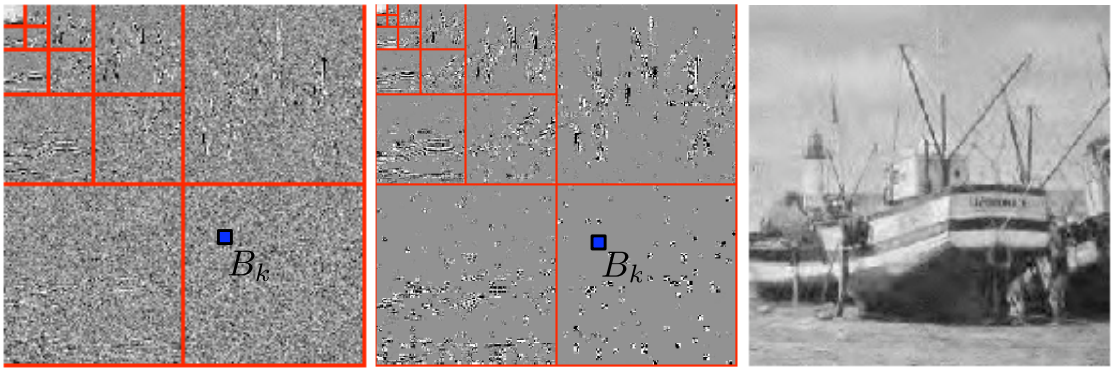


Figure 5.24: *Left: wavelet coefficients, center: block thresholded coefficients, right: denoised image.*

Figure 5.25, left, compares the three block thresholding obtained for $q \in \{0, 1, \text{stein}\}$. Numerically, on natural images, Stein block thresholding gives the best results. Figure 5.25, right, compares the block size for the Stein block thresholder. Numerically, for a broad range of images, a value of $s = 4$ works well.

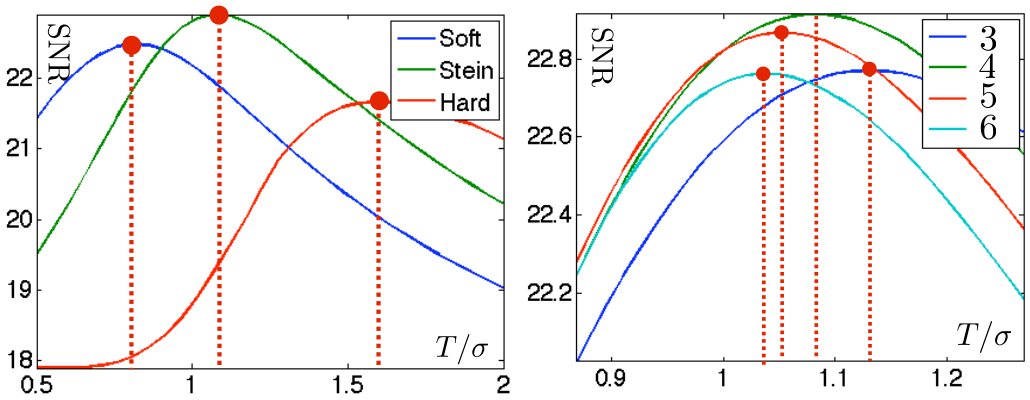


Figure 5.25: *Curve of SNR with respect to T/σ (left) and comparison of SNR for different block size (right).*

Figure 5.26 shows a visual comparison of the denoising results. Block stein thresholding of orthogonal wavelet coefficients gives a result nearly as good as a translation invariant wavelet hard thresholding, with a faster algorithm. The block thresholding strategy can also be applied to wavelet coefficients in translation invariant tight frame, which produces the best results among all denoisers detailed in this book.

Code 14 implement this block thresholding.

SNR=23.4dB 22.8dB 23.8dB

Figure 5.26: *Left: translation invariant wavelet hard thresholding, center: block orthogonal Stein thresholding, right: block translation invariant Stein thresholding.*

One should be aware that more advanced denoisers use complicated statistical models that improves over the methods proposed in this book, see for instance [23].

5.4 Data-dependant Noises

For many imaging devices, the variance of the noise that perturbs $f_0[n]$ depends on the value of $f_0[n]$. This is a major departure from the additive noise formation model considered so far. We present here two popular examples of such non-additive models.

5.4.1 Poisson Noise

Many imaging devices sample an image through a photons counting operation. This is for instance the case in digital camera, confocal microscopy, TEP and SPECT tomography.

```
n = size(f,1);
% Forward wavelet transform.
fw = perform_wavelet_transf(f,j0,+1);
% Compute indexing of the blocks.
[dX,dY,X,Y] = ndgrid(0:s-1,0:s-1,1:s:n-s+1,1:s:n-s+1);
I = X+dX + (Y+dY-1)*n;
% Extract the blocks
H = reshape(fw(I(:)),size(I));
% Compute the average energy of each block, and duplicate.
v = mean(mean(abs(H).^2,1),2);
v = repmat( max3(v,1e-15), [w w]);
% Stein threshold the blocks.
HT = max3(1-T^2*v.^(-1),0) .* H;
% Reconstruct the thresholded coefficients.
fw(I(:)) = HT(:);
% Inverse wavelet transform.
f1 = perform_wavelet_transf(fw,j0,-1);
```

Matlab code 14: *Denoising by Stein block thresholding. Input: noisy image* `f`*, coarse scale* `j0`*, threshold* `T`*, size* `s` *of the blocks, output: denoised image* `f1`*.*

Poisson model. The uncertainty of the measurements for a quantized unknown image $f_0[n] \in \mathbb{N}$ is then modeled using a Poisson noise distribution

$$f[n] \sim \mathcal{P}(\lambda) \quad \text{where} \quad \lambda = f_0[n] \in \mathbb{N},$$

and where the Poisson distribution $\mathcal{P}(\lambda)$ is defined as

$$\mathbb{P}(f[n] = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

and thus varies from pixel to pixel. Figure 5.27 shows examples of Poisson distributions.

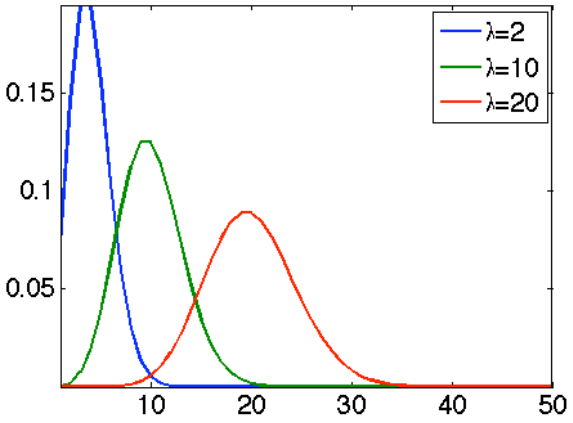


Figure 5.27: *Poisson distributions for various* λ*.*

One has

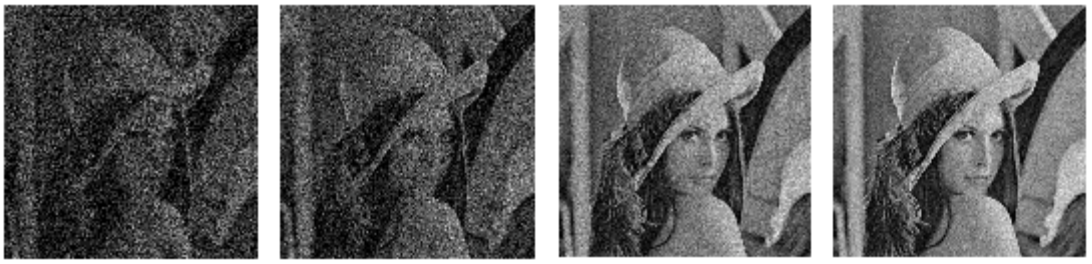
$$\mathbb{E}(f[n]) = \lambda = f_0[n] \quad \text{and} \quad \mathrm{Var}(f[n]) = \lambda = f_0[n]$$

so that the denoising corresponds to estimating the mean of a random vector from a single observation, but the variance now depends on the pixel intensity. This shows that the noise level increase with the intensity of the pixel (more photons are coming to the sensor) but the relative variation $(f[n] - f_0[n])/f_0[n]$ tends to zero in expectation when $f_0[n]$ increases.

Figure 5.28 shows examples of a clean image f_0 quantized using different values of $\lambda_{\max}$ and perturbed with the Poisson noise model.

Variance stabilization. Applying thresholding estimator

$$\mathcal{D}(f) = \sum_m S_T^q(\langle f, \psi_m \rangle)\psi_m$$



$\lambda_{\max} = 5$ $\lambda_{\max} = 50$ $\lambda_{\max} = 50$ $\lambda_{\max} = 100$

Figure 5.28: *Noisy image with Poisson noise model, for various* $\lambda_{\max} = \max_n f_0[n]$.

to f might give poor results since the noise level fluctuates from point to point, and thus a single threshold T might not be able to capture these variations. A simple way to improve the thresholding results is to first apply a variance stabilization non-linearity $\phi : \mathbb{R} \to \mathbb{R}$ to the image, so that $\phi(f)$ is as close as possible to an additive Gaussian white noise model

$$\phi(f) \approx \phi(f_0) + w \tag{5.8}$$

where $w[n] \sim \mathcal{N}(0, \sigma)$ is a Gaussian white noise of fixed variance σ^2.

Perfect stabilization is impossible, so that (5.8) only approximately holds for a limited intensity range of $f_0[n]$. Two popular variation stabilization functions for Poisson noise are the Anscombe mapping

$$\phi(x) = 2\sqrt{x + 3/8}$$

and the mapping of Freeman and Tukey

$$\phi(x) = \sqrt{x+1} + \sqrt{x}.$$

Figure 5.29 shows the effect of these variance stabilizations on the variance of $\phi(f)$.

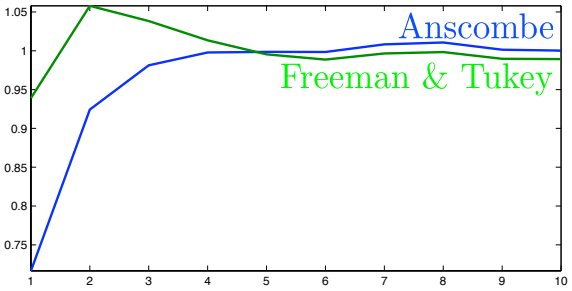


Figure 5.29: *Comparison of variariance stabilization: display of* $Var(\phi(f[n]))$ *as a function of* $f_0[n]$.

A variance stabilized denoiser is defined as

$$\Delta^{\text{stab},q}(f) = \phi^{-1}(\sum_m S_T^q(\langle \phi(f), \psi_m \rangle)\psi_m)$$

where ϕ^{-1} is the inverse mapping of ϕ.

Figure 5.30 shows that for moderate intensity range, variance stabilization improves over non-stabilized denoising.

5.5 Multiplicative Noise

Multiplicative image formation. A multiplicative noise model assumes that

$$f[n] = f_0[n]w[n]$$

Figure 5.30: *Left: noisy image, center: denoising without variance stabilization, right: denoising after variance stabilization.*

where w is a realization of a random vector with $\mathbb{E}(w) = 1$. Once again, the noise level depends on the pixel value

$$\mathbb{E}(f[n]) = f_0[n] \quad \text{and} \quad \mathrm{Var}(f[n]) = f_0[n]^2\sigma^2 \quad \text{where} \quad \sigma^2 = \mathrm{Var}(w).$$

Such a mutiplicative noise is a good model for SAR satellite imaging, where f is obtained by averaging S images

$$\forall\, 0 \leqslant s < K, \quad f^{(s)}[n] = f_0[n]w^{(s)}[n] + r^{(s)}[n]$$

where $r^{(s)}$ is a Gaussian white noise, and $w^{(s)}[n]$ is distributed according to a one-sided exponential distribution

$$\mathcal{P}(w^{(s)}[n] = x) \propto e^{-x}\, \mathbb{I}_{x>0}.$$

For K large enough, averaging the images cancels the additive noise and one obtains

$$f[n] = \frac{1}{K}\sum_{s=1}^{K} f^{(s)}[n] \approx f_0[n]w[n]$$

where w is distributed according to a Gamma distribution

$$w \sim \Gamma(\sigma = K^{-\frac{1}{2}}, \mu = 1) \quad \text{where} \quad \mathbb{P}(w = x) \propto x^{K-1}e^{-Kx},$$

One should note that increasing the value of K reduces the overall noise level.

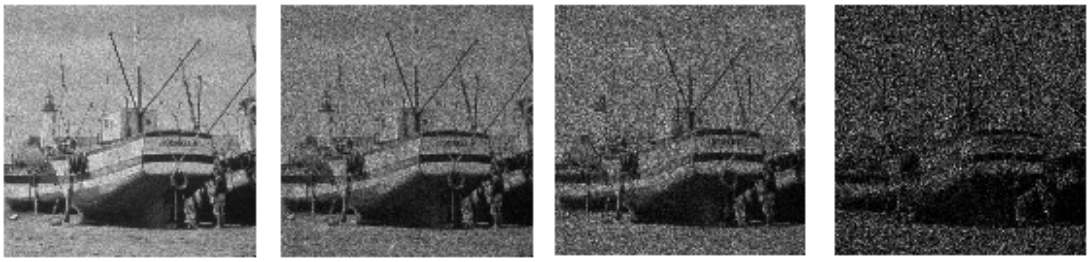


Figure 5.31: *Noisy images with multiplicative noise, with varying σ.*

Figure **??** shows an example of such image formation for a varying number $K = 1/\sigma^2$ of averaged images.

A simple variance stabilization transform is

$$\phi(x) = \log(x) - c$$

where

$$c = \mathrm{E}(\log(w)) = \psi(K) - \log(K) \quad \text{where} \quad \psi(x) = \Gamma'(x)/\Gamma(x)$$

and where Γ is the Gamma function that generalizes the factorial function to non-integer. One thus has

$$\phi(f)[n] = \phi(f_0)[n] + z[n],$$

where $z[n] = \log(w) - c$ is a zero-mean additive noise.

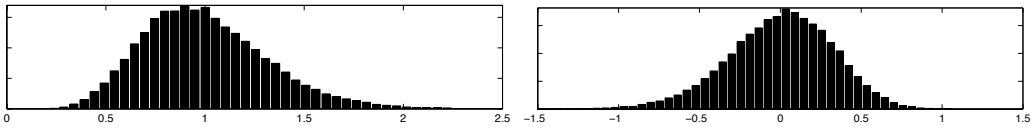


Figure 5.32: *Histogram of multiplicative noise before (left) and after (right) stabilization.*

Figure 5.32 shows the effect of this variance stabilization on the repartition of w and z.

Figure 5.33 shows that for moderate noise level σ, variance stabilization improves over non-stabilized denoising.

Figure 5.33: *Left: noisy image, center: denoising without variance stabilization, right: denoising after variance stabilization.*

Chapter 6

Denoising with Variational Minimization

To solve denoising and more general inverse problems, variational methods consider a prior J that assigns a score $J(f)$ to each signal or image f. In this Section, we focus in particular on images, $d = 2$, and the proposed variational priors are easily extended to arbitrary dimensions.

6.1 Sobolev and Total Variation Priors

The simplest prior are obtained by integrating local differential quantity over the image. They corresponds to norms in functional spaces that imposes some smoothness on the signal of the image. We detail here the Sobolev and the total variation priors, that are the most popular in image processing.

6.1.1 Continuous Priors

In the following, we consider either continuous functions $f \in \mathrm{L}^2([0,1]^2)$ or discrete vectors $f \in \mathbb{R}^N$, and consider continuous priors and there discrete counterparts in Section 6.1.2.

Sobolev prior. The prior energy $J(f) \in \mathbb{R}$ is intended to be low for images in a class $f \in \Theta$. The class of uniformly smooth functions detailed in Section 4.2.1 corresponds to functions in Sobolev spaces. A simple prior derived from this Sobolev class is thus

$$J_{\mathrm{Sob}}(f) = \frac{1}{2}\|f\|_{\mathrm{Sob}}^2 = \frac{1}{2}\int \|\nabla f(x)\|^2 \mathrm{d}x, \tag{6.1}$$

where ∇f is the gradient in the sense of distributions.

Total variation prior. To take into account discontinuities in images, one considers a total variation energy, introduced in Section 4.2.3. It was introduced for image denoising by Rudin, Osher and Fatemi [25]

The total variation of a smooth image f is defined as

$$J_{\mathrm{TV}}(f) = \|f\|_{\mathrm{TV}} = \int \|\nabla_x f\| \mathrm{d}x. \tag{6.2}$$

This energy extends to non-smooth functions of bounded variations $f \in \mathrm{BV}([0,1]^2)$. This class contains indicators functions $f = 1_\Omega$ of sets Ω with a bounded perimeter $|\partial\Omega|$.

The total variation norm can be computed alternatively using the co-area formula (4.12), which shows in particular that $\|1_\Omega\|_{\mathrm{TV}} = |\partial\Omega|$.

6.1.2 Discrete Priors

An analog image $f \in \mathrm{L}^2([0,1]^2)$ is discretized through an acquisition device to obtain a discrete image $f \in \mathbb{R}^N$. Image processing algorithms work on these discrete data, and we thus need to define discrete priors for finite dimensional images.

Discrete gradient. Discrete Sobolev and total variation priors are obtained by computing finite differences approximations of derivatives, using for instance forward differences

$$\delta_1 f[n_1, n_2] = f[n_1 + 1, n_2] - f[n_1, n_2]$$
$$\delta_2 f[n_1, n_2] = f[n_1, n_2 + 1] - f[n_1, n_2],$$

and one can use higher order schemes to process more precisely smooth functions. One should be careful with boundary conditions, and we consider here for simplicity periodic boundary conditions, which correspond to computing the indexes $n_i + 1$ modulo N. More advanced symmetric boundary conditions can be used as well to avoid boundary artifacts.

A discrete gradient is defined as

$$\nabla f[n] = (\delta_1 f[n], \delta_2 f[n]) \in \mathbb{R}^2$$

which corresponds to a mapping from images to vector fields

$$\nabla : \mathbb{R}^N \longrightarrow \mathbb{R}^{N \times 2}.$$

Figure 6.1 shows examples of gradient vectors. They point in the direction of the largest slope of the function discretized by f. Figure 6.2 shows gradients and their norms displayed as an image. Regions of high gradients correspond to large intensity variations, and thus typically to edges or textures.

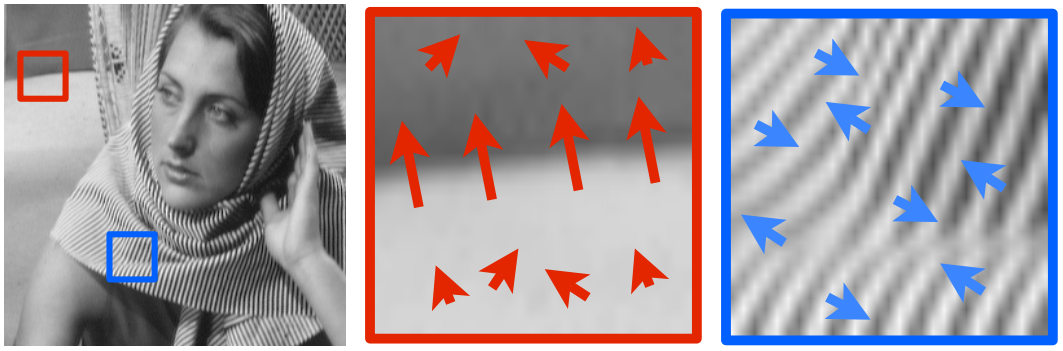


Figure 6.1: *Discrete gradient vectors.*

Discrete divergence. One can also use backward differences,

$$\tilde{\delta}_1 f[n_1, n_2] = f[n_1, n_2] - f[n_1 - 1, n_2]$$
$$\tilde{\delta}_2 f[n_1, n_2] = f[n_1, n_2] - f[n_1, n_2 - 1].$$

They are dual to the forward differences, so that

$$\delta_i^* = -\tilde{\delta}_i,$$

which means that

$$\forall\, f, g \in \mathbb{R}^N, \quad \langle \delta_i f,\, g \rangle = -\langle f,\, \tilde{\delta}_i g \rangle,$$

which is a discrete version of the integration by part formula

$$\int_0^1 f' g = -\int_0^1 f g'$$

for smooth periodic functions on $[0,1]$

A divergence operator is defined using backward differences,

$$\mathrm{div}(v)[n] = \tilde{\delta}_1 v_1[n] + \tilde{\delta}_2 v_2[n],$$

and corresponds to a mapping from vector fields to images

$$\mathrm{div} : \mathbb{R}^{N\times 2} \longrightarrow \mathbb{R}^N.$$

It is related to the dual of the gradient

$$\mathrm{div} = -\nabla^*$$

which means that

$$\forall\, f \in \mathbb{R}^N,\ \forall\, v \in \mathbb{R}^{N\ \times 2},\quad \langle \nabla f,\, v\rangle_{\mathbb{R}^{N\times 2}} = -\langle f,\, \mathrm{div}(v)\rangle_{\mathbb{R}^N}$$

which corresponds to a discrete version of the divergence theorem.

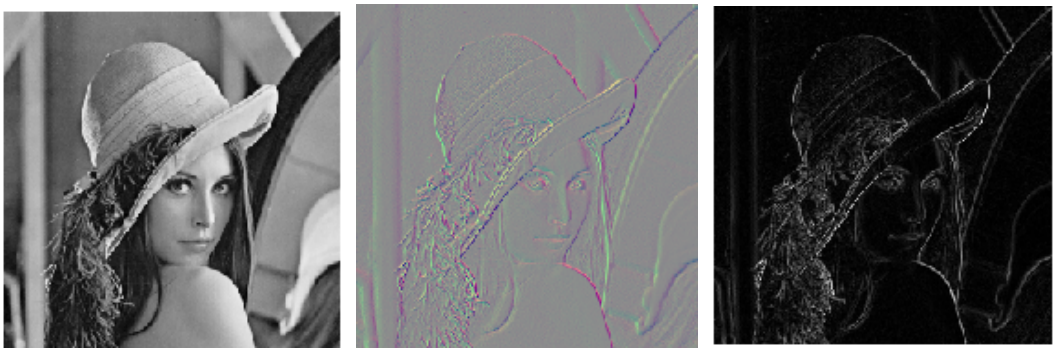


Image f $\quad$ ∇f $\quad$ $\|\nabla f\|$

Figure 6.2: *Discrete operators.*

Discrete laplacian. A general definition of a Laplacian is

$$\Delta f = \mathrm{div}(\nabla f),$$

which corresponds to a semi-definite negative operator.

For discrete images, and using the previously defined gradient and divergence, it is a local high pass filter

$$\Delta f[n] = \sum_{p\in V_4(n)} f[p] - 4f[n], \tag{6.3}$$

that approximates the continuous second order derivative

$$\frac{\partial^2 f}{\partial x_1^2}(x) + \frac{\partial^2 f}{\partial x_2^2} \approx N^2 \Delta f[n] \quad \text{for} \quad x = n/N.$$

Lapalacian operators thus correspond to filterings. A continuous Laplacian is equivalently defined over the Fourier domain in diagonal form as

$$g = \Delta f \quad \Longrightarrow \quad \hat{g}(\omega) = \|\omega\|^2 \hat{f}(\omega)$$

and the discrete Laplacian (6.3) as

$$g = \Delta f \quad \Longrightarrow \quad \hat{g}[\omega] = \rho[\omega]^2 \hat{f}(\omega) \quad \text{where} \quad \rho[\omega]^2 = \sin\left(\frac{\pi}{N}\omega_1\right)^2 + \sin\left(\frac{\pi}{N}\omega_2\right)^2. \tag{6.4}$$

Discrete energies. A discrete Sobolev energy is obtained by using the ℓ^2 norm of the discrete gradient vector field

$$J_{\mathrm{Sob}}(f) = \frac{1}{2}\sum_n (\delta_1 f[n])^2 + (\delta_2 f[n])^2 = \frac{1}{2}\|\nabla f\|^2. \tag{6.5}$$

Similarly, a discrete TV energy is defined as the ℓ^1 norm of the gradient field

$$J_{\mathrm{TV}}(f) = \sum_n \sqrt{(\delta_1 f[n])^2 + (\delta_2 f[n])^2} = \|\nabla f\|_1 \tag{6.6}$$

where the ℓ^1 norm of a vector field $v \in \mathbb{R}^{N\times 2}$ is

$$\|v\|_1 = \sum_n \|v[n]\| \tag{6.7}$$

where $v[n] \in \mathbb{R}^2$.

6.2 PDE and Energy Minimization

Image smoothing is obtained by minimizing the prior using a gradient descent.

6.2.1 General Flows

The gradient of the prior $J : \mathbb{R}^N \to \mathbb{R}$ is a vector $\operatorname{grad} J(f)$. It describes locally up to the first order the variation of the prior

$$J(f+\varepsilon) = J(f) + \langle \varepsilon, \operatorname{grad} J(f)\rangle + o(\|\varepsilon\|^2).$$

If J is a smooth function of the image f, a discrete energy minimization is obtained through a gradient descent

$$f^{(k+1)} = f^{(k)} - \tau \operatorname{grad} J(f^{(k)}), \tag{6.8}$$

where the step size τ must be small enough to guarantee convergence.

For infinitesimal step size τ, one replaces the discrete parameter k by a continuous time, and the flow

$$t > 0 \longmapsto f_t \in \mathbb{R}^N$$

solves the following partial differential equation

$$\frac{\partial f_t}{\partial t} = -\operatorname{grad} J(f_t) \quad \text{and} \quad f_0 = f. \tag{6.9}$$

The gradient descent can be seen as an explicit discretization in time of this PDE at times $t_k = k\tau$.

6.2.2 Heat Flow

The heat flow corresponds to the instantiation of the generic PDE (6.9) to the case of the Sobolev energies $J_{\mathrm{Sob}}(f)$ defined for continuous function in (6.1) and for discrete images in (6.5).

Since it is a quadratic energy, its gradient is easily computed

$$J(f+\varepsilon) = \frac{1}{2}\|\nabla f + \nabla\varepsilon\|^2 = J(f) - \langle \Delta f, \varepsilon\rangle + o(\|\varepsilon\|^2),$$

so that

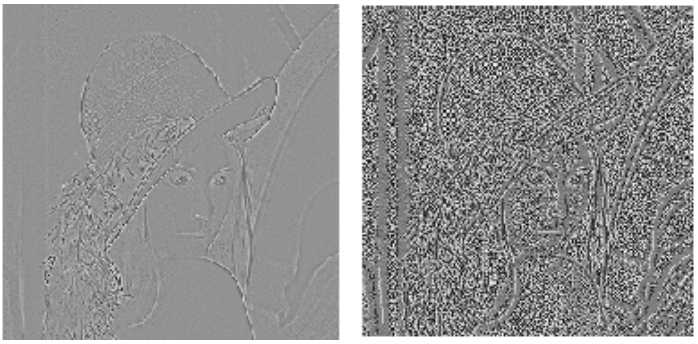
$$\operatorname{grad} J_{\mathrm{Sob}}(f) = -\Delta f.$$

Figure 6.4, left, shows an example of Laplacian. It is typically large (positive or negative) near edges.

The heat flow is thus

$$\frac{\partial f_t}{\partial t}(x) = -(\operatorname{grad} J(f_t))(x) = \Delta f_t(x) \quad \text{and} \quad f_0 = f. \tag{6.10}$$

Figure 6.3: *Display of f_t for increasing time t for heat flow (top row) and TV flow (bottom row).*



$\mathrm{div}(\nabla f)$ $\mathrm{div}(\nabla f/\|\nabla f\|)$

Figure 6.4: *Discrete Laplacian and discrete TV gradient.*

Continous in space. For continuous images and an unbounded domain $\mathbb{R}^2$, the PDE (6.10) has an explicit solution as a convolution with a Gaussian kernel of increasing variance as time increases

$$f_t = f \star h_t \quad \text{where} \quad h_t(x) = \frac{1}{4\pi t} e^{-\frac{-\|x\|^2}{4t}}. \tag{6.11}$$

This shows the regularizing property of the heat flow, that operates a blurring to make the image more regular as time evolves.

Discrete in space. The discrete Sobolev energy (6.5) minimization defined a PDE flow that is discrete in space

$$\frac{\partial f_t}{\partial t}[n] = -(\mathrm{grad}\, J(f_t))[n] = \Delta f_t[n].$$

It can be further discretized in time as done in (6.8) and leads to a fully discrete flow

$$f^{(k+1)}[n] = f^{(k)}[n] + \tau\Big(\sum_{p \in V_4(n)} f[p] - 4f[n]\Big) = f \star h[n]$$

where $V_4(n)$ are the four neighbor to a pixel n. The flow thus corresponds to iterative convolutions

$$f^{(k)} = f \star h \star \ldots \star h = f \star^k h.$$

where h is a discrete filter.

It can be shown to be stable and convergent if $\tau < 1/4$.

6.2.3 Total Variation Flows

Total variation gradient. The total variation energy J_{TV}, both continuous (6.2) and discrete (6.6) is not a smooth function of the image. For instance, the discrete J_{TV} is non-differentiable at an image f such that there exists a pixel n where $\nabla f[n] = 0$.

If $\nabla f[n] \neq 0$, one can compute the gradient of the TV energy specialized at pixel n as

$$(\operatorname{grad} J(f))[n] = -\operatorname{div}\left(\frac{\nabla f}{\|\nabla f\|}\right)[n]$$

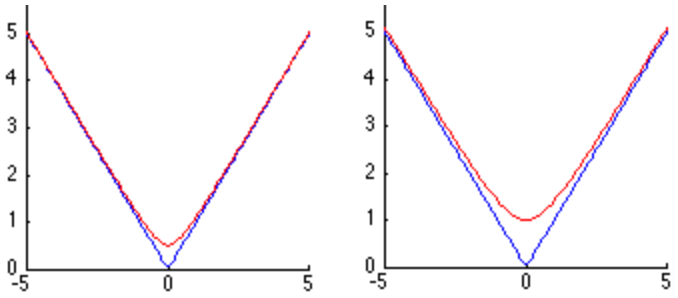
which exhibits a division by zero singularity for a point with vanishing gradient. Figure 6.4 shows an example of TV gradient, which appears noisy in smooth areas, because $\|\nabla f[n]\|$ is small in such regions.

This non-differentiabilty makes impossible the definition of a gradient descent and a TV flow.

Regularized total variation. To avoid this issue, one can modify the TV energy, and define a smoothed TV prior

$$J^{\varepsilon}_{\mathrm{TV}}(f) = \sum_n \sqrt{\varepsilon^2 + \|\nabla f[n]\|^2} \tag{6.12}$$

where $\varepsilon > 0$ is a small regularization parameter. Figure 6.5 shows this effect of this regularization on the absolute value function.



