TAMPERE UNIVERSITY OF TECHNOLOGY

MATTEO MAGGIONI

VIDEO FILTERING USING SEPARABLE FOUR-DIMENSIONAL NONLOCAL SPATIOTEMPORAL TRANSFORMS

Master of Science Thesis

Examiners: Prof. Karen Egiazarian
Dr. Alessandro Foi

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
MATTEO MAGGIONI : Video Filtering Using Separable Four-Dimensional Nonlocal Spatiotemporal Transforms
Master of Science Thesis, 132 pages, 16 enclosure pages
October 2010
Major: Signal Processing
Examiners: Prof. Karen Egiazarian, Dr. Alessandro Foi
Keywords: Video filtering, nonlocal methods, adaptive transforms, motion estimation

The large number of practical application involving digital videos has motivated a significant interest in restoration or enhancement solutions to improve the visual quality under the presence of noise. We propose a powerful video denoising algorithm that exploits temporal and spatial redundancy characterizing natural video sequences to reduce the effects of noise. The algorithm implements the paradigm of nonlocal grouping and collaborative filtering, where a four-dimensional transform-domain representation is leveraged to enforce sparsity and thus regularize the data. Moreover we present an extension of our algorithm that can be effectively used as a deblocking and deringing filter to reduce the artifacts introduced by most of the popular video compression techniques.

Our algorithm, termed V-BM4D, at first constructs three-dimensional volumes, by tracking blocks along trajectories defined by the motion vectors, and then groups together mutually similar volumes by stacking them along an additional fourth dimension. Each group is transformed through a decorrelating four-dimensional separable transform, and then it is collaboratively filtered by coefficients shrinkage. The effectiveness of shrinkage is due to the sparse representation of the transformed group. Sparsity is achieved because of different type of correlation among the groups: local correlation along the two dimensions of the blocks, temporal correlation along the motion trajectories, and nonlocal spatial correlation along the fourth dimension. As a conclusive step, the different estimates of the filtered groups are adaptively aggregated and subsequently returned to their original position, to produce a final estimate of the original video.

The proposed filtering procedure leads to excellent results in both objective and subjective visual quality, since in the restored video sequences the effect of the noise or of the compression artifacts is noticeably reduced, while the significant features are preserved. As demonstrated by experimental results, V-BM4D outperforms the state of the art in video denoising.

ACKNOWLEDGEMENTS

I would like to show my gratitude to my advisor and reviewer, Giacomo Boracchi, for his support during the development of this Thesis from the earliest stages. He helped me in the understanding of the subject, and in producing a more coherent and readable text. A deep thank goes to Alessandro Foi, whose invaluable guidance during my staying in Finland made possible the fulfillment of my work, and to Karen Egiazarian for welcoming me in his group in the Department of Signal Processing at Tampere University of Technology.

During the last months spent abroad, I have had the chance to learn a lot about my field of study and about life in general. I am grateful to all the new friends I made in Tampere, Finland gave us the opportunity to live unforgettable experiences, as dipping in a frozen lake or having inhuman saunas in the middle of the woods. I am also truly indebted to my friends and colleagues at Politecnico di Milano, who had shared with me the joys and sorrows of being a computer science student during the last crazy five years spent in campus together.

Finally, I want to thank my family and my closest and deepest friends in Milan. I have always felt their love and support in both the good and the hard times of my life. Beyond a shadow of a doubt, I would not be the person I am now without them.

Tampere, August 2010

Matteo Maggioni

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. IMAGE FORMATION MODELS

1.1 Preliminaries

A digital image, for the purpose of this Thesis, is considered as a two-dimensional discrete positive function y formally defined as:

$$y : X \to \mathbb{R}, \text{ where } X \subset \mathbb{Z}^2. \tag{1.1}$$

The elements belonging to the domain, that is every $x \in X \subset \mathbb{Z}^2$, are two-dimensional spatial coordinates, the pixels, composing the image y, and $y(x)$ represents the intensity value of y at the position indexed by x. In other words every pixel value $y(x)$ is the result of a light intensity measurement made by a CCD matrix. Each captor of the CCD is roughly a square wherein the number of incoming photons is being counted for a fixed period of time, the so-called obturation time. This count is not deterministic because, even in ideal condition, where the light source is constant, the number of photons reaching the sensor fluctuates around its average in accordance with the Central Limit Theorem, and, additionally, each captor, if not adequately cooled, receives heat spurious photons.

As it can be seen in Equation (1.1), the codomain of y is $\mathbb{R}$, thus we will consider greyscale digital images only. Moreover the intensity value will be normalized, without loss of generality, to the common range $[0, 1]$. Greyscale images measure the light intensity, from the minimum brightness of the value 0 to the maximum of 1. However in a digital, discrete, world not every light intensity can be fully represented, thus pixel values need to be quantized to L levels, usually a power of 2. Common greyscale images have $L = 256 = 2^8$ different grey levels, using 8 bits to describe every pixel value. In Figure 1.1 is illustrated an example of a digital image supported by a grid X with 15×9 pixels, note that by convention the origin is on the top-left corner of the chosen reference system (u, v).

Digital images, like any other signal acquired by real word sensors, typically contain noise. Gaussian noise is part of almost any signal. For example, the familiar white noise of a non-tuned television station is well modeled as Gaussian. Since image sensors must count photons and the number of photons is a random quantity, especially in low-light situation images often have photon counting noise. Thus noise is caused by several faults in the image acquisition process, among which we can cite the following:

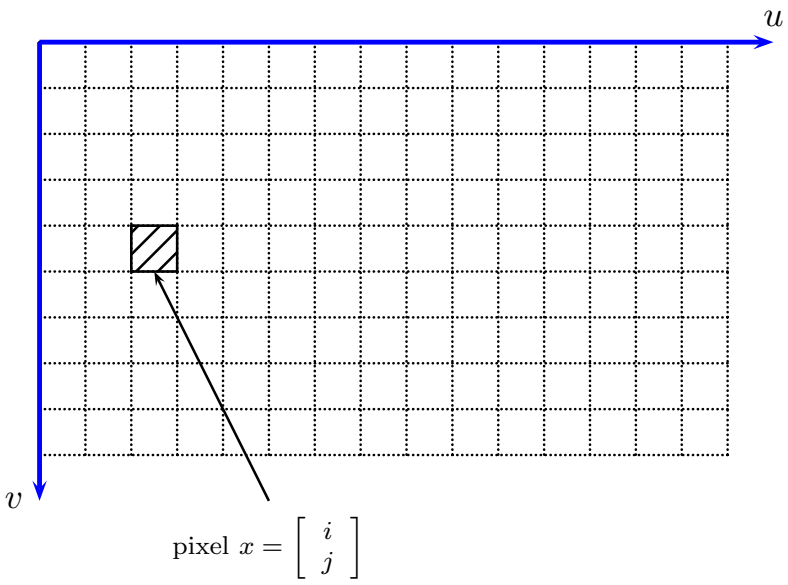


Figure 1.1: Visual representation of a digital image.

- Photon counting errors;
- Electric and thermal noise;
- Quantization noise due to ADCs;
- Data transmission errors.

For these reasons pixel values in the sensed image could be different than the true intensities taken by an ideal camera in ideal conditions. The most general model of noisy image z can be expressed as an additive decomposition of two components, the original uncorrupted signal and the noise:

$$z(x) = y(x) + \eta(x),\ x \in X, \tag{1.2}$$

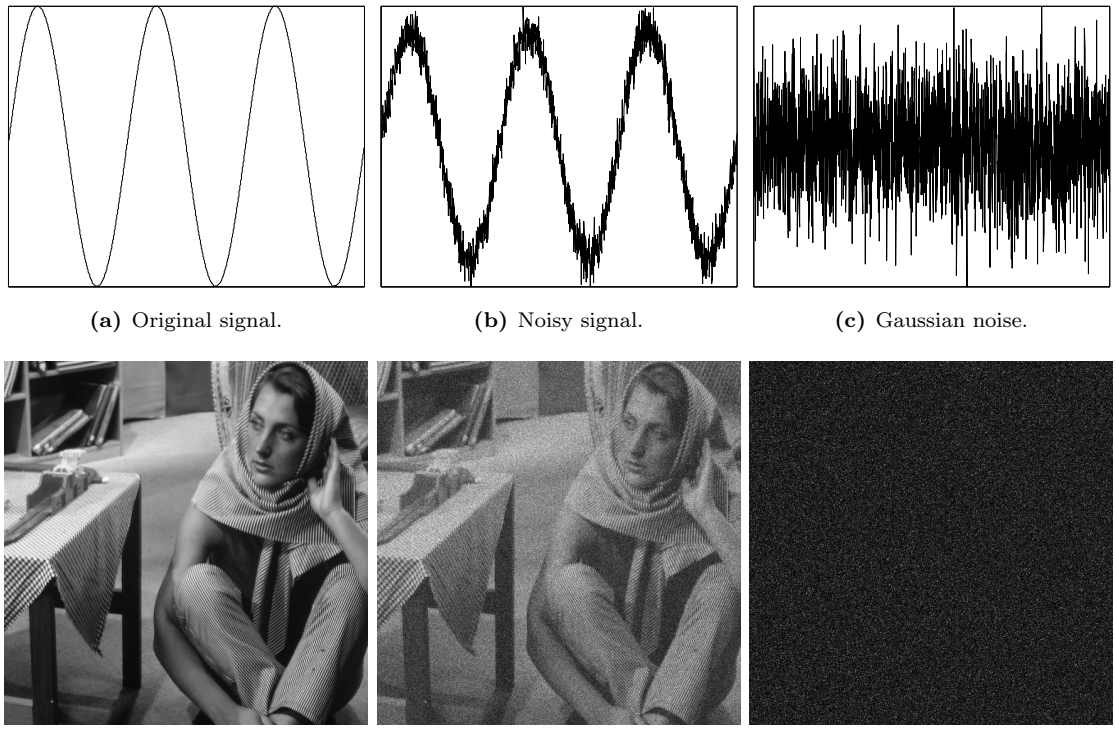
where y is the unknown true image and η is a random variable that describes the noise corrupting the signal at every given pixel $x \in X \subset \mathbb{Z}^2$ of the image. Another common model, called multiplicative for obvious reason, defines the noisy image z as:

$$z(x) = y(x) \cdot \eta(x),\ x \in X. \tag{1.3}$$

Note that the former model can be transformed into the latter by using exponentiations, the opposite direction can be taken as well using logarithms:

$$e^{z(x)} = e^{y(x)+\eta(x)} = e^{y(x)}e^{\eta(x)},\ x \in X$$
$$\log z(x) = \log(y(x) \cdot \eta(x)) = \log y(x) + \log \eta(x),\ x \in X.$$

Here the focus is on the additive model of Equation (1.2). Noise is an unwanted



(a) Original signal. **(b)** Noisy signal. **(c)** Gaussian noise.

(d) Original image. **(e)** Noisy image. **(f)** Gaussian noise (absolute value).

Figure 1.2: Example of Gaussian noise on a one-dimensional and two-dimensional signal. The Gaussian noise uniformly corrupts the original signal, as it is independent from the underlying signal value.

random component in an observed signal, thus its behavior is best described by using random variables typically following a Gaussian or a Poisson distribution, or a combination of both. In the remaining part of this chapter we will analyze every alternative image formation model mentioned above.

1.2 Gaussian Noise

The Gaussian distribution is undoubtedly the most common probabilistic model used to approximate noise in corrupted images. Referring to Equation (1.2), the noise component η is now defined as an independent and identically distributed zero-mean Gaussian random variable with variance σ^2. This is the white noise model:

$$\eta(\cdot) \sim \mathcal{N}(0, \sigma^2). \tag{1.4}$$

Thereby this simple, widely used, noise model is a realization of a zero-mean independent and identically distributed Gaussian random vector added to the original signal y. In Figure 1.2 there are represented two examples of signals degraded by white Gaussian noise with standard deviation $\sigma = 0.1$. The one-dimensional signal is the function $y(x) = \sin(x)$ with $x \in [0, 6\pi]$ while the two-dimensional signal is a 512×512 greyscale image, the well known test image of *Barbara*. Note that the

average intensity value of the signal does not change after the application of the Gaussian noise and, furthermore, the expected value of the noisy signal z is actually the original signal y:

$$\begin{aligned}\mathbb{E}[z(x)] &= \mathbb{E}[y(x) + \eta(x)] \\ &= \mathbb{E}[y(x)] + \mathbb{E}[\eta(x)] \\ &= y(x),\ x \in X,\end{aligned}$$

where the last equality holds because $y(x)$ is a deterministic value, that is $\mathbb{E}[y(x)] = y(x)$ for all $x \in X$, and η is the random Gaussian variable with zero-mean defined in Equation (1.4), thus $\mathbb{E}[\eta(x)] = 0$ for all $x \in X$.

1.3 Poissonian Noise

Earlier works in image and signal processing is broadly dominated by the assumption of dealing with white Gaussian noise, despite the fact that several important physical events may be better described by non-Gaussian degradation. In fact, in a significant number of applications such as medical or astronomical imaging, that is where images are acquired by photon-counting devices, noise can be better modeled by a Poisson distribution.

Formally each observation $z(x)$ of the noisy signal z, with $x \in X \subset \mathbb{Z}^2$, can be defines as an independent random variable drawn from a Poisson distribution of parameter proportional to the original signal $y(x)$ [45]:

$$z(x) \sim \mathcal{P}(\lambda \cdot y(x)),\ x \in X, \tag{1.5}$$

where λ is a positive real number.

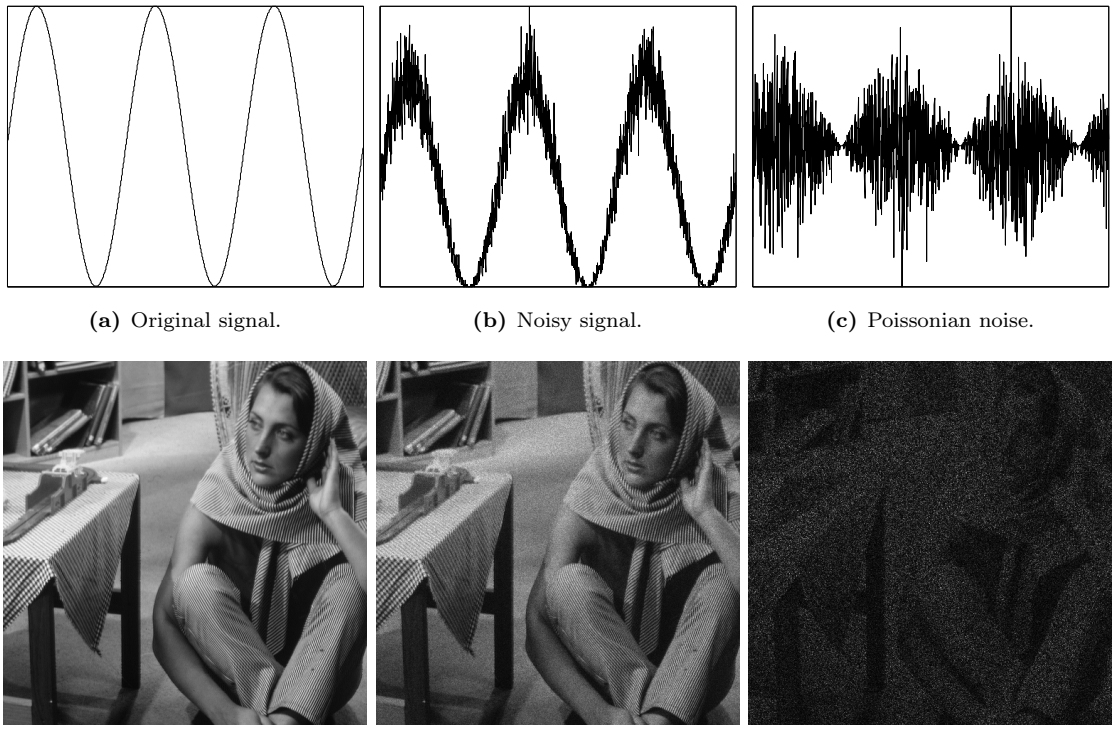
From now on, to simplify the notation, without any advice stating the opposite, we will not explicitly write the conditioning on y of the expectations, standard-deviations and variances formulas, for example by $\mathbb{E}[\cdot]$ we mean $\mathbb{E}[\cdot|y]$.

The main issue with this model is that the noise can be no longer assumed independent from the signal. Since the parameter of a Poisson distribution is both the expected value and the variance of the random variable, the magnitude of Poissonian noise is proportional to the underlying original signal intensity value. As a matter of fact, both the expected value and the variance of z are the underlying intensity value to be estimated.

$$\mathbb{E}[z(x)] = \mathbb{V}\mathrm{ar}[z(x)] = \lambda \cdot y(x),\ x \in X. \tag{1.6}$$

Therefore the noise η can be formally defined as:

$$\eta(x) = z(x) - \mathbb{E}[z(x)].$$



(a) Original signal. **(b)** Noisy signal. **(c)** Poissonian noise.

(d) Original image. **(e)** Noisy image. **(f)** Poissonian noise (absolute value).

Figure 1.3: Example of Poissonian noise on a one-dimensional and two-dimensional signal. This type of noise is not independent from the original signal, in fact, as shown in the left-most figures, its intensity increases as the underlying signal gets higher, i.e. in correspondence with the high-peaks of the sin function or with the face of *Barbara*. Thus, in case of images, brighter areas suffer from heavier noise corruption, while darker areas are less impaired.

Moreover, remembering Equation (1.6) and recalling that $y(x)$ is a deterministic value for each $x \in X$, we can formally define the statistics of η as follows:

$$\begin{aligned}
\mathbb{E}\left[\eta(x)\right] &= \mathbb{E}\left[z(x) - \mathbb{E}\left[z(x)\right]\right] \\
&= \mathbb{E}\left[z(x) - y(x)\right] \\
&= 0 \\
\mathbb{V}\mathrm{ar}\left[\eta(x)\right] &= \mathbb{V}\mathrm{ar}\left[z(x) - \mathbb{E}\left[z(x)\right]\right] \\
&= \mathbb{V}\mathrm{ar}\left[z(x) - y(x)\right] \\
&= y(x).
\end{aligned}$$

This behavior is illustrated in Figure 1.3. Looking at Figures 1.3(c) and 1.3(f), it is clearly shown that the noise level increases with the intensity of the original signal, in strong contrast to the uniformly spread Gaussian noise depicted in Figure 1.2(c). For example the legs of the table next to Barbara present an almost trifling noise level, while other areas, such as the table napkin or the floor in the background, are significantly more affected by noise inasmuch as the underlying signal is brighter in

the original image.

An interesting characteristic of the Poisson distribution is that for sufficiently large[1] values of the parameter λ, the distribution can be approximated with a Normal distribution of both mean and variance equal to λ:

$$\mathcal{P}(\lambda) \approx \mathcal{N}(\lambda, \lambda). \tag{1.7}$$

The accuracy of the previous approximation increases with λ, thus a Poissonian process can be approximated as a special signal-dependent Gaussian distribution defined as [10]:

$$\eta(x) \sim \mathcal{N}(0, y(x)),\ x \in X.$$

1.4 Noise in Camera Raw Data

The last model of this brief review is perhaps the most general one, it can be used to accurately describe the statistical behavior of noise corrupting raw data generated by cameras. Raw data means a raw digital image file produced by a digital camera's sensor that has not yet been processed and, thus, that could have a wider dynamic range than the eventual processed final image. A common raw image processing pipeline comprehends for example denoising, pixel error recovering, white balancing and compression. Intuitively it is composed by two terms: a signal-dependent Poissonian part and a signal-independent Gaussian part respectively used to model the photon counting process and the remaining stationary disturbances, such as those described in Section 1.1.

First of all the noise model η of Equation (1.2) must be extended to define a generic signal-dependent model[10, 25]:

$$z(x) = y(x) + \sigma(y(x))\xi(x),\ x \in X, \tag{1.8}$$

where, as usual, $z : X \to \mathbb{R}$ is the observed noisy signal, $y : X \to \mathbb{R}$ is the original signal and x is a two-dimensional spatial coordinate, i.e. a pixel position in the image.

The innovative part is the definition of the noise composed by two components: the term $\xi(x) : X \to \mathbb{R}$ is zero-mean independent random noise with unitary variance assumed for simplicity as $\xi \sim \mathcal{N}(0, 1)$, and the function $\sigma : \mathbb{R} \to \mathbb{R}^+$ that gives the standard deviation of the whole noise component. Since $\mathbb{E}[\xi(x)] = 0$, then the original signal y can be expressed as the expected value of the noisy observation z: $\mathbb{E}[z(x)] = y(x)$. Furthermore the standard deviation of the noise is a function, called σ, of the expected value of the noisy signal: $\text{std}[z(x)] = \sigma(\mathbb{E}[z(x)])$.

The noise model in Equation (1.8) can be expressed with two mutually indepen-

[1] The actual value of λ depends on the type of application and on the desired accuracy. Empirical observations prove that good results can be obtained with $\lambda > 20$.

dent component:

$$\sigma(y(x))\xi(x) = \eta_p(y(x)) + \eta_g(x),\ x \in X, \tag{1.9}$$

where η_p is the signal-dependent Poissonian part and η_g is the signal-independent Gaussian part, characterized as follows:

$$\chi(y(x) + \eta_p(y(x))) \sim \mathcal{P}(\chi y(x)),\ \chi > 0,\ x \in X,$$
$$\eta_g(x) \sim \mathcal{N}(0, b),\ b \geq 0,\ x \in X.$$

From the definition of the Poisson distribution, the expected value and variance of the random variable can be derived as follow:

$$\mathbb{E}\left[\chi(y(x) + \eta_p(y(x)))\right] = \mathbb{V}\mathrm{ar}\left[\chi(y(x) + \eta_p(y(x)))\right] = \chi y(x). \tag{1.10}$$

Eventually, from Equation (1.10), it is possible to derive the statistics of the Poissonian component η_p of the noise modeled in (1.9). The expected value is defined as:

$$\begin{aligned}\mathbb{E}\left[\chi(y(x) + \eta_p(y(x)))\right] &= \mathbb{E}\left[\chi y(x)\right] + \mathbb{E}\left[\chi\eta_p(y(x))\right] \\ &= \chi y(x) + \chi\mathbb{E}\left[\eta_p(y(x))\right] = \chi y(x),\end{aligned}$$

and, from the last equality, we can state that:

$$\mathbb{E}\left[\eta_p(y(x))\right] = 0 \tag{1.11}$$

while the variance is:

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\chi(y(x) + \eta_p(y(x)))\right] &= \mathbb{V}\mathrm{ar}\left[\chi y(x) + \chi\eta_p(y(x))\right] \\ &= \chi^2\mathbb{V}\mathrm{ar}\left[\eta_p(y(x))\right] = \chi y(x),\end{aligned}$$

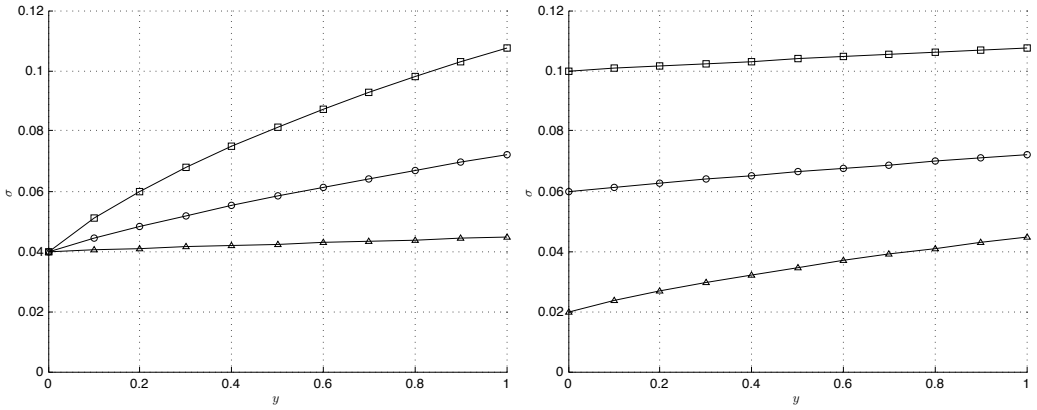
rewriting the last line we obtain:

$$\mathbb{V}\mathrm{ar}\left[\eta_p(y(x))\right] = \frac{y(x)}{\chi} = ay(x), \tag{1.12}$$

where a is generated by the substitution $a = \chi^{-1}$.

As expected, the variance of η_p is proportional to the value of the signal $y(x)$, while the Gaussian component η_g has a constant variance equal to b. Thus, the whole variance of the noisy signal z in Equation (1.8) can be defined as:

$$\mathbb{V}\mathrm{ar}\left[z(x)\right] = \sigma^2(y(x)) = ay(x) + b. \tag{1.13}$$



(a) Parameters: $a = 0.02^2$ (△), 0.06^2 (○), 0.1^2 (□) and $b = 0.04^2$.

(b) Parameters: $a = 0.04^2$ and $b = 0.02^2$ (△), 0.06^2 (○), 0.1^2 (□).

Figure 1.4: Example of the signal-dependent standard deviation function $\sigma(y) = \sqrt{ay + b}$.

Consequently the standard deviation is:

$$\text{std}\left[z(x)\right] = \sigma(y(x)) = \sqrt{ay(x) + b}.$$

The parameters a and b depend on the hardware characteristic of sensors and on the acquisition settings, such as quantum efficiency, pedestal parameter, analog gain, ISO value[2], temperature, etc [10]. As well as we did in Equation (1.7), it is possible to approximate the Poisson distribution with a signal-dependent Normal distribution as follows:

$$\begin{aligned}\sigma(y(x))\xi(x) = \sqrt{ay(x) + b}\xi(x) &\simeq \eta_h(y(x)) \\ \eta_h(x) &\sim \mathcal{N}(0, ay(x) + b).\end{aligned}$$

1.4.1 Clipping

In real world the dynamic range of acquisition, transmission and storage system is always limited. As said in Section 1.1, in this work the range is normalized to the interval $[0, 1]$, where the upper and lower-bound correspond to the darkest and the brightest intensity value in the signal respectively. Given that the original, noise-free, signal y is actually limited in that range, the same assumptions no longer holds for z, because the noise η added to y can produce an under or over-exposure in the original signal. For these reasons some values in the noisy signal z may exceed the bounds, thus, to deal with this problem, we must define a new observation model, called the clipped observation model [10, 25]:

$$\tilde{z}(x) = \max(0, \min(z(x), 1)),\ x \in X. \tag{1.14}$$

[2]Large ISO numbers produce worse signal-to-noise ratio. On the other hand, small ISO values assure better SNR but, as a drawback, yield to darker images if the exposure time is not long enough.