$\varepsilon = 0.1$ $\varepsilon = 0.01$

Figure 6.5: *Regularized absolute value* $x \mapsto \sqrt{x^2 + \varepsilon^2}$.

This smoothed TV energy is a differentiable function of the image, and its gradient is

$$\operatorname{grad} J^{\varepsilon}_{\mathrm{TV}}(f) = -\operatorname{div}\left(\frac{\nabla f}{\sqrt{\varepsilon^2 + \|\nabla f\|^2}}\right). \tag{6.13}$$

One can see that this smoothing interpolate between TV and Sobolev, as

$$\operatorname{grad}^{\varepsilon}_f \sim -\Delta/\varepsilon \quad \text{when} \quad \varepsilon \to +\infty.$$

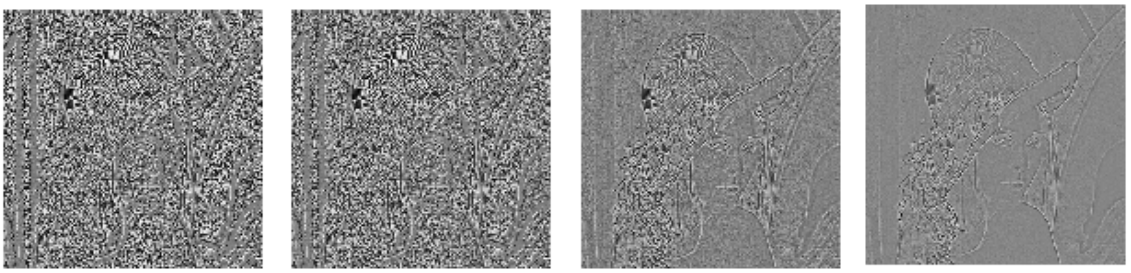
Figure 6.6 shows the evolution of this gradient for several value of the smoothing parameter.

Regularized total variation flow. The smoothed total variation flow is then defined as

$$\frac{\partial f_t}{\partial t} = \operatorname{div}\left(\frac{\nabla f_t}{\sqrt{\varepsilon^2 + \|\nabla f_t\|^2}}\right). \tag{6.14}$$

Choosing a small ε makes the flow closer to a minimization of the total variation, but makes the computation unstable.

In practice, the flow is computed with a discrete gradient descent (6.8). For the smoothed total variation flow to converge, one needs to impose that $\tau < \varepsilon/4$, which shows that being more faithful to the TV energy requires smaller time steps and thus slower algorithms.



$\varepsilon = 0.01$ $\varepsilon = 0.05$ $\varepsilon = 0.1$ $\varepsilon = 0.5$

Figure 6.6: *Regularized gradient norm* $\sqrt{\|\nabla f(x)\|^2 + \varepsilon^2}$.

Figure 6.3 shows a comparison between the heat flow and the total variation flow for a small value of ε. This shows that the TV flow smooth less the edges than heat diffusion, which is consistent with the ability of the TV energy to better characterize sharp edges.

Table 15 details the implementation of this smoothed TV flow.

```
% step size
tau = epsilon/5;
% number of iterations to reach T.
niter = round(T/tau);
% initial solution for time t=0.
ftv = f;
for i=1:niter
    % gradient and norm of gradient
    Gr = grad(ftv); d = sqrt(sum3(Gr.^2,3));
    G = -div( Gr ./ repmat( sqrt( epsilon^2 + d.^2 ) , [1 1 2]) );
    % descent step
    ftv = ftv - tau*G;
end
```

Matlab code 15: *Smoothed total variation flow, computed by gradient descent. The input is the image* `f`*, the output is* `ftv`*, and approximation of the solution* f_T *of the smoothed TV flow* (6.14) *at time* $t = T$*. The regularization parameter* ε *is stored in* `epsilon`*.*

6.2.4 PDE Flows for Denoising

PDE flows can be used to remove noise from an observation $f = f_0 + w$. As detailed in Section 5.1.2 a simple noise model assumes that each pixel is corrupted with a Gaussian noise $w[n] \sim \mathcal{N}(0, \sigma)$, and that these perturbations are independent (white noise).

The denoising is obtained using the PDE flow within initial image f at time $t = 0$

$$\frac{\partial f_t}{\partial t} = -\operatorname{grad}_{f_t} J \quad \text{and} \quad f_{t=0} = f.$$

An estimator $\tilde{f} = f_{t_0}$ is obtained for a well chose $t = t_0$. Figure 6.7 shows examples of Sobolev and TV flows for denoising.

Since f_t converges to a constant image when $t \to +\infty$, the choice of t_0 corresponds to a tradeoff between removing enough noise and not smoothing too much the edges in the image. This is usually a difficult task. During simulation, if one has access to the clean image f_0, one can monitor the denoising error $\|f_0 - f_t\|$ and choose the $t = t_0$ that minimizes this error. Figure 6.10, top row, shows an example of this oracle estimation of the best stopping time.

6.3 Sparsity Priors

6.3.1 Ideal sparsity prior.

As detailed in Chapter 4, it is possible to use an orthogonal basis $\mathcal{B} = \{\psi_m\}_m$ to efficiently approximate an image f in a given class $f \in \Theta$ with a few atoms from $\mathcal{B}$.

Figure 6.7: *Denoising using f_t displayed for various time t for Sobolev (top) and TV (bottom) flows.*

To measure the complexity of an approximation with $\mathcal{B}$, we consider the ℓ^0 prior, which counts the number of non-zero coefficients in $\mathcal{B}$

$$J_0(f) = \#\{m \setminus \langle f, \psi_m \rangle \neq 0\} = \|a\|_0 \quad \text{where} \quad a[m] = \langle f, \psi_m \rangle.$$

We have introduced the ℓ^0 pseudo-norm $\|a\|_0$, which we treat here as an ideal sparsity measure for the coefficients a of f in $\mathcal{B}$.

Natural images are not exactly composed of a few atoms, but they can be well approximated by a function f_M with a small ideal sparsity $M = J_0(f)$. In particular, the best M-term approximation defined in (4.3) is defined by

$$f_M = \sum_{|\langle f, \psi_m \rangle| > T} \langle f, \psi_m \rangle \psi_m \quad \text{where} \quad M = \#\{m \setminus |\langle f, \psi_m \rangle| > T\}.$$

As detailed in Section 4.2, discontinuous images with bounded variation have a fast decay of the approximation error $\|f - f_M\|$. Natural images f are well approximated by images with a small value of the ideal sparsity prior J_0.

Figure 6.8 shows an examples of decomposition of a natural image in a wavelet basis, $\psi_m = \psi^{\omega}_{j,n}$ $m = (j, n, \omega)$. This shows that most $\langle f, \psi_m \rangle$ are small, and hence the decomposition is quite sparse.

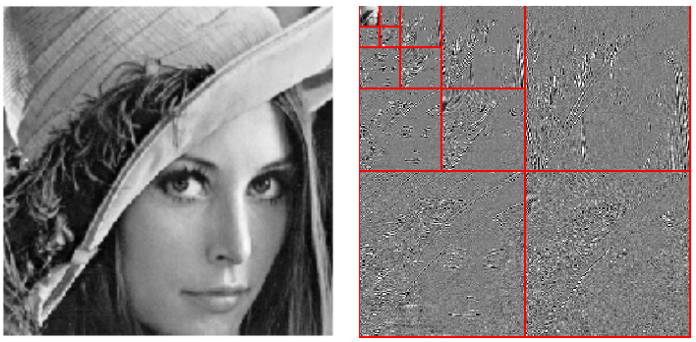


Image f Coefficients $\langle f, \psi_m \rangle$

Figure 6.8: *Wavelet coefficients of natural images are relatively sparse.*

6.3.2 Convex relaxation

Unfortunately, the ideal sparsity prior J_0 is difficult to handle numerically because $J_0(f)$ is not a convex function of f. For instance, if f and g have non-intersecting supports of there coefficients in $\mathcal{B}$, then $J_0((f+g)/2) = J_0(f) + J_0(g)$, which shows the highly non-convex behavior of J_0.

This ideal sparsity J_0 is thus not amenable to minimization, which is an issue to solve general inverse problems considered in Section 7.

We consider a family of ℓ^q priors for $q > 0$, intended to approximate the ideal prior J_0

$$J_q(f) = \sum_m |\langle f, \psi_m \rangle|^q.$$

As shown in Figure 6.9, the unit balls in $\mathbb{R}^2$ associated to these priors are shrinking toward the axes, which corresponds to the unit ball for the ℓ^0 pseudo norm. In some sense, the J_q priors are becoming closer to J_0 as q tends to zero, and thus J_q favors sparsity for small q.

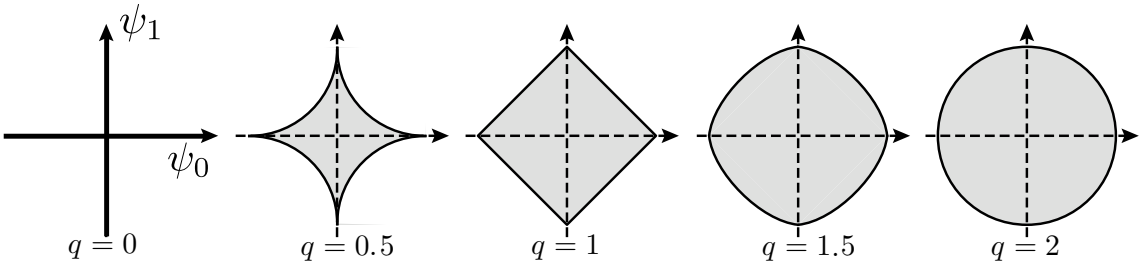


Figure 6.9: *ℓ^q balls $\{x \setminus J_q(x) \leqslant 1\}$ for varying q.*

The prior J_q is convex if and only if $q \geqslant 1$. To reach the highest degree of sparsity while using a convex prior, we consider the ℓ^1 sparsity prior J_1, which is thus defined as

$$J_1(f) = \|(\langle f, \psi_m \rangle)\|_1 = \sum_m |\langle f, \psi_m \rangle|. \tag{6.15}$$

In the following, we consider discrete orthogonal bases $\mathcal{B} = \{\psi_m\}_{m=0}^{N-1}$ of $\mathbb{R}^N$.

6.4 Regularization for Denoising

Instead of performing a gradient descent flow for denoising as detailed in Section 6.2.4 and select a stopping time, one can formulate an optimization problem. The estimator is then defined as a solution of this optimization. This setup has the advantage as being well defined mathematically even for non-smooth priors such as the TV prior J_{TV} or the sparsity prior J_1. Furthermore, this regularization framework is also useful to solve general inverse problems as detailed in Chapter 7.

6.4.1 Regularization

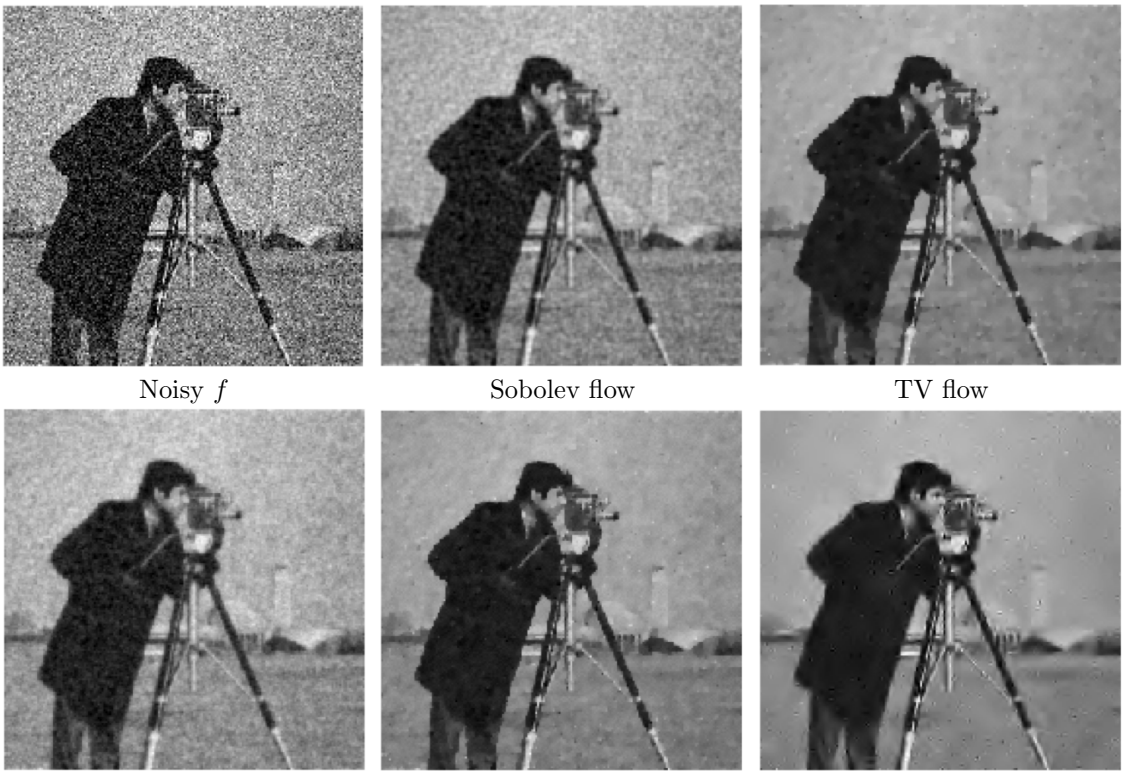
Given some noisy image $f = f_0 + w$ of N pixels and a prior J, we consider the convex optimization problem

$$f_\lambda^\star \in \operatorname*{argmin}_{g \in \mathbb{R}^N} \frac{1}{2}\|f - g\|^2 + \lambda J(g), \tag{6.16}$$

where $\lambda > 0$ is a Lagrange multiplier parameter that weights the influence of the data fitting term $\|f - g\|^2$ and the regularization term $J(g)$.

If one has at his disposal a clean original image f_0, one can minimize the denoising error $\|f_\lambda^\star - f_0\|$, but it is rarely the case. In practice, this parameter should be adapted to the noise level and to the regularity of the unknown image f_0, which might be a non trivial task.

We note that since we did not impose J to be convex, the problem (6.16) might have several optimal solutions.



Noisy f | Sobolev flow | TV flow

Sobolev reg | TV reg | TI wavelets

Figure 6.10: *Denoising using PDE flows and regularization.*

An estimator is thus defined as

$$\tilde{f} = f_\lambda^\star$$

for a well chosen λ.

If J is differentiable and convex, one can compute the solution of (6.16) through a gradient descent

$$f^{(k+1)} = f^{(k)} - \tau \left(f^{(k)} - \lambda \operatorname{grad} J(f^{(k)}) \right) \tag{6.17}$$

where the descent step size $\tau > 0$ should be small enough. This gradient descent is similar to the time-discretized minimization flow (6.8), excepted that the data fitting term prevent the flow to converge to a constant image.

Unfortunately, priors such as the total variation J_{TV} or the sparsity J_1 are non-differentiable. Some priors such as the ideal sparsity J_0 might even be non-convex. This makes the simple gradient descent not usable to solve for (6.16). In the following Section we show how to compute $f_\lambda^\star$ for several priors.

6.4.2 Sobolev Regularization

The discrete Sobolev prior defined in (6.5) is differentiable, and the gradient descent (6.17) reads

$$f^{(k+1)} = (1-\tau) f^{(k)} + \tau f - \tau\lambda\Delta J(f^{(k)}.$$

Since $J(f) = \|\nabla f\|^2$ is quadratic, one can use a conjugate gradient descent, which converges faster.

The solution $f_\lambda^\star$ can be computed in closed form as the solution of a linear system

$$f_\lambda^\star = (\mathrm{Id}_N - \lambda\Delta)^{-1} f,$$

which shows that the regularization (6.16) is computing an estimator that depends linearly on the observations f.

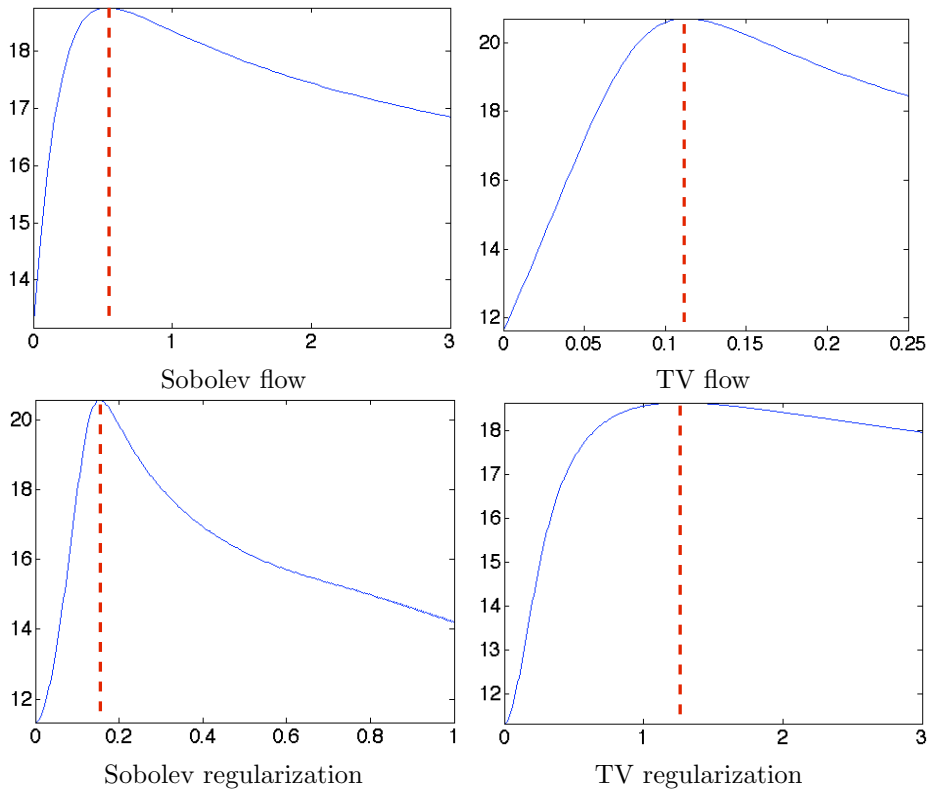


Figure 6.11: *SNR as a function of time t for flows (top) and λ for regularization (bottom).*

If the differential operators are computed with periodic boundary conditions, this linear system can be solved exactly over the Fourier domain

$$\hat{f}^\star_\lambda[\omega] = \frac{1}{1+\lambda\rho_\omega^2}\hat{f}[\omega] \tag{6.18}$$

where ρ_ω depends on the discretization of the Laplacian, see for instance (6.4).

Equation (7.17) shows that denoising using Sobolev regularization corresponds to a low pass filtering, whose strength is controlled by λ. This should be related to the solution (6.11) of the heat equation, which also corresponds to a filtering, but using a Gaussian low pass kernel parameterized by its variance t^2.

This Sobolev regularization denoising is a particular case of the linear estimator considered in Section 5.2. The selection of the parameter λ is related to the selection of an optimal filter as considered in Section 5.2.2, but with the restriction that the filter is computed in a parametric family.

6.4.3 TV Regularization

The total variation prior J_{TV} defined in (6.6) is non-differentiable. One can either use a smoothed approximation of the prior, or use an optimization algorithm not based on a gradient descent.

Smoothed TV. The TV prior can be approximated to obtain the prior $J^\varepsilon_{\mathrm{TV}}(g)$ defined in (6.12), which is differentiable with respect to g. Using the gradient of this prior defined in (6.13), one obtains the following instantiation of the gradient descent (6.17)

$$f^{(k+1)} = (1-\tau)f^{(k)} + \tau f + \lambda\tau \,\mathrm{div}\left(\frac{\nabla f_t}{\sqrt{\varepsilon^2 + \|\nabla f_t\|^2}}\right). \tag{6.19}$$

which converge to the unique minimizer $f^\star_\lambda$ of (6.16).

Chambolle's algorithm. Chambolle in [5] detail an algorithm to minimize exactly the TV denoising problem

$$f_\lambda^\star = \operatorname*{argmin}_{g\in\mathbb{R}^N} \frac{1}{2}\|f-g\|^2 + \lambda\|g\|_{\mathrm{TV}}. \tag{6.20}$$

It uses a relationship between the vectorial ℓ^1 and ℓ^∞ norms

$$\|v\|_1 = \sum_{m=0}^{N-1} \|v[m]\| \quad \text{and} \quad \|v\|_\infty = \max_{0\leqslant m<N} \|v[m]\|$$

where each $v[m]\in\mathbb{R}^2$ and $v\in\mathbb{R}^{N\times 2}$. One has

$$\|v\|_1 = \max_{\|w\|_\infty\leqslant 1} \langle w,\, u\rangle$$

which allows one to re-write the optimization (6.20) as

$$\min_{g\in\mathbb{R}^N} \max_{\|w\|_\infty\leqslant 1} \frac{1}{2}\|f-g\|^2 + \lambda\langle w,\, \nabla g\rangle.$$

Exchanging the roles of the min and the max, one proves that the solution of (6.20) is re-written as

$$f_\lambda^\star = f + \lambda\,\mathrm{div}(w^\star) \tag{6.21}$$

where

$$w^\star \in \operatorname*{argmin}_{\|w\|_\infty\leqslant 1} \|f+\lambda\,\mathrm{div}(w^\star)\|^2. \tag{6.22}$$

The convex optimization problem (6.22) computes a dual vector field $w^\star\in\mathbb{R}^{N\times 2}$, from which the denoised image is recovered using (6.21).

The dual problem (6.22) is the minimization of a quadratic functional subject to a convex ℓ^∞ constraint. It can thus be solved using for instance a projected gradient descent

$$w^{(k+1)}[m] = \frac{\tilde{w}^{(k)}[m]}{\max(|\tilde{w}^{(k)}[m]|,1)} \quad \text{where} \quad \tilde{w}^{(k)} = w^{(k)} + \tau\nabla(f/\lambda + \mathrm{div}(w^{(k)})).$$

If the gradient step size satisfy $0<\tau<1/4$, one can prove that

$$f+\lambda\,\mathrm{div}(w^{(k)}) \longrightarrow f_\lambda^\star \quad \text{when} \quad k\to+\infty.$$

```
% Initialization of the dual variables.
G = zeros(n,n,2);
for i=1:niter
    % Gradient of the energy
    dG = grad( div(G) - f/lambda );
    % gradient descent
    G = G + tau*dG;
    % Projection on Linfty constraints
    d = repmat( sqrt(sum(G.^2,3)), [1 1 2] ); % norm of the vectors
    G = G ./ max(d,ones(n,n,2));
end
% Final solution from the dual variables.
ftv = f-lambda*div(G);
```

Matlab code 16: *Total variation regularization computed using Chambolle's algorithm. The input is the noisy image* `f`*, the output is* `ftv`*, the solution* f_λ *of* (6.20)*. The regularization parameter* λ *is stored in* `lambda`*.*

6.4.4 Sparse Regularization and Thresholding

Given some orthogonal basis $\{\psi_m\}_m$ of $\mathbb{R}^N$, the denoising by regularization (6.16) is written using the sparsity J_0 and J_1 as

$$f^\star_{\lambda,q} = \operatorname*{argmin}_{g\in\mathbb{R}^N} \frac{1}{2}\|f-g\|^2 + \lambda J_q(f)$$

for $q=0$ or $q=1$. It can be re-written in the orthogonal basis as

$$f^\star_{\lambda,q} = \sum_m a^\star_{\lambda,q}[m]\psi_m$$

$$a^\star_{\lambda,q} = \operatorname*{argmin}_{b\in\mathbb{R}^N} \sum_m \frac{1}{2}|a[m]-b[m]|^2 + \lambda\phi_q(b[m])$$

where $a[m] = \langle f, \psi_m\rangle$, and with

$$\phi_1(x) = |x| \quad \text{and} \quad \phi_0(x) = \begin{cases} 0 & \text{if } x=0, \\ 1 & \text{otherwise.} \end{cases}$$

Each coefficients of the denoised image is the solution of

$$a^\star_{\lambda,q}[m] = \operatorname*{argmin}_{\alpha\in\mathbb{R}} \frac{1}{2}|a[m]-\beta|^2 + \lambda\phi_q(\beta)$$

and one can shows that this optimization is solved exactly in closed form using thresholding

$$a^\star_{\lambda,q}[m] = S^q_T(a[m]) \quad \text{where} \quad \begin{cases} T=\sqrt{2\lambda} & \text{for } q=0, \\ T=\lambda & \text{for } q=1, \end{cases} \tag{6.23}$$

where S^0_T is the hard thresholding introduced in (5.3), and S^1_T is the soft thresholding introduced in (5.4).

One thus has

$$f_{\lambda,q} = \sum_m S^q_T(\langle f, \psi_m\rangle)\psi_m.$$

As detailed in Section 5.3, these denoising methods has the advantage that the threshold is simple to set for Gaussian white noise w of variance σ^2. Theoritical values indicated that $T=\sqrt{2\log(N)}\sigma$ is asymptotically optimal, see Section 5.3.3. In practice, one should choose $T\approx 3\sigma$ for hard thresholding (ℓ^0 regularization), and $T\approx 3\sigma/2$ for soft thresholding (ℓ^1 regularization), see Figure 5.14.

Chapter 7

Inverse problems

7.1 Inverse Problems Regularization

Increasing the resolution of signals and images requires to solve an ill posed inverse problem. This corresponds to inverting a linear measurement operator that reduces the resolution of the image. This chapter makes use of convex regularization introduced in Chapter 6 to stabilize this inverse problem.

7.1.1 Inverse Problem in Imaging

Solving a discrete inverse problems corresponds to recovering a high resolution signal or image $f_0 \in \mathbb{R}^N$ from a small number of P noisy measurements

$$y = \Phi f_0 + w \in \mathbb{R}^P.$$

Since $P \ll N$, the linear operator $\Phi : \mathbb{R}^N \mapsto \mathbb{R}^P$ is not invertible, and has a large kernel of dimension $N - P$.

The difficulty of this inversion is further increased by the addition of some measurement noise $w \in \mathbb{R}^P$. The simplest noise model assume that w is a Gaussian white noise of variance σ^2, see Section 5.1.2.

Denoising. The case of the identity operator $\Phi = \mathrm{Id}_N$, $P = N$ corresponds to the classical denoising problem, already treated in Chapters 5 and 6.

For a general operator Φ, the recovery of f_0 is more challenging, and this requires to perform both an inversion and a denoising. For many problem, this two goals are in contradiction, since usually inverting the operator increases the noise level.

De-blurring. This is for instance the case for the deblurring problem, where Φ is a translation invariant operator, that corresponds to a low pass filtering with some kernel h

$$\Phi f = f \star h. \tag{7.1}$$

In this case $N = P$, and this operator is invertible if

$$\forall\, \omega, \quad \hat{h}[\omega] \neq 0,$$

and applying the inverse filter over the Fourier domain computes $f^+ \in \mathbb{R}^N$ defined as

$$\hat{f}^+[\omega] = \hat{y}[\omega]/\hat{h}[\omega] = \hat{f}_0[\omega] + \hat{w}[\omega]/\hat{h}[\omega]. \tag{7.2}$$

For low pass filter, the Fourier transform $\hat{h}[\omega]$ is small for high frequency, and the estimation f^+ is bad because of high frequency explosion of the noise.

This shows the necessity to replace the brute force inversion (7.2) by a more gentle regularization. Doing so performs a denoising that reduces the performance of the inversion but is mandatory to avoid the noise explosion at high frequencies.

Super-resolution. Super-resolution corresponds to an even more ill-posed inverse problem, where the acquisition blur is followed by a coarse sub-sampling

$$\Phi f = f \star h \downarrow_k, \quad \text{where} \quad (a \downarrow_k)[n] = a[nk] \tag{7.3}$$

so that $P = N/k^d$ where d is the dimension of the data ($d = 2$ for images). Figure 7.1, middle, shows an example of a low resolution image Φf_0.

Inverting the operator defined in (7.3) has important industrial application to upsample the content of digital photos and to compute high definition videos from low definition videos.

Inpainting. Inpainting corresponds to interpolating missing pixels in an image. This is modeled by a diagonal operator over the spacial domain

$$(\Phi f)(x) = \begin{cases} 0 & \text{if} \quad x \in \Omega, \\ f(x) & \text{if} \quad x \notin \Omega. \end{cases} \tag{7.4}$$

where $\Omega \subset \{0, \ldots, N-1\}$ is a set of missing pixels, and $P = N - |\Omega|$. Figure 7.1, right, shows an example of damaged image Φf_0.

Original f_0 Low resolution Φf_0 Masked Φf_0

Figure 7.1: *Example of inverse problem operators.*

7.1.2 Inverse Problem Regularization

The ill-posed problem of recovering an approximation of the high resolution image $f_0 \in \mathbb{R}^N$ from noisy measures $y \in \mathbb{R}^P$ is regularized by solving a convex optimization problem

$$f^\star \in \operatorname*{argmin}_{f \in \mathbb{R}^N} \frac{1}{2}\|y - \Phi f\|^2 + \lambda J(f) \tag{7.5}$$

where $\|y - \Phi f\|^2$ is the data fitting term and $J(f)$ is a convex prior.

The Lagrange multiplier λ weights the importance of these two terms, and is in practice difficult to set. Simulation can be performed on high resolution signal f_0 to calibrate the multiplier by minimizing the super-resolution error $\|f_0 - \tilde{f}\|$, but this is usually difficult to do on real life problems.

In the case where there is no noise, $\sigma = 0$, the Lagrange multiplier λ should be set as small as possible. In the limit where $\lambda \to 0$, the unconstrained optimization problem (7.5) becomes a constrained optimization

$$f^\star = \operatorname*{argmin}_{f \in \mathbb{R}^N, \Phi f = y} J(f). \tag{7.6}$$

7.1.3 L^2 regularization

The simplest prior, that avoids the exposition of the noise during the inversion, is the ℓ^2 norm

$$J(f) = \frac{1}{2}\|f\|^2$$

that ensures that the recovered signal or image has a bounded energy.

In the noise-free setting, one obtains the pseudo inverse operator that compute the solution $f^\star$

$$f^\star = \operatorname*{argmin}_{\Phi f = y} \|f\|^2 = \Phi^+ y \quad \text{where} \quad \Phi^+ = \Phi^*(\Phi\Phi^*)^{-1}. \tag{7.7}$$

It corresponds to inverting the operator on the complementary of its kernel.