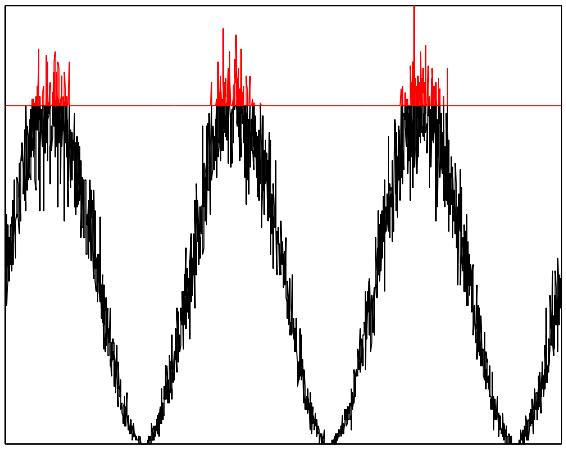


Figure 1.5: Clipped signal corrupted by Poissonian noise. The part of the signal above the red line is dropped, and the remaining blue part is the clipped signal.

In Equation (1.14) the noisy signal z is clipped, or, in other words, every value of z above or below the bounds shall be substituted with the value of the exceeded bound and a new noisy signal is produced: the clipped noisy signal called $\tilde{z}$. For an example refer to Figure 1.5, where it is represented a sine function corrupted with Poissonian noise wherein the blue noisy wave is the clipped signal $\tilde{z}$ and the red term, those above the value 1, is the part of the original noisy observation z exceeding the upper-bound.

The noise model for the clipped observation model of Equation (1.14) is then:

$$\tilde{z}(x) = \tilde{y}(x) + \tilde{\sigma}(\tilde{y}(x))\tilde{\xi}(x), \tag{1.15}$$

where $\tilde{y}(x) = \mathbb{E}\left[\tilde{z}(x)\right] \in \tilde{Y} \subseteq [0,1]$ and $\mathbb{E}\left[\tilde{\xi}(x)\right] = 0$, $\mathbb{V}\mathrm{ar}\left[\tilde{\xi}(x)\right] = 1$. The function $\tilde{\sigma} : \tilde{Y} \to \mathbb{R}^+$ gives the standard deviation of the clipped observation: $\tilde{\sigma}(\tilde{y}(x)) = \mathrm{std}\left[\tilde{z}(x)\right]$. In general the original signal y is different from the underlying signal $\tilde{y}$ of the clipped observation, and consequently the statistics of the corresponding noisy signals are different as well:

$$\begin{aligned} \tilde{y}(x) = \mathbb{E}\left[\tilde{z}(x)\right] &\neq \mathbb{E}\left[z(x)\right] = y(x) \\ \tilde{\sigma}(\tilde{y}(x)) = \mathrm{std}\left[\tilde{z}(x)\right] &\neq \mathrm{std}\left[z(x)\right] = \sigma(y(x)). \end{aligned}$$

Note that it is possible to define a transformation from the functions y and σ given $\tilde{y}$ and $\tilde{\sigma}$ and vice versa [25, 10, 24]. Assuming, for simplicity, the independence of the two clippings, $\tilde{z} = 0\ \forall z < 0$ and $\tilde{z} = 1 \forall z > 1$, which is actually false only in cases of extremely powerful noise, let's introduce a random variable $\nu \sim \mu, \infty$ and its corresponding clipped -from below- $\tilde{\nu} = \max \nu, 0$. The probability density

function $f_{\tilde{\nu}}$ of $\tilde{\nu}$ is defined as:

$$f_{\tilde{\nu}} = \begin{cases} \phi(t-\mu) + \Phi(-\mu)\delta_0(t) & t \geq 0 \\ 0 & t < 0 \end{cases},$$

where ϕ is the probability density function, Φ the cumulative density function of the standard normal distribution $\mathcal{N}(0,1)$ and δ_0 is the Dirac delta impulse. Thus it can be proved that the statistics of the clipped $\tilde{\nu}$ are:

$$\mathbb{E}[\tilde{\nu}] = \mathcal{E}_m(\mu) = \Phi(\mu) + \phi(\mu) \tag{1.16}$$

$$\begin{aligned} \mathbb{V}\mathrm{ar}[\tilde{\nu}] &= \mathcal{S}_m^2(\mu) = \Phi(\mu) + \mathcal{E}_m(\mu)\mu - \mathcal{E}_m^2(\mu) \\ &= \Phi(\mu) + \phi(\mu)\mu - \phi^2(\mu) + \Phi(\mu)\mu(\mu - \Phi(\mu)\mu - 2\phi(\mu)). \end{aligned} \tag{1.17}$$

Using these functions, recalling that $y = \mathbb{E}[z]$ and $\sigma(y) = \mathrm{std}[z]$, the corresponding expectation $\tilde{y} = \mathbb{E}[\tilde{z}]$ and $\sigma(\tilde{y}) = \mathrm{std}[\tilde{z}]$ of the model (1.15) are obtained from the direct transformations defined below:

$$\tilde{y} = \mathcal{A}(y, \sigma(y)) = \sigma(y)\mathcal{E}_m\left(\frac{y}{\sigma(y)}\right) - y + 1 - \sigma(y)\mathcal{E}_m\left(\frac{1-y}{\sigma(y)}\right) \tag{1.18}$$

$$\tilde{\sigma}(\tilde{y}) = \mathcal{B}(y, \sigma(y)) = \sigma(y)\mathcal{S}_m\left(\frac{y}{\sigma(y)}\right)\mathcal{S}_m\left(\frac{1-y}{\sigma(y)}\right). \tag{1.19}$$

On the other hand, the non-clipped signal y can be computed from $\tilde{y}$ and $\tilde{\sigma}$, through the following indirect transformation:

$$y == \mathcal{C}(\tilde{y}, \tilde{\sigma}(\tilde{y})) = \tilde{y}\mathcal{E}_r\left(\frac{\tilde{y}}{\tilde{\sigma}(\tilde{y})}\right) - \tilde{y} + 1 - (1-\tilde{y})\mathcal{E}_r\left(\frac{1-\tilde{y}}{\tilde{\sigma}(\tilde{y})}\right), \tag{1.20}$$

where $\mathcal{E}_r$ is defined as a function of $\rho = \frac{\mathcal{E}_m(\mu)}{\mathcal{S}_m(\mu)} = \frac{\mathbb{E}[\tilde{\nu}]}{\mathrm{std}[\tilde{\nu}]}$ as $\mathcal{E}_r = \frac{\mu}{\mathcal{E}_m(\mu)}$. In Figure 1.6 it is shown the standard deviation function $\sigma(y)$ having $a = 0.01$ and $b = 0.04$ in solid line, and in dashed line the corresponding standard deviation function $\tilde{\sigma}(\tilde{y})$. The solid grey segments express the mapping between the points in $\sigma(y)$ and their counterparts in $\tilde{\sigma}(\tilde{y})$, in other words the segments are the mapping $\sigma(y) \mapsto \tilde{\sigma}(\tilde{y})$. Note that the small black triangles indicates the points $(\tilde{y}, \tilde{\sigma}(\tilde{y}))$ that corresponds to the point in $\sigma(y)$ where $y = \{0, 1\}$.

Note that the clipped signal $\tilde{z}$, with regard to the true signal y, is corrupted by an error having a non-zero expected value. To clarify this statement, let's rewrite the model in Equation (1.15) by adding and subtracting y:

$$\tilde{z}(x) = y(x) + \left[\tilde{y}(x) - y(x) + \tilde{\sigma}(\tilde{y}(x))\tilde{\xi}(x)\right], x \in X,$$

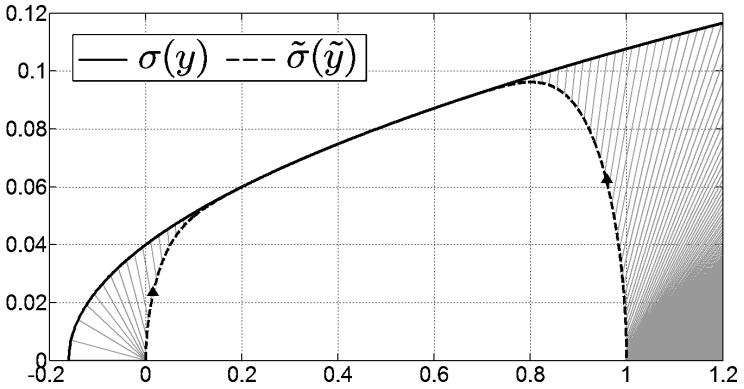


Figure 1.6: Correspondence between the standard deviation functions of the original signal $\sigma(y)$ and the standard deviation function of the clipped signal $\tilde{\sigma}(\tilde{y})$.

where the noise has become the term inside the square brackets:

$$\tilde{\eta}(x) = \left[\tilde{y}(x) - y(x) + \tilde{\sigma}(\tilde{y}(x))\tilde{\xi}(x)\right].$$

As a last comment, observe that, even if the variance of ξ is equal in the clipped and non-clipped observation: $\mathbb{V}\mathrm{ar}\left[\xi(x)\right] = \mathbb{V}\mathrm{ar}\left[\tilde{\xi}(x)\right] = 1$, their distribution are different. More specifically, assuming for simplicity $\xi(x) \sim \mathcal{N}(0,1)$, the random variable $\tilde{\xi}$ follows a doubly censored (clipped) normal distribution supported on $\left[\frac{-\tilde{y}}{\tilde{\sigma}(\tilde{y})}, \frac{1-\tilde{y}}{\tilde{\sigma}(\tilde{y})}\right]$.

1.5 Qualitative Measures

It is useful in this section to analyze qualitative measures of noise in degraded signals. More specifically this measures used in this work will be the Mean Square Error (MSE), Signal-to-Noise Ratio (SNR) and Peak-Signal-to-Noise Ratio (PSNR).

Generally speaking these measures are used to compare a non-processed signal versus the corresponding processed one, i.e. an uncompressed versus a compressed audio track or an original image versus the denoised one. Usually the results are given in a logarithmic decibel (dB) scale, because signals may have a very wide dynamic range. It is important to note that the great majority of signal processing applications do not require an exact reconstruction of the original signal, which is seldom possible, thus a reasonable amount of loss is allowed in the reconstructed data.

Before moving on, it is useful to anticipate the concept of denoising which will be extensively covered in Chapter 3. By denoising we mean reconstruct an estimate $\hat{y}$ of the original signal y from the noisy observation z using an algorithm generally denoted with D:

$$\hat{y} = \mathsf{D}(z). \tag{1.21}$$

The goal is of course to remove the noise component η, while keeping the estimate

$\hat{y}$ as close as possible to the original y.

1.5.1 Mean Square Error

The Mean Square Error, MSE, is the most common technique to measure the difference between two similar signals. It corresponds to the expected value of the square error loss, where the error is the magnitude of the dissimilarity between the original signal and the estimated one. For our purpose, when y is an image, a two-dimensional discrete function defined for all $x \in X \subset \mathbb{Z}^2$ and $\hat{y}$ is its estimate, then the MSE becomes:

$$MSE = \frac{1}{|X|} \sum_{x \in X} \left(y(x) - \hat{y}(x) \right)^2 . \tag{1.22}$$

Observe that, in general, we don't have the original signal y, unless we are in test environments where the noisy signal z is produced adding synthetic noise to a chosen noise-free signal y.

1.5.2 Signal-to-Noise Ratio

The Signal-to-Noise Ratio, SNR, can be viewed as the ratio of a signal power to the noise power corrupting the signal itself, higher outcomes express better quality of the signal. In general this measure is defined as the power ratio between a signal y and the background noise η:

$$SNR = 10 \log_{10} \frac{P_y}{P_\eta}$$

In case of images, the SNR measures the amount of noise with its standard deviation $\mathrm{std}\,[\eta]$ and with the empirical standard deviation of the original signal $\mathrm{std}\,[y]$:

$$SNR = 10 \log_{10} \frac{\mathrm{std}\,[y]}{\mathrm{std}\,[\eta]} \tag{1.23}$$

where $\mathrm{std}\,[y]$ is defined as:

$$\mathrm{std}\,[\eta] = \left(\frac{1}{X} \sum_{x \in X} (y(x) - \bar{y})^2 \right)^{\frac{1}{2}} , \tag{1.24}$$

and $\bar{y}$ is the average grey level value:

$$\bar{y} = \frac{1}{X} \sum_{x \in X} y(x). \tag{1.25}$$

On the other hand the standard deviation of the noise can be either computed as an empirical measurement or formally obtained when the noise model and parameters are known.

1.5.3 Peak-Signal-to-Noise Ratio

The Peak-Signal-to-Noise Ratio, PSNR, is the reference measure used in this Thesis. It expresses the ratio between the maximum possible power of a signal versus the power of the corrupting noise. As usual The PSNR can be easily defined from the MSE, defined in Equation (1.22), as follows:

$$PSNR = 10 \log_{10} \frac{M^2}{MSE}, \tag{1.26}$$

where M is the maximum possible value of the signal, i.e. for normalized images $M = 1$.

It is important to note that high SNR or PSNR do not always correspond to a signal with perceptually high quality, the measures are thus somewhat subjective.

2. TRANSFORM BASED IMAGE REPRESENTATION

2.1 Mathematical background

This section introduces the necessary mathematical background to better comprehend the topics that will be discussed in the continuation of this Thesis. Sometimes will be also provided basic examples using vectors belonging to $\mathbb{R}^2$ to support the formal discussion of the concept hereby defined. A more detailed view of the arguments can be found in [19, 46].

2.1.1 Hilbert Spaces

A vector space is, in general, a set whose elements, the vectors, can be combined and manipulated in certain ways. Throughout the section we merge our definitions for real and complex scalars using the unified notation $\alpha \in \mathbb{F}$ to express a number belonging to $\mathbb{R}$ or $\mathbb{C}$.

Definition 2.1.1 (Vector space)**.** *A vector space $\mathcal{V}$ over the set $\mathbb{F}$ is a set of vectors together with vector addition and scalar multiplication operations. These operations must satisfy the usual following properties for any $u, v, w \in \mathcal{V}$ and $\alpha, \beta \in \mathbb{F}$:*

i. Commutativity: $u + v = v + u$;

ii. Associativity: $(u + v) + w = u + (v + w)$ and $(\alpha\beta)u = \alpha(\beta u)$;

iii. Distributivity: $\alpha(u + v) = \alpha u + \alpha v$ and $(\alpha + \beta)u = \alpha u + \beta u$.

Furthermore for every $u \in \mathcal{V}$ there exists:

iv. Additive identity: An element $\mathbf{0}$, such that $u + \mathbf{0} = u$;

iiv. Additive inverse: A unique element $-u \in \mathcal{V}$ such that $u + (-u) = \mathbf{0}$;

iiiv. Multiplicative identity: An element $\mathbf{1}$ such that $\mathbf{1} \cdot u = u$.

Note that $\mathbf{0}$ is the zero-vector, different in general from the zero scalar.

In finite Euclidean space, like $\mathbb{F}^k$, the elements of $\mathcal{V}$ are k-tuples and the operations of addition and multiplication are defined respectively as:

$$u + v = \begin{bmatrix} u_1 \\ \vdots \\ u_k \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} u_1 + v_1 \\ \vdots \\ u_k + v_k \end{bmatrix},$$

and:

$$\alpha u = \alpha \begin{bmatrix} u_1 \\ \vdots \\ u_k \end{bmatrix} = \begin{bmatrix} \alpha u_1 \\ \vdots \\ \alpha u_k \end{bmatrix}.$$

Definition 2.1.2 (Vector subspace)**.** *A subset $\mathcal{W}$ of a vector space $\mathcal{V}$ is a subspace if:*

i. For all $u, v \in \mathcal{W}$, $u + v \in \mathcal{W}$;

ii. For all $u \in \mathcal{W}$ and $\alpha \in \mathbb{F}$, $\alpha u \in \mathcal{W}$.

To clarify the concept of subspace, we can think of a line $r \subset \mathbb{R}^2$ passing through the origin of the cartesian plane. The line r can be completely described by one of its non-zero point, thus it is one of the simplest case of subspace. If we apply an orthogonal projection of a vector u onto some subspace, we get the closest vector $\hat{u}$ to u in the subspace, moreover the vector $(u - \hat{u})$ is orthogonal to every vector v belonging to the subspace of $\hat{u}$. When the norm $||v||$ of the element v specifying the subspace is unitary, the projection is defined as $\hat{u} = \langle u, v \rangle v$. For example the projection of a vector u onto one of the standard axis e of the plane $\mathbb{R}^2$ gives the corresponding component of u.

Definition 2.1.3 (Inner product)**.** *An inner product on a vector space $\mathcal{V}$ is a function $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to \mathbb{F}$ with the following properties:*

i. $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$;

ii. $\langle \alpha u, v \rangle = \alpha \langle u, v \rangle$;

iii. $\langle u, v \rangle^* = \langle v, u \rangle$;

iv. $\langle u, u \rangle \geq 0$ *and* $\langle u, u \rangle = 0$ *if and only if* $u = 0$.

Note that the second and the third condition imply $\langle u, \alpha v \rangle = \alpha^* \langle u, v \rangle$, where α^* is the complex conjugate of α, thus the inner product is linear in the first argument and conjugate-linear in the second argument. The inner product, also named scalar or dot product, of two vectors $u = [\ u_1 \quad u_2\ ]^T$ and $v = [\ v_1 \quad v_2\ ]^T$ in the real plane is defined as:

$$\langle u, v \rangle = u_1 v_1 + u_2 v_2 \text{ with } u, v \in \mathbb{R}^2. \tag{2.1}$$

The standard inner product for $\mathbb{F}^k$ as a generalization of the one provided in Equation (2.1) is:

$$\langle u, v \rangle = \sum_{i=i}^{k} u_i v_i^*. \tag{2.2}$$

Note that the inner product defined in Equation (2.2) is not the only valid inner product for $\mathbb{F}^k$. Another example, relating to the vector space of complex-valued

functions in $\mathbb{F}$, could be:

$$\langle u, v \rangle = \int_{-\infty}^{\infty} u(t)v^*(t)dt.$$

As an interesting note, in $\mathbb{R}^k$, there is an equivalent expression of the standard inner product defined in Equation (2.2):

$$\langle u, v \rangle = ||u||^2 \, ||v||^2 \cos\theta.$$

This formulation is also used to define the angle θ between two vectors u and v in k dimensions. Thus the inner product can bu used as a measure of similarity between a pair of vectors. Fixing the length of two vectors, the greater their inner product, the closer the vectors are in orientation. For example when the distance is minimized when the angle $\cos\theta$ is equal to 0 or 1, conversely the vectors are the farthest when they are antiparallel, that is when $\theta = \pi$. The inner product can be then seen as the measure of vector's length along with relative orientation.

Definition 2.1.4 (Inner product space)**.** *A vector space equipped with the inner product is called an inner product space.*

Definition 2.1.5 (Norm)**.** *A norm defined on a vector space $\mathcal{V}$ in $\mathbb{F}$ is a real-valued function $||\cdot|| : \mathcal{V} \to \mathbb{R}$ such that the following holds for any $u, v \in \mathcal{V}$ and $\alpha \in \mathbb{F}$:*

i. $||u|| \geq 0$ *and* $||u|| = 0$ *if and only if* $u = \mathbf{0}$*;*

ii. $||\alpha u|| = |\alpha| \, ||u||$*;*

iii. $||u + v|| \leq ||u|| + ||v||$ *(triangular inequality) with equality if and only if* $v = \alpha u$*.*

Intuitively the norm is a function that assigns a real value to each vector $u \in \mathcal{V}$ and such value is commonly referred to as length of u. Observe that an inner product can be used to define a norm, and such norms are called the norms induced by the inner product. For example, to clarify this concept, the norm induced by the standard inner product of Equation (2.2) is called the Euclidean norm, yielding the conventional notion of length for vectors in $\mathbb{F}^k$:

$$||u||_2 = \sqrt{\langle u, u \rangle} = \left(\sum_{i=1}^{k} |u_i|^2 \right)^{\frac{1}{2}}.$$

Theorem 2.1.1. *In any inner product space the following inequality holds:*

$$| \langle u, v \rangle | \leq ||u|| \, ||v|| \,, \tag{2.3}$$

and this is called the Cauchy-Schwarz inequality.

Definition 2.1.6 (Normed vector space)**.** *A vector space on which a norm is defined is called a normed vector space.*

For $\mathbb{F}^k$, any $p \in [0, \infty)$ has an associated p-norm defined as:

$$||u||_p = \sqrt[p]{\langle u, u \rangle} = \left(\sum_{i=1}^{k} |u_i|^p \right)^{\frac{1}{p}}. \tag{2.4}$$

For $p = 1$, this norm is called the taxicab or Manhattan norm, and for $p = 2$, we get our usual Euclidean square norm (induced by the standard inner product). Note that when $p = \infty$ the norm is defined as the maximum element of u:

$$||u||_\infty = \max\{|u_1|, \ldots, |u_k|\}.$$

As we already mentioned in an inner product space $\mathcal{V}$ the function $||u|| = \langle u, u \rangle^{\frac{1}{2}}$ can be used to define a norm for $\mathcal{V}$. Intuitively the length of a vector given by the norm function can be thought of as the vector's distance from the origin, thus, generalizing this concept, the norm $||\cdot||$ can be also used as a distance function, also called metric, between elements in $\mathcal{V}$.

Definition 2.1.7 (Vector distance)**.** *In a normed vector space, the distance between vectors u and v is the norm of their difference:*

$$d(u, v) = ||u - v||. \tag{2.5}$$

At this point could be useful to make some examples of the most common normed vector spaces. The simplest we can think about is the finite dimensional spaces $\mathbb{F}^k$. As mentioned before, the elements of the space are k-tuples and the operations are component wise. We recall that the standard inner product and the induced norm are:

$$\langle u, v \rangle = \sum_{i=1}^{k} u_i v_i^*$$

$$||u|| = \sqrt{\langle u, u \rangle} = \left(\sum_{i=1}^{k} |u_i|^2 \right)^{\frac{1}{2}}.$$

Definition 2.1.8 (Sequence space)**.** *A sequence space is a vector space whose elements are infinite sequences of real or complex numbers.*

Another interesting space are the sequence spaces $\ell^p(\mathbb{Z})$, i.e. the vector spaces of all complex-valued sequences $(u_i)_{i \in \mathbb{Z}}$ having finite ℓ^p-norm, where the ℓ^p-norm for $p \in [1, \infty)$ is defined as:

$$||u||_p = \left(\sum_{i \in \mathbb{Z}} |u_i|^p \right)^{\frac{1}{p}}, \tag{2.6}$$

and the ℓ^∞-norm is:

$$||u||_\infty = \sup_{i\in\mathbb{Z}} |u_i|.$$

Moreover these spaces satisfy the following inclusion property:

$$p < q \text{ implies } \ell^p(\mathbb{Z}) \subset \ell^q(\mathbb{Z}), \tag{2.7}$$

thus the larger the value of p, the larger is the set of vectors with a given norm. In other words if a sequence has a finite ℓ^1-norm then it also has a finite ℓ^2-norm, while the opposite is not necessarily true. The space $\ell^2(\mathbb{Z})$ is often referred to as the space of square-summable, or finite-energy, sequences. In that case the inner product and the relative induced norm are:

$$\langle u, v\rangle = \sum_{i\in\mathbb{Z}} u_i v_i^*$$
$$||u|| = \sqrt{\langle u, u\rangle}.$$

Definition 2.1.9 (Sequence convergence)**.** *Let $\{u_i\}_{i=1}^\infty$ be a sequence in a vector space $\mathcal{V}$, then $\{u_i\}$ it is said to converge to $u \in \mathcal{V}$, and u is called the limit of the sequence, if given any $\epsilon > 0$ there exist N such that $||u - u_n|| < \epsilon$ $\forall n \geq N$, or equivalently:*

$$\lim_{i\to\infty} ||u_i - u|| = 0.$$

Definition 2.1.10 (Cauchy sequence)**.** *The sequence $\{u_i\}_{i=1}^\infty$ is a Cauchy sequence if, for every $\epsilon > 0$, there exist $N \in \mathbb{N}$ such that for all $n, m \geq N$ we have:*

$$||u_n - u_m|| < \epsilon.$$

From the previous definitions we can say that, in convergent sequences the elements eventually stay arbitrarily close to the limit of the sequence u and, in case of Cauchy sequences the elements stay eventually close to each other. Even if it can seem counter intuitive, not all Cauchy sequences must converge, this is always true for real-valued sequences, but it is not for all vector spaces. Complete vector spaces are those in which sequences that intuitively ought to converge (the Cauchy sequences) have a limit in the space. We now define the relevant terms. Note that a metric is always needed to define convergence. Furthermore we limit our attention to metrics induced by norms.

Definition 2.1.11 (Completeness, Banach space)**.** *A normed space $\mathcal{V}$ is called complete if every Cauchy sequence converges to an element u in $\mathcal{V}$. A complete normed space is also called a Banach space.*

Definition 2.1.12 (Hilbert space)**.** *An inner product space, complete in the norm arising from the inner product, i.e. a complete inner product space, is called a Hilbert*

space.

2.1.2 Linear Operators

As a last concept of this Section, let us introduce the notion of linear operator. A natural definition of linear operators could be the generalization of finite-dimensional matrix multiplications.

Definition 2.1.13 (Linear operator)**.** *A function $A : \mathcal{H}_1 \to \mathcal{H}_2$ is called a linear operator if for all $u, v \in \mathcal{H}_1$ and $\alpha \in \mathbb{F}$:*

i. $A(u + v) = Au + Av$;

ii. $A(\alpha u) = \alpha(Au)$.

Conventionally we write Au instead of $A(u)$, just as like as matrix multiplications. In fact linear operators from $\mathbb{F}^N$ to $\mathbb{F}^M$ and matrices in $\mathbb{F}^{N+M}$ are exactly the same thing. Also many other concepts can be transposed from finite-dimensional linear algebra:

i. Kernel: $\ker(A) = \{u \in \mathcal{H}_1 : Au = \mathbf{0}\}$;

ii. Range: $\text{range}(A) = \{Au \in \mathcal{H}_2 : u \in \mathcal{H}_1\}$.