For noisy measures, on performs a quadratic regularization

$$f^\star = \operatorname*{argmin}_{f \in \mathbb{R}^N} \|y - \Phi f\|^2 + \lambda \|f\|^2$$

whose closed form solution is obtained by solving a regularized (non-singular) linear system

$$f^\star = (\Phi^*\Phi + \lambda \mathrm{Id}_N)^{-1}\Phi^* y. \tag{7.8}$$

7.1.4 Sobolev Regularization

The discrete Sobolev prior introduced in (6.5) regularizes the inverse by computing a linear denoising. This corresponds to minimizing

$$f^\star \in \operatorname*{argmin}_{f \in \mathbb{R}^N} \|y - \Phi f\|^2 + \lambda \|\nabla f\|^2.$$

The solution depends linearly on the data

$$f^\star = (\Phi^*\Phi - \lambda\Delta)^{-1}\Phi^* y, \tag{7.9}$$

and the parameter λ controls the amount of denoising.

The solutions of (7.8) and (7.9) depend linearly on the measures y, and can be computed numerically using a conjugate gradient descent. For convolution operator, the solution can be computed directly over the Fourier domain, see Section 7.2.1.

7.1.5 Total Variation Regularization

Exact total variation. The discrete total variation prior $J_{\mathrm{TV}}(f)$ defined in (6.6) is a convex but non differentiable function of the image f, so that the regularization problem (7.5)

$$f^\star \in \operatorname*{argmin}_{f \in \mathbb{R}^N} \frac{1}{2}\|y - \Phi f\|^2 + \lambda \sum_n \|\nabla f[n]\| \tag{7.10}$$

cannot be solved using a gradient descent. Section 7.1.7 details a class of algorithm that can solve (7.5) for both TV and sparse regularization. We note that since (7.5) is not a strictly convex, the minimizer is not unique in general.

Smoothed total variation. One can use the smoothed total variation prior $J^\varepsilon_{\mathrm{TV}}$ for some small parameter $\varepsilon > 0$ and solve (7.5) using a gradient descent that generalize (6.19) for inverse problems

$$f^{(k+1)} = f^{(k)} - \tau\Phi^*(\Phi f^{(k)} - y) + \lambda\tau \operatorname{div}\left(\frac{\nabla f_t}{\sqrt{\varepsilon^2 + \|\nabla f_t\|^2}}\right)$$

that converges to $f^\star$ that minimizes (7.5) if

$$0 < \tau < (\|\Phi^*\Phi\| + 4/\varepsilon)^{-1}.$$

7.1.6 Sparse Regularization

Sparse ℓ^1 regularization in an orthogonal basis $\{\psi_m\}_m$ of $\mathbb{R}^N$ makes use of the J_1 prior defined in (6.15), so that the inversion is obtained by solving the following convex program

$$f^\star \in \operatorname*{argmin}_{f\in\mathbb{R}^N} \frac{1}{2}\|y-\Phi f\|^2 + \lambda \sum_m |\langle f,\, \psi_m\rangle|. \tag{7.11}$$

This corresponds to the basis pursuit denoising for sparse approximation introduced by Chen, Donoho and Saunders in [6]. The resolution of (7.11) can be perform using an iterative thresholding algorithm as detailed in Section 7.1.7.

For noiseless measurements $y = \Phi f_0$, one solves a constraint basis pursuit problem

$$f^\star \in \operatorname*{argmin}_{\Phi f=y} \sum_m |\langle f,\, \psi_m\rangle|.$$

This can be recasted as a convex linear program, which can in turn by solved by various solver such as simplex, interior points, or Douglas-Rachford iterations.

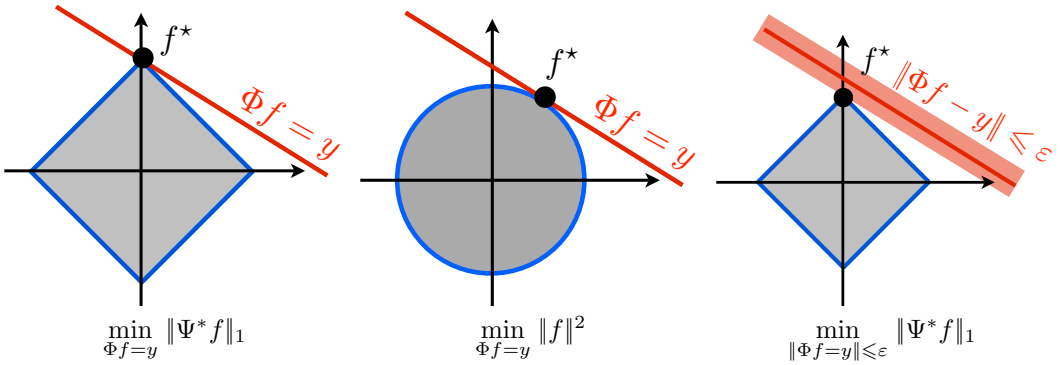


Figure 7.2: *Geometry of convex optimizations.*

7.1.7 Proximal Algorithms for TV and Sparsity

This section details an iterative algorithm that compute a solution of (7.5) for either the TV prior $J = J_{\mathrm{TV}}$ or the sparse ℓ^1 prior, which corresponds respectively to the minimizations (7.10) and (7.11).

This algorithm was derived by several authors, among which [14, 11], and belongs to the general family of forward-backward splitting in proximal iterations [8]. We note that faster algorithms can be used, such as Nesterov scheme [22].

Surrogate functionals. The energy to minimize in (7.10) and (7.11) is written as

$$E(f) = \frac{1}{2}\|\Phi f - y\|^2 + \lambda J(f).$$

The difficulty is the presence of the operator Φ in the ℓ^2 norm, which makes this problem significantly more difficult than the simple denoising by regularization (6.16).

To derive an iterative algorithm, we modify the energy $E(f)$ to obtain a surrogate functional $E(f, f^{(k)})$ whose minimization corresponds to a simpler denoising problem.

Given some guess $f^{(k)} \in \mathbb{R}^N$ of the solution $f^\star$, the surrogate functional is defined as

$$E(f, f^{(k)}) = E(f) - \frac{1}{2}\|\Phi f - \Phi f^{(k)}\|^2 + \frac{1}{2\tau}\|f - f^{(k)}\|^2.$$

One has

$$E(f, f^{(k)}) \geqslant E(f) \quad \text{and} \quad E(f^{(k)}, f^{(k)}) = E(f^{(k)}) \tag{7.12}$$

so that $E(f, f^{(k)})$ is a proxy for the minimization of $E(f)$.

Proximal iterations. A proximal iterative algorithm computes

$$f^{(k+1)} = \operatorname*{argmin}_{f\in\mathbb{R}^N} E(f, f^{(k)}).$$

Property (7.12) guarantees that E is decaying

$$E(f^{(k+1)}) \leqslant E(f^{(k)}).$$

Furtheremore, one has

$$E(f, f^{(k)}) = C + \frac{1}{2\tau}\|f - f^{(k)} + \tau\Phi^*(\Phi f^{(k)} - y)\|^2 + \lambda\sum_m |\langle f, \psi_m\rangle|$$

where C is independent of f. Defining a proximal operator

$$\operatorname{prox}_{\lambda J}(\tilde{f}) = \operatorname*{argmin}_{f\in\mathbb{R}^N} \frac{1}{2}\|\tilde{f} - f\|^2 + \lambda J(f), \tag{7.13}$$

that corresponds to the variational denoiser introduced in (6.16), one thus has

$$f^{(k+1)} = \operatorname{prox}_{\lambda\tau J}\left(\tilde{f}^{(k)}\right) \quad \text{where} \quad \tilde{f}^{(k)} = f^{(k)} - \tau\Phi^*(\Phi f^{(k)} - y).$$

One can prove, see [], that if $\tau < 2/\|\Phi^*\Phi\|_S$, then

$$f^{(k)} \to f^\star.$$

Noiseless case. If $\sigma = 0$, so that one observe noiseless measures $y = \Phi f_0$, an heuristic to compute approximately the solution $f^\star$ of (7.6) is to use a decaying value of $\lambda = \lambda^{(k)}$ during the iterations. One can for instance use $\lambda_k = \lambda_{\max}/k$, although there is no proof of convergence to $f^\star$.

constrained problem. The constrained problem

$$f_\varepsilon^\star \in \operatorname*{argmin}_{\|\Phi f - y\|\leqslant\varepsilon} \sum_m |\langle f, \psi_m\rangle|.$$

is equivalent to the problem (7.5), in the sense that $f^\star$ is a solution of (7.5) for a suitable value of λ that depends on ε. Unfortunately, the correspondence between λ and ε is unknown and depends on y.

An heuristic to automatically find this correspondence is to iteratively update the value of $\lambda = \lambda^{(k)}$

$$\lambda^{(k+1)} = \lambda^{(k)}\frac{\varepsilon}{\|\Phi f^{(k)} - y\|}.$$

Sparse regularization. For the case of $J = J_1$, the proximal denoising operator (7.13) is computed in closed form using a soft thresholding, as already noticed in (6.23).

The resulting proximal iterative algorithm corresponds to the iterative soft thresholding algorithm, that alternates a gradient descent step

$$\tilde{f}^{(k)} = f^{(k)} - \tau\Phi^*(\Phi f^{(k)} - y). \tag{7.14}$$

and soft thresholding

$$f^{(k+1)} = \sum_m S^1_{\lambda\tau}(\langle\tilde{f}^{(k)}, \psi_m\rangle)\psi_m. \tag{7.15}$$

Table 17 details the implementation of this method, when the data is assumed to be sparse in the trivial identity basis.

TV regularization. For the case of $J = J_{\mathrm{TV}}$, the proximal operator (7.13) does not have a closed form solution. One thus has to use inner iteration of Chambolle's scheme (6.22) to compute the proximal map.

```
% gradient descent step size
tau = 1.9/norm(Phi)^2;
% iterative thresholding
fspars = y; % initialization
for i=1:niter
    fspars = perform_thresholding(fspars + tau*Phi'*(y-Phi*fspars), lambda*tau, 'soft');
end
```

Matlab code 17: *Iterative thresholding (iteration of* (7.14) *and* (7.15)*) to solve the inverse problem regularization with ℓ^1 prior, in the case where the sparsity basis is the identity. The input is the operator Φ given as a matrix* `Phi`*, the observation y stored in* `y` *and the regularization parameter λ stored in* `lambda`*. The output is the solution* `fspars`*.*

7.2 Example of Inverse Problems

We detail here some inverse problem in imaging that can be solved using quadratic regularization or non-linear TV and sparse regularization.

7.2.1 Deconvolution

The blurring operator (7.1) is diagonal over Fourier, so that quadratic regularization are easily solved using Fast Fourier Transforms when considering periodic boundary conditions.

The pseudo-inverse $f^\star = f^+$ defined in (7.7) is computed as

$$\hat{f}^\star[\omega] = \begin{cases} \hat{y}[\omega]/\hat{h}[\omega] & \text{if} \quad \hat{h}[\omega] \neq 0, \\ 0 & \text{if} \quad \hat{h}[\omega] = 0. \end{cases}$$

The quadratic regularization defined in (7.8) is computed as

$$\hat{f}^\star[\omega] = \frac{\hat{h}[\omega]^*}{|h[\omega]|^2 + \lambda} \hat{y}[\omega] \tag{7.16}$$

and the Sobolev regularization defined in (7.9) satisfy

$$\hat{f}^\star[\omega] = \frac{\hat{h}[\omega]^*}{|h[\omega]|^2 - \lambda \rho^2[\omega]} \hat{y}[\omega] \tag{7.17}$$

where $\rho[\omega]^2$ depends on the discretization of the Laplacian operator, and is given in (6.4) for a finite difference implementation. Table 18 details the implementation of both regularization.

```
% Fourier transform of the filter
phiF = fft2(phi);
% Perform the L2 inversion.
fl2 = real( ifft2( yF .* conj(phiF) ./ ( abs(phiF).^2 + epsilon) ) );
% Compute the Sobolev prior penalty (rescale to [0,1]).
x = [0:n/2-1, -n/2:-1];
[Y,X] = meshgrid(x,x);
\rho = (X.^2 + Y.^2)*(2/n)^2;
% Perform the sobolev inversion.
fsob = real( ifft2( fft2(y) .* conj(phiF) ./ ( abs(phiF).^2 + lambda*\rho) ) );
```

Matlab code 18: *ℓ^2 and Sobolev regularization. The filter ϕ is given in* `phi`*, the regularization parameter λ in* `lambda` *and the obervation y in* `y`*. The output are the ℓ^2 regularization* `fl2` *and the Sobolev regularization* `fsob`*.*

Both TV and sparse regularization cannot be solved as easily and necessitate iterative proximal algorithm for their resolution. We now give two example of such deconvolution for a spike and wavelet orthogonal bases.

Sparse Spikes. Sparse spikes deconvolution makes use of sparsity in the spacial domain, which corresponds to the orthogonal basis of Diracs $\psi_m[n] = \delta[n-m]$. This sparsity was first introduced in the seismic imaging community [], where the signal f_0 represent the change of density in the underground and is assumed to be composed of a few Diracs impulse.

In a simplified linearized 1D set-up, ignoring multiple reflexions, the acquisition of underground data f_0 is modeled as a convolution $y = h \star f_0 + w$, where h is a so-called "wavelet" signal sent in the ground. This should not be confounded with the construction of orthogonal wavelet bases detailed in Chapter 3, although the term "wavelet" originally comes from seismic imaging.

The wavelet filter h is typically a band pass signal that perform a tradeoff between space and frequency concentration especially tailored for seismic exploration. Figure (7.3) shows a typical wavelet that is a second derivative of a Gaussian, together with its Fourier transform. This shows the large amount of information removed from f during the imaging process.

The sparse ℓ^1 regularization in the Dirac basis reads

$$f^\star = \operatorname*{argmin}_{f \in \mathbb{R}^N} \frac{1}{2}\|f \star h - y\|^2 + \lambda \sum_m |f[m]|.$$

Figure 7.3 shows the result of ℓ^1 minimization for a well chosen λ parameter, that was optimized in an oracle manner to minimize the error $\|f^\star - f_0\|$.

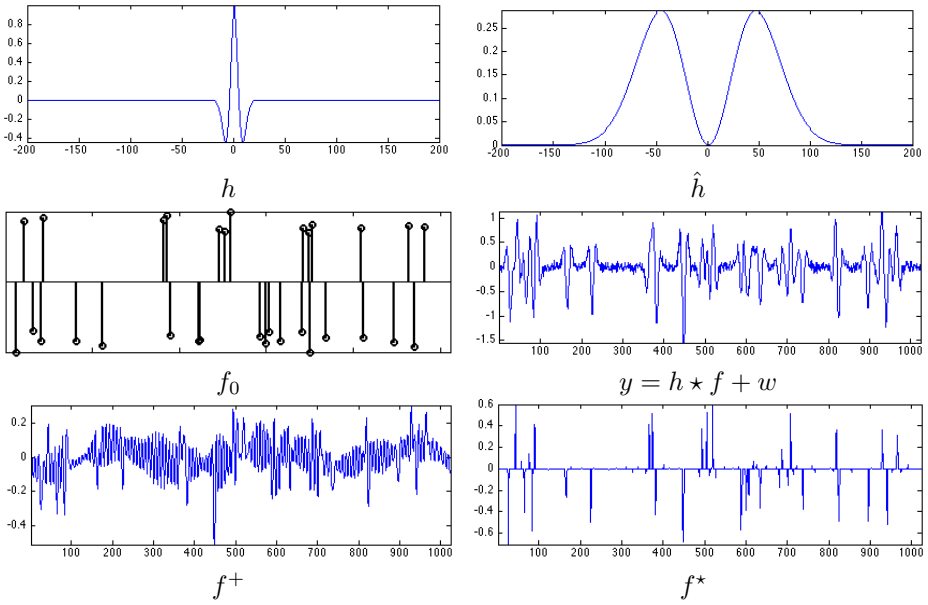


Figure 7.3: *Pseudo-inverse and ℓ^1 sparse spikes deconvolution.*

The iterative soft thresholding for sparse spikes inversion iterates

$$\tilde{f}^{(k)} = f^{(k)} - \tau h \star (h \star f^{(k)} - y)$$

and

$$f^{(k+1)}[m] = S^1_{\lambda\tau}(\tilde{f}^{(k)}[m])$$

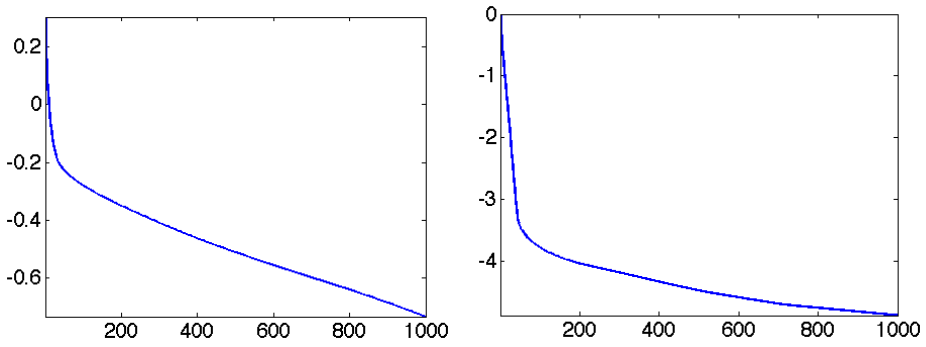
where the step size should obeys

$$\tau < 2/\|\Phi^*\Phi\| = 2/\max_\omega |\hat{h}(\omega)|^2$$

to guarantee convergence. Figure 7.4 shows the progressive convergence of the algorithm, both in term of energy minimization and iterates. Since the energy is not strictly convex, we note that convergence in energy is not enough to guarantee convergence of the algorithm.

Sparse in wavelets. Signal and image acquired by camera always contain some amount of blur because of objects being out of focus, movements in the scene during exposure, and diffraction. A simplifying assumption assumes a spatially invariant blur, so that Φ is a convolution

$$y = f_0 \star h + w.$$



$\log_{10}(E(f^{(k)})/E(f^\star)-1)$ $\log_{10}(\|f^{(k)}-f^\star\|/\|f_0\|)$

Figure 7.4: *Decay of the energy and convergence through the iterative thresholding iterations.*

In the following, we consider h to be a Gaussian filter of width $\mu > 0$. The number of effective measurements can thus be considered to be $P \sim 1/\mu$, since Φ nearly set to 0 large enough Fourier frequencies. Table 19 details the implementation of the sparse deconvolution algorithm.

Figures 7.5 and 7.6 shows examples of signal and image acquisition with Gaussian blur.

Sobolev regularization (7.17) improves over ℓ^2 regularization (7.16) because it introduces an uniform smoothing that reduces noise artifact. It however fail to recover sharp edge and thus does a poor job in inverting the operator. To recover sharper transition and edges, one can use either a TV regularization or a sparsity in an orthogonal wavelet basis.

Figure 7.5 shows the improvement obtained in 1D with wavelets with respect to Sobolev. Figure 7.6 shows that this improvement is also visible for image deblurring. To obtain a better result with fewer artifact, one can replace the soft thresholding in orthogonal wavelets in during the iteration (7.15) by a thresholding in a translation invariant tight frame as defined in (5.6).

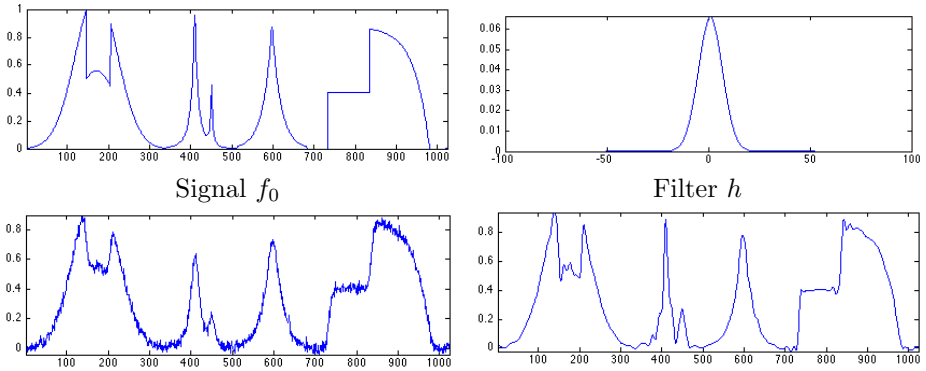


Figure 7.5: *Sparse 1D deconvolution using orthogonal wavelets.*

Figure 7.7 shows the decay of the SNR as a function of the regularization parameter λ. This SNR is computed in an oracle manner since it requires the knowledge of f_0. The optimal value of λ was used in the reported experiments.

7.2.2 Noiseless Inpainting

For the inpainting problem, the operator defined in (7.4) is diagonal in space

$$\Phi = \mathrm{diag}_m(\delta_{\Omega^c}[m]),$$

and is an orthogonal projector $\Phi^* = \Phi$.

In the noiseless case, to constrain the solution to lie in the affine space $\{f \in \mathbb{R}^N \setminus y = \Phi f\}$, we

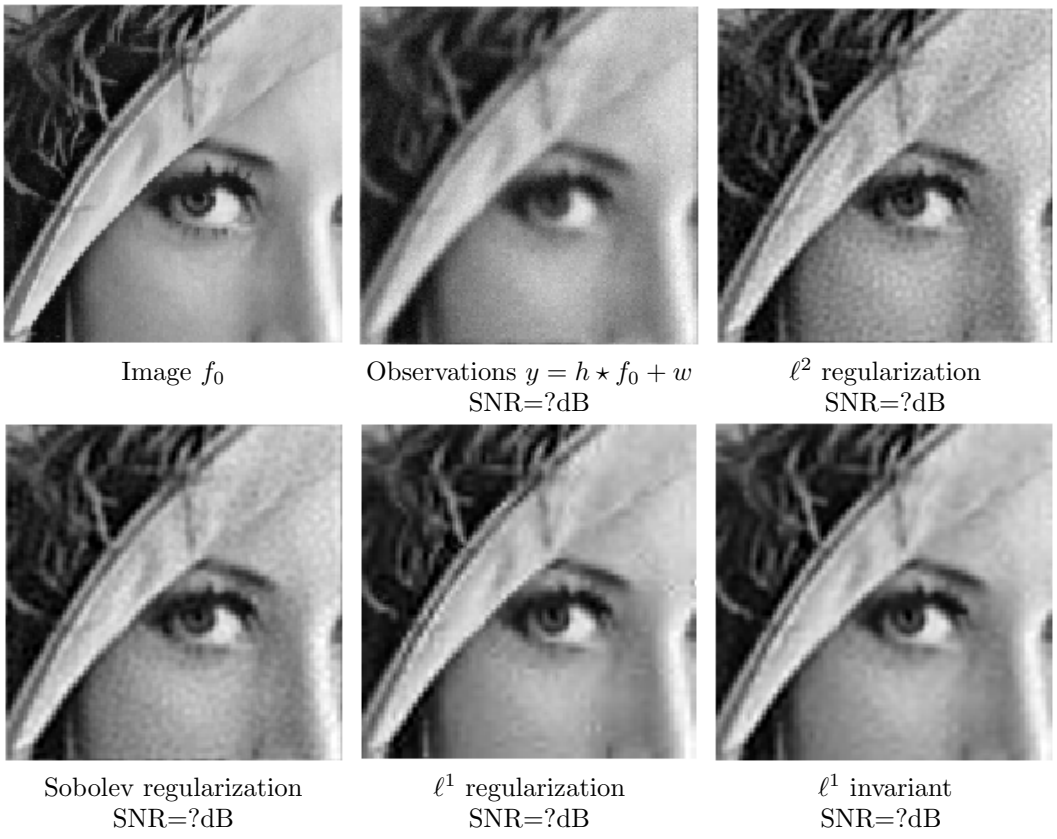


Figure 7.6: *Image deconvolution.*

use the orthogonal projector

$$\forall\, x, \quad P_y(f)(x) = \begin{cases} f(x) & \text{if} \quad x \in \Omega, \\ y(x) & \text{if} \quad x \notin \Omega. \end{cases}$$

Sobolev and TV inpainting. In the noiseless case, the recovery (7.6) is solved using a projected gradient descent. For the Sobolev energy, the algorithm iterates

$$f^{(k+1)} = P_y(f^{(k)} + \tau \Delta f^{(k)}).$$

which converges if $\tau < 2/\|\Delta\| = 1/4$. Figure 7.8 shows some iteration of this algorithm, which progressively interpolate within the missing area. Table 20 details the implementation of the inpainting with the Sobolev prior.

Figure 7.9 shows an example of Sobolev inpainting to achieve a special effect.

For the smoothed TV prior, the gradient descent reads

$$f^{(k+1)} = P_y\left(f^{(k)} + \tau \div \left(\frac{\nabla f^{(k)}}{\sqrt{\varepsilon^2 + \|\nabla f^{(k)}\|^2}}\right)\right)$$

which converges if $\tau < \varepsilon/4$.

Figure 7.10 compare the Sobolev inpainting and the TV inpainting for a small value of ε. The SNR is not improved by the total variation, but the result looks visually slightly better.

Sparsity inpainting. To inpaint using a sparsity prior without noise, we use a small value for λ. The iterative thresholding algorithm (7.15) is written as follow for $\tau = 1$,

$$f^{(k+1)} = \sum_m S^1_\lambda(\langle P_y(f^{(k)}),\, \psi_m\rangle)\psi_m$$

Figure 7.11 shows the improvevement obtained by the sparse prior over the Sobolev prior if one uses soft thresholding in a translation invariant wavelet frame.

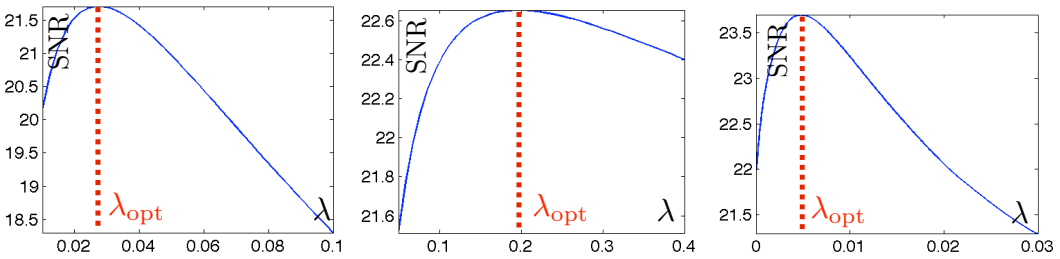


Figure 7.7: *SNR as a function of* λ.

```
% Fourier transform of the filter, assumed to be symetric
hF = fft2(h);
% Shortcut for the filtering operator
filt = @(x)real(ifft2(fft2(x).*hF));
% Iterative soft thresholding in wavelets.
fspars = y; % initialization
tau = 1.5/max(abs(hF(:)));
for i=1:niter
    fspars = fspars + tau * filt( y-filt(fspars) );
    fW = perform_wavelet_transf(fspars, Jmin, +1,options);
    fW = perform_thresholding( fW, lambda*tau, 'soft' );
    fspars = perform_wavelet_transf(fW, Jmin, -1,options);
end
```

Matlab code 19: *Deconvolution with* ℓ^1 *regularization in a wavelet basis. The filter* ϕ *is given in* `phi`*, the observations in* `y`*, the regularization parameter* λ *is* `lambda`*. The solution is given in* `fspars`*.*

```
% step size
tau = 1/5;
% initialization
fsob = y;
for i=1:niter
    % laplacien
    L = div( grad(fsob) );
    % gradient descent
    fsob = fsob + tau * L;
    % project on constraints
    fsob(mask==0) = y(mask==0);
end
```

Matlab code 20: *Inpainting with Sobolev regularization. The mask* Ω *is given in* `mask`*, so that the masked indices are* `f(mask)`*, the observations in* `y`*. The solution is given in* `fsob`*.*

7.2.3 Tomography Inversion

In medical imaging, a scanner device compute projection of the human body along rays $\Delta_{t,\theta}$ defined

$$x \cdot \tau_\theta = x_1 \cos\theta + x_2 \sin\theta = t$$

where we restrict ourself to 2D projection to simplify the exposition.

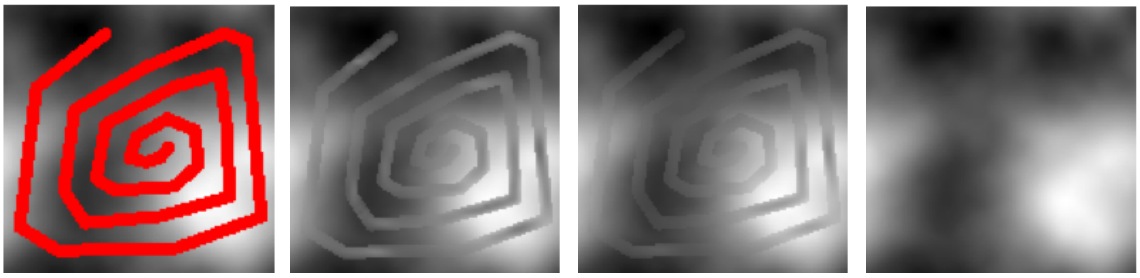
The scanning process computes a Radon transform, which compute the integral of the function to acquires along rays

$$\forall\, \theta \in [0,\pi), \forall\, t \in \mathbb{R}, \quad p_\theta(t) = \int_{\Delta_{t,\theta}} f(x)\, ds = \iint f(x)\, \delta(x \cdot \tau_\theta - t)\, dx$$

see Figure (7.12)

The Fourier slice theorem relates the Fourier transform of the scanned data to the 1D Fourier transform of the data along rays

$$\forall\, \theta \in [0,\pi)\ ,\ \forall\, \xi \in \mathbb{R} \quad \hat{p}_\theta(\xi) = \hat{f}(\xi\cos\theta, \xi\sin\theta). \tag{7.18}$$



$k = 1$ $k = 10$ $k = 20$ $k = 100$

Figure 7.8: *Sobolev projected gradient descent algorithm.*

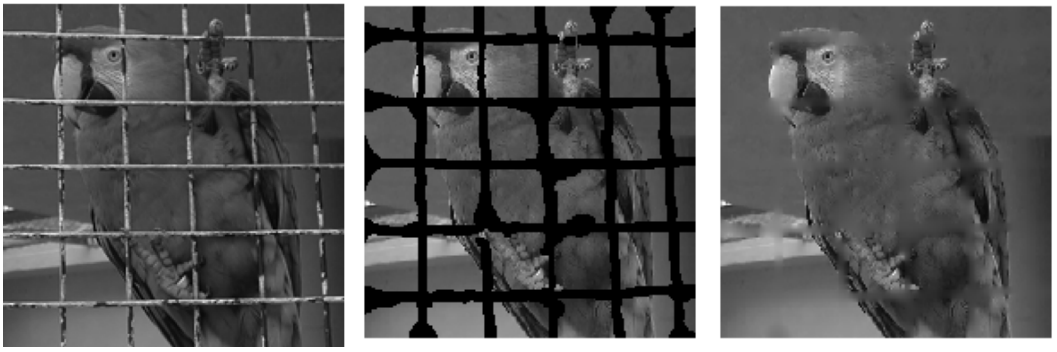


Image f_0 Observation $y = \Phi f_0$ Sobolev $f^\star$

Figure 7.9: *Inpainting the parrot cage.*

This shows that the pseudo inverse of the Radon transform is computed easily over the Fourier domain using inverse 2D Fourier transform

$$f(x) = \frac{1}{2\pi} \int_0^\pi p_\theta \star h(x \cdot \tau_\theta)\, d\theta$$

with $\hat{h}(\xi) = |\xi|$.