A linear operator is called bounded when its norm is finite. The operator norm is then defined as:

$$||A|| < \sup_{||u||=1} ||Au|| .$$

A bounded linear operator $A : \mathcal{H}_1 \to \mathcal{H}_2$ is called invertible if there exist a bounded linear operator $A^{-1} : \mathcal{H}_2 \to \mathcal{H}_1$, called the inverse of A, such that:

$$A^{-1}Au = u \text{ for every } u \in \mathcal{H}_1$$
$$AA^{-1}v = v \text{ for every } v \in \mathcal{H}_2.$$

The linear operator $A^* : \mathcal{H}_2 \to \mathcal{H}_1$ is called the adjoint of $A : \mathcal{H}_1 \to \mathcal{H}_2$ when:

$$\langle Au, v\rangle_{\mathcal{H}_2} = \langle u, A^*v\rangle_{\mathcal{H}_1} \text{ for every } u \in \mathcal{H}_1, v \in \mathcal{H}_2.$$

If $A = A^*$, the operator A is called a self-adjoint or Hermitian operator. Note that every bounded linear operator has a unique adjoint and the operators AA^* and A^*A are always self-adjoint, moreover $||A|| = ||A^*||$. Finally a bounded linear operator $A : \mathcal{H} \to \mathcal{H}$ is called unitary if it satisfies:

$$AA^* = A^*A = I$$

An interesting property of the unitary operator is that it preserves the inner product and, thus, the induced norm, for example the Euclidean norm:

$$\langle Au, Av\rangle = A\langle u, Av\rangle = \langle u, A^*Av\rangle = \langle u, v\rangle$$
$$||Au||^2 = \langle Au, Au\rangle = \langle u, u\rangle = ||u||^2 .$$

2.1.3 Bases

In any vector space $\mathcal{V}$ it is not unusual to combine certain elements belonging to $\mathcal{V}$ to produce others elements.

Definition 2.1.14 (Linear combination)**.** *Let $u_1, \ldots, u_k$ be vectors in a given vector space $\mathcal{V}$ and let $\alpha_1, \ldots, \alpha_k$ be scalars in $\mathbb{F}$. A linear combination of the vectors is a sum of the form:*

$$u = \alpha_1 u_1 + \cdots + \alpha_k u_k.$$

Definition 2.1.15 (Linear independence)**.** *A set of vectors $\{u_i\}_{i=1}^k$ in a vector space $\mathcal{V}$ is called linearly independent if there exist scalars $\alpha_1, \alpha_2, \ldots, \alpha_k$, which are not all zero, such that $\sum_{i=1}^k = \alpha_i u_i = 0$. Otherwise the set $\{u_i\}_{i=1}^k$ is called linearly independent if it is not linearly dependent, in such case the scalars for which $\sum_{i=1}^k = \alpha_i u_i = 0$ must be all zeros $\alpha_1 = \alpha_2 = \ldots = \alpha_k = 0$.*

A nonempty infinite set $\mathcal{S}$ of vectors in a vector space $\mathcal{V}$ is called linearly independent if at least one of the vectors in the set $\mathcal{S}$ can be written as a linear combination of the other vectors in the set. If this condition does not holds for the set $\mathcal{S}$, then $\mathcal{S}$ is said to be linearly independent. Note that if a set $\mathcal{S}$ has a linearly dependent subset, then the set $\mathcal{S}$ itself is linearly dependent as well. Moreover if $\mathcal{S}$ is linearly independent, then every subset of $\mathcal{S}$ is also linearly independent.

Definition 2.1.16 (Linear span)**.** *Let $\mathcal{S}$ be a subset of a vector space $\mathcal{V}$. The linear span of $\mathcal{S} \subset \mathcal{V}$ is the set of all the finite linear combination of the elements in $\mathcal{S}$:*

$$span(\mathcal{S}) = \left\{ \sum_{i=1}^{k} \alpha_i u_i : \alpha_i \in \mathbb{F}, u_i \in \mathcal{S}, k \in \mathbb{N} \right\}. \tag{2.8}$$

The coefficients α_i are called the expansion coefficients of u with regard to the basis $\mathcal{S}$. Note that a span is always a subspace and that the sum has a finite number of terms even if the set $\mathcal{S}$ is infinite. Additionally we call spanning set for $\mathcal{V}$ a set $\mathcal{S}$ whose linear span is $\mathcal{V}$.

Observe that, given a spanning set $\mathcal{S} = \{u_i\}_{i=1}^k$ for a vector space $\mathcal{V}$, the set $\mathcal{S} \cup \{v\}$ is linearly dependent for any vector $v \in \mathcal{V}$, because from the definition of linear span the vector v can be expressed as a linear combination of the elements of the spanning set.

Definition 2.1.17 (Basis)**.** *A basis for a vector space $\mathcal{V}$ is a linearly independent spanning set $\mathcal{B} \subset \mathcal{V}$ for $\mathcal{V}$. The number of elements in $\mathcal{B}$ is the dimension of $\mathcal{V}$ and it may be infinite.*

If $\mathcal{B} = \{u_i\}_{i=1}^{k}$ is a basis for $\mathcal{V}$, then the coefficients $\alpha_1, \ldots, \alpha_k$ in the linear expansion of any vector $uin\mathcal{V}$ must be unique, otherwise the condition of linear independence of $\mathcal{B}$ would not hold anymore [19]. Observe that if $\mathcal{S}$ is a spanning set for $\mathcal{V}$, then there exists a set $\mathcal{B} \subset \mathcal{S}$ such that $\mathcal{B}$ is a bases for the vector space $\mathcal{V}$.

Theorem 2.1.2. *Let $\mathcal{V}$ be a finite-dimensional vector space and suppose that $\mathcal{V}$ has a basis containing k elements. Then the following holds:*

i. Any other basis for $\mathcal{V}$ has also k elements;

ii. Any linearly independent set of k vectors is also a basis for $\mathcal{V}$.

Consequentiality from the previous theorem, we can define the dimension of a vector space $\mathcal{V}$ as the cardinality of a basis for $\mathcal{V}$. For example, the so-called standard basis for a vector space $\mathbb{F}^k$ is the set $\mathcal{B} = \{e_1, \ldots, e_k\}$, where:

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, e_k = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.$$

Then we see that the dimension of $\mathbb{F}^k$ is k and $|\mathcal{B}| = k$ as well.

2.1.4 Orthonormal Bases

As we mentioned before, a basis allows to represent all the vectors as linear combinations of basis elements. Note that there may exist several sets having the same span, thus it is of fundamental interest to find the smallest set with a particular span. The choice of a basis has dramatic consequences, since computational complexity, stability, and approximation accuracy of signal expansions depend on the properties of the bases that are employed. This leads to the concept of dimension of a vector space, which depends on concepts of linear independence and bases defined in the previous section.

Definition 2.1.18 (Orthogonality)**.** *Two vectors u and v are said to be orthogonal when $\langle u, v \rangle = 0$, written as $x \perp v$.*

The concept of orthogonality between vectors can be used to define also the following:

i. $u \perp S$: a vector u is said to be orthogonal to a set of vectors S, when $x \perp s$ for all $s \in S$;

ii. $S_1 \perp S_2$: two sets of vectors S_1 and S_2 are orthogonal when $s_1 \perp S_2$ for all $s_1 \in S_1$;

iii. A set of vectors S is orthogonal when $s_1 \perp s_2$ for every $s_1, s_2 \in S$, such that $s_1 \neq s_2$;

iv. A set of vectors S is orthonormal when is orthogonal and $\langle s, s \rangle = 1$ for every $s \in S$;

v. Given a subspace $\mathcal{W}$ of a vector space $\mathcal{V}$, the orthogonal complement of $\mathcal{W}$ is the set $\{x \in \mathcal{V} : x \perp \mathcal{W}\}$.

Vectors in an orthonormal set $\{u_i\}_{i \in \mathbb{Z}}$ are linearly independent since $\mathbf{0} = \sum_i \alpha_i u_i$ implies that $0 = \langle \sum_i \alpha_i u_i, u_j \rangle = \sum_i \alpha_i \langle u_i, u_j \rangle = \alpha_j$ for any j.

Definition 2.1.19 (Orthonormal basis)**.** *An orthonormal basis for a Hilbert space $\mathcal{H}$ is a basis $\mathcal{B}$ such that every pair of elements belonging to $\mathcal{B}$ are orthogonal unit vectors.*

Theorem 2.1.3. *Let $\mathcal{B} = \{e_1, \ldots, e_k\}$ be an orthonormal basis for a Hilbert space $\mathcal{H}$ and let u any vector belonging to $\mathcal{H}$. The unique coefficients for the expansion of u in terms of $\mathcal{B}$ are given by the inner products $\langle u, e_i \rangle$:*

$$u = \sum_i \langle x, e_i \rangle \, e_i, \; \forall u \in \mathcal{H}. \tag{2.9}$$

For example, the elements belonging to the real plane are vectors defined by two coordinates, representing the left-right and bottom-top distances respectively. Implicitly we use the standard basis $e_1 = [\ 1 \quad 0\ ]^T$ and $e_2 = [\ 0 \quad 1\ ]^T$, which is a particular orthonormal basis of $\mathbb{R}^2$. The vectors e_1 and e_2 are orthogonal and of unit length, for these reason they are called orthonormal. The expansion of u with respect to the basis $\mathcal{S} = \{e_1, e_2\}$ is defined as:

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = u_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + u_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = u_1 e_1 + u_2 e_2. \tag{2.10}$$

Obviously, for this basis, an expansion exists for any u since the coefficients of the expansion u_1 and u_2 are simply the values defining the vector u itself. However in general the existence of an expansion like (2.10) for any vector u depends on the set of vectors $\mathcal{S}$ spanning the whole space and its uniqueness depends on the linear independence of the elements of $\mathcal{S}$. In finite dimensions, as $\mathbb{F}^k$, a sufficient condition to build an orthonormal basis is to have an orthonormal set of size k. This becomes false in infinite dimensions, since having an infinite orthonormal set is not enough to guarantee a proper orthonormal basis.

Theorem 2.1.4. *Given an orthonormal set $\mathcal{S} = \{u_i\}_{i=1}^k$ in $\mathcal{H}$, the following are equivalent:*

i. The set $\mathcal{S}$ is an orthonormal basis for $\mathcal{H}$;

ii. There is no nonzero $v \in \mathcal{H}$ such that $\langle u, v\rangle$ for all $u \in \mathcal{S}$;

iii. The spanning set span($\mathcal{S}$) is dense in $\mathcal{H}$, that is, every vector $v \in \mathcal{H}$ is a limit of a sequence of vectors in span($\mathcal{S}$);

iv. For every $v \in \mathcal{H}$ the Parseval's equality holds:

$$||v||^2 = \sum_i |\langle v, u_i\rangle|^2 . \tag{2.11}$$

v. For every $v_1, v_2 \in \mathcal{H}$ the generalized Parseval equality holds:

$$\langle v_1, v_2\rangle = \sum_i \langle v_1, u_i\rangle \langle v_2, u_i\rangle^* . \tag{2.12}$$

2.1.5 Non-Orthogonal Bases

Although the advantages of using an orthonormal basis are clear, there are still cases where the basis set of the problem we have to cope with could or should be not made orthogonal. In such situations we can still use similar expansion formulas used in the standard orthogonal case, by exploiting a dual basis set $\bar{\mathcal{S}}$, whose elements are not orthogonal to each other, but to the corresponding element of the expansion set.

Definition 2.1.20 (Biorthogonal basis)**.** *A biorthogonal basis is a pair set of equal length $\mathcal{B} = \{u_i\}_{i=1}^k$, $\bar{\mathcal{B}} = \{\bar{u}_i\}_{i=1}^k$, whose elements belong to an Hilbert space $\mathcal{H}$, such that:*

$$\langle u_k, \bar{u}_i\rangle = \delta_{k-i} = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{if } k \neq i \end{cases}, \tag{2.13}$$

where $u_k \in \mathcal{B}$ and $\bar{u}_i \in \bar{\mathcal{B}}$. Moreover there exist strictly positive constants A and B such that for all $v \in \mathcal{H}$:

$$A\,||v||^2 \leq \sum_i |\langle v, u_i\rangle|^2 \leq B\,||v||^2 \tag{2.14}$$

$$\frac{1}{B}\,||v||^2 \leq \sum_i |\langle v, \bar{u}_i\rangle|^2 \leq \frac{1}{A}\,||v||^2 . \tag{2.15}$$

Bases satisfying the above constraints are called Riesz bases.

Clearly, designing a biorthogonal basis pair has more degrees of freedom than designing an orthogonal basis. The disadvantage is that computations can become numerically unstable as the elements of the expansion get closer to colinearity or, in general, are strongly correlated [19]. The calculation of the expansion coefficients using the inner product is called the analysis part of the overall process, while the

calculation of the original vector from those coefficients and the basis vectors is the synthesis part. In finite dimensions, analysis and synthesis are simply matrix-vector multiplications. If the expansion vectors are a basis then the synthesis matrix is squared and non singular and its columns are the basis vectors themselves.