Imaging devices only capture a limited number of equispaced rays at orientations $\{\theta_k = \pi/k\}_{0 \leqslant k < K}$. This defines a tomography operator which corresponds to a partial Radon transform

$$Rf = (p_{\theta_k})_{0 \leqslant k < K}.$$

Relation (7.18) shows that knowing Rf is equivalent to knowing the Fourier transform of f along rays,

$$\{\hat{f}(\xi \cos(\theta_k), \xi \sin(\theta_k))\ \}_k.$$

We thus simply the acquisition process over the discrete domain and model it as computing directly samples of the Fourier transform

$$\Phi f = (\hat{f}[\omega])_{\omega \in \Omega} \in \mathbb{R}^P$$

where Ω is a discrete set of radial lines in the Fourier plane, see Figure 7.13, right.

In this discrete setting, recovering from Tomography measures $y = Rf_0$ is equivalent in this setup to inpaint missing Fourier frequencies, and we consider partial noisy Fourier measures

$$\forall\, \omega \in \Omega, \quad y[\omega] = \hat{f}[\omega] + w[\omega]$$

where $w[\omega]$ is some measurement noise, assumed here to be Gaussian white noise for simplicity.

The peuso-inverse $f^+ = R^+ y$ defined in (7.7) of this partial Fourier measurements reads

$$\hat{f}^+[\omega] = \begin{cases} y[\omega] & \text{if} \quad \omega \in \Omega, \\ 0 & \text{if} \quad \omega \notin \Omega. \end{cases}$$

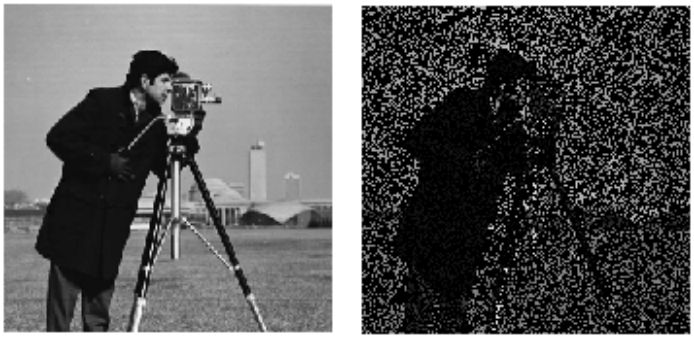


Image f_0 Observation $y = \Phi f_0$

Sobolev $f^\star$ SNR=?dB TV $f^\star$ SNR=?dB

Figure 7.10: *Inpainting with Sobolev and TV regularization.*

Figure 7.14 shows examples of pseudo inverse reconstruction for increasing size of Ω. This reconstruction exhibit serious artifact because of bad handling of Fourier frequencies (zero padding of missing frequencies).

The total variation regularization (7.10) reads

$$f^\star \in \operatorname*{argmin}_f \frac{1}{2} \sum_{\omega \in \Omega} |y[\omega] - \hat{f}[\omega]|^2 + \lambda \|f\|_{\mathrm{TV}}.$$

It is especially suitable for medical imaging where organ of the body are of relatively constant gray value, thus resembling to the cartoon image model introduced in Section 4.2.4. Figure 7.15 compares this total variation recovery to the pseudo-inverse for a synthetic cartoon image. This shows the hability of the total variation to recover sharp features when inpainting Fourier measures. This should be contrasted with the difficulties that faces TV regularization to inpaint over the spacial domain, as shown in Figure 7.11.

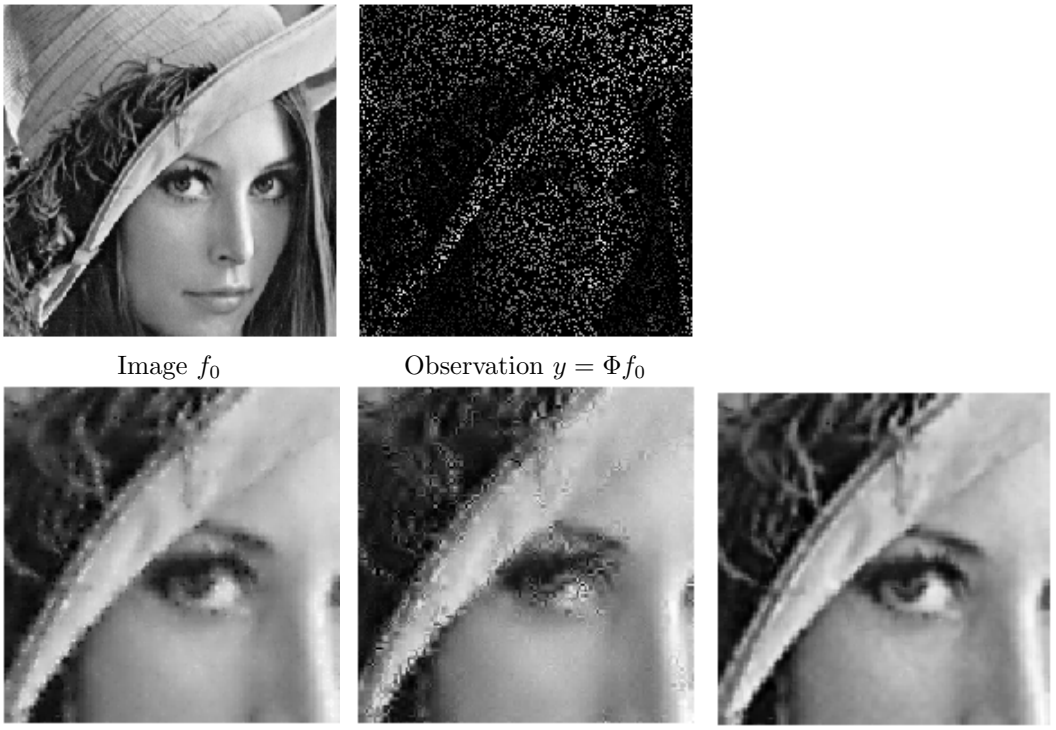


Figure 7.11: *Inpainting with Sobolev and sparsity.*

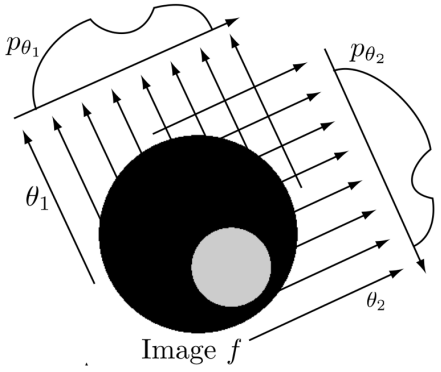


Figure 7.12: *Principle of tomography acquisition.*

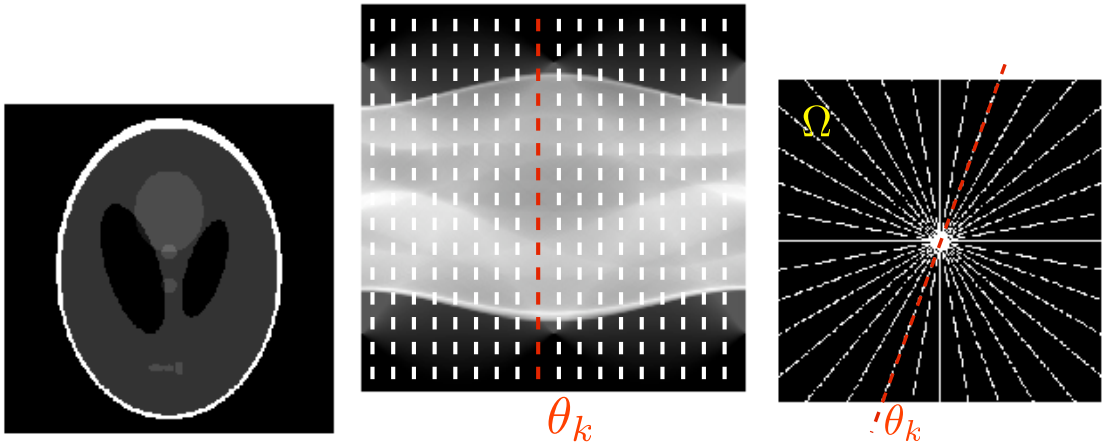


Image f Radon sub-sampling Fourier domain

Figure 7.13: *Partial Fourier measures.*

Image f_0 13 projections 32 projections.

Figure 7.14: *Pseudo inverse reconstruction from partial Radon projections.*

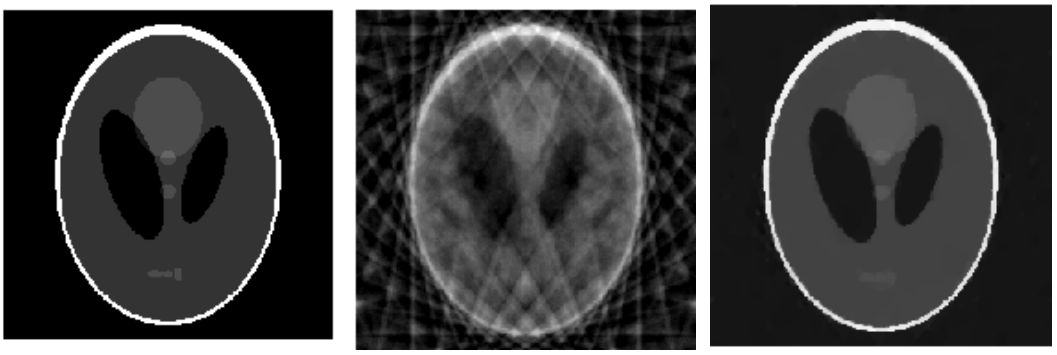


Image f_0 Pseudo-inverse TV

Figure 7.15: *Total variation tomography inversion.*

Chapter 8

Linear Mesh Processing

This chapter exposes the basics of surface approximation with 3D meshes and the way to process such meshes with linear operators. In particular, it studies filtering on 3D meshes and explains how a Fourier theory can be built to analyze these filters.

8.1 Surface Discretization with Triangulated Mesh

8.1.1 Continuous Geometry of Surfaces

In this course, in order to simplify the mathematical description of surfaces, we consider only globally parameterized surfaces. We begin by considering surfaces embedded in euclidean space $\mathcal{M} \subset \mathbb{R}^k$.

Definition 1 (Parameterized surface)**.** *A parameterized surface is a mapping*

$$u \in \mathcal{D} \subset \mathbb{R}^2 \mapsto \phi(u) \in \mathcal{M}.$$

Of course, most surfaces do not benefit from such a simple parameterization. For instance, a sphere should be split into two parts in order to be mapped on two disks $\mathcal{D}_1, \mathcal{D}_2$. These topological difficulties require the machinery of manifolds in order to incorporate a set of charts $\mathcal{D} = \{\mathcal{D}_i\}_i$ that overlap in a smooth manner. All the explanations of this course extend seamlessly to this multi-charts setting.

A curve is defined in parameter domain as a 1D mapping $t \in [0,1] \mapsto \gamma(t) \in \mathcal{D}$. This curve can be traced over the surface and its geometric realization is $\bar{\gamma}(t) \stackrel{\text{def.}}{=} \phi(\gamma(t)) \in \mathcal{M}$. The computation of the length of γ in ambient k-dimensional space $\mathbb{R}^k$ follows the usual definition, but to do the computation over the parametric domain, one needs to use a local metric defined as follow.

Definition 2 (First fundamental form)**.** *For an embedded manifold $\mathcal{M} \subset \mathbb{R}^k$, the first fundamental form is*

$$I_\phi = \left(\langle \frac{\partial \phi}{\partial u_i}, \frac{\partial \phi}{\partial u_j} \rangle \right)_{i,j=1,2}.$$

This local metric I_ϕ defines at each point the infinitesimal length of a curve as

$$L(\gamma) \stackrel{\text{def.}}{=} \int_0^1 \|\bar{\gamma}'(t)\| \mathrm{d}t = \int_0^1 \sqrt{\gamma'(t)^{\mathrm{T}} I_\phi(\gamma(t)) \gamma'(t)} \mathrm{d}t.$$

This fundamental form is an intrinsic invariant that does not depends on how the surfaces is isometrically embedded in space (since the length depends only on this tensor field I_ϕ). In contrast, higher order differential quantities such as curvature might depend on the bending of the surface and are thus usually not intrinsic (with the notable exception of invariants such as the gaussian curvature). In this course, we restrict ourselves to first order quantities since we are mostly interested in lengths and the intrinsic study of surfaces.

Example 1 (Isometry and conformality). *A surface $\mathcal{M}$ is locally isometric to the plane if $I_\phi = \mathrm{Id}_2$. This is for instance the case for a cylinder. The mapping ϕ is said to be conformal if $I_\phi(u) = \lambda(u)\mathrm{Id}_2$. It means that the length of a curve over the plane is only locally scaled when mapped to the surface. In particular, the angle of two interescting curves is the same over the parametric domain and over the surface. This is for instance the case for the stereographic mapping between the plane and a sphere.*

8.1.2 Discretization of Surfaces with Triangulations

Mesh Data Structure A triangulated mesh is a discrete structure that can be used to approximate a surface embedded in Euclidean space $\mathbb{R}^k$. It is composed of a topological part $M = (V, E, F)$ and a geometrical realization $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$. It is important to make the distinction between these two parts since many algorithms rely only on geometry (point clouds processings such as dimension reduction) or on topology (such as compression).

The topology M of the mesh is composed of

– *Vertices* (0D): this is an abstract set of indices $V \simeq \{1, \ldots, n\}$.

– *Edges* (1D): this is a set of pair of vertices $E \subset V \times V$. This set is assumed to be symmetric

$$(i, j) \in E \quad \Longleftrightarrow \quad i \sim j \Leftrightarrow (j, i) \in E.$$

– *Faces* (2D): this is a collection of 3-tuples of vertices $F \subset V \times V \times V$, with the additional compatibility condition

$$(i, j, k) \in F \quad \Longrightarrow \quad (i, j), (j, k), (k, i) \in E.$$

We further assumer that there is no isolated edges

$$\forall\, (i, j) \in E, \quad \exists\, k,\ (i, j, k) \in F.$$

The set of edges can be stored in a symmetric matrix $A \in \mathbb{R}^{n \times n}$ such that $A_{ij} = 1$ if $(i, j) \in E$ and $A_{ij} = 0$ otherwise. This matrix is often stored as a sparse matrix since the number of edges is usually much smaller than n^2. The set of vertices and edges form a non-oriented graph $\mathcal{G} = (V, E)$. Faces are often stored as a matrix $A_F \in \{1, \ldots, n\}^{3 \times m}$ where m is the number of faces and a column $((A_F)_{i,1}, (A_F)_{i,2}, (A_F)_{i,3})$ stores the indices of a face. In a triangulation, the face matrix A_F allows to recover the edge incidence matrix A. The face data structure allows to really capture the 2D geometry of surfaces, which is not possible with graphs alone.

The geometric realization $\mathcal{M}$ is defined through a spacial localization of the vertices (for instance in 3D space)

$$\mathcal{V} \stackrel{\text{def.}}{=} \{x_i \setminus i \in V\} \subset \mathbb{R}^3.$$

This allows to define a piecewise linear mesh

$$\mathcal{F} \stackrel{\text{def.}}{=} \bigcup_{(i,j,k) \in F} \mathrm{Conv}(x_i, x_j, x_k) \subset \mathbb{R}^3,$$

where the convex envelop $\mathrm{Conv}(x, y, z)$ of three points is the Euclidean triangle generated by (x, y, z).

This piecewise linear realization $\mathcal{M}$ can be displayed as a 3D surface on a computer screen. This is performed through a perspective projection of the points and a linear interpolation of color and light inside the triangle. Figure 8.1 shows an example of 3D display, with a zoom on the faces of the mesh.

Adjacency Relationships From the basis topological information given by $M = (V, E, F)$, one can deduce several adjacency data-structures that are important to navigate over the triangulation.

Definition 3 (Vertex 1-ring). *The vertex 1-ring of a vertex $i \in V$ is*

$$V_i \stackrel{\text{def.}}{=} \{j \in V \setminus (i, j) \in E\} \subset V.\text{‘} \tag{8.1}$$

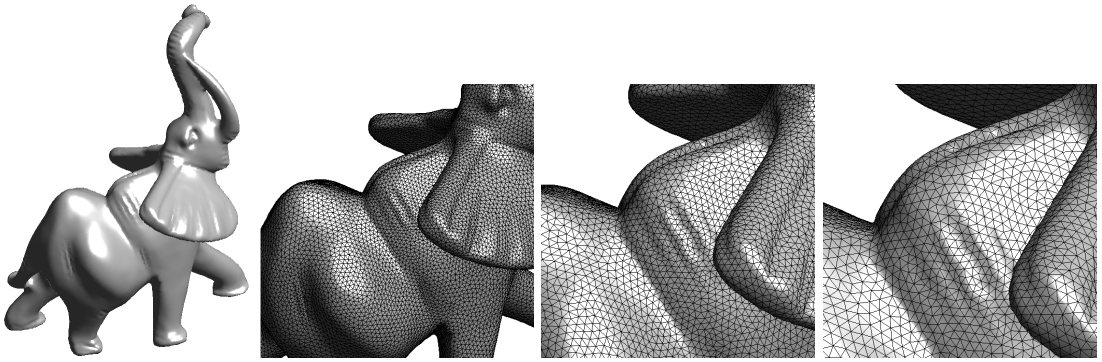


Figure 8.1: *Example of display of a 3D mesh.*

The s-ring is defined by induction as

$$\forall\, s > 1, \quad V_i^{(s)} = \left\{ j \in V \setminus (k,j) \in E \quad \textit{and} \quad k \in V_i^{(s-1)} \right\}. \tag{8.2}$$

Definition 4 (Face 1-ring)**.** *The face 1-ring of a vertex $i \in V$ is*

$$F_i \stackrel{\text{def.}}{=} \{(i,j,k) \in F \setminus i,j \in V\} \subset F.$$

The geometrical realization of a vertex 1-ring is

$$\mathcal{V}_i = \bigcup_{(i,j,k)\in V_i} \mathrm{Conv}(x_i, x_j, x_k).$$

A triangulated mesh is a manifold mesh if all the rings $\mathcal{V}_i$ for $i \in V$ are homeomorphic to either a disk (for interior vertices) or to a half disk (for boundary vertices). This ensures that the geometrical mesh really has the topology of a 2D surface embedded in $\mathbb{R}^3$ (possibly with boundaries). In particular, it implies that there is at most two faces connected to each edge

$$\forall\, (i,j) \in E, \quad \#\{k \setminus (i,j,k) \in F\} \leqslant 2.$$

As an application of these local rings, one can compute a normal at each point using a simple rule

$$\forall\, f = (i,j,k) \in F, \quad \overrightarrow{n_f} \stackrel{\text{def.}}{=} \frac{(x_j - x_i) \wedge (x_k - x_i)}{\|(x_j - x_i) \wedge (x_k - x_i)\|}.$$

and where

$$\forall\, i \in V, \quad \overrightarrow{n_i} \stackrel{\text{def.}}{=} \frac{\sum_{f\in F_i} \overrightarrow{n_f}}{\|\sum_{f\in F_i} \overrightarrow{n_f}\|}.$$

These normals are used to define for instance a light intensity $I(i) = \max(\langle n_i, \ell(i), , \rangle 0)$, where $\ell(i)$ is the incident light. In practice one uses a infinite light source $\ell(i) = \ell =$constant or a local spot located at position $s \in \mathbb{R}^3$ through $\ell(i) = (v_i - s)/\|v_i - s\|$. This light intensity is interpolated on the whole mesh during display.

8.2 Linear Mesh Processing

The light intensity I is a particular example of a function defined at each vertex of the mesh. Mesh processing is intended to process such functions and we thus define carefully vector spaces and operators on meshes.

8.2.1 Functions on a Mesh

In this course, a function is a discrete set of values defined at each vertex location.

Definition 5 (Linear space on a mesh)**.** *A function on a mesh is a mapping $f \in \ell^2(\mathcal{V}) \simeq \ell^2(V) \simeq \mathbb{R}^n$ and can be viewed equivalently as*

$$f : \left\{ \begin{array}{ccc} \mathcal{V} & \longrightarrow & \mathbb{R} \\ x_i & \longmapsto & f(x_i) \end{array} \right. \iff f : \left\{ \begin{array}{ccc} V & \longrightarrow & \mathbb{R} \\ i & \longmapsto & f_i \end{array} \right. \iff f = (f_i)_{i \in V} \in \mathbb{R}^n.$$

The linear space of the functions on a mesh is equipped with an Hilbert space structure that allows to quantify approximation error and compute projections of functions.

Definition 6 (Inner product and norm)**.** *One defines the following inner product and norm for vector $f, g \in \mathbb{R}^n$*

$$\langle f, g \rangle \stackrel{\text{def.}}{=} \sum_{i \in V} f_i g_i \quad \textit{and} \quad \|f\|^2 = \langle f, f \rangle.$$

In order to modify (process) functions on a mesh (such as a light intensity I), this course considers only linear operations that are defined through a large matrix.

Definition 7 (Linear operator A)**.** *A linear operator A is defined as*

$$A : \ell^2(V) \to \ell^2(V) \iff A = (a_{ij})_{i,j \in V} \in \mathbb{R}^{n \times n} \textit{ (matrix).}$$

and operate on a function f as follow

$$(Af)(x_i) = \sum_{j \in V} a_{ij} f(x_j) \iff (Af)_i = \sum_{j \in V} a_{ij} f_j.$$

Example 2. *If the coordinates of the point of a mesh are written $x_i = (x_i^1, x_i^2, x_i^3) \in \mathbb{R}^3$, then the X-coordinate defines a function $f : i \in V \mapsto x_i^1 \in \mathbb{R}$. A geometric mesh $\mathcal{M}$ is thus 3 functions defined on M.*

Mesh processing is the task of modifying functions $f \in \ell^2(V)$. For instance, one can denoise a mesh $\mathcal{M}$ as 3 functions on M. The usual strategy applies a linear operator $f \mapsto Af$. Sometimes, A can computed from M only (for instance for compression) but most of the times it requires both M and $\mathcal{M}$.

8.2.2 Local Operators

In most applications, one can not store and manipulate a full matrix $A \in \mathbb{R}^{n \times n}$. Furthermore, one is usually interested in exploiting the local redundancies that exist in most usual functions $f \in \mathbb{R}^n$ defined on a mesh. This is why we restrict our attention to local operators that can be conveniently stored as sparse matrices (the zeros are not kept in memory).

Definition 8 (Local operator)**.** *A local operator $W \in \mathbb{R}^{n \times n}$ satisfies $w_{ij} = 0$ if $(i,j) \notin E$.*

$$(Wf)_i = \sum_{(i,j) \in E} w_{ij} f_j.$$

A particularly important class of local operators are local smoothings (also called filterings) that perform a local weighted sum around each vertex of the mesh. For this averaging to be consistent, we define a normalized operator $\tilde{W}$ whose set of weights sum to one.

Definition 9 (Local averaging operator)**.** *A local normalized averaging is $\tilde{W} = (\tilde{w}_{ij})_{i,j \in V} \geqslant 0$ where*

$$\forall\, (i,j) \in E, \quad \tilde{w}_{ij} = \frac{w_{ij}}{\sum_{(i,j) \in E} w_{ij}}.$$

It can be equivalently expressed in matrix form as

$$\tilde{W} = D^{-1} W \quad \textit{with} \quad D = \mathrm{diag}_i(d_i) \quad \textit{where} \quad d_i = \sum_{(i,j) \in E} w_{ij}.$$

The smoothing property corresponds to $\tilde{W}1 = 1$ which means that the unit vector is an eigenvector of W with eigenvalue 1.

Example 3. *In practice, we use three popular kinds of averaging operators.*

– *Combinatorial weights: they depends only on the topology (V, E) of the vertex graph*

$$\forall\,(i,j) \in E, \quad w_{ij} = 1.$$

– *Distance weights: they depends both on the geometry and the topology of the mesh, but do not require faces information,*

$$\forall\,(i,j) \in E, \quad w_{ij} = \frac{1}{\|x_j - x_i\|^2}.$$

– *Conformal weights: they depends on the full geometrical realization of the 3D mesh since they require the face information*

$$\forall\,(i,j) \in E, \quad w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}). \tag{8.3}$$

Figure 8.2 shows the geometrical meaning of the angles α_{ij} and β_{ij}

$$\alpha_{ij} = \angle(x_i, x_j, x_{k_1}) \qquad \text{and} \qquad \beta_{ij} = \angle(x_i, x_j, x_{k_2}),$$

where $(i,j,k_1) \in F$ and $(i,j,k_2) \in F$ are the two faces adjacent to edge $(i,j) \in E$. We will see in the next section the explanation of these celebrated cotangent weights.

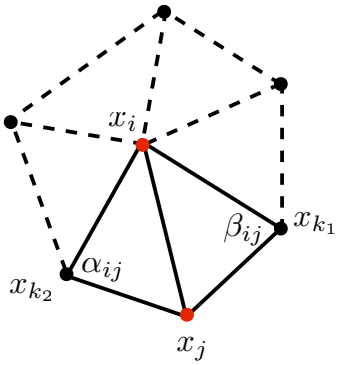


Figure 8.2: *One ring around a vertex i, together with the geometrical angles α_{ij} and β_{ij} used to compute the conformal weights.*

One can use iteratively a smoothing in order to further filter a function on a mesh. The resulting vectors $\tilde{W}f, \tilde{W}^2, \ldots, \tilde{W}^k f$ are increasingly smoothed version of f. Figure 8.3 shows an example of such iterations applied to the three coordinates of mesh. The sharp features of the mesh tend to disappear during iterations. We will make this statement more precise in the following, by studying the convergence of these iterations.

8.2.3 Approximating Integrals on a Mesh

Before investigating algebraically the properties of smoothing operators, one should be careful about what are these discrete operators really approximating. In order for the derivation to be simple, we make computation for a planar triangulation M of a mesh $\mathcal{M} \subset \mathbb{R}^2$.

In the continuous domain, filtering is defined through integration of functions over the mesh. In order to descretize integrals, one needs to define a partition of the plane into small cells centered around a vertex or an edge.

Definition 10 (Vertices Voronoi)**.** *The Voronoi diagram associated to the vertices is*

$$\forall\, i \in V, \quad E_i = \{x \in \mathcal{M} \setminus \forall\, j \neq i, \|x - x_i\| \leqslant \|x - x_j\|\}$$

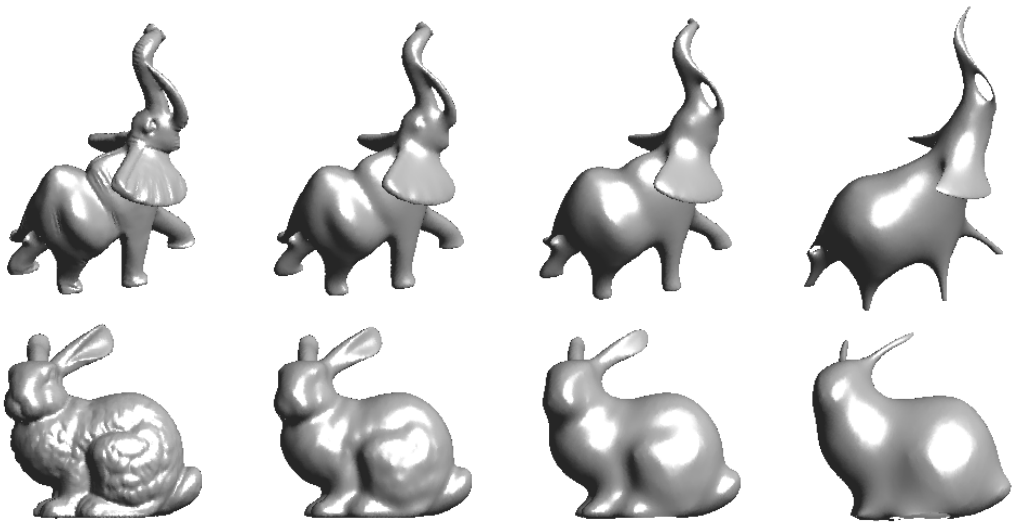


Figure 8.3: *Examples of iterative smoothing of a 3D mesh.*

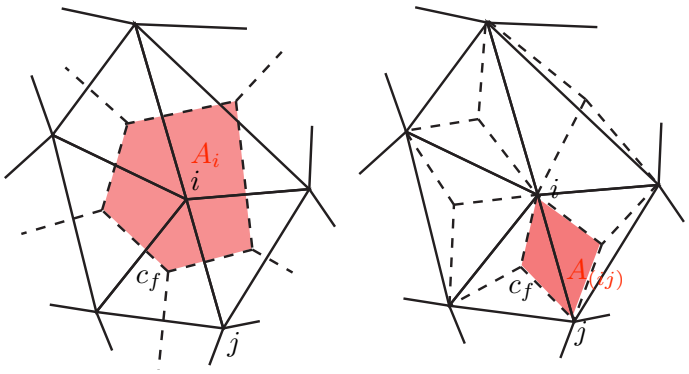


Figure 8.4: *Left: vertex Voronoi cell, right: delaunay Voronoi cell. The point c_f is the orthocenter of a face $f = (i, j, k)$.*

Definition 11 (Edges Voronoi)**.** *The Voronoi diagram associated to the edges is*

$$\forall\, e = (i,j) \in E, \quad E_e = \{x \in \mathcal{M} \setminus \forall\, e' \neq e, d(x,e) \leqslant d(x,e')\}$$

These Voronoi cells indeed form a partition of the mesh

$$\mathcal{M} = \bigcup_{i \in V} E_i = \bigcup_{e \in E} E_e.$$

The following theorem gives the formula for the area of these cells.

Theorem 1 (Voronoi area formulas)**.** *For all $e = (i,j) \in E$, $\forall\, i \in V$, one has*

$$A_e = \mathit{Area}(E_e) = \frac{1}{2}\|x_i - x_j\|^2 \left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right)$$

$$A_i = \mathit{Area}(E_i) = \frac{1}{2} \sum_{j \in N_i} A_{(ij)}.$$

With these areas, one can approximate integrals on vertices and edges using

$$\int_{\mathcal{M}} f(x)\mathrm{d}x \approx \sum_{i \in V} A_i\, f(x_i) \approx \sum_{e=(i,j) \in E} A_e\, f([x_i, x_j]).$$

Of particular interest is the approximation of the so-called Dirichelet energy $\int_{\mathcal{M}} \|\nabla_x f\|^2 \mathrm{d}x$. In order to compute it on a triangular mesh, one can use a finite difference approximation of the gradient of a function at the point $x_{ij} = (x_i + x_j)/2$ along an edge (i, j)

$$\langle \nabla_{x_{ij}} f, \frac{x_i - x_j}{\|x_i - x_j\|} \rangle \approx \frac{f(x_i) - f(x_j)}{\|x_i - x_j\|}.$$

This leads to the following approximation of the Dirichlet energy

$$\int_{\mathcal{M}} \|\nabla_x f\|^2 \mathrm{d}x \approx \sum_{(i,j)\in E} A_{(i,j)} \langle \nabla_{x_{ij}} f, \frac{x_i - x_j}{\|x_i - x_j\|} \rangle^2 \approx \sum_{(i,j)\in E} A_{(i,j)} \frac{|f(x_j) - f(x_i)|^2}{\|x_j - x_i\|^2} \tag{8.4}$$

$$= \sum_{(i,j)\in E} w_{ij} |f(x_j) - f(x_i)|^2 \qquad \text{where} \qquad w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}). \tag{8.5}$$

This discrete formulation shows that the correct weights to approximate the Dirichlet energy are the cotangent one, already introduced in equation (8.3).

8.2.4 Example on a Regular Grid

A regular grid is an uniform discretization with n points of $[0, 1)$ (in 1D) or $[0, 1)^2$ (in 2D). One usually assumes periodic boundary conditions, which means that each side of the square is associated with its opposite.

Since the geometry of a regular grid is invariant under translation, local averaging operators can be computed as convolution on $D = (\mathbb{Z}/p\mathbb{Z})^d$ where $n = p^d$ for d the dimension of the domain ($d = 1$ or $d = 2$)

$$\forall\, i \in D, \quad \tilde{W} f(i) = \sum_{k\in D} f(k) \tilde{w}(i - k),$$

where the operation $+$ and $-$ should be computed modulo p and $\tilde{w}(k) = \tilde{W}(0, k)$ is the convolution kernel.

Example 4 (Averaging). *The uniform averaging filter is defined as*

$$\tilde{W} f(i) = \frac{1}{|N|} \sum_{k\in N} f(i + k),$$

where N is the set of neighbors of the point 0 and $|N| = 2^d$. In this case, in dimension 1, $\tilde{w} = (1, 0, 1)/2$, where this notation assumes that $\tilde{w}$ is centered at the point 0.

In order to study translation invariant operators like local filtering, one needs to use the discrete Fourier transform that diagonalizes these operators.

Definition 12 (Discrete Fourier transform). *The 1D discrete Fourier transform $\Phi(f) \in \mathbb{C}^n$ of the vector $f \in \mathbb{C}^n$*

$$\Phi(f)(\omega) = \hat{f}(\omega) \overset{\text{def.}}{=} \frac{1}{n} \sum_k f_k e^{\frac{2\imath\pi}{n} k\omega}.$$

A similar definition can be given for the 2D discrete Fourier transform. The main property of the Fourier transform is the following diagonalization result.

Theorem 2 (Convolution and Fourier). *For any vector f, one has*

$$\Phi(\tilde{W}^k f) = \Phi(\tilde{w} * \ldots * \tilde{w} * f) \quad \Longrightarrow \quad \Phi(\tilde{W}^k f)(\omega) = \widehat{\tilde{w}}(\omega)^k \, \hat{f}(\omega).$$

The main interest of this tools is that $\Phi(f)$ can be computed in $O(n \log(n))$ operations with the FFT algorithm. Using the following theorem, it gives an alternative expression of a local filtering. This expression in the Fourier domain can be used to speed up the computation of $\tilde{w} * f$ if $\tilde{w}$ has a lot of non zero entries (which is not the case in our setting of local operators). It is also useful to analyze theoretically the behavior of iterated filterings.

Theorem 3 (Convergence). *For any function f defined on a regular grid in 1D or 2D, one has*

$$\tilde{W}^k f \overset{k\to+\infty}{\longrightarrow} \frac{1}{|V|}\sum_{i\in V} f_i$$

This Fourier theory can only be developed for domains that have a group structure that enables translation invariant filtering. In particular, it does not carry over easily to an arbitrary surface. In the remaining, we define a corresponding theory for graphs and triangulated surfaces using the eigenvector of Laplacian operators. This Fourier transform on meshes enables the analysis of the convergence of many filtering schemes.

8.2.5 Gradients and Laplacians on Meshes

Gradient operator A gradient operator defines directional derivatives on a triangulation. It maps functions defined on vertices to functions defined on the set of oriented edges

$$\bar{E} \overset{\text{def.}}{=} \{(i,j) \in E \setminus i > j\}.$$

Definition 13 (Gradient). *Given a local averaging W, the gradient operator G is defined as*

$$\forall\,(i,j)\in E,\ i<j, \quad (Gf)_{(i,j)} \overset{\text{def.}}{=} \sqrt{w_{ij}}(f_j - f_i) \in \mathbb{R}.$$

This mapping can be viewed equivalently as

$$\begin{array}{lll} G : \ell^2(V) \longrightarrow \ell^2(E), & \text{or} & G : \mathbb{R}^n \longrightarrow \mathbb{R}^p \quad \text{where} \quad p = |E|, \\ & \text{or} & G \in \mathbb{R}^{n\times p} \quad \text{(a matrix)}. \end{array}$$

The value of $(Gf)_e$ for an edge $e=(i,j)$ can be thought as a derivative along direction $\overrightarrow{x_i x_j}$.

Example 5. *For the local averaging based on square distances, one has*

$$w_{ij} = \|x_i - x_j\|^{-2}, \quad (Gf)_{(i,j)} = \frac{f(x_j)-f(x_i)}{\|x_i - x_j\|}.$$

which is exactly the finite difference discretization of a directional derivative.

One a regular grid, one can note that
– Gf discretizes $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^{\mathrm{T}}$.
– $G^{\mathrm{T}} v$ discretizes $\div(v) = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}$.

Laplacian Operator A Laplacian operator is a discrete version of a second order derivative operator.

Definition 14 (Laplacian). *Given a local averaging W, the Laplacian operator D is defined as*

$$L \overset{\text{def.}}{=} D - W, \quad \textit{where} \quad D = \mathrm{diag}_i(d_i), \quad \textit{with} \quad d_i = \sum_j w_{ij}.$$

In the remaining, we also make use of normalized operators, which have an unit diagonal.

Definition 15 (Normalized Laplacian). *The normalized Laplacian is defined as*

$$\tilde{L} \overset{\text{def.}}{=} D^{-1/2} L D^{-1/2} = \mathrm{Id}_n - D^{-1/2} W D^{1/2} = \mathrm{Id}_n - D^{1/2}\tilde{W} D^{-1/2}.$$

This normalized Laplacian correspond to the weighted graph Laplacian used in graph theory, see for instance [7].

Remark 1. *One can note that*
– *Laplacians are symmetric operators $L, \tilde{L} \in \mathbb{R}^{n\times n}$.*

– *L acts like a (second order) derivative since $L1 = 0$.*
– *in contrast, the normalized Laplacian is not a real derivative since $\tilde{L}1 \neq 0$ in general.*

The main interest of the gradient operator is that it factorizes the Laplacian as follow.

Theorem 4 (Laplacian factorization)**.** *One has*

$$L = G^{\mathrm{T}}G \quad \textit{and} \quad \tilde{L} = (GD^{-1/2})^{\mathrm{T}}(GD^{-1/2}).$$

This theorem proves in particular that L and $\tilde{L}$ are symmetric positive definite operators. The inner product defined by the Laplacian can be expressed as an energy summed over all the edges of the mesh

$$\langle Lf,\, f\rangle = \|Gf\|^2 = \sum_{(i,j)\in E} w_{ij}\|f_i - f_j\|^2.$$

In the particular case of the cotangent weights introduced in equation (8.3), one can see that the Laplacian norm $\langle Lf,\, f\rangle$ is exactly the finite differences approximation of the continuous Dirichlet energy $\int_{\mathcal{M}} |\nabla_x f|\mathrm{d}x$ derived in equation (8.5). This is why these cotangent weights are the best choice to compute a Laplacian that truly approximates the continuous Laplace Beltrami operator (see definition 16).

A similar expression is derived for the normalized laplacian

$$\langle \tilde{L}f,\, f\rangle = \|GD^{-1/2}f\|^2 = \sum_{(i,j)\in E} w_{ij}\Big\|\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\Big\|^2.$$

Of particular interest for the study of filtering on meshes is the behavior of the spectrum of the Laplacian. We can first study its kernel.

Theorem 5 (Kernel of the Laplacian)**.** *If M is connected, then*

$$\textit{ker}(L) = \textit{span}(1) \quad \textit{and} \quad \textit{ker}(L) = \textit{span}(D^{1/2}).$$

8.2.6 Examples in 1D and 2D

In 1D, all local weights are equivalent since the points are equi-spaced. The corresponding Laplacian is a convolution that can be written as

$$(Lf)_i = \frac{1}{h^2}\left(2f_i - f_{i+1} - f_{i-1}\right) = \frac{1}{h^2} f * (-1, 2, 1)\,,$$

where it is important to remember that the notation $(-1, 2, 1)$ means that the vector is centered around 0.

This discrete 1D Laplacian is the finite difference approximation of the continuous Laplacian on the torus $\mathcal{T}$ of the segment $[0, 1)$ modulo 1. Up to a minus sign, this Laplacian is just the second order derivative

$$L \xrightarrow{h\to 0} -\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x_i)$$

One should be careful with our notation that consider positive semi-definite Laplacian, that have the opposite sign with respect to second order derivative operators (which are definite negative).

The gradient operator corresponds to a discretization of the first order derivative $f \mapsto f'$ (which is anti symmetric). The continuous counterpart of the factorization $L = G^{\mathrm{T}}G$ is the integration by part formula on the torus

$$\int_{\mathcal{T}} f''(x)g(x)\mathrm{d}x = -\int_{\mathcal{T}} -f(x)g'(x)\mathrm{d}x \quad \Longrightarrow \quad \int_{\mathcal{T}} f''(x)f(x)\mathrm{d}x = -\int_{\mathcal{T}} |f'(x)|^2 \leqslant 0.$$

The discrete Laplacian on a 2D grid can also be written as a 2D convolution

$$(Lf)_i = \frac{1}{h^2}\left(4f_i - f_{j_1} - f_{j_2} - f_{j_3} - f_{j_4}\right) = \frac{1}{h^2} f * \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

where $\{j_k\}_k$ are the four neighbors of the point i. This operator is the finite difference approximation to the continuous 2D Laplacian

$$L \overset{h\to 0}{\longrightarrow} -\frac{\partial^2 f}{\partial x^2}(x_i) - \frac{\partial^2 f}{\partial y^2}(x_i) = -\Delta f(x_i).$$

The factorization $Lf = G^{\mathrm{T}} G f$ corresponds to the decomposition $\Delta f = \div(\nabla f)$.

8.2.7 Example of a Parametric Surface

We recall that a parameterized surface is a mapping $u \in \mathcal{D} \subset \mathbb{R}^2 \mapsto \phi(u) \in \mathcal{M}$. Whereas the continuous Laplacian is simple to define on the plane using partial derivatives, its definition on a surface requires the intervention of an arbitrary parameterization ϕ which makes its expression cumbersome.

Definition 16 (Laplace-Beltrami)**.** *The Laplace-Beltrami operator on a parametric surface $\mathcal{M}$ is defined as*

$$\sqrt{g}\Delta_{\mathcal{M}} \overset{\text{def.}}{=} \frac{\partial}{\partial u_1}\left(\frac{g_{22}}{\sqrt{g}}\frac{\partial}{\partial u_1} - \frac{g_{12}}{\sqrt{g}}\frac{\partial}{\partial u_2}\right) + \frac{\partial}{\partial u_2}\left(\frac{g_{11}}{\sqrt{g}}\frac{\partial}{\partial u_2} - \frac{g_{12}}{\sqrt{g}}\frac{\partial}{\partial u_1}\right)$$

where $g = \det(I_\phi)$ and $I_\phi = (g_{ij})_{i,j=1,2}$.

The Laplacian is however an intrinsic operator that does not depends on the chosen parameterization, as shown by the following approximation theorem.

Remark 2 (Laplacian using averaging)**.**

$$\Delta_{\mathcal{M}} f(x) = \lim_{h\to 0} \frac{1}{|B_h(x)|}\int_{y\in\mathcal{M}} f(y)\mathrm{d}y \quad \textit{where} \quad B_h(x) = \{y \setminus d_{\mathcal{M}}(x,y) \leqslant h\}$$

where $d_{\mathcal{M}}$ is the geodesic distance on $\mathcal{M}$ and $h = \max_{(i,j)\in E} \|x_i - x_j\|$ is the discretization precision.

8.3 Diffusion and Regularization on Surfaces

8.3.1 Heat Diffusion

The main linear PDE for regularization of functions is the heat equation that governs the isotropic diffusion of the values of a function in time.

Definition 17 (Heat diffusion)**.** *$\forall\, t > 0$, one defines $F_t : M \to \mathbb{R}$ solving*

$$\frac{\partial F_t}{\partial t} = -D^{-1}LF_t = -(\mathrm{Id}_n - \tilde{W})F_t \qquad \textit{and} \qquad \forall\, i \in V,\ F_0(i) = f(i)$$

In order to compute numerically the solution of this PDE, one can fix a time step $\delta > 0$ and use an explicit discretization in time $\bar{F}_k$ as $F_0 = f$ and

$$\frac{1}{\delta}\left(\bar{F}_{k+1} - \bar{F}_k\right) = -D^{-1}L\bar{F}_k \quad \Longrightarrow \quad \bar{F}_{k+1} = \bar{F}_k - \delta D^{-1}L\bar{F}_k = (\mathrm{Id} - \delta)\bar{F}_k + \delta\tilde{W}\bar{F}_k. \tag{8.6}$$

If δ is small enough, one hopes that the discrete solution $\bar{F}_k$ is close to the continuous time solution F_t for $t = \delta k$. This is indeed the case as proven later in these notes.

Remark 3. *In order for this scheme to be stable, one needs $\delta < 1$. This is be proven later using the extension of Fourier theory to meshes.*

Remark 4. *If $\delta = 1$, then the discretization of the Heat equation corresponds to iterative smoothing since $\bar{F}_k = \tilde{W}^k f$. In this case stability is not guaranteed but only pathological meshes give unstable filtering (see theorem 13).*

Instead of using the explicit discretization in time (8.6), one can use an implicit scheme which compute an approximate solution $\tilde{F}_k$ at step k by solving

$$\frac{1}{\delta}\left(\tilde{F}_{k+1} - \tilde{F}_k\right) = -D^{-1}L\tilde{F}_{k+1} \quad \Longrightarrow \quad ((\delta+1)\mathrm{Id}_n - \delta\tilde{W})\tilde{F}_{k+1} = \tilde{F}_k. \tag{8.7}$$

Computing $\tilde{F}_k$ requires the solution of a sparse linear system at each step k. The implicit scheme (8.7) is thus computationally more involved than the explicit scheme (8.6). We will however see later that the implicit scheme is always stable for any value of $\delta \leqslant 1$.

Example 6 (Mesh smoothing)**.** *In order to smooth a mesh whose points are $x_i = (x_i^1, x_i^2, x_i^3)$, one can perform a heat diffusion for each component $f_i = (x_i^k), k = 1, 2, 3$. Figure 8.5 shows an example of such a smoothing.*

In practice, mesh smoothing is used to denoise a function $f = f_0 + \sigma g$ where $g \in \mathbb{R}^n$ is a realization of a gaussian white noise (each entry $g(i)$ are independent and follow a gaussian law with unit variance). The difficult task it to find an optimal stopping time t to minimize $\|F_t - f_0\|$, which is not available since one does not know f_0. For uniformly smooth surfaces, the theory predicts that a linear filtering such as the heat equation requires a stopping time proportional to the noise level σ. This is however false for more complex surfaces such as the one used in computer graphics. In these case, alternate non linear diffusions such as non-linear PDE or wavelet thresholding usually perform better, see [20] for an overview of these methods in image processing.

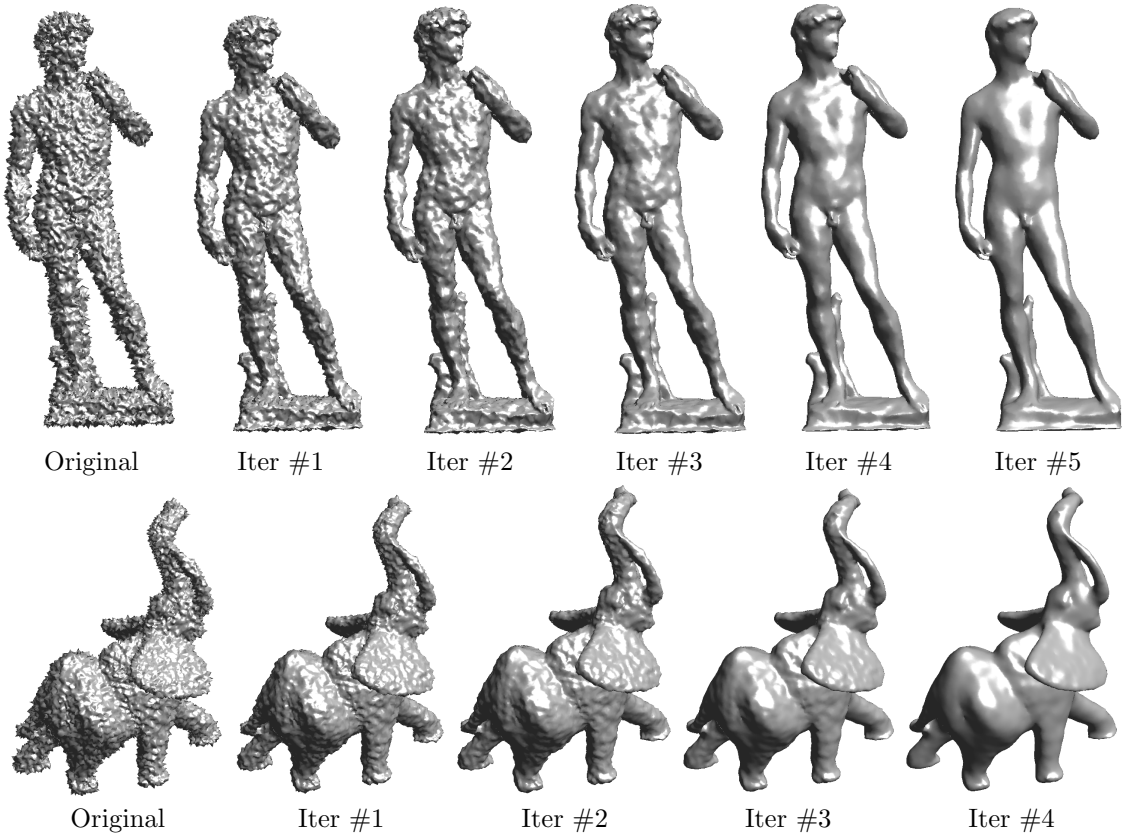


Figure 8.5: *Examples of mesh denoising with the heat equation.*

Other differential equations. One can solve other partial differential equations involving the Laplacian over a 3D mesh $M = (V, E, F)$. For instance, one can consider the wave equation, which defines, for all $t > 0$, a vector $F_t \in \ell^2(V)$ as the solution of

$$\frac{\partial^2 F_t}{\partial t^2} = -D^{-1}LF_t \qquad \text{and} \qquad \begin{cases} F_0 = f \in \mathbb{R}^n, \\ \frac{\mathrm{d}}{\mathrm{d}t}F_0 = g \in \mathbb{R}^n, \end{cases} \tag{8.8}$$

In order to compute numerically the solution of this PDE, one can fix a time step $\delta > 0$ and use an explicit discretization in time $\bar{F}_k$ as $F_0 = f$, $F_1 = F_0 + \delta g$ and for $k > 1$

$$\frac{1}{\delta^2}\left(\bar{F}_{k+1} + \bar{F}_{k-1} - 2\bar{F}_k\right) = -D^{-1}L\bar{F}_k \quad \Longrightarrow \quad \bar{F}_{k+1} = 2\bar{F}_k - \bar{F}_{k-1} - \delta^2 D^{-1}L\bar{F}_k.$$

Figure 8.6 shows examples of the resolution of the wave equation on 3D meshes.

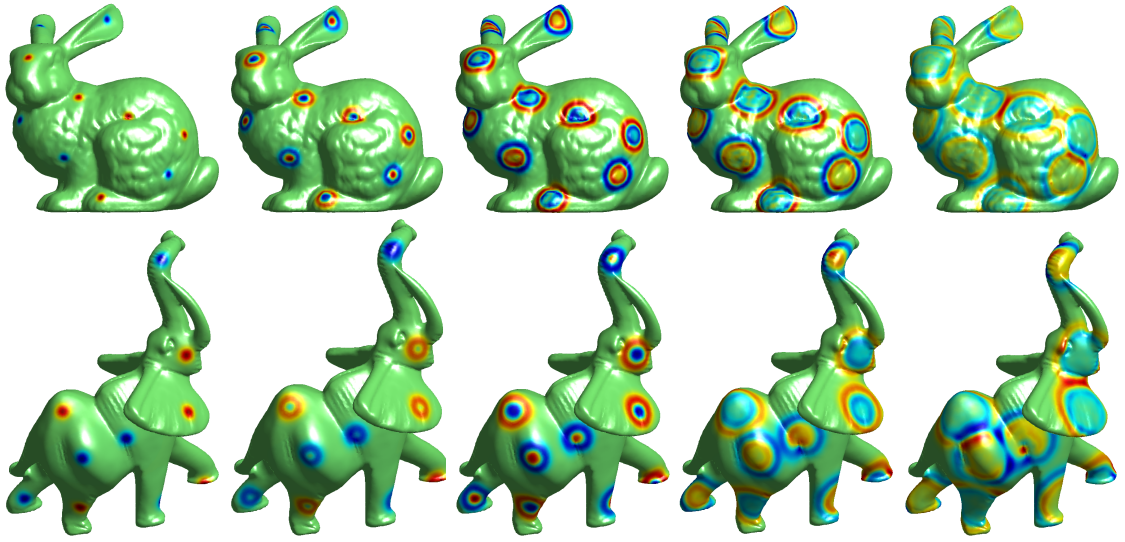


Figure 8.6: *Example of evolution of the wave equation on 3D mesh. The initial condition f is a superposition of small positive and negative gaussians.*

8.3.2 Spectral Decomposition

In order to better understand the behavior of linear smoothing on meshes, one needs to study the spectral content of Laplacian operators. This leads to the definition of a Fourier theory for meshes. The decomposition $\tilde{L} = (GD^{-1/2})^{\mathrm{T}}(GD^{-1/2})$ of the Laplacian implies that it is a positive semi-definite operator. One can thus introduce the following orthogonal factorization.

Theorem 6 (Eigen-decomposition of the Laplacian)**.** *It exists a matrix U, $U^{\mathrm{T}}U = \mathrm{Id}_n$ such that*

$$\tilde{L} = U\Lambda U^{\mathrm{T}} \qquad \textit{where} \qquad \Lambda = \mathrm{diag}_\omega(\lambda_\omega), \qquad \lambda_1 \leqslant \ldots \leqslant \lambda_n.$$

The eigenvalues λ_ω correspond to a frequency index that ranks the eigenvectors u_ω of $U = (u_\omega)_\omega$. One can first state some bounds on these eigenvalues.

Theorem 7 (Spectral bounds)**.** $\forall\, i,\ \lambda_i \in [0, 2]$ *and*
– *If M is connected then $0 = \lambda_1 < \lambda_2$.*
– *$\lambda_n = 2$ if and only if M is 2-colorable.*

We recall the definition of a colorable graph next.

Definition 18 (Colorable graph)**.** *A graph (V, E) is k-colorable if it exist a mapping $f : V \to \{1, \ldots, k\}$ such that*

$$\forall\,(i, j) \in E, \quad f(i) \neq f(j).$$

A 2-colorable graph is also called bi-partite. A 2-colorable mesh is pathological for filtering since one can split the set of vertices into two parts without inner connexions. The filtering process can oscillate by exchanging values between these sets, thus never converging.

The orthogonal eigen-basis $U = (u_\omega)_\omega$ is an orthogonal basis of the space $\mathbb{R}^n \simeq \ell^2(V)$, which can be written as

$$u_\omega : \left\{ \begin{array}{ccc} V & \longrightarrow & \mathbb{R} \\ i & \longmapsto & u_\omega(x_i) \end{array} \right.$$

The orthogonality means that $\langle u_\omega, u_{\omega'} \rangle = \delta_\omega^{\omega'}$. This basis allows to compute an orthogonal decomposition of any functions f

$$\forall\, f \in \ell^2(V), \quad f = \sum_\omega \langle f, u_\omega \rangle u_\omega.$$

Having such a tool allows to split a function f in elementary contributions $\langle f, u_\omega \rangle$ with a control in the energy because of orthogonality

$$\|f\|^2 = \sum_\omega |\langle f, u_\omega \rangle|^2.$$

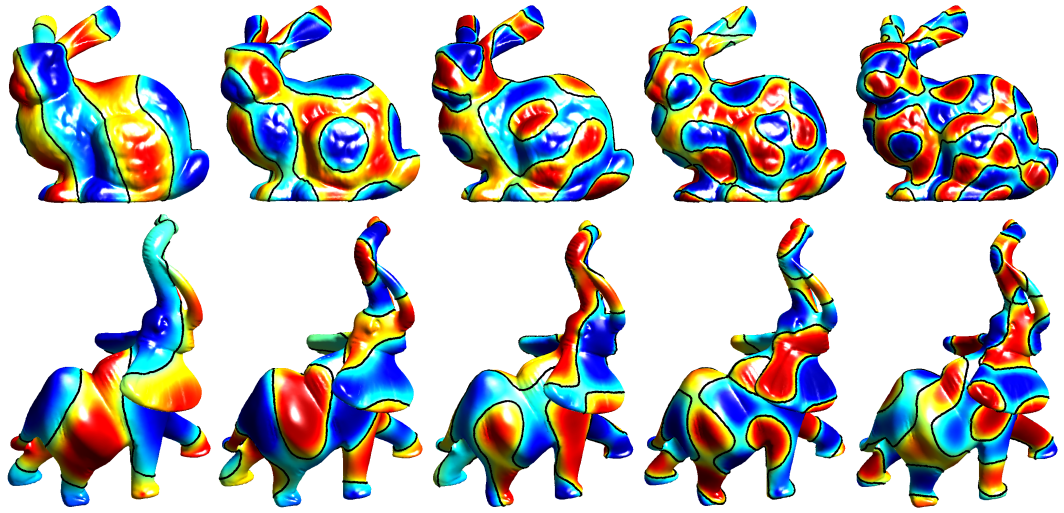


Figure 8.7: *Examples of eigenvectors u_ω of the Laplacian $\tilde{L}$. The blue colors indicated negative values, red colors positive ones. The black curve is the 0 level set of the eigenvector.*

Figure 8.7 shows some examples of eigenfunctions depicted using color ranging from blue (negative values of the eigenfunction) to red (positive values). One can see that these functions are oscillating, in a way similar to the traditional Fourier basis. In some sense (made more precise latter), this basis is the extension of the Fourier basis to meshes. A function u_ω corresponding to a large spectral value λ_ω is highly oscillating and corresponds thus intuitively to a high frequency atom.

Extracting numerically eigenvectors from a large matrix is a difficult problem. If the matrix is sparse, a method of choice consists in using iterative powers of a shifted version of the laplacian. One starts from a random initial vector v_0 and iterates

$$v_{k+1} = \frac{w_{k+1}}{\|w_{k+1}\|} \qquad \text{where} \qquad w_{k+1} = (\tilde{L} - \lambda \mathrm{Id}_n)^{-1} v_k. \tag{8.9}$$

These iterates converges to the eigenvectors corresponding to the eigenvalue the closest to λ, as staten in the following theorem.

Theorem 8 (Inverse iterations)**.** *For a given shift λ, lets denote*

$$\omega^\star = \operatorname*{argmin}_\omega |\lambda - \lambda_\omega| \qquad \textit{and} \qquad \omega^+ = \operatorname*{argmin}_{\omega \neq \omega^\star} |\lambda - \lambda_\omega|$$

If $|\lambda - \lambda_{\omega^\star}| < |\lambda - \lambda_{\omega^+}|$, *then*

$$v_k \overset{k\to+\infty}{\longrightarrow} u_{\omega^\star} \quad \textit{and} \quad \langle Lv_k, v_k\rangle \overset{k\to+\infty}{\longrightarrow} \lambda_{\omega^\star}.$$

The speed of convergence of these inverse iterations is governed by the conditioning of $(\tilde{L} - \lambda \mathrm{Id}_n)^{-1}$ since

$$\|v_k - u_{\omega^\star}\| \leqslant C\rho(\lambda)^k \qquad \text{where} \qquad \rho(\lambda) \overset{\text{def.}}{=} \frac{|\lambda - \lambda_{\omega^\star}|}{|\lambda - \lambda_{\omega^+}|} < 1.$$

The smallest $\rho(\lambda)$ is, the faster the method converges.

In order to compute an iteration (8.9) of the method, one needs to solve a sparse linear system $Aw_{k+1} = v_k$ whith $A = \tilde{L} - \lambda \mathrm{Id}_n$. In order to do so, one can use a direct method such as LU factorization. The advantage of such an approach is that the factorization is computed once for all and can be re-used to solve very quickly at each step k. These factorization are however quite slow to compute especially for large matrices. For large problems, one can solve this linear system using an iterative algorithm such as conjugate gradient. These iterative method are attractive for sparse matrices, but a fast convergence requires $1/\rho(\lambda)$, the conditioning of $\tilde{L} - \lambda \mathrm{Id}_n$ to be not large, with is contradictory with the constraint for iterations 8.9 to converge fast.

8.3.3 Spectral Theory on a Regular Grid

In the particular case of a 1D or 2D lattice, the eigenfunctions defined earlier correspond exactly to the Fourier basis used in the discrete Fourier transform.

Theorem 9 (Spectrum in 1D). *For a 1D regular lattice,*

$$u_\omega(k) = \frac{1}{\sqrt{n}} \exp\left(\frac{2\imath\pi}{n} k\omega\right) \qquad \textit{and} \qquad \lambda_\omega = 4\sin^2\left(\frac{2\pi}{n}\omega\right).$$

Theorem 10 (Spectrum in 2D). *For a 2D regular lattice,* $n = n_1 n_2$, $\omega = (\omega_1, \omega_2)$

$$u_\omega(k) = \frac{1}{\sqrt{n}} \exp\left(\frac{2\imath\pi}{n} \langle k, \omega\rangle\right) \qquad \textit{and} \qquad \lambda_\omega = 4\left(\sin^2\left(\frac{2\pi}{n_1}\omega_1\right) + \sin^2\left(\frac{2\pi}{n_2}\omega_2\right)\right).$$

As already mentioned, on a mesh, the eigenvectors of $\tilde{L}$ correspond to a extension of the Fourier basis to meshes. The definition of the Fourier transform on meshes requires a little care since a diagonal normalization by D is used as defined next.