For example, if we want to represent a vector $v \in \mathbb{R}^2$ as an expansion $\alpha_1 u_1 + \alpha_2 u_2$ with respect to $u_1 = [\ 1 \quad 0\ ]$ and $u_2 = [\ \frac{1}{2} \quad 1\ ]$, we should use the non standard element u_2 to match the vertical component of v, that is $\alpha_2 = u_2$, then we correct the horizontal component as well as $\alpha_1 = u_1 - \frac{1}{2} v_2$. In other words: $\alpha_1 = \langle v, \bar{u}_1 \rangle$ and $\alpha_2 = \langle v, \bar{u}_2 \rangle$, where $\bar{u}_1 = [\ 1 \quad -\frac{1}{2}\ ]^T$ and $\bar{u}_2 = [\ 0 \quad 1\ ]^T$. Thus the produced expansion formula is:

$$v = \langle v, \bar{u}_1 \rangle\, u_1 + \langle v, \bar{u}_2 \rangle\, u_2.$$

In general for a k-dimensional problem the expansion of a general vector $v \in \mathbb{F}^k$ becomes the following:

$$v = \sum_i \langle v, \bar{u}_i \rangle\, u_i = \sum_i \langle v, u_i \rangle\, \bar{u}_i.$$

If the basis $\mathcal{B}$ is orthogonal then it is its own dual basis and the expansion formula becomes the usual orthogonal expansion given in Equation (2.9). Additionally, there exist similar equivalences to the ones listed in Theorem 2.1.4 that hold for the biorthogonal case as well. The generalized Parseval's relations then becomes:

$$||v||^2 = \sum_i \langle v, u_i \rangle \langle v, \bar{u}_i \rangle^* \tag{2.16}$$

$$\begin{aligned} \langle v_1, v_2 \rangle &= \sum_i \langle v_1, u_i \rangle \langle v_2 \bar{u}_i \rangle^* \\ &= \sum \langle v_1, \bar{u}_i \rangle \langle v_2, u_i \rangle^* . \end{aligned} \tag{2.17}$$

2.1.6 Frames

In the previous Sections the vectors forming the bases have always been considered as being linearly independent. Obviously it is possible to add more vectors to such bases, thus forming other sets in which the condition of linear dependence no longer holds. Note that it is also possible to build completely new bases of linearly dependent vectors which are still able to span the vector space. The motivations of having more vectors than what we strictly need, are the prospect to relax the constraints that their linearly independent counterparts and, on the other hand, the possibility to provide better solutions. Moreover the condition of linear independence for a basis of a given Hilbert space $\mathcal{H}$ may be sometimes too much restrictive, even if it can be still possible to build a set that, although not satisfying the basis conditions, are still able to span $\mathcal{H}$. In this section we will introduce the basic concepts of a more flexible type of spanning set called frames. Intuitively a frame for a Hilbert

space $\mathcal{H}$ is a set $\mathcal{S}$ that are able to span $\mathcal{H}$, note that if both the dimension of $\mathcal{H}$ and the cardinality of $\mathcal{S}$ are equal to k, the set $\mathcal{S}$ is simply a basis for $\mathcal{H}$.

Theorem 2.1.5. *Suppose that $\mathcal{H}$ is a finite-dimensional Hilbert space and $\{u_i\}_{i=1}^k$ is a finite collection of vectors from $\mathcal{H}$. The the following are equivalent:*

i. $\{u_i\}_{i=1}^k$ is a frame for $\mathcal{H}$;

ii. $span\{u_i\}_{i=1}^k = \mathcal{H}$.

Definition 2.1.21 (Frame)**.** *A collection of vectors $\mathcal{S} = \{u_i\}_{i=1}^k$ in an Hilbert space $\mathcal{H}$ is called a frame if there exist two constants $A > 0$ and $B < \infty$ such that, for all $v \in \mathcal{H}$:*

$$A \left|\left|v\right|\right|^2 \leq \sum_i \left|\langle u, u_i\rangle\right|^2 \leq B \left|\left|v\right|\right|^2. \tag{2.18}$$

The constants A and B are called frame bounds, and when they are equal the frame $\mathcal{S}$ is called a tight frame. In a tight frame we have:

$$\sum_i \left|\langle v, u_i\rangle\right|^2 = A \left|\left|v\right|\right|^2, \forall v \in \mathcal{H}, \tag{2.19}$$

which can be rewritten, by putting $\frac{1}{\sqrt{A}}$ into the summation, as:

$$\sum_i \left|\left\langle v, A^{-\frac{1}{2}} u_i \right\rangle\right|^2 = \left|\left|v\right|\right|^2. \tag{2.20}$$

In other words, any tight frame can be rescaled to be a tight frame with frame bound 1, yielding a Parseval tight frame $\{u_i\}_{i=1}^k$. In such frames a vector, or signal, $v \in \mathcal{H}$ can be expanded as follows:

$$v = \sum_i \langle v, u_i\rangle \, u_i. \tag{2.21}$$

Note how the expansion in (2.21) resembles the standard formula in the case of an orthonormal basis. However a frame in general does not constitute an orthonormal basis. If all the vectors in a tight frame have unit norm, then the constant A gives the redundancy ratio (for example, $A = 2$ means there are twice as many vectors as needed to cover the space). Note that if $A = B = 1$, and $||v_i|| = 1$ for all i, then the frame is an orthonormal basis, conversely an orthonormal basis is always a Parseval frame.

As a consequence of the linear dependence of the elements of a frame, the expansion of a vector v is not unique. In particular there exist a set of coefficients β_i, not all equal to zero such that $\sum_i \beta_i u_i = 0$. Thus, if can write v as:

$$v = \sum_i \alpha_i u_i,$$

then we can add β_i to each coefficient α_i without changing the correctness of the expansion.

Definition 2.1.22 (Frame redundancy)**.** *In a k-dimensional Hilbert space $\mathcal{H}$, let $\{u_i\}_{i=1}^n$ be a frame for $\mathcal{H}$ having n elements. The redundancy of the frame is the quantity $\frac{n}{k} > 1$.*

The redundancy of the frame gives a description of the size of the frame with regard to the spanned vector space. In application where the representation of signals is degraded by noise, such redundancy may help to distinguish the relevant part of the observed phenomena reducing the effects of noise in the signals themselves. For example if we code a signal with a series of coefficients, having a certain degree of redundancy leads to a better chance of understanding the underlying signal even if the observation presents some type of corruption due to noise. Furthermore, because of this redundancy, recalling the expansion formula for Parseval frames of Equation (2.21), note that there may exist other collections of vectors $\{w_i\}_{i=1}^k \in \mathcal{H}$ that are equally capable to properly reconstruct the vector v:

$$v = \sum_i \langle u, w_i \rangle \, u_i \, , \forall v \in \mathcal{H}.$$

Definition 2.1.23 (Dual frame)**.** *Let $\{u_i\}_{i=1}^k$ be a frame for an Hilbert space $\mathcal{H}$. A sequence $\{w_i\}_{i=1}^k \in \mathcal{H}$ is called a dual frame for $\{u_i\}_{i=1}^k$ if $\{w_i\}_{i=1}^k$ satisfies the expansion formula for all $x \in \mathcal{H}$:*

$$v = \sum_i \langle u, w_i \rangle \, u_i. \tag{2.22}$$

If we define a linear operator A by the Parseval's expansion formula as:

$$Av = \sum_i \langle v, u_i \rangle \, u_i. \tag{2.23}$$

Since A is positive and invertible on $\mathcal{H}$, we can replace v by $A^{-1}v$, obtaining:

$$v = AA^{-1}v = \sum_i \left\langle A^{-1}v, u_i \right\rangle u_i = \sum_i \left\langle v, A^{-1}u_i \right\rangle u_i.$$

Additionally if we apply A^{-1} to both members of Equation (2.23) we get:

$$v = A^{-1}Av = A^{-1} \sum_i \langle v, u_i \rangle \, u_i = \sum_i \langle v, u_i \rangle \, A^{-1}u_i.$$

Combining the preceding equations we finally arrive to the expansion formulas:

$$v = \sum_i \left\langle v, A^{-1}u_i \right\rangle u_i = \sum_i \langle v, u_i \rangle \, A^{-1}u_i, \tag{2.24}$$

where the collection of vectors $\{w_i\}_{i=1}^k = \{S^{-1}u_i\}_{i=1}^k$ is called the canonical dual frame of $\{u_i\}_{i=1}^k$, that is $w_i = S^{-1}u_i$ for all $i \in \{1, \ldots, k\}$. In fact, given the invertibility of A, the collection $\{A^{-1}u_i\}_{i=1}^k$ is a frame and, thus, it spans all $\mathcal{H}$. Similarly $\{u_i\}_{i=1}^k$ is the canonical dual frame of $\{S^{-1}u_i\}_{i=1}^k$. Conversely if the condition $w_i = S^{-1}u_i$ for all $i \in \{1, \ldots, k\}$ does not hold the frame $\{w_i\}_{i=1}^k$ is called the alternate dual frame.

Theorem 2.1.6. *If $\{u_i\}_{i=1}^k$ is a basis for a Hilbert space $\mathcal{H}$, then its dual frame is unique.*

Theorem 2.1.7. *If $\{u_i\}_{i=1}^k$ is a frame for a Hilbert space $\mathcal{H}$, then $\{u_i\}_{i=1}^k$ has a unique dual frame if and only if $\{u_i\}_{i=1}^k$ is a basis.*

Theorem 2.1.8. *Let $\{u_i\}_{i=1}^k$ be a frame for a Hilbert space $\mathcal{H}$ and let $\{w_i\}_{i=1}^k$ be a dual frame for $\{u_i\}_{i=1}^k$. Then $\{w_i\}_{i=1}^k$ is the canonical dual frame if and only if:*

$$\sum_{i=1}^{k} |\langle v, w_i \rangle|^2 \leq \sum_{i=1}^{k} |\langle v, \bar{w}_i \rangle|^2, \forall v \in \mathcal{H}, \tag{2.25}$$

for all frames $\{\bar{w}_i\}_{i=1}^k$ which are duals of $\{u_i\}_{i=1}^k$.

2.2 Transform Operators

In this section there will be presented the main transform operators applicable to continuos and digital signals, from the standard one-dimensional case to the more interesting, in this Thesis, multi-dimensional case, i.e. the transformation of digital images. A signal may exist in its original form as a continuous entity or it may be also generated directly as a discrete sequence. Continuous signals may be converted into a quantized form as a sequence of sampled values that describe the realization of the signal at discrete points in time. Images are examples of signals that are originated in a continuous form, even if they are generally processed as discrete sequences.

The signal output in the transformed domain may be analyzed, interpreted, and further processed for implementing diverse image processing tasks. These transformations are widely used, since by using these transformations, it is possible to express an image as a combination of a set of basic signals. In fact signal analysis usually is focused on the decomposition of signals into a linear combination of basis functions.

Such transformations are motivated by, essentially, the uncovering of several meaningful information, which can be hardly seen in the original domain of the signal. For example, these methods could be used in applications of signal compression and transmission where the original observation is transformed in a significantly reduced representation. It is obvious, by means of the signal compression example,

that a reverse transformation is needed to reconstruct the original signal rapidly and without loss of information.

In the following sections we will introduce the most common used transformations with a particular focus on their application on discrete two-dimensional signals, i.e. digital images, presenting the necessary formal notion supported by some visual examples. However for an exhaustive investigation of the topics, it is useful to refer to the dedicated literature [46].

2.2.1 Fourier Transform

Fourier's theorem states that it is possible to construct any generic one-dimensional continuous function $f(x)$ as a summation of sine and cosine terms of increasing frequency multiplied by specific coefficients. Thus, in this transform, the basis functions are sinusoids with different periods which describe the spatial frequencies in an image. Instead of representing the signal amplitude as a function of time, in case of continuous one-dimensional $f(x)$, or as a function of space, in case of two-dimensional discrete images, the transform represents the signal by how much information is contained at different frequencies. As a first introductory example, refer to Figure 2.1 where a simple linear combination of sinusoids is shown, more specifically the function $y(x) = \cos(2\pi\omega_0 x) + 4\cos(2\pi\omega_1 x)$. As expected the transform produce two impulse in the frequency domain in correspondence with the characteristic $\omega_{0,1} = \{10Hz, 2Hz\}$ of the single term that compose $y(x)$, and the amplitude values are represented as well: in fact the magnitude of the lower-frequency cosine is four times bigger than the other, higher frequency, term.

Using Fourier transform, thus it is possible to analyze a signal as a set of sinusoids, wherein each sinusoid has a precise frequency. However for a proper understanding of what we are going to use, the two-dimensional discrete Fourier transform, it is useful to recall the Fourier transform for a continuous one-dimensional function $f(x)$.

Let $f(x)$ denote a continuous-time signal. Moreover, assume that $f(x)$ is a finite-energy square-integrable function, that is $f(x) \in L^2(\mathbb{R})$. This means that the following relation holds:

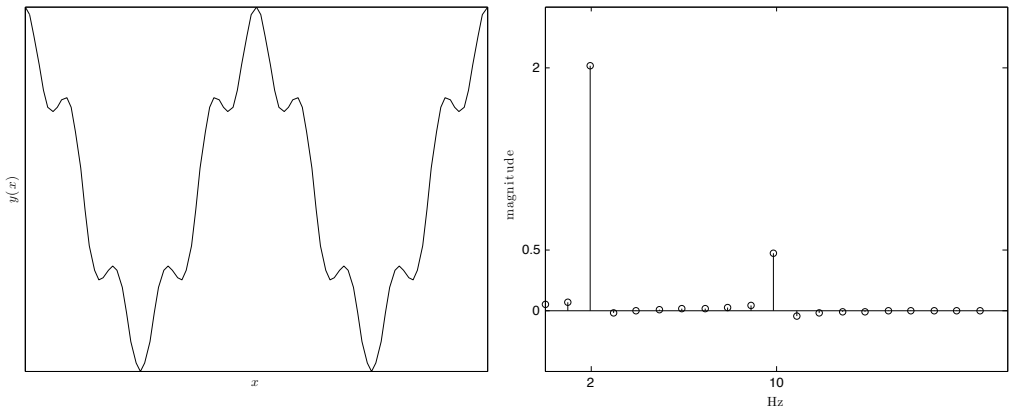
$$f(x) = \int_{-\infty}^{+\infty} |f(x)|^2 dx < \infty.$$

Observe that this condition is not very restrictive because it is satisfied by almost every practically relevant function.

The Fourier transform, FT, is a function describing the amount of each frequency term that must be added together to build back the original signal $f(x)$ and it is defined as follow:

$$F(\omega) = \langle f, e_\omega \rangle = \int_{-\infty}^{+\infty} f(x) e^{-i2\pi\omega x} dx, \tag{2.26}$$

where ω is a real variable called the frequency, $i = \sqrt{-1}$ is the standard imaginary



(a) Original function $y(x)$.

(b) Coefficients of the Fourier transform. As expected we obtain two coefficients located at $\omega_{0,1} = \{10Hz, 2Hz\}$ with magnitude 4 and 1 respectively.

Figure 2.1: Fourier Transform for a linear combination of two sinusoids $y(x) = c_0 \cos(2\pi\omega_0 x) + c_1 \cos(2\pi\omega_1 x)$, where $\omega_{0,1} = \{10Hz, 2Hz\}$ and $c_{0,1} = \{1, 4\}$.

unit and the exponential term, denoted by e_ω, using the well-known Euler's formula, represents the building blocks of the transform:

$$e^{-i2\pi\omega x} = \cos(2\pi\omega xc) - i\sin(2\pi\omega x).$$

The result $F(\omega)$ is a finite energy function on the frequency domain. The original signal may be exactly reconstructed from its spectrum by the inverse Fourier transform, IFT, defined as:

$$f(x) = \int_{-\infty}^{+\infty} F(\omega) e^{+i2\pi\omega x} d\omega. \tag{2.27}$$

To denote such Fourier transform pair, we could write:

$$f(x) \underset{IFT}{\overset{FT}{\longleftrightarrow}} F(\omega). \tag{2.28}$$

Equations (2.26) and (2.27) establish a one-to-one correspondence between a signal and its Fourier transform. Note also that, in some sense, the function $F(\omega)$ is the inner product between the original function $f(x)$ and an element of a Fourier basis $y_\omega = e^{-i2\pi\omega x}$ and the inverse relation of Equation (2.27) is analogous to the reconstruction formula based an orthogonal basis of Equation (2.9), of course replacing the summation $\sum$ with the integral $\int$. Equation (2.26) can be decomposed into its real and imaginary term, and from that we can compute its magnitude, also

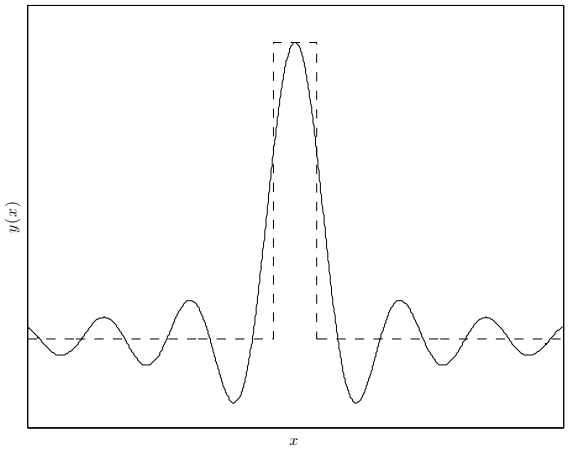


Figure 2.2: Countinuos Fourier transform for the rectangular pulse signal $y(x) = \text{rect}(x)$.

called the Fourier spectrum, and its phase as follows:

$$\begin{aligned} F(\omega) &= R(\omega) + iI(\omega) \\ |F(\omega)| &= \sqrt{R^2(\omega) + I^2(\omega)} \\ \Phi(\omega) &= \tan^{-1}\left(\frac{I(\omega)}{R(\omega)}\right). \end{aligned}$$

In Figure 2.2 it is presented in dashed line the rectangular signal defined as follows:

$$\text{rect}(x) = \begin{cases} 0 & \text{if } |x| > 0 \\ \frac{1}{2} & \text{if } |x| = \frac{1}{2} \\ 1 & \text{if } |x| < 0 \end{cases},$$

and its Fourier transform drawn in solid black line, which is actually the sinc function, hereby defined:

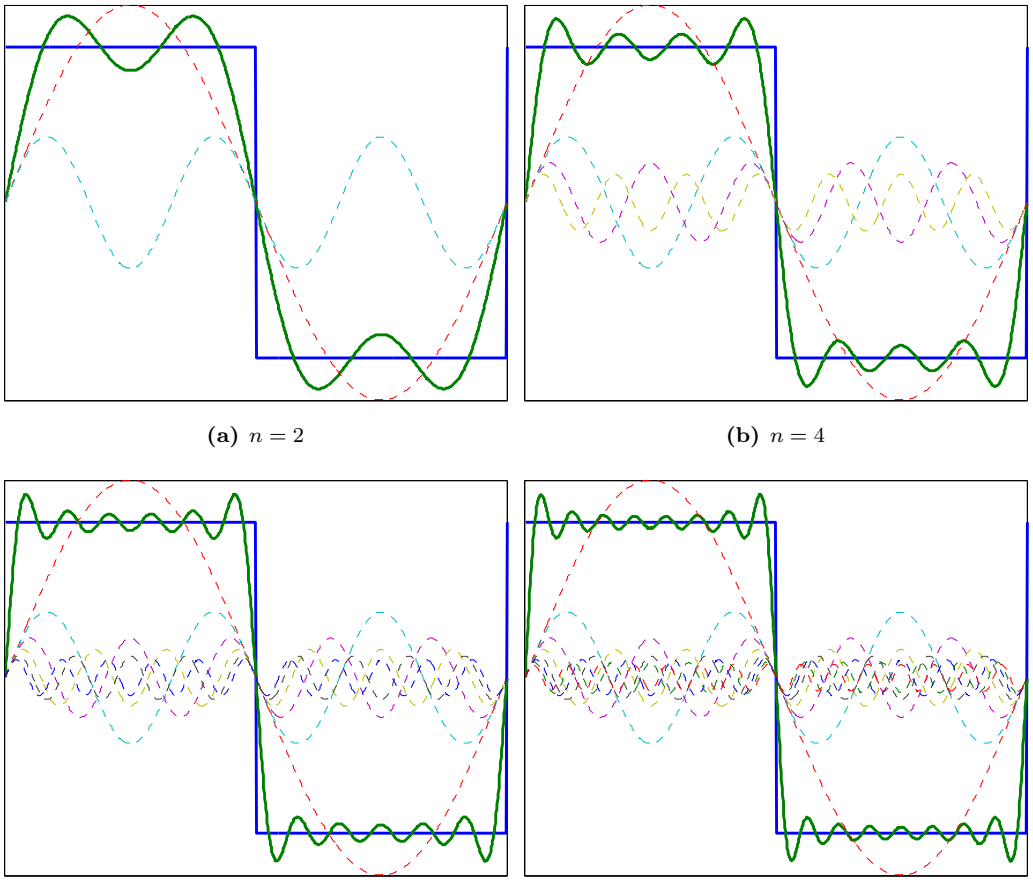
$$\text{sinc}(x) = \begin{cases} 1 & \text{if } x = 0 \\ \frac{\sin(x)}{x} & \text{otherwise} \end{cases}.$$

Another common example is depicted in Figure 2.3, where it is shown the expansion of a square wave of length $L = 2\pi$ using the first $n = \{2, 4, 6, 8\}$ term of the Fourier transform:

$$f(x) = \frac{4}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n} \sin\left(\frac{n\pi x}{L}.\right) \tag{2.29}$$

The original and the synthesized wave are displayed with a thick blue and a thick green line respectively, obviously the quality of the approximation increases with n.

Extending the concept of one-dimensional Fourier transform, the two-dimensional



(a) $n = 2$ **(b)** $n = 4$

(c) $n = 6$ **(d)** $n = 8$

Figure 2.3: Approximation of a square wave with period 2π using the first n sinusoids of the expansion. The single sinusoid terms used in each expansion are illustrated separately in dashed lines.

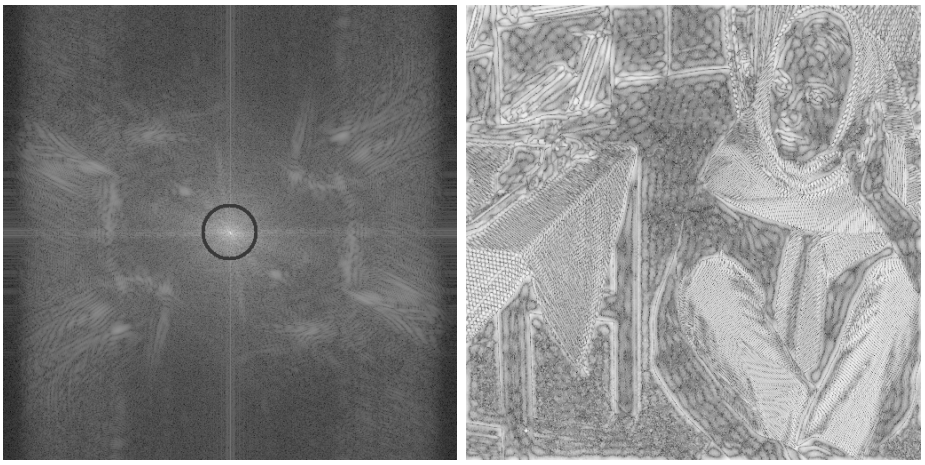
Fourier transform of a continuous function $f(x_1, x_2)$ is denoted by:

$$F(\omega, \psi) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x_1, x_2) e^{-i2\pi(\omega x_1 + \psi x_2)} dx_1 dx_2. \tag{2.30}$$

The variable ω in Equation (2.30) indicates the frequency, i.e. the number of waves per unit length in the horizontal direction, and ψ indicates the number of waves along the vertical direction. For a given pair of values of these frequency components the integral yields just the amplitude of the chosen component. The corresponding inverse relation is:

$$f(x_1, x_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(\omega, \psi) e^{+i2\pi(\omega x_1 + \psi x_2)} d\omega d\psi. \tag{2.31}$$

Finally, when the function or signal is represented in discrete domain using a sequence of discrete samples, such as $f(x) = \{f(0), \ldots, f(N-1)\}$, the corresponding Fourier Transform of the discrete signal is the Discrete Fourier Transform, also called



(a) High-pass filter on the coefficients of the image *Barbara*.

(b) Result of the inverse DFT of the filtered coefficients.

Figure 2.4: Examples of two-dimensional DFT and a high-pass filter application on the image *Barbara*. The left image is the result of the inversion of the Fourier transform applied on the filtered coefficients illustrated in the right image. The high-pass filtering consists in the zeroing of the coefficients lying inside the black circle, i.e. the ones carrying the low-frequency information.

simply DFT. Since the signal is discretized, integrations in the continuous case are replaced by summations.

The one-dimensional discrete Fourier transform of a function $f(x)$ of size N with the independent variable x, i.e. the index, goes from 0 to $N-1$, and the corresponding inverse are defined as:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-i\frac{2\pi ux}{N}} \tag{2.32}$$

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{-i\frac{2\pi ux}{N}}. \tag{2.33}$$

The two-dimensional discrete Fourier transform of a two-dimensional signal $f(x_1, x_2)$ of dimension $M \times N$ with integer indices x_1 and x_2 running from 0 to $M-1$ and 0 to $N-1$, and its inverse are represented by:

$$F(u,v) = \frac{1}{MN} \sum_{x_1=0}^{M-1} \sum_{x_2=0}^{N-1} f(x_1, x_2) e^{-i2\pi\left(\frac{ux_1}{N} + \frac{vx_2}{M}\right)} \tag{2.34}$$

$$f(x_1, x_2) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{-i2\pi\left(\frac{ux_1}{N} + \frac{vx_2}{M}\right)}. \tag{2.35}$$

Observe that each term $F(u,v)$ contains all values of $f(x_1, x_2)$, modulated by the exponential term, thus the value at $(u,v) = (0,0)$, placed at the center of the transform, corresponds to the zero frequency sinusoid in the spatial domain, that is the average gray level or the DC component of the image. As we move away from

the origin the DFT shows the magnitude of the sinusoids of increasing frequency. Low frequencies in the transform domain correspond to slowly varying areas in the spatial image and, conversely, the higher frequencies represent fast changes in the original textures, as edges or even noise. In other words the low-frequency terms represent the overall shape of the signal, while the high-frequency terms are used to sharpen edges and define every fine detail.