Definition 19 (Manifold-Fourier transform). *For* $f \in \ell^2(V)$,

$$\Phi(f)(\omega) = \hat{f}(\omega) \overset{\text{def.}}{=} \langle D^{1/2} f, u_\omega\rangle \quad \Longleftrightarrow \quad \Phi(f) = \hat{f} = U^{\mathrm{T}} D^{1/2}.$$

where $(u_\omega)_\omega$ *are the eigenvectors of* $\tilde{L}$.

One can note that there is still a degree of freedom in designing this Fourier transform since one can use any local weighting (for instance combinatorial, distance or conformal). Depending on the application, one might need to use weights depending only on the topology of the mesh (combinatorial for mesh compression).

A major theoretical interest of this Fourier transform is that it diagonalizes local averaging operators.

Theorem 11 (Spectral smoothing). *One has* $\Phi \tilde{W} \Phi^{-1} = \mathrm{Id}_n - \Lambda$ *and thus for any function* f

$$\widehat{\tilde{W} f}(\omega) = (1 - \lambda_\omega)\hat{f}(\omega)$$

This diagonalization allows to prove the convergence of iterative smoothing.

Theorem 12 (Convergence of iterated smoothing). *If* $\lambda_n < 2$ *(i.e. M is not 2-colorable), then for any function* f

$$\tilde{W}^k f \overset{k\to+\infty}{\longrightarrow} \frac{1}{n}\sum_{i\in V} f_i.$$

8.3.4 Spectral Resolution of the Heat Diffusion

Recall that the heat diffusion is defined as

$$\forall\, t>0, \quad \frac{\partial F_t}{\partial t} = -D^{-1}LF_t = -(\mathrm{Id}_n - \tilde{W})F_t$$

Using the manifold Fourier expansion $\hat{F}_t \overset{\text{def.}}{=} U^{\mathrm{T}} D^{1/2} F_t$, this differential equation can be re-written as

$$\frac{\partial \hat{F}_t(\omega)}{\partial t} = -\lambda_\omega \hat{F}_t(\omega) \quad \Longrightarrow \quad \hat{F}_t(\omega) = \exp(-\lambda_\omega t)\hat{f}(\omega). \tag{8.10}$$

This allows to study the convergence of the continuous heat equation.

Theorem 13 (Convergence of heat equation)**.** *If $\mathcal{M}$ is connected,*

$$F_t \overset{t\to+\infty}{\longrightarrow} \frac{1}{n}\sum_{i\in V} f_i.$$

Recall that the heat equation is discretized using the following explicit and implicit schemes, equations (8.6) and (8.7)

$$\begin{cases} \bar{F}_k = (1-\delta)\bar{F}_k + \delta\tilde{W}\bar{F}_k, \\ ((1+\delta)\mathrm{Id}_n - \delta\tilde{W})\tilde{F}_{k+1} = \tilde{F}_k. \end{cases}$$

These filtering iterations can be re-written over the Fourier domain as

$$\begin{cases} \widehat{\bar{F}_{k+1}}(\omega) = (1-\delta\lambda_\omega)\widehat{\bar{F}_k}(\omega), \\ \widehat{\tilde{F}_{k+1}}(\omega) = \frac{1}{(1+\delta\lambda_\omega)}\widehat{\bar{F}_k}(\omega). \end{cases}$$

This allows to state the stability and convergence of the finite difference discretization.

Theorem 14 (Convergence of discretization)**.** *The explicit scheme is stable if $\delta < 1$. The implicit scheme is always stable. One has*

$$\begin{cases} \bar{F}_{t/\delta} \overset{\delta\to 0}{\longrightarrow} F_t, \\ \tilde{F}_{t/\delta} \overset{\delta\to 0}{\longrightarrow} F_t. \end{cases}$$

with the restriction that for the explicit scheme, the mesh must not be 2-colorable.

Other Differential Equations. The manifold Fourier transform can also be used to solve the wave equation (8.8) since

$$\frac{\partial^2 \hat{F}_t(\omega)}{\partial t^2} = -\lambda_\omega \hat{F}_t(\omega) \quad \Longrightarrow \quad \hat{F}_t(\omega) = \cos(\sqrt{\lambda_\omega}t)\hat{f}(\omega) + \frac{1}{\sqrt{\lambda_\omega}}\sin(\sqrt{\lambda_\omega}t)\hat{g}(\omega).$$

8.3.5 Quadratic Regularization

Instead of using a PDE for regularization, one can try to find a new function that is both close to the original one f and that is smooth in a certain sense. This leads to the notion of quadratic regularization, where one uses a Laplacian as a smoothness prior on the recovered function.

Definition 20 (Quadratic regularizer)**.** *For $t>0$, one defines*

$$F_t^q = \operatorname*{argmin}_{g\in\mathbb{R}^n} \|f-g\|^2 + t\|\tilde{G}g\|^2 \qquad \textit{where} \qquad \tilde{G} = GD^{-1/2}.$$

This optimization replaces $f \in \ell^2(V)$ by $F_t^{\mathrm{q}} \in \ell^2(V)$ with small gradients. This optimization can be found in closed form by inverting a sparse linear system.

Theorem 15 (Solution of quadratic regularization)**.** *F_t^q is unique and*

$$F_t^q = (\mathrm{Id}_n + t\tilde{L})^{-1} f.$$

Over the Fourier domain, this inversion reads

$$\hat{F}_t^{\mathrm{q}}(\omega) = \frac{1}{1+t\lambda_\omega}\hat{f}(\omega).$$

This corresponds to an attenuation of the high frequency content of f, in a way very similar to equation (8.10).

Once again, similarly to the heat equation, the spectral expression of the quadratic regularizer allows to study its convergence for large t.

Theorem 16 (Convergence of quadratic regularization)**.** *If $\mathcal{M}$ is connected,*

$$F_t^q \overset{t\to+\infty}{\longrightarrow} \frac{1}{n}\sum_{i\in V} f_i.$$

8.3.6 Application to Mesh Compression

We have shown how the Fourier basis on meshes can be used to compute in a diagonal fashion filtering, heat diffusion and quadratic regularization. This Fourier transform is however of little interest in practice, since the original filterings (or finite difference approximation of the heat equation) are usually faster to compute directly than over the Fourier domain. The Fourier transform is thus mainly of theoretical interest in these cases since it allows to prove convergence results.

Another class of applications makes use of an orthogonal expansion such as the Fourier one to perform mesh compression. This section shows how to compute a linear M-term approximation in this Fourier basis and to do mesh compression. We refer to the survey [1] for more advanced non-linear mesh compression methods.

The orthogonal basis $U = (u_\omega)_\omega$ of $\ell^2(V) \simeq \mathbb{R}^n$, where $\tilde{L} = U\Lambda U^{\mathrm{T}}$ allows to define a linear approximation as followed.

Definition 21 (Linear M-term approximation)**.** *For any $M > 0$, the linear M-term approximation of f is*

$$f = \sum_{\omega=1}^{n} \langle f,\, u_\omega\rangle u_\omega \quad \overset{M\text{-term approx.}}{\Longrightarrow} \quad f_M \overset{\text{def.}}{=} \sum_{\omega=1}^{M} \langle f,\, u_\omega\rangle u_\omega.$$

The quality of the approximation is measured using the error decay, which can in turn be estimated using the removed coefficients

$$E(M) \overset{\text{def.}}{=} \|f - f_M\|^2 = \sum_{\omega>M} |\langle f,\, u_\omega\rangle|^2.$$

A good orthogonal basis U is a basis for which $E(M)$ decays fast on the signals of interest. Equivalently, a fast decay of E with M corresponds to a fast decay of $|\langle f,\, u_\omega\rangle|$ for large ω. Figure 8.8 shows the decay of the Fourier spectrum for two different functions defined on a 3D mesh. The smooth function (left in the figure) exhibits a fast decay of its spectrum, meaning that it can be well approximated with only a few Fourier coefficients.

We recall that the Fourier atoms

$$\forall\, \omega \in \mathbb{Z}, \quad u_\omega(x) = \frac{1}{\sqrt{2\pi}} e^{\imath\omega x}$$

are the eigenvectors of the compact, symmetric, semi-definite negative operator $f \mapsto f''$ (that should be defined on the Hilbert space of twice Sobolev derivable functions). This set of function is also an Hilbert basis of the space $\mathrm{L}^2(\mathbb{R}/(2\pi\mathbb{Z}))$ of 2π-periodic square integrable functions and a Fourier coefficient is $\hat{f}(\omega) \overset{\text{def.}}{=} \langle f,\, u_\omega\rangle$.

Approximation theory studies this linear error decay for classical functional spaces. One can for instance study the Fourier expansion over euclidean spaces.

Theorem 17 (Fourier in 1D)**.** *If f is C^α regular on $\mathbb{R}/(2\pi\mathbb{Z})$,*

$$|\hat{f}(\omega)| \leqslant \|f^{(\alpha)}\|_\infty |\omega|^{-\alpha}.$$

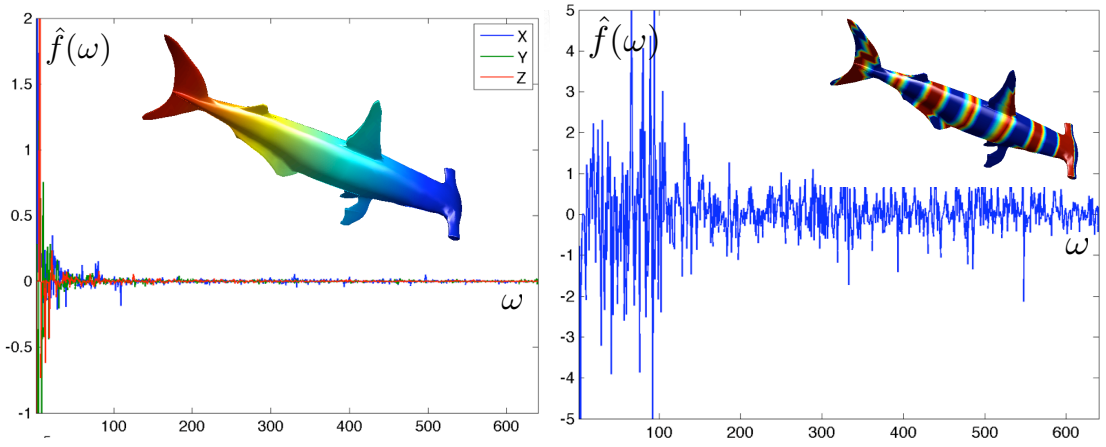


Figure 8.8: *Examples of Fourier spectrum for a smooth and a non-smooth function.*

This result can be proven with a simple integration by parts. A slightly more difficult result shows that the linear approximation error decays like $M^{-\alpha}$.

Theorem 18 (Fourier approximation). *If f is C^α on $\mathbb{R}/(2\pi\mathbb{Z})$, then it exist $C > 0$ such that*

$$\sum_\omega |\omega|^{2\alpha} |\langle f, u_\omega \rangle|^2 < +\infty \quad \Longrightarrow \quad E(M) \leqslant C M^{-\alpha}.$$

This kind of results can be extended to continuous surfaces thanks to the continuous Laplacian. We suppose that $\mathcal{M}$ is a surface parameterized by ϕ, and a function $f = \phi \circ \bar{f}$ is defined on it. By definition, this function f is C^α if $\bar{f}$ is C^α in euclidean space. For a compact surface $\mathcal{M}$, the Laplace-Beltrami operator $\Delta_\mathcal{M}$ is symmetric (for the inner product on the surface), is negative semi-definite and has a discrete spectrum $\Delta_\mathcal{M} u_\omega = -\lambda_\omega u_\omega$ for $\omega \in \mathbb{N}$. The functions $\{u_\omega\}_\omega$ are an orthogonal basis for function of finite energy on the surface $\mathrm{L}^2(\mathcal{M})$. The inner product of an arbitrary smooth function $f \in \mathrm{C}^\alpha(\mathcal{M})$ can be bounded using integration by parts

$$\langle f, u_\omega \rangle = \frac{1}{\lambda_\omega^k} \langle \Delta_\mathcal{M}^k f, u_\omega \rangle \quad \Longrightarrow \quad |\langle f, u_\omega \rangle| \leqslant \frac{\|f\|_{\mathrm{C}^\alpha}}{\lambda_\omega^{\alpha/2}}.$$

This proves the efficiency of the Fourier basis on surfaces to approximate smooth functions.

When computing the M-term approximation f_M of f one removes the small amplitude Fourier coefficients of the orthogonal expansion of f. Figure 8.9 shows some examples of mesh approximation where one retains an increasing number of Fourier coefficients. Mesh compression is only a step further, since one also need to code the remaining coefficients. This requires first quantifying the coefficients up to some finite precision and then binary code these coefficients into a file.

8.3.7 Application to Mesh Parameterization

This section is restricted to the study of meshes that can be globally parameterized on a plane. It means that they are topologically equivalent to a 2D disk. More complex meshes should be first segmented in cells that are equivalent to a disk.

A parameterization of a continuous surface $\mathcal{M}$ is a bijection

$$\psi : \mathcal{M} \longrightarrow \mathcal{D} \subset \mathbb{R}^2.$$

A similar definition applies to a discrete mesh where one computes a 2D position $\psi(i)$ for all the vertices $i \in V$ and then interpolates linearly the mapping to the whole piecewise linear geometric mesh. This section explains the basics of linear methods for mesh parameterization. We refer to various surveys [15, 29] for more details on mesh parameterization.

Usually, a 2D mesh is computed from range scanning or artistic modeling, so it does not come with such a parameterization. In order to perform texture mapping or more general mesh deformations, it is however important to use such a parameterization. Since many bijections are

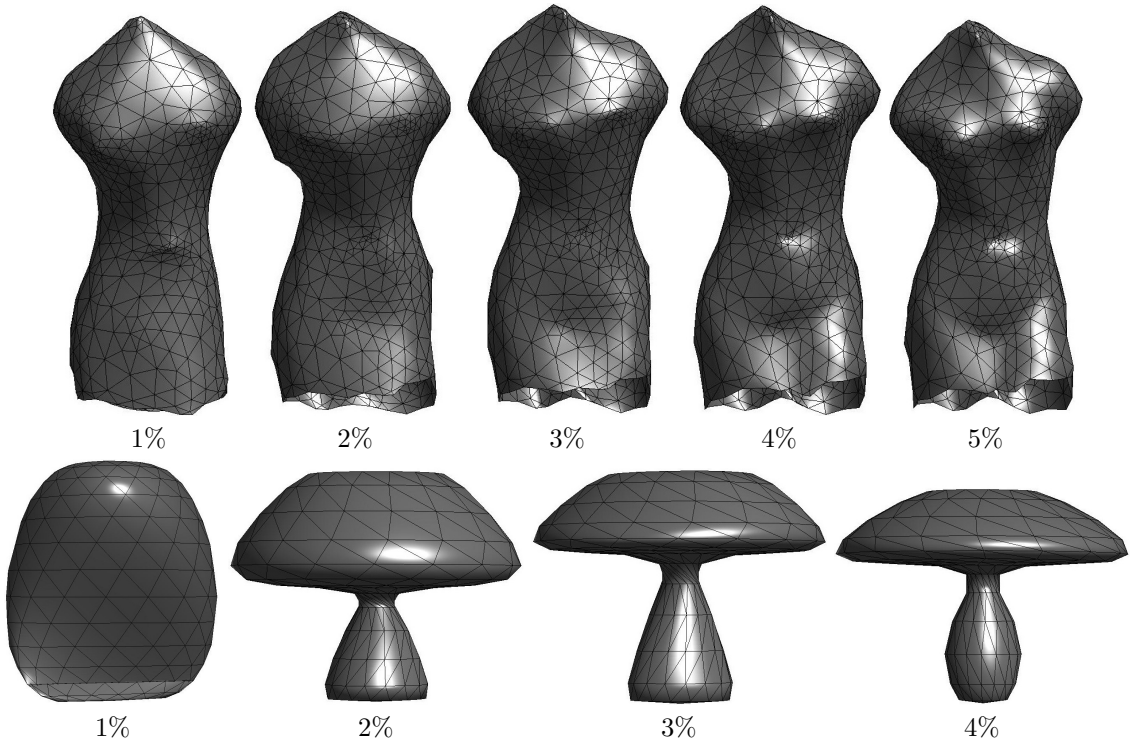


Figure 8.9: *Examples of spectral mesh compression.*

possible to layout the mesh in 2D, the mapping ψ has to satisfy additional smoothness assumptions. Classically, one requires that each coordinate of ψ has a vanishing Laplacian (it is thus harmonic) outside a set of constrained vertices that enforce boundary conditions.

More precisely, $\psi = (\psi_1, \psi_2)$ is the solution of

$$\begin{cases} \forall\, i \notin \partial\mathcal{M}, & (L\psi_1)(i) = (L\psi_2)(i) = 0 \\ \forall\, i \in \partial\mathcal{M}, & \psi(i) = \psi^0(i) \in \partial\mathcal{D}, \end{cases}$$

where $\partial\mathcal{M}$ is the boundary of the mesh, which consists in vertices whose face ring is not homeomorphic to a disk but rather to a half disk. This formulation requires the solution of two sparse linear systems (one for each coordinate of ψ).

The boundary condition $\psi^0(i)$ for $i \in \partial\mathcal{M}$ describes a 1D piecewise linear curve in the plane, that is fixed by the user. In the following, we will see that this curve should be convex for the parameterization to be bijective.

Remark 5. *For such an harmonic parameterization, each point is the average of its neighbors since*

$$\forall\, i, \quad \psi(i) = \frac{1}{\sum_j w_{ij}} \sum_{(i,j)\in E} w_{i,j}\psi(j).$$

The powerful feature of this linear parameterization method is that it can be proven to produce a valid (bijective) parameterization as long as the constrained position (boundary values of ψ) are along a convex curve.

Theorem 19 (Tutte theorem)**.** *If $\forall\,(i,j) \in E,\ w_{ij} > 0$, and if $\partial\mathcal{D}$ is a convex curve, then ψ is a bijection.*

Figure 8.10 shows several examples of parameterizations. One is free to use any laplacian (combinatorial, distance or conformal) as long as it produces positive weights. There is a issue with the conformal weights, which can be negative if the mesh contains obtuse triangles. In practice however it leads to the best results. The efficiency of a parameterization can be measured by some amount of distortion induced by the planar mapping. Linear methods cannot hope to cope with large isoperimetric distortions (for instance large extrusions in the mesh) since harmonicity leads to clustering of vertices.

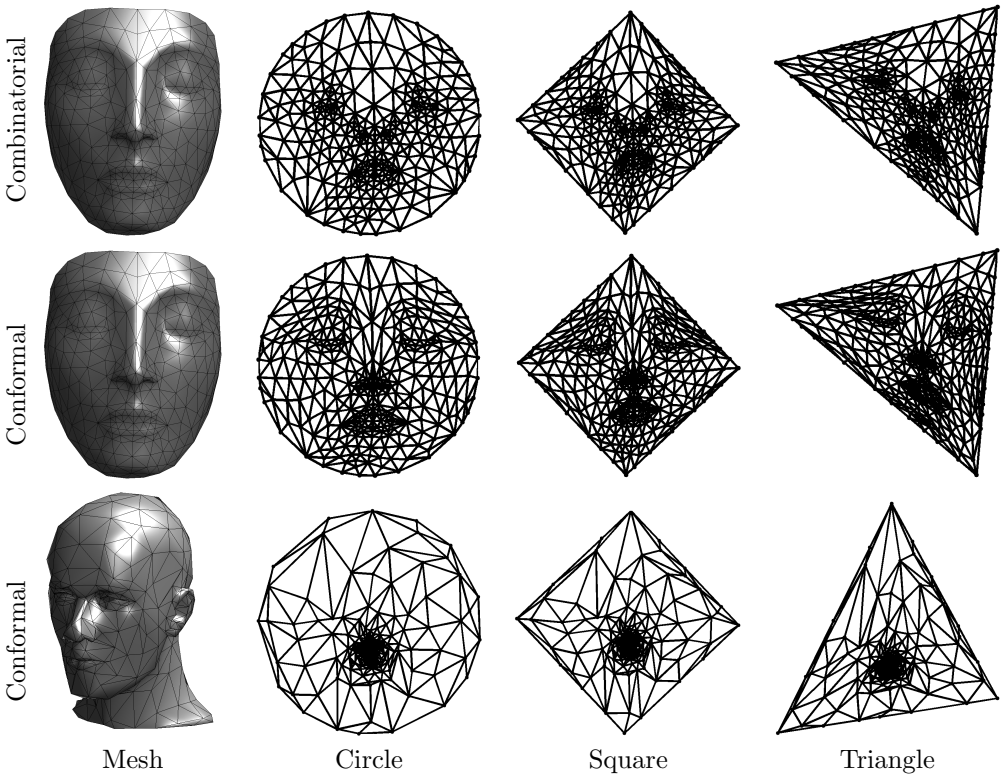


Figure 8.10: *Examples of mesh parameterizations.*

8.3.8 Application to Mesh Flattening

One of the difficulty with linear parameterization methods is that they require to set up the positions of the vertices along the boundary of the mesh. In order to let the boundary free to evolve and find some optimal shape, one can replace the fixed point constraint by a global constraints of unit variance as follow

$$\min_{\psi_1,\psi_2\in\mathbb{R}^n} \|\tilde{G}\psi_1\|^2 + \|\tilde{G}\psi_2\|^2 \qquad \text{with} \qquad \begin{cases} \|\psi_i\| = 1, \\ \langle \psi_1, \psi_2 \rangle = 0, \\ \langle \psi_i, 1 \rangle = 0. \end{cases}$$

This optimization problem also has a simple global solution using eigenvectors of the Laplacian.

Theorem 20 (Mesh flattening solution)**.** *The mesh flattening solution is given by*

$$Span(\psi_1, \psi 2) = Span(u_1, u_2) \quad \textit{where} \quad \tilde{L} = U\Lambda U^{\mathrm{T}}.$$

In order to compute this flattening, one thus needs to extract 2 eigenvectors from a sparse matrix. Note however that, in contrast to linear parameterization schemes, this flattening is not ensured to be bijective. Figure 8.11 shows that for meshes with large distortion, this flattening indeed leads to wrong parameterizations.

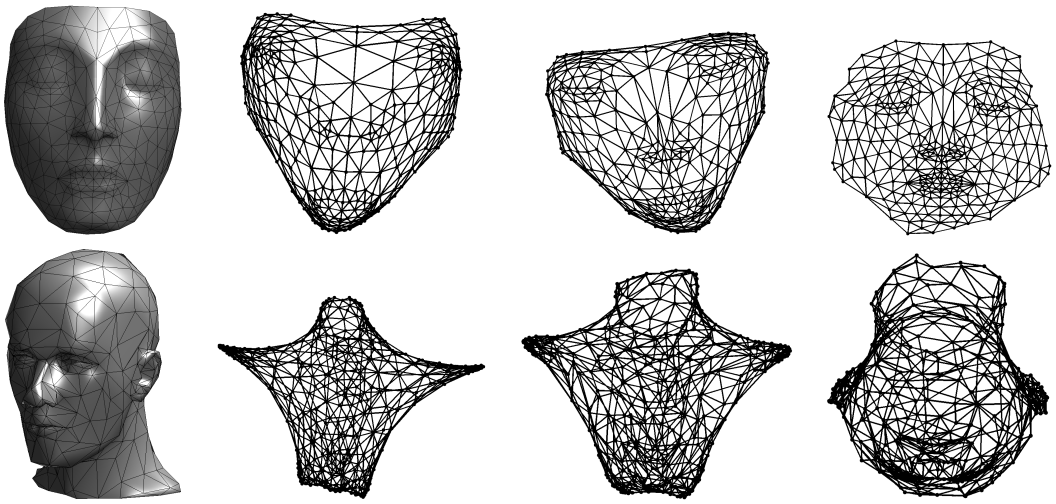


Figure 8.11: *Examples of mesh flattening.*

Chapter 9

Multiresolution Mesh Processing

This chapter shows how computations on a mesh can be performed in a multiscale manner, by considering meshes of increasing resolutions. This leads to the notion of subdivision surfaces and wavelet transform, which are two different tools to interpolates and decompose functions on meshes. Both methods rely on a special kind of meshes whose triangulations can be obtained by applying a regular refinement rule.

9.1 Semi-regular Meshes

9.1.1 Nested Multiscale Grids.

In order to perform multiscale mesh processing, one needs to pack the vertices V of a topological mesh $M = (V, E, F)$ in sets of increasing resolution. As explained in section 8.1.2, it is important to remember that this construction is purely combinatorial, in that no geometrical information (such as actual positions of the vertices in $\mathbb{R}^3$) is required to build the set of multi-resolution meshes. In fact these multiscale grids can be used to actually process the geometrical realization $\mathcal{M}$ of the mesh M as three real valued functions (the three coordinates of the points).

We thus consider a set of nested indexes

$$V_0 \subset V_{-1} \subset \ldots \subset V_L = V$$

which are split according to

$$V_j = V_{j+1} \cup H_{j+1}.$$

Next section describes how to actually compute this set of nested grids using a triangular split, but most of the mathematical tools are in fact valid for arbitrary set of indices, as long as they are embedded in one each other through scales.

For mesh processing, an index $\ell \in V_j$ corresponds to a vertex $x_\ell \in \mathcal{V} \subset \mathbb{R}^3$. The signals to be processed are vectors $f \in \mathbb{R}^n$ of size $n = |V_L|$ defined on the grid V_L. We sometimes write $f \in \ell^2(V_L)$ instead of $f \in \mathbb{R}^n$ to emphasis the domain on which f is indexed. This chapter describes transforms for signals $f \in \ell^2(V_L)$ sampled on the finest grid V_L.

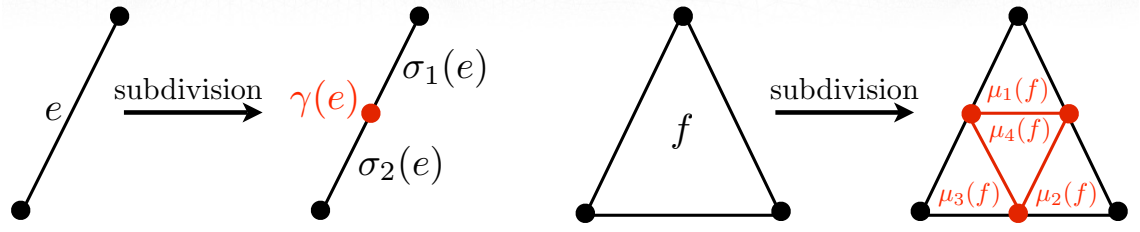


Figure 9.1: *Edge-splitting subdivision.*

9.1.2 Semi-regular Triangulation.

The combinatorial structure of a triangular mesh is defined in section 8.1.2. This chapter considers only a certain class of meshes $M = (V, E, F)$ that can be obtained by a regular split of faces, starting from an initial coarse triangulation. This splitting leads to a set of multiresolution meshes $M_j = (V_j, E_j, F_j)$ for $J \leqslant j \leqslant 0$, where the full mesh is $M_J = M$.

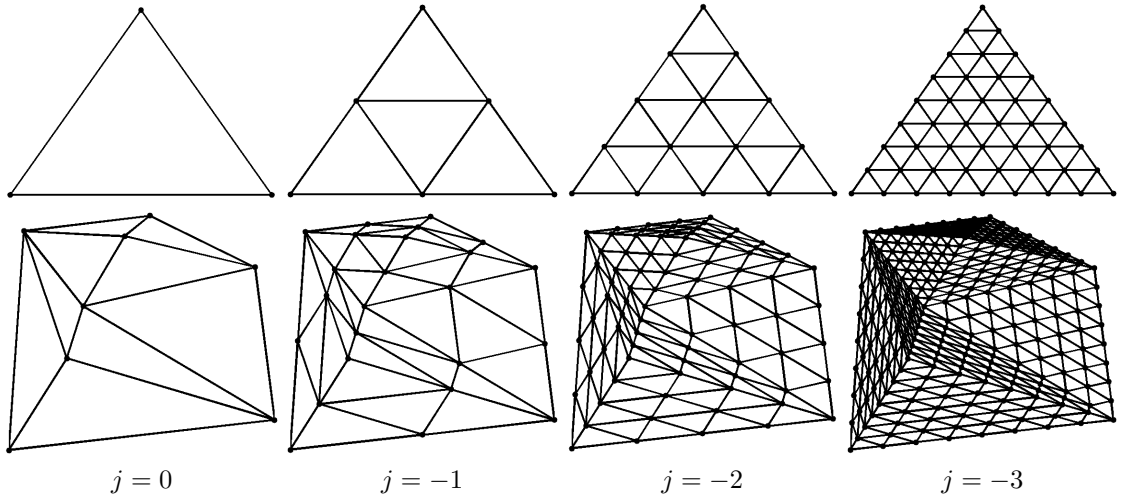


Figure 9.2: *Regular subdivision 1:4 of a single triangle. Regular subdivision of a planar triangulation M_0.*

Starting from this coarse triangulation, one defines by subdivision a multiscale triangulation $(V_j, E_j, F_j)_{L \leqslant j \leqslant 0}$ where

– For each edge $e \in E_j$, a central index $\gamma(e) \in V_{j-1}$ is added to the vertices

$$V_{j-1} = V_j \cup \{\gamma(e) \setminus e \in E_j\}.$$

– Each edge is subdivided into two finer edges

$$\forall\, e = (a, b) \in E_j, \quad \sigma_1(e) = (a, \gamma(e)) \quad \text{and} \quad \sigma_2(e) = (b, \gamma(e)).$$

The subdivided set of edges is then

$$E_{j-1} = \{\sigma_i(e) \setminus i = 1, 2 \quad \text{and} \quad e \in E_j\}.$$

– Each face $f = (a, b, c) \in F_j$ is subdivided into four faces

$$\begin{cases} \mu_1(f) = (a, \gamma(a, b), \gamma(a, c)), \ \mu_2(f) = (b, \gamma(b, a), \gamma(b, c)), \\ \mu_3(f) = (c, \gamma(c, a), \gamma(c, b)), \ \mu_4(f) = (\gamma(a, b), \gamma(b, c), \gamma(c, a)). \end{cases}$$

The subdivided set of faces is then

$$F_{j-1} = \{\mu_i(f) \setminus i = 1, 2, 3, 4 \quad \text{and} \quad f \in F_j\}.$$

Figure 9.1 shows the notations related to the subdivision process. Figure 9.2 shows an example of recursive splitting of a triangle and a coarse triangulation. Figure 9.3 shows examples of semi-regular triangulation using a geometric realization (position of the vertices) to create a 3D surface.

The set of vertices can be classified as

– **Regular vertices** are those who belong neither to the coarse mesh V_0 nor to a boundary of a mesh M_j. These vertices have always 6 neighbors.
– **Extraordinary vertices** are the initial vertices of V_0. They exhibit arbitrary connectivity.
– **Boundary vertices** are those belonging to a mesh boundary. Boundary vertices not in V_0 always have 4 immediate neighbors.

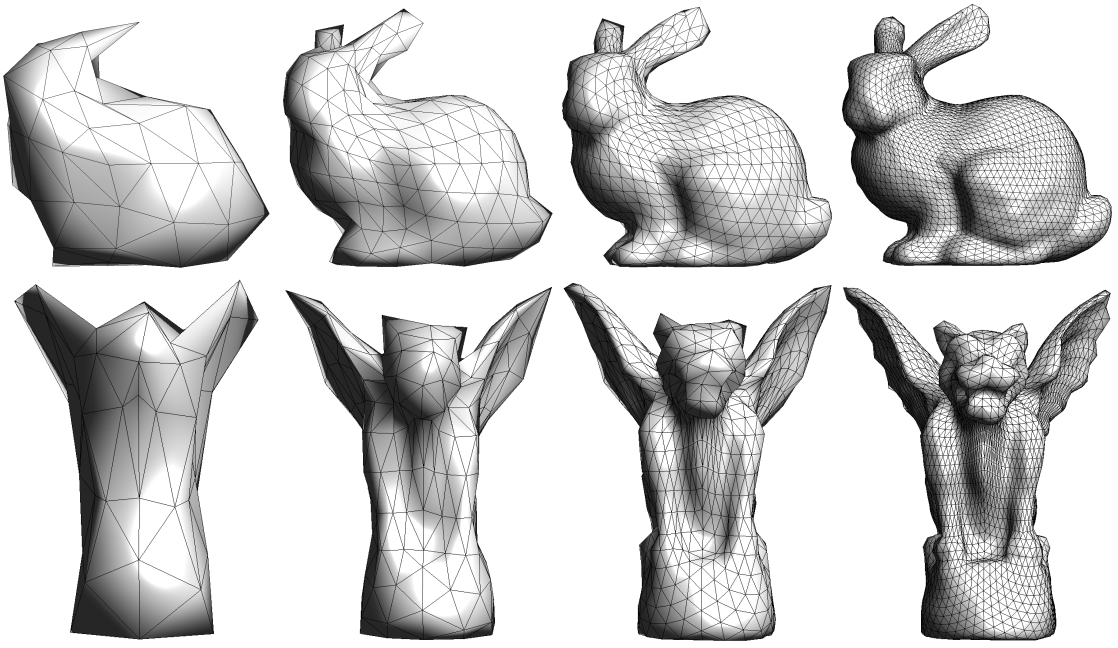


Figure 9.3: *Examples of semi-regular meshes $(V_j)_j$ for increasing scale j (from left to right).*

Obviously not every meshes can be obtained from such a subdivision process. In practice, an arbitrary mesh, obtained from CAD design or range scanning usually does not have any multiscale structure. It is thus necessary to remesh it in order to modify the connectivity of the mesh. During this process, the position of the vertices in $\mathbb{R}^3$ is modified in order for the geometrical realization to stay close from the original piecewise linear surface. One can see [2] fur a survey of various semi-regular remeshing methods.

9.1.3 Spherical Geometry Images

Starting from some input surfaces $\mathcal{S} \subset \mathbb{R}^3$, one typically wants to compute a semi-regular meshes $(M_j)_{j \geqslant L}$ that approximate $\mathcal{S}$. In most case, the surface $\mathcal{S}$ is actually given as an arbitrary triangulated mesh and this process corresponds to a semi-regular remeshing. Many algorithm have been devised for surface remeshing and we describe here a method [24] that works for surfaces that have the topology of a sphere. It means that the surface has genus 0, without boundary and without handles.

This methods works by computing several intermediate surface-wise parameterization.

– *Spherical parameterization:* each points of the original triangulation of $\mathcal{S}$ is mapped onto the unit sphere. This create a bijective parameterization

$$\phi_S : S^2 \to \mathcal{S}.$$

This is a non-linear process that differ from the planar parameterization introduced in section 8.3.7. We do not give the details of such a process, but it requires minimizing the smoothness of the mapping ϕ_S^{-1} under the constraint that it maps points of $\mathcal{S}$ to unit length vectors (point on the sphere S^2). The algorithm is explained in details in [24].

– *Spherical-tetraedron flattening:* one flatten each quadrant (1/8) of the sphere in order to have a mapping

$$\phi_T : \text{Octaedron} \to S^2.$$

One can use for instance a mapping between spherical barycentric coordinate on each quadrant and Euclidean barycentric coordinates on each face of the octahedron.

– *Tetraedron unfolding:* One maps each equilateral face of the octaedron on a rectangular triangle that corresponds to 1/8th of the square $[0,1]^2$

$$\phi_U : [0,1]^2 \to \text{Octaedron}.$$

– *Regular sampling:* the geometry image is obtained by regularly sampling the square on a uniform grid

$$x_\ell = \phi_S \circ \phi_T \circ \phi_U(\ell/n) \quad \text{for} \quad \ell_i = 0, \ldots, n-1.$$

The mapping $\ell \mapsto x_\ell \in \mathbb{R}^3$ is the geometry image, which can be stored as a 3-channel (color) image.

From such a geometry image x_ℓ, one can easily compute a semi-regular mesh by simply performing a regular 1:4 subdivision of the octaedron. Figure 9.4 shows the steps of the construction of a geometry image, and the resulting semi-regular mesh.

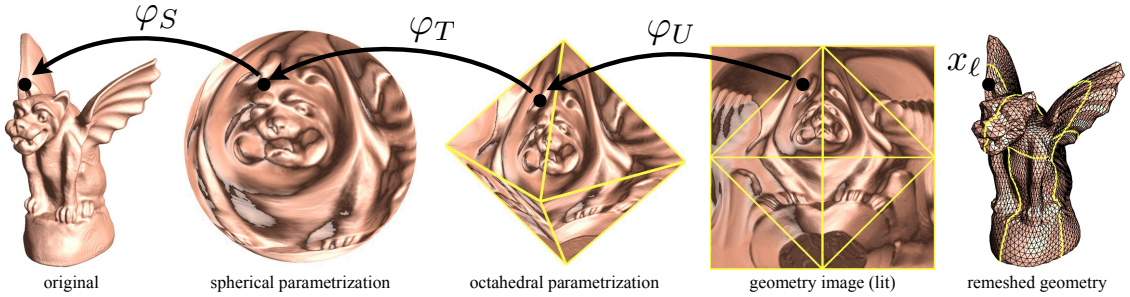


Figure 9.4: *Spherical geometry image construction, taken from [24].*

9.2 Subdivision Curves

Before getting into the detail of subdivision surfaces, we describe the subdivision process in the simpler setting of 1D signals. This leads to the construction of subdivision of 1D functions and subdivision curves.

In this 1D setting, the grid point indexes are dyadic sub-grids of $\mathbb{Z}$

$$\forall\, j \geqslant L, \quad V_j = \left\{ \ell 2^{j-L} \setminus 0 \leqslant \ell < s_0 2^{-j} \right\},$$

where $s_0 = |V_0|$ is the size of the initial vector f_0 to be subdivided.

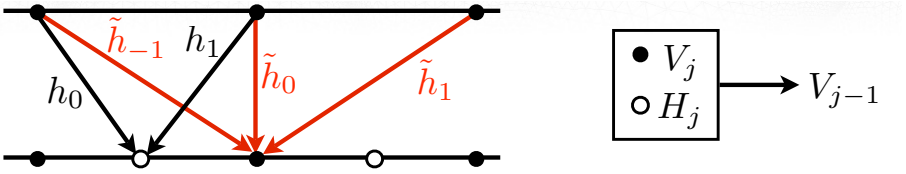


Figure 9.5: *1D subdivision scheme with filters h and $\tilde{h}$. The red curve represent the original signal f^0.*

Each subdivision steps computes, from a set $f_j(\ell) \in \ell^2(V_j)$ of coarse values, a refined vector $f_{j-1} \in \ell^2(V_{j-1})$ defined by

$$\begin{cases} \forall\, k \in H_j,\ f_{j-1}(k) = \sum_t f_j((k-1)/2 + t) h(t), \\ \forall\, \ell \in V_j,\ f_{j-1}(\ell) = \sum_t f_j(\ell + t) \tilde{h}(t). \end{cases}$$

where the set of weights h and $\tilde{h}$ acts as local averaging operators. This averaging should be corrected at the boundary, and we use here cyclic boundary conditions which identifies 0 and $s_0 2^{-j}$ in V_j. Figure 9.5 shows a graphical display of these averaging operators.

One can write this subdivision steps as convolution by introducing the global set of weights

$$g = [\ldots, \tilde{h}(-1), h(0), \tilde{h}(0), h(1), \tilde{h}(1), \ldots]$$

since one has

$$f_{j-1} = (f_j \uparrow 2) * g \quad \text{where} \quad a \uparrow 2 = [\ldots, 0, a(-1), 0, a(0), 0, a(1), 0, \ldots].$$

This corresponds to the traditional description of the wavelet low-pass filtering [20].

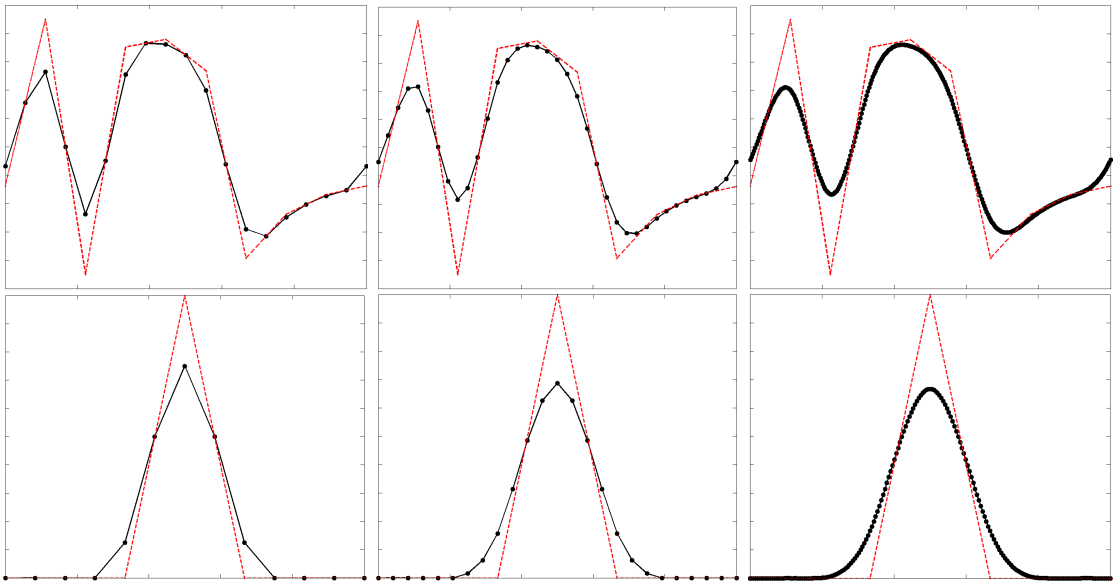


Figure 9.6: *1D subdivision of a signal. Bottom row shows the subdivision from an impulse signal, converging to the scaling function ϕ.*

Figure 9.6 shows several steps of subdivision, starting from an initial vector of size $|V_0| = 10$. One can apply this subdivision of functions to a pair of signals

$$(X_0, Y_0) : V_0 \to \mathbb{R}^2$$

which is a control polygon composed of points located in the plane. The subdivision curve converges to the limiting curve

$$(X_j, Y_j) \overset{j \to -\infty}{\longrightarrow} (X(t), Y(t))_{t=0}^{1} \subset \mathbb{R}^2.$$

An interesting property is that this curve is included in the convex hull of the control polygon

$$(X(t), Y(t))_t \subset \mathrm{Conv}(X_0, Y_0).$$

Figure 9.7 shows examples of subdivision curves.

9.3 Subdivision Surfaces

Subdivision schemes allows to compute a set of progressively refined vectors on a semi-regular mesh. More precisely, from an initial vector $f_0 \in \mathbb{R}^{|V_0|}$ defined on the coarse mesh M_0, local interpolation kernels computes iteratively vectors $f_j \in \mathbb{R}^{|V_j|}$ of finer resolution. When applied to 3 function $(f_0^i)_{i=1,2,3}$ defining the geometrical position of points in $\mathbb{R}^3$, this hierarchical construction defines a subdivision surface. These subdivisions surfaces are used extensively in computer aided geometry and computer graphics. One can see [9] for a survey of subdivision surfaces and their applications.

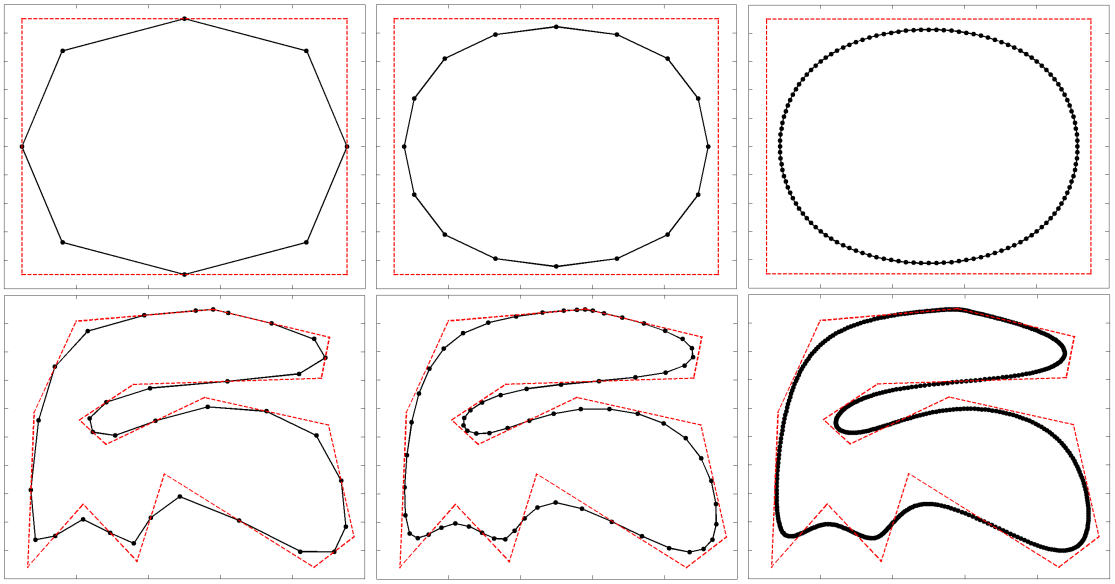


Figure 9.7: *Two examples of subdivision curves. The red curve represent the original curve* (X^0, Y^0).

9.3.1 Interpolation Operators

In order to refine a vector $f_j \in \mathbb{R}^{|V_j|}$ defined on the vertex V_j of the mesh M_j, one uses two interpolators

$$P_j : \ell^2(V_j) \longrightarrow \ell^2(H_j) \qquad \text{and} \qquad \tilde{P}_j : \ell^2(V_j) \longrightarrow \ell^2(V_j). \tag{9.1}$$

A new refined function $f_{j-1} \in \mathbb{R}^{|V_{j-1}|}$ defined on the vertices $V_{j-1} = V_j \cup H_j$ of M_{j-1} is defined by applying these two refinement operators:

$$\forall\, \ell \in V_{j-1}, \quad f_{j-1}(\ell) = \begin{cases} (P_j f_j)(\ell) & \text{if} \quad \ell \in V_j, \\ (\tilde{P}_j f_j)(\ell) & \text{if} \quad \ell \in H_j. \end{cases}$$

Since $V_j \subset V_{j-1}$, the operator $\tilde{P}_j$ only modify slightly the value at vertex in V_j. On the other hand, the operator P_j creates new value at the vertices of H_j that are inserted between V_j and V_{j-1}.

In practical applications, these interpolating operators are local, meaning that the value of $(P_j f_j)(\ell)$ and $(\tilde{P}_j f_j)(\ell)$ depends only on values $f_j(\ell')$ for $\ell' \in V_j$ being close to $\ell \in V_{j-1}$, typically in the 1-ring or 2-ring vertex neighborhood.

A particularly important setting for subdivision scheme is when one apply the subdivision steps in parallel to three vectors (X_j, Y_j, Z_j) starting from three initial vectors describing the position in 3D space of a coarse mesh M_0. This allows to defines finer and finer spacial localization for the vertex of the refined meshes M_j. Figure 9.9 shows an example of such a subdivision surface. In order for the resulting infinitely refined surface to have good properties such as being continuous and even smooth, one needs to design carefully the interpolation operators. Next section gives examples of such operators.

9.3.2 Some Classical Subdivision Stencils

In order to define the interpolation operators P_j and $\tilde{P}_j$ of equation (9.1), one needs to use a naming convention for the neighborhoods of vertices.

For a vertex $\ell \in V_j$, the one ring neighborhood V_ℓ has already been defined in equation (8.1). It is the set of vertices adjacent to ℓ. In a regular point (that does not belongs to V_0 and not on a boundary of the mesh), its size is $|V_\ell| = 6$ since a point has 6 neighbors. This 1-ring is used to define $\tilde{P}_j$.

For a vertex $k \in H_j \subset V_{j-1}$, the butterfly neighborhood is a set of vertices in V_j close to k. This neighborhood is used to define P_j. The two immediate neighbors are

$$(v_k^1, v_k^2) \overset{\text{def.}}{=} \{v \in V_j \setminus (v, k) \in E_{j-1}\}.$$

Two other vertices (w_k^1, w_k^2) are defined using the two faces adjacent to edge $(v_k^2, v_k^2) \in E_j$

$$f_k^1 = (v_k^1, v_k^2, w_k^1) \in F_j \quad \text{and} \quad f_k^2 = (v_k^1, v_k^2, w_k^2) \in F_j.$$

For edges E_j on the boundary of M_j, one one face is available, in which case we implicitly assume that $f_1 = f_2$ (reflecting boundary conditions). The four last vertices are defined using faces adjacent to f_1 and f_2:

$$\forall\, i,j = 1,2, \quad f_k^{i,j} \stackrel{\text{def.}}{=} (z_k^{i,j}, v_k^j, w_k^j) \in F_j \quad \text{with} \quad f_k^{i,j} \neq f_j.$$

Once again, reflecting boundary condition are applied for faces on the boundary of the mesh. The butterfly neighborhood is depicted on figure 9.8.

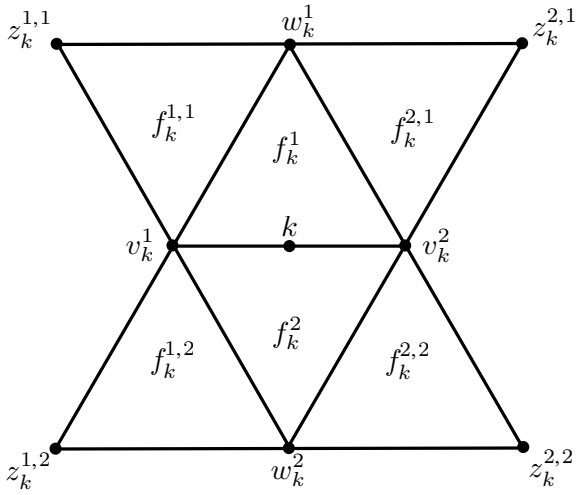


Figure 9.8: *The butterfly neighborhood of a vertex $k \in H_j$.*

Linear Interpolating Scheme The simplest subdivision rule compute values along edge mide point using a simple linear interpolation as follow

$$\begin{cases} \forall\, k \in H_j, & (P_j f_j)(k) = \frac{1}{2}(f(v_k^1) + f(v_k^2)), \\ \forall\, \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = f_j(\ell). \end{cases} \tag{9.2}$$

Since $\tilde{P}_j$ is the identity operator, this scheme is called interpolating. It means that value of f_0 on points of the coarse triangulation are kept during iteration of the subdivision.

Butterfly Interpolating Scheme The linear scheme creates function that are piecewise linear on each face of the coarse triangulation F_0. In order to create smooth surface, one needs to use more points in the butterfly neighborhood as follow

$$\begin{cases} \forall\, k \in H_j, & (P_j f_j)(k) = \frac{1}{2}\sum_{i=1}^2 f(v_k^i) + \frac{1}{8}\sum_{i=1}^2 f(w_k^i) - \frac{1}{16}\sum_{i,j=1}^2 f(z_k^{i,j}), \\ \forall\, \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = f_j(\ell). \end{cases} \tag{9.3}$$

Loop Approximating Scheme In order to gain flexibility in the subdivision design, one can also modify points in V_j during the iterations. This means that $\tilde{P}_j$ is not any more the identity, and that all the values will evolves during the iterations. The question of wether these iterated modification actually converge to a limit value is studied in the next section.

The Loop subdivision rule is defined as

$$\begin{cases} \forall\, k \in H_j, & (P_j f_j)(k) = \frac{3}{8}\sum_{i=1}^2 f(v_k^i) + \frac{1}{8}\sum_{i=1}^2 f(w_k^i), \\ \forall\, \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = (1 - |V_\ell|\beta_{|V_\ell|}) f_j(\ell) + \beta_{|V_\ell|}\sum_{\ell' \in V_\ell} f_j(\ell'). \end{cases}$$

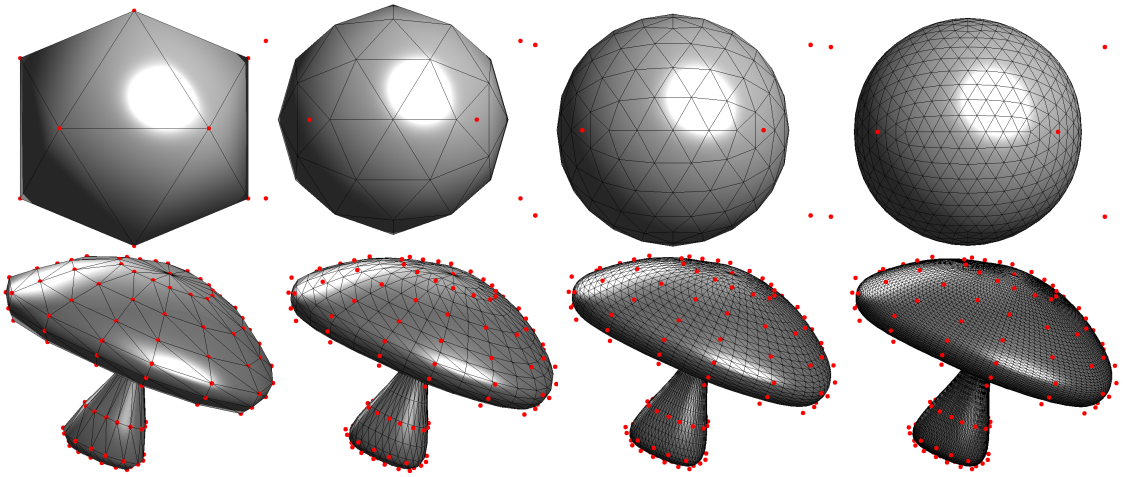


Figure 9.9: *Examples of iterative subdivision using Loop scheme. The points* (X_0, Y_0, Z_0) *of the initial coarse mesh* M_0 *are shown in red.*

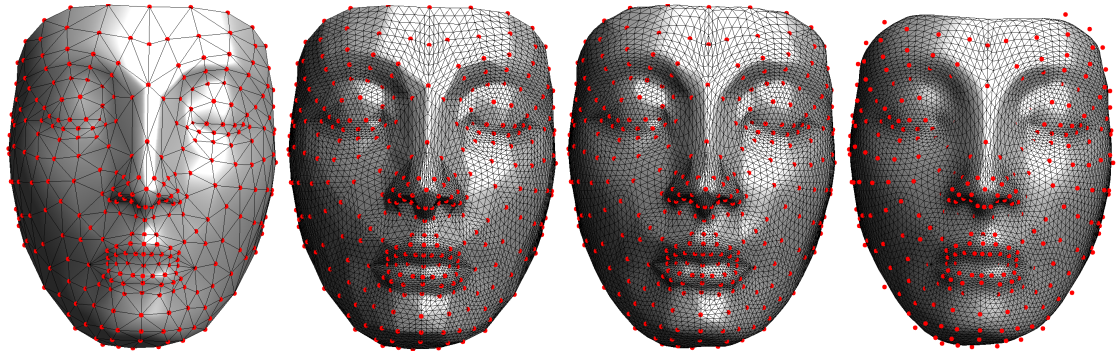


Figure 9.10: *Examples of subdivision schemes. The points* (X_0, Y_0, Z_0) *of the initial coarse mesh* M_0 *are shown in red. Since the linear and butterfly scheme are interpolating, these points actually belongs to the limiting surface.*

where the weights depends on the number of neighbors and are defined as

$$\beta_m \overset{\text{def.}}{=} \frac{1}{m}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos(2\pi/m)\right)^2\right).$$

Other schemes. It is possible to define subdivision schemes using rules that do not involve a regular 1:4 splitting of each coarse face. For instance, in dual schemes such as the one depicted in figure 9.11, the faces of F_j are not included in F_{j-1} but only in F_{j-2}.

9.3.3 Invariant Neighborhoods

In order to study the convergence of subdivision schemes, one needs to consider independently each vertex $x \in V_{j_0(x)}$, where $j_0(x)$ is the coarser scale at which x appears

$$j_0(x) = \max\ \{j \setminus x \in V_j\}.$$

Original vertices satisfy $j_0(x) = 0$ and are the only one (except boundary vertices) that have a non-regular connectivity.

The vertex x belongs to the mesh $M_{j_0(x)}$ which is going to be refined through scales $j < j_0(x)$. In order to analyze this refinement, one needs to define an invariant neighborhood $V_j^x \subset V_j$ of x for each scale $j \leqslant j_0(x)$. These neighborhood are the set of points that are required to compute the

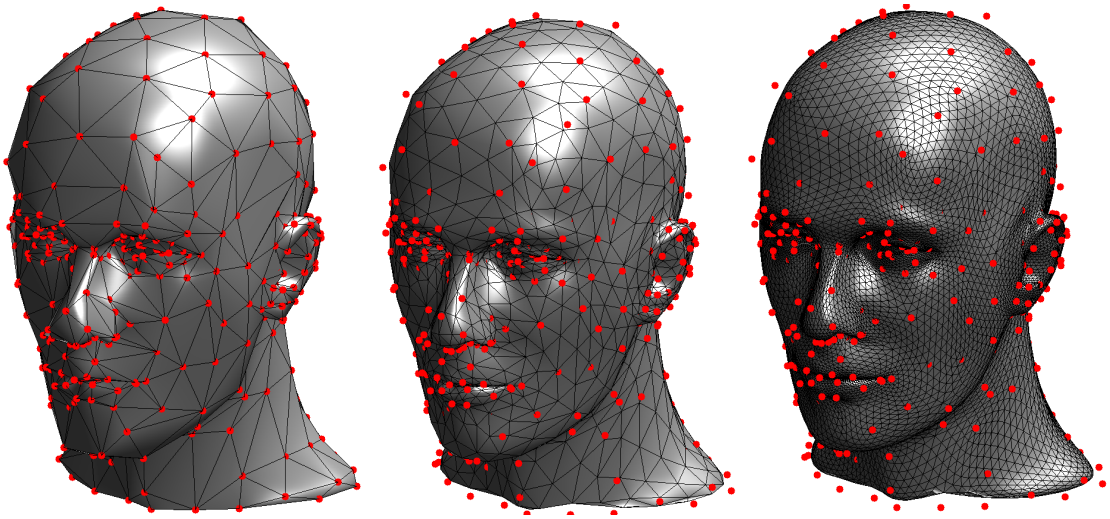


Figure 9.11: *Surface after 0, 1 and 3 step of $\sqrt{3}$ subdivision [18].*

operators P_j and $\tilde{P}_j$. More precisely, given a vector $f \in \ell^2(V_{j-1})$, the neighborhoods are required to satisfy

$$\begin{cases} \forall\, \ell \in V_{j-1}^x \cap V_j, & (\tilde{P}_j f)(\ell) \text{ depends only on } V_j^x \\ \forall\, k \in V_{j-1}^x \cap H_j, & (P_j f)(k) \text{ depends only on } V_j^x. \end{cases}$$

We further impose that all the invariant neighborhoods have the same size

$$\forall\, j \leqslant j_0(x), \quad \#V_j^x = m_x.$$

Figure 9.12 shows an example of invariant neighborhood which corresponds to the 2-ring $V_\ell^{(2)}$, as defined in (8.2).

Thanks to the invariance of these neighborhood systems, one can restrict the predictors around x and define

$$P_j^x : V_j^x \longrightarrow V_{j-1}^x \cap V_j \qquad \text{and} \qquad \tilde{P}_j^x : V_j^x \longrightarrow V_{j-1}^x \cap H_j.$$

The subdivision matrix $S_j^x \in \mathbb{R}^{m_x \times m_x}$ is then defined as matrix of the following mapping

$$(\tilde{P}_j^x, P_j^x) : V_x^j \longrightarrow V_x^{j-1}.$$

All the subdivision schemes studied in this chapter are invariant, meaning that the subdivision rule does not change through the scales j. This impose that the subdivision matrices are constant $S_j^x = S^x$. In fact, in all the examples given in the previous section, they only depends on the number $|V_x|$ of neighbors in the one ring of x.

9.3.4 Convergence of Subdivisions

The value at $x \in V_{j_0(x)}$ of a function $f_j \in \ell^2(V_j)$ obtained by subdividing at scale $j \leqslant j_0(x)$ an initial vector $f_0 \in \ell^2(V_0)$ can be computed as

$$f_j(x) = \left(S^x f_{j+1}^x\right)(x) = \left((S^x)^{j_0(x)-j} f_{j_0(x)}^x\right)(x),$$

where the vector $f_i^x \in \mathbb{R}^{m_x}$ is the restriction of f_i to the set V_j^x.

In order to analyze the limiting function resulting from an infinite number of subdivision, one can use the eigen vector decomposition of the matrix S^x

$$S^x = \tilde{\Phi} V \Lambda \Phi^{\mathrm{T}} \quad \text{where} \quad \begin{cases} \Phi^{\mathrm{T}} = \tilde{\Phi}^{-1}, \\ \Lambda = \mathrm{diag}(\lambda_i), \lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_{m_x}. \end{cases}$$

Since the subdivision matrix S_x is not symmetric, some of the eigenvalues might be complex, and we shall ignore this difficulty here. The fact that P_j and $\tilde{P}_j$ are predictor implies that the

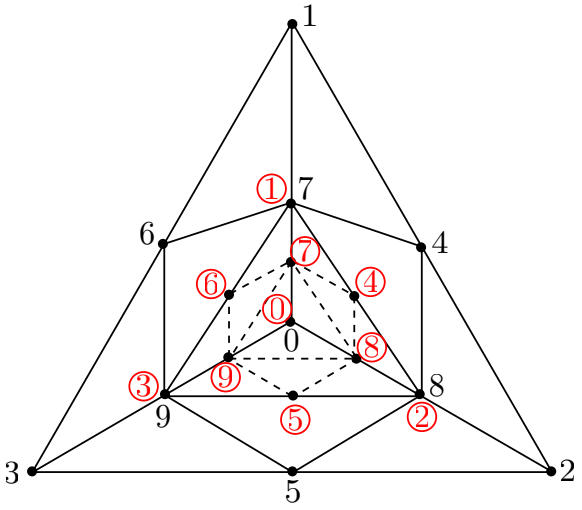


Figure 9.12: *Invariant neighborhood V_j^x and V_{j-1}^x (indexing with red circles) of the Loop subdivision scheme for a vertex of valence $|V_\ell| = 0$. The number in $\{0, \dots, 9\}$ refers to the numbering of the vertices in V_j^x and V_{j-1}^x*

subdivision matrix has to satisfy $S^x 1 = 1$, meaning that $\tilde{\phi}_1 = 1$ is an eigenvector associated to the eigenvalue 1. In the following we further makes the following assumption

$$1 = \lambda_1 < \lambda \stackrel{\text{def.}}{=} \lambda_2 = \lambda_3 < \lambda_4. \tag{9.4}$$

This hypothesis is satisfied by all the subdivision rules introduced in the previous section.

If one write $\Phi = (\phi_i)_{i=1}^m$ and $\Phi = (\phi_i)_{i=1}^m$, one has the following decomposition of a vector $f \in \mathbb{R}^{m_x}$

$$f = \sum_{i=1}^{m_x} \langle f, \phi_i \rangle \tilde{\phi}_i \quad \text{and} \quad (S^x)^k(x) = \sum_{i=1}^{m_x} \lambda_i^k \langle f, \phi_i \rangle \tilde{\phi}_i.$$

One thus has the following asymptotic expansion

$$\frac{1}{\lambda^k} (f - \langle f, \phi_1 \rangle 1) = \langle f, \phi_2 \rangle \tilde{\phi}_2 + \langle f, \phi_3 \rangle \tilde{\phi}_3 + o(1). \tag{9.5}$$

This expression describes the asymptotic behavior of the subdivision scheme at zero order (position) and first order (tangents).

Theorem 21 (Convergence of the subdivision scheme)**.** *If the subdivision matrix S^x of a point x satisfies* (9.4) *then the subdivision process converges at x to the value*

$$f^j(x) \stackrel{j \to -\infty}{\longrightarrow} \langle f^x_{j_0(x)}, \phi_1 \rangle.$$

The smoothness of the resulting function is more difficult to analyze. A particularly important setting is when one computes the subdivision of 3 function $p_0 = (X_0, Y_0, Z_0) \in \ell^2(V_0)^3$ corresponding to the position in $\mathbb{R}^3$ (geometrical realization) of a coarse mesh M_0. In this case, the subdivided functions $p_j = (X_j, Y_j, Z_j)$ gives refined 3D meshes that converge uniformly to a continuous surfaces

$$p(x) = (X(x), Y(x), Z(x)) = (\langle X^x_{j_0}, \phi_1 \rangle, \langle Y^x_{j_0}, \phi_1 \rangle, \langle Z^x_{j_0}, \phi_1 \rangle).$$

Condition (9.4) nearly implies that the resulting surface is smooth. Indeed, the asymptotic expansion (9.5) shows that for a point x' near x in the subdivision domain, the differential vector can be well approximated as a projection on a 2D plane

$$p(x) - p(x') + o(1) \in \text{Span}(\tau_2^x, \tau_3^x) \quad \text{where} \quad \tau^i(x) \stackrel{\text{def.}}{=} (\langle X^x_{j_0}, \phi_i \rangle, \langle Y^x_{j_0}, \phi_i \rangle, \langle Z^x_{j_0}, \phi_i \rangle).$$

If the vectors τ_2^x and τ_3^x are linearly independent, they form a basis of the tangent plane at $p(x)$.

Example of the Loop subdivision. For the Loop interpolation operators defined in equation (9.3.2), the invariant neighborhood V_j^x correspond to the 2-ring of x in the triangulation G_j, as shown in figure 9.12. For a vertex with k neighbors, $|V_x| = k$, the size of these invariant neighborhood is $m_x = 3k + 1$. A particular neighboring for $k = 3$ is depicted in figure 9.12, together with an indexing in $\{0, \dots, 3k = 9\}$ of the points in V_j^x and V_{j-1}^x. For this indexing, the subdivision matrix reads

$$\begin{pmatrix}
7 & & & & & & 3 & 3 & 3 \\
1 & 1 & & 1 & & 1 & 10 & 1 & 1 \\
1 & & 1 & & 1 & 1 & & 1 & 10 & 1 \\
1 & & & 1 & & 1 & 1 & 1 & 1 & 10 \\
1 & & & & 1 & & & 3 & 3 & \\
1 & & & & & 1 & & & 3 & 3 \\
1 & & & & & & 1 & 3 & & 3 \\
1 & & & & & & & 3 & 1 & 1 \\
1 & & & & & & & 1 & 3 & 1 \\
1 & & & & & & & 2 & 1 & 3
\end{pmatrix}$$

where the 0's have been omitted and where the rows should be rescaled to sum to 1. The eigenvalues of this matrix satisfy $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 1/3 > \lambda_4$.

9.4 Wavelets on Meshes

9.4.1 Multiscale Biorthogonal Bases on Meshes

The transforms considered in this section are multiscale and indexed by the set of nested grids $(V_j)_{L<j\leqslant J}$. This corresponds to computing a set of coefficients $(d_j)_{L<j\leqslant J} \cup f_J$ from an initial input signal f. These coefficients corresponds to inner products with basis vectors

$$\begin{cases} d_j \in \ell^2(H_j) & \text{where} \quad \forall\, k \in H_j,\ d_j(k) = \langle f,\, \psi_{j,k} \rangle, \\ f_J \in \ell^2(V_J) & \text{where} \quad \forall\, \ell \in V_J,\ f_J(\ell) = \langle f,\, \phi_{J,\ell} \rangle. \end{cases}$$

By analogy with the wavelet setting, the vectors $\psi_{j,k} \in \mathbb{R}^n$ corresponds to primal wavelets and are intended to capture the details present in the signal f at a scale j, whereas the scaling vectors $\phi_{J,k} \in \mathbb{R}^n$ capture the missing coarse approximation of f at scale J. This decomposition is stopped at any coarse scale $L < J \leqslant 0$.

In order to reconstruct the function f from this set of transformed coefficients, one needs to use a set of bi-orthogonal basis vectors

$$f = \sum_{L<j\leqslant J, k\in H_j} d_j(k)\tilde{\psi}_{j,k} + \sum_{\ell \in V_J} f_J(\ell)\tilde{\phi}_{J,\ell}.$$

If this reconstruction formula holds for any scale $L < J \leqslant 0$, the set of vectors

$$(\psi_{j,k}, \phi_{j,\ell})_{k\in H_j, \ell \in V_j}^{L<j\leqslant 0} \qquad \text{and} \qquad (\tilde{\psi}_{j,k}, \tilde{\phi}_{j,\ell})_{k\in H_j, \ell \in V_j}^{L<j\leqslant 0}, \tag{9.6}$$

is said to be a pair of primal and dual multiscale bases (together with their scaling functions).

The following paragraph shows how one can modify such a pair of multiscale bases while still maintaining the biorthogonality property. This lifting process is useful to design multiscale bases with various properties on complicated domains.

9.4.2 The Lifting Scheme

The lifting scheme is a construction of multiscale biorthogonal bases introduced by Sweldens [31, 32]. It extends the traditional construction of wavelets in two main directions:

– As explained in [12], it allows to implement already existing filter banks more efficiency by splitting the computation into elementary blocks. This computational gain is described at the end of the section together with the factorization of wavelets into lifting steps.

– It allows to define multiscale transforms over domains that are not translation invariant. This section gives two examples of such transforms: a non-separable 2D wavelet transform and wavelets on triangulated meshes.

In order to build wavelets on triangulation, one can specialize the lifting scheme to a particular setting where only two lifting steps are applied.

Forward lifting scheme. The forward algorithm performs the transform

$$(f_{j-1}(\ell))_{\ell\in V_{j-1}} \quad \longrightarrow \quad (d_j(k))_{k\in H_j} \cup (f_j(\ell))_{\ell\in V_j}$$

by applying the following steps

– *Splitting:* this corresponds selecting the coefficient of $f_{j-1}(\ell)$ that are in V_j or in H_j

$$(f_{j-1}(\ell))_{\ell\in V_{j-1}} = (f_j(\ell))_{\ell\in V_j} \cup (f_j(\ell))_{\ell\in H_j}.$$

These two sets of coefficients are treated differently in the two remaining steps of the transform.

– *Predict step:* creates wavelets coefficients d_j by computing local differences between each coefficient in V_j and its neighbors in H_j

$$\forall\, k \in H_j, \quad d_j(k) = f_{j-1}(k) - \sum_{\ell\in V_j} p_j(k,\ell) f_{j-1}(\ell).$$

The coefficients $p_j(k,\ell)$ are weights that determine the predict operator

$$P_j : \left\{ \begin{array}{ccc} \ell^2(V_j) & \longrightarrow & \ell^2(H_j) \\ g & \longmapsto & h = P_j g \end{array} \right. \quad \text{where} \quad h(k) = \sum_{k\in H_j} p_j(k,\ell) g(\ell).$$

– *Update step:* enhance the properties of each remaining low pass coefficients $f_{j-1}(\ell)$ for $\ell \in V_j$ by pooling locally the wavelets coefficients $d_j(k)$ for k around ℓ

$$\forall\, \ell \in V_j, \quad f_j(\ell) = f_{j-1}(\ell) + \sum_{k\in H_j} u_j(\ell,k) d_j(k).$$

The coefficients $u_j(\ell,k)$ are weights that determine the update operator

$$U_j : \left\{ \begin{array}{ccc} \ell^2(H_j) & \longrightarrow & \ell^2(V_j) \\ h & \longmapsto & g = U_j h \end{array} \right. \quad \text{where} \quad g(\ell) = \sum_{k\in H_j} u_j(\ell,k) h(k).$$

Figure 9.13, top row, shows the block diagram associated to this forward lifting wavelet transform.

The iterations of the forward lifting transform can also be written in vector and operator format

$$\left\{ \begin{array}{l} d_j = f_{j-1}^{H_j} - P_j f_{j-1}^{V_j}, \\ f_j = f_{j-1}^{V_j} + U_j d_j = (\mathrm{Id}_{V_j} - U_j P_j) f_{j-1}^{V_j} + U_j f_{j-1}^{H_j}, \end{array} \right.$$

where g^A is the restriction of some vector g to the set A.

Backward lifting scheme. The backward transform algorithm does the reverse computation

$$(d_j(k))_{k\in H_j} \cup (f_j(\ell))_{\ell\in V_j} \quad \longrightarrow \quad (f_{j-1}(\ell))_{\ell\in V_{j-1}}$$

One of the main feature of the lifting scheme is that this is achieved by simply reversing the order of the lifting steps and interchanging +/- signs.

– *Inverse update step:*

$$\forall\, \ell \in V_j, \quad f_{j-1}(\ell) = f_j(\ell) - \sum_{k\in H_j} u_j(\ell,k) d_j(k).$$

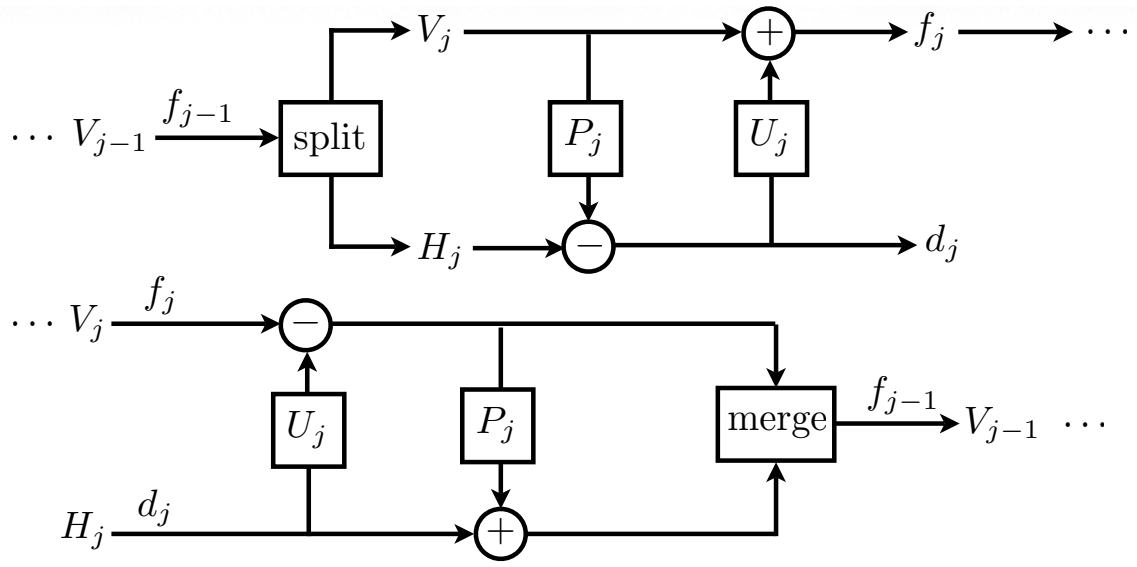


Figure 9.13: *Block diagrams for the forward and backward lifting scheme.*

– *Inverse predict step:*

$$\forall\, k \in H_j, \quad f_{j-1}(k) = d_j(k) + \sum_{\ell \in V_j} p_j(k,\ell) f_{j-1}(\ell).$$

– *Merging:* makes the union of the coefficients computed in the two previous steps

$$(f_{j-1}(\ell))_{\ell \in V_{j-1}} = (f_j(\ell))_{\ell \in V_j} \cup (f_j(\ell))_{\ell \in H_j}.$$

Figure 9.13, top row, shows the block diagram associated to this backward lifting wavelet transform.

The lifting scheme is more general than the algorithm described in this section since several passes of predict/update steps can be applied to further enhance the properties of the resulting transform. However, the steps beyond the two initial ones are difficult to analyze, except in the notable exception of points sampled evenly on a 1D axes, where a factorization algorithm [12] allows to recover traditional wavelet filters.

9.4.3 Imposing vanishing moments.

The operator P_j is called a predictor since the values of $P_j f_{j-1}^{V_j}$ should typically be close to $f_{j-1}^{H_j}$ for the wavelet coefficients d_j to be small. Such predictors have already been constructed in equations (9.2), (9.3) and (9.3.2).

The operator U_j is called an update operator since the additional term $U_j d_j$ should enhance the properties of $f_{j-1}^{V_j}$. This update steps does not appears in the theory of subdivision surface and this section considers a local update operator which guaranty the conservation of the mean value when switching from f_{j-1} to f_j.

Polynomial vectors. In order to select predict and update operator that have good properties, one follow the insight gained from the analysis of the wavelet approximation of signal on the real line. In order to do so, one need analyze the effect of a lifting wavelet transform on polynomials. The most basic constraint enforces one vanishing moment by imposing orthogonality with the constant vector $\Phi_0 = 1$. This constraint does not require to known the spacial location x_ℓ of each index $\ell \in V_L$. In order to impose higher order vanishing moments, one needs to assume some sampling pattern, for instance

$$\forall\, \ell \in V_L, \quad f(\ell) = \bar{f}(x_\ell) \qquad \text{where} \qquad x_\ell \in \mathbb{R}^q$$

and where $\bar{f}$ is a function defined on $\mathbb{R}^q$. For instance, the points x_ℓ might corresponds to a regular sampling of the line (this is the traditional wavelet setting) or to an irregular sampling of a 2D

surface embedded in $\mathbb{R}^3$. The next paragraphs describe several situations with different sampling grids. Once the precise locations of the samples are known, one can for instance select Φ_s as some monomials of degree $(s_1, \ldots, s_q)$ over $\mathbb{R}^q$.

Vanishing moment and polynomials reproduction. Having defined these polynomial vectors, one requires that the following constraints are fulfilled.

– *Vanishing moments:* the wavelet coefficients of a low order polynomial should be 0, which implies that

$$\forall\, k \in H_j,\ \langle \Phi_s,\, \psi_{j,\ell} \rangle = 0. \tag{9.7}$$

– *Polynomial reproduction:* coarse coefficients f_j computed from a polynomial f_{f-1} should also be polynomials, which implies that

$$\forall\, \ell \in V_j,\ \langle \Phi_s,\, \phi_{j,\ell} \rangle = \Phi_s(\ell) \tag{9.8}$$

In order for the wavelets and scaling function to satisfy conditions (9.7) and (9.8), the predict operator P_j and update operator U_j should be designed carefully. One can impose these constraint from the fine scale $j = L$ until the coarse scale $j = 0$. Indeed, if $(\phi_{j-1,\ell}, \psi_{j-1,\ell})_{k,\ell}$ satisfy conditions (9.7) and (9.8), then, for the scale j

$$\forall\, s \in S, \quad \begin{cases} (9.7) & \Longleftrightarrow & P_j \Phi_s^{V_j} = \Phi_s^{H_j}, \\ (9.8) & \Longleftrightarrow & {U_j}^{\mathrm{T}} \left(\Phi_s^{V_j} + {P_j}^{\mathrm{T}} \Phi_s^{H_j} \right) = \Phi_s^{H_j}. \end{cases}$$

where $\Phi_s^A \in \ell^2(A)$ is the restriction of Φ_s to A.

In contrast, the constraint (9.8) on the update operator P_j is more involved and the next section shows how to handle it on a triangulation situations for only one vanishing moment $|S| = 1$.

9.4.4 Lifted Wavelets on Meshes

The lifted wavelet bases can be used to process signals $f \in \ell^2(V_L)$ where $\ell \in V_L$ index a sampling x_ℓ of an arbitrary surface. The construction of biorthogonal wavelets on triangulated mesh has been first proposed by Lounsbery et al. [19] and re-casted into the lifting scheme framework by Schroeder and Sweldens [26, 27].

Designing predict operators. The constraints (9.7) on the predictor P_j is easily solved. For instance, for each k, one selects only $|S|$ non vanishing weights $(p_j(k,\ell))_\ell$ and solves a small $|S| \times |S|$ linear system. Furthermore, in the case of a regular triangulation with edges of constant length, predictors with several vanishing moments have been already defined in (9.2), (9.3) and (9.3.2). Figure 9.14 shows the weights for these predictors.

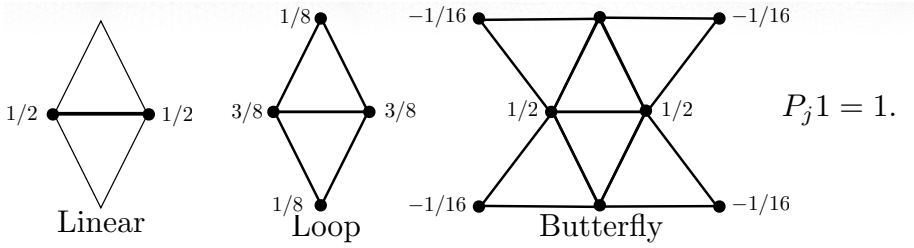


Figure 9.14: *Predict operators on a triangulation.*

One can choose any of these operators, and creates respectively linear, butterfly and Loop wavelets bases. All these predictors have one vanishing moment since they satisfy $P_j 1^{H_j} = 1^{V_j}$. In fact they have more vanishing moments if one consider polynomials Φ_s sampled at points $x_\ell \in \mathbb{R}^2$ of an hexagonal tiling with constant edge length. In practice, if the triangulation under consideration have edges with smoothly varying length, the resulting predictor are efficient to predict the value of smooth functions on the triangulation.

Designing update operators. In order to ensure the reproduction of constant polynomials, we design the update operator so that it depends only on the direct neighbors in H_j of each point in V_j

$$\forall\, \ell \in V_j, \quad V_\ell = \{\gamma(\ell,\ell') \setminus (\ell,\ell') \in E_j\}\,.$$

One then looks for a valid update operator in the following form

$$\forall\, h \in \ell^2(H_j),\ \forall\, \ell \in V_j, \quad (U_j h)(\ell) = \lambda_\ell \sum_{k \in V_\ell} h(k), \tag{9.9}$$

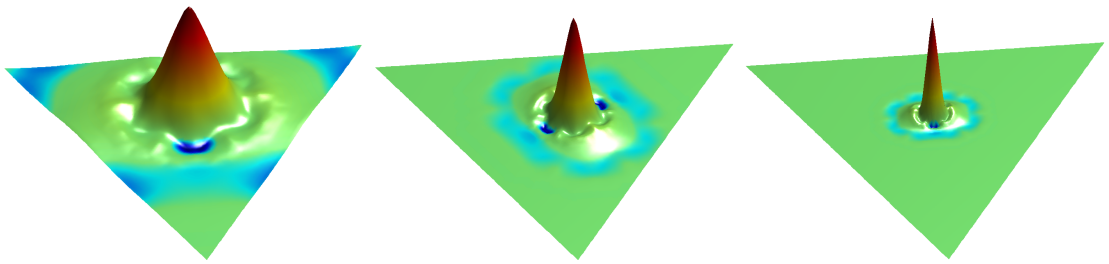
where each λ_ℓ should be fixed in order for condition (9.8) to be satisfied.

In an semi-regular triangulation, $|V_\ell| = 6$ excepted maybe for some points in the coarse grid $\ell \in V_0$. In this setting, the values of λ_ℓ can be computed by a recursion through the scales. In an ideal triangulation where $|V_\ell| = 6$ for all ℓ, one can use a constant weight $\lambda_\ell = \lambda$.

For the predictors defined in (9.2), (9.3) and (9.3.2), one has

$$P_j{}^{\mathrm{T}} 1^{H_j} = 3 \times 1^{V_j} \qquad \text{and} \qquad U_j{}^{\mathrm{T}} 1^{V_j} = 6\lambda 1^{H_j}$$

so setting $\lambda_\ell = 1/24$ solves equation (9.8). Figure 9.15 shows examples of butterfly wavelets on a planar semi-regular triangulation.



$j = -2$ $\qquad$ $j = -3$ $\qquad$ $j = -4$

Figure 9.15: *Example of wavelets $\psi_{j,k}$ on a semi-regular triangulation. The height over the triangle (together with the color) indicates the value of the wavelet vector.*

9.4.5 Non-linear Mesh Compression

These wavelets can be used to perform an approximation of a function $f \in \ell^2(V_L)$ defined on the fine triangulation. For instance a wavelet approximation can be applied to each coordinate $f_i, i = 1,2,3$ of the actual position $x_\ell = (f_1(\ell), f_2(\ell), f_3(\ell)) \in \mathbb{R}^3$ of the surface points, as done in [16, 17]. This leads to a scheme to approximate and compress a 3D surface using the lifted biorthogonal wavelets associated to the semi-regular triangulation. This is possible because these wavelets depend only on the combinatorial grids V_j and not on the precise position of the samples x_ℓ in 3D.

In order to perform a wavelet approximation in this biorthogonal basis, one uses a non-linear thresholding at $T > 0$

$$f = \sum_{(j,k) \in I_T} \langle f,\, \psi_{j,k} \rangle \tilde{\psi}_{j,k}$$

$$\text{where} \quad I_T = \left\{ (j,k) \setminus k \in H_j \quad \text{and} \quad |\langle f,\, \psi_{j,k} \rangle| > T |\operatorname{supp}(\psi_{j,k})|^{-1/2} \right\}.$$

Note that for each coefficient the threshold T is scaled according to the size of the support of the wavelet in order to approximately normalize the wavelets in $\ell^2(V_L)$ norm.

Figure 9.16 shows an example of compression of the position of a vertex in 3D spaces as 3 functions defined on a semi-regular mesh. Figure 9.17 shows an example of compression of a spherical texture map which is a single function defined at each vertex of a semi-regular mesh obtained by subdividing an icosaedron.

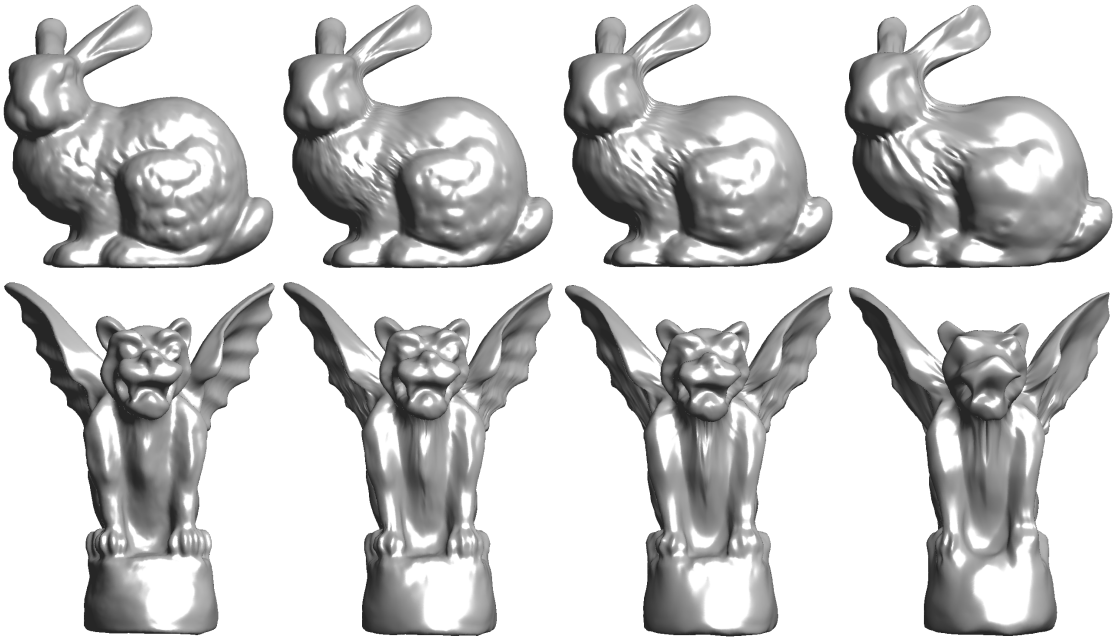


Figure 9.16: *Non-linear wavelet mesh compression with a decreasing number of coefficients.*

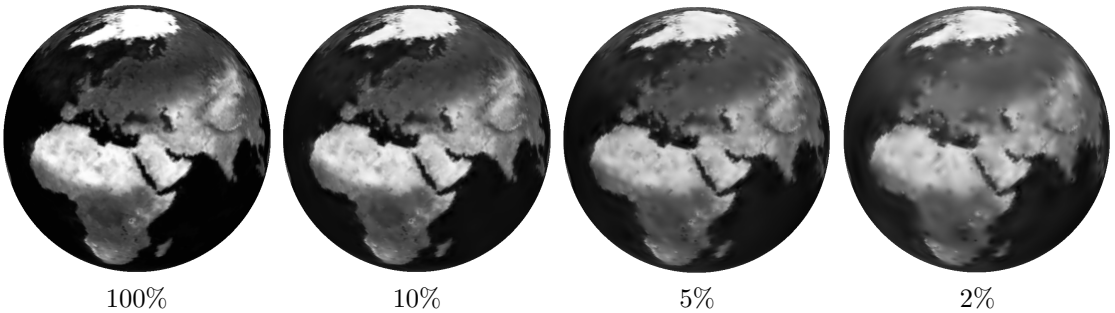


Figure 9.17: *Non-linear spherical wavelet compression with a decreasing number of coefficients.*

Bibliography

[1] P. Alliez and C. Gotsman. Recent advances in compression of 3d meshes. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 3–26. Springer Verlag, 2005.

[2] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. In *AIM@SHAPE repport*. 2005.

[3] E. Candès and D. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. *Commun. on Pure and Appl. Math.*, 57(2):219–266, 2004.

[4] E. J. Candès, L. Demanet, D. L. Donoho, and L. Ying. Fast discrete curvelet transforms. *SIAM Multiscale Modeling and Simulation*, 5:861–899, 2005.

[5] A. Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97, 2004.

[6] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[7] F. R. K. Chung. Spectral graph theory. *Regional Conference Series in Mathematics, American Mathematical Society*, 92:1–212, 1997.

[8] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4(4), 2005.

[9] P. Schroeder et al. D. Zorin. Subdivision surfaces in character animation. In *Course notes at SIGGRAPH 2000*, July 2000.

[10] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, PA, 1992.

[11] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. on Pure and Appl. Math.*, 57:1413–1541, 2004.

[12] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[13] D. Donoho and I. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, Dec 1994.

[14] M. Figueiredo and R. Nowak. An EM Algorithm for Wavelet-Based Image Restoration. *IEEE Trans. Image Proc.*, 12(8):906–916, 2003.

[15] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

[16] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In Alyn Rockwood, editor, *Proceedings of the Conference on Computer Graphics (Siggraph99)*, pages 325–334. ACM Press, August8–13 1999.

[17] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pages 271–278, New York, July 23–28 2000. ACMPress.

[18] L. Kobbelt. $\sqrt{3}$ subdivision. In Sheila Hoffmeyer, editor, *Proc. of SIGGRAPH'00*, pages 103–112, New York, July 23–28 2000. ACMPress.

[19] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph.*, 16(1):34–73, 1997.

[20] S. Mallat. *A Wavelet Tour of Signal Processing, 3rd edition.* Academic Press, San Diego, 2009.

[21] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. on Pure and Appl. Math.*, 42:577–685, 1989.

[22] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1, Ser. A):127–152, 2005.

[23] J. Portilla, V. Strela, M.J. Wainwright, and Simoncelli E.P. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Proc.*, 12(11):1338–1351, November 2003.

[24] E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Transactions on Graphics*, 22(3):340–349, July 2003.

[25] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

[26] P. Schröder and W. Sweldens. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In *Proc. of SIGGRAPH 95*, pages 161–172, 1995.

[27] P. Schröder and W. Sweldens. Spherical wavelets: Texture processing. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*. Springer Verlag, Wien, New York, August 1995.

[28] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[29] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.

[30] G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Boston, 1996.

[31] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computation Harmonic Analysis*, 3(2):186–200, 1996.

[32] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.

[33] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding.* Prentice-Hall, Englewood Cliffs, NJ, 1995.