Figure 2.4 shows an examples of application of the DFT on images. Each pair of points in the frequency domain corresponds to a sinusoid in the spatial domain. The right-most figure shows the result of the application of an high-pass filter. The original DFT and the filtered one is presented in Figure 2.4(a). The filter drops all coefficients within a specified distance from the origin, illustrated as a black circle, hence some of the low frequencies component of the image are zeroed. For a better recognition of the enhanced details, the filtered image produced by the inverse DFT is presented in a logarithmic scale. As expected, after the application of the inverse DFT, the produced image shows only the highly varying areas of the original image, such as the stripes on Barbara's trousers or the edges of the printed square on the table napkin.

2.2.2 Discrete Cosine Transform

The Discrete Cosine Transform, DCT, is the basis for many image and video compression algorithms, specifically it is behind the JPEG and MPEG standards for compression of still and video images respectively. The intuitive idea behind the DCT is that every finite sequence of digital samples can be expressed as a sum of cosine functions, or, equivalently, the DCT function bases are cosines having different frequencies.

The one-dimensional forward discrete cosine transform of N samples is formulated as:

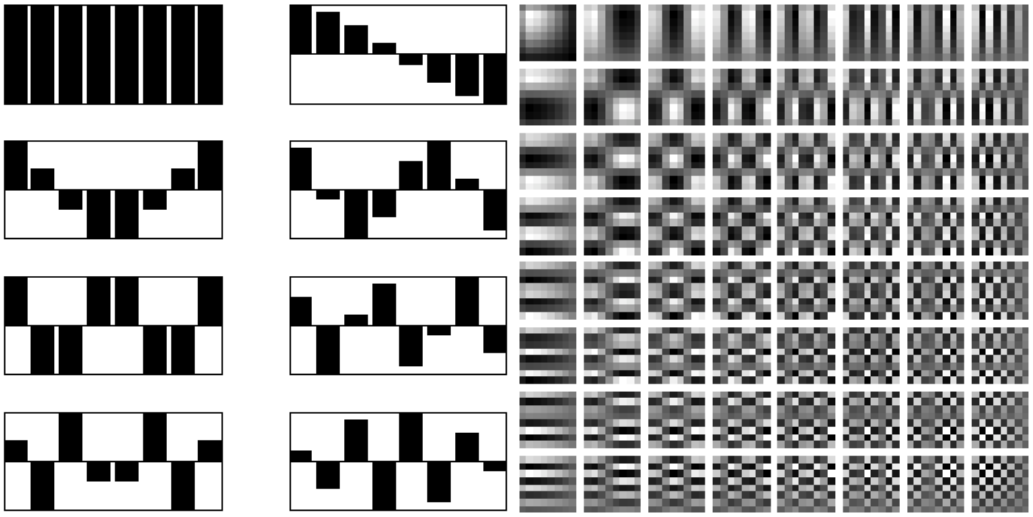
$$F(u) = \langle f, e_u \rangle = \sqrt{\frac{2}{N}} C(u) \sum_{x=0}^{N-1} f(x) \cos\left(\frac{\pi(2x+1)u}{2N}\right), \tag{2.36}$$

for $u = \{0, 1, \ldots, N-1\}$, where the function $C(u)$ is defined as follows:

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.37}$$

The function $f(x)$ represents the value of the x^{th} sample of the input signal while $F(u)$ represents a DCT coefficient for $u = \{0, 1, \ldots, N-1\}$. The inverse relation is formulated in a similar way as:

$$f(x) = \sqrt{\frac{2}{N}} \sum_{u=0}^{N-1} C(u) F(u) \cos\left(\frac{\pi(2x+1)u}{2N}\right). \tag{2.38}$$



(a) Basis of a one-dimensional DCT with $N = 8$. Frequencies increase left-right and top-bottom.

(b) Basis of a two-dimensional DCT with $N = 8$. Frequencies increase from top-left to bottom-right.

Figure 2.5: Illustration of the DCT transform basis functions.

for $x = \{0, 1, \ldots, N-1\}$.

The two-dimensional DCT can be computed using the one-dimensional DCT horizontally and then vertically across the signal because DCT is a separable function. The two-dimensional forward discrete Cosine transform applied to a block of $M \times N$ samples of a two-dimensional signal $f(x_1, x_2)$ is then defined as:

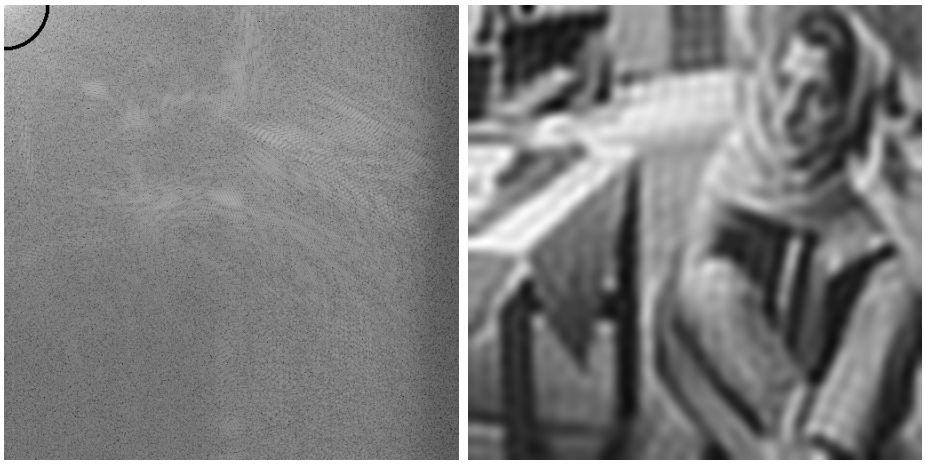
$$F(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{x_1=0}^{N-1} \sum_{x_2=0}^{M-1} f(x_1, x_2) \cos\left(\frac{\pi(2x_1+1)u}{2N}\right) \cos\left(\frac{\pi(2x_2+1)u}{2M}\right), \tag{2.39}$$

for $u = \{0, 1, \ldots, N-1\}$ and $v = \{0, 1, \ldots, M-1\}$, where the function C are equal to the one defined in Equation (2.37).

The function $f(x_1, x_2)$ represents the value of the x_1^{th} sample in the x_2^{th} row of a two-dimensional signal, while $F(u, v)$ is a two-dimensional transformed coefficient for $u = \{0, 1, \ldots, N-1\}$ and $v = \{0, 1, \ldots, M-1\}$. To prove that the DCT transform is a separable function, let's rewrite Equation (2.39) as:

$$F(u,v) = \frac{2}{\sqrt{M}} C(v) \sum_{x_2=0}^{M-1} \left\{ \frac{2}{\sqrt{N}} C(u) \sum_{x_1=0}^{N-1} f(x_1, x_2) \cos\left(\frac{\pi(2x_1+1)u}{2N}\right) \right\} \cos\left(\frac{\pi(2x_2+1)u}{2M}\right).$$

Thus it is clear that we can accomplish the two-dimensional DCT of a signal $f(x_1, x_2)$ by a cascade of two one-dimensional DCT (in two distinct steps). At first a one-dimensional DCT is applied row-wise in all the rows independently to obtain



(a) Low-pass filter of the DCT coefficients of the image *Barbara*.

(b) Inverse DCT applied to the filtered coefficients.

Figure 2.6: Examples of two-dimensional DCT and a low-pass filter on the image *Barbara*. The coefficients outside the black circle in the right image, i.e. those carrying the high frequency information, are zeroed. The result of the inverse transformation of the filtered coefficient is shown in the right image.

an hybrid transform $F(u, y)$:

$$F(u,y) = \frac{2}{\sqrt{N}} C(u) \sum_{x_1=0}^{N-1} f(x_1, x_2) \cos\left(\frac{\pi(2x_1+1)u}{2N}\right),$$

for $u = \{0, 1, \ldots, N-1\}$. Then, in the second step, another one-dimensional DCT is applied column-wise in all the columns of the hybrid transform $F(u, y)$ to obtain the desired result $F(u, v)$:

$$F(u,v) = \frac{2}{\sqrt{M}} C(v) \sum_{x_2=0}^{M-1} F(u,y) \cos\left(\frac{\pi(2x_2+1)u}{2M}\right)$$

for $v = \{0, 1, \ldots, M-1\}$. The inverse relation, the two-dimensional inverse DCT, is defined similarly as follows:

$$f(x_1,x_2) = \frac{2}{\sqrt{MN}} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} C(u)C(v)F(u,v) \cos\left(\frac{\pi(2x_1+1)u}{2N}\right) \cos\left(\frac{\pi(2x_2+1)u}{2M}\right), \tag{2.40}$$

for $x_1 = \{0, 1, \ldots, N-1\}$ and $x_2 = \{0, 1, \ldots, M-1\}$. The above function is again a separable function similar to what we have shown for the two-dimensional DCT. As a result, the two-dimensional inverse DCT can be computed in exactly the opposite way, first applying a one-dimensional inverse DCT row-wise and then applying a second one-dimensional inverse column-wise. The steps have not to be explicitly reported since they are dual of the ones presented in the forward case.

In Figure 2.6 it is represented an example of application of the DCT to the

image *Barbara*, specifically in Figure 2.6(a) there are presented the coefficients of the standard DCT transform, note that the component of higher frequency are displayed in the top-left corner, in the same figure it is shown the application of a low-pass filter, illustrated as a black circle, and, finally, the inverse DCT of the coefficients in Figure 2.6(a) is displayed in the last image wherein only the highly varying component of the original image has been smoothed out as a consequence of the low-pass filtering.

2.2.3 Windowed Fourier Transform

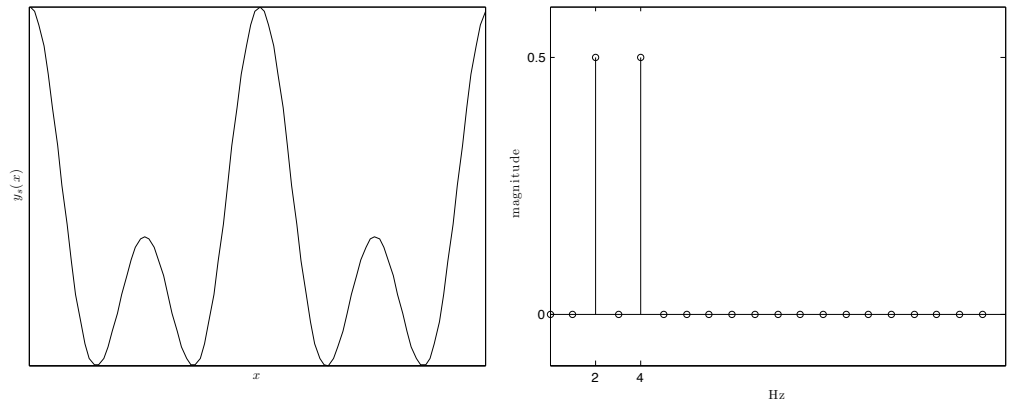
The Fourier transform is an analysis of global frequency content in the signal. The common Fourier representation of signals works well only if the spectral properties of signals are stationary, that is when the frequency components do not change throughout the realization of the signals over time. The stationarity of a signal is a consequence of the statistical invariance of the signal over time. If the probability of the signal f having a certain value at a certain time is constant, then the signal is said stationary, otherwise if transient events occur during the realization of f the signal becomes non stationary because of the unpredictability of those events. Observe for example that in the analysis of the Fourier spectrum of a signal f every frequency component exists throughout the entire duration of the signal, because, referring for example to Equation (2.26), every transform coefficient is affected by all values of the original signal f. Thus transient and non stationary signals require a basis that is more localized in both time and frequency.

For example in Figure 2.7 it is represented a stationary sinusoids y_s and a non-stationary one y_{ns} defined respectively as:

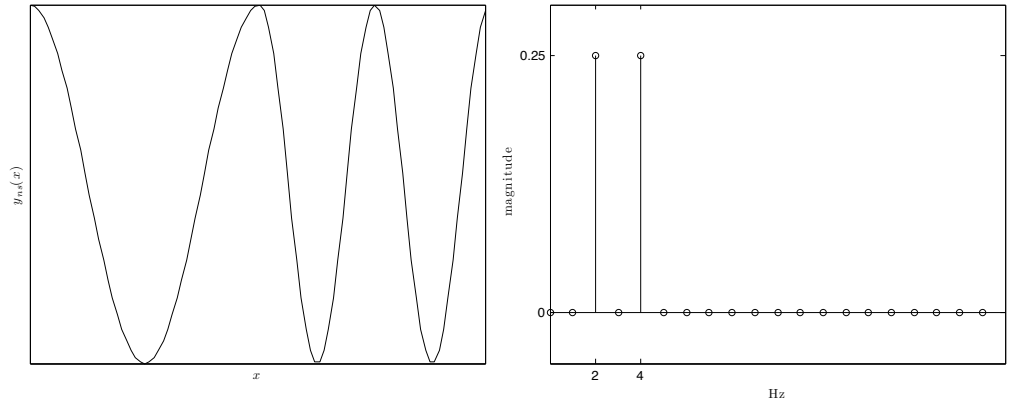
$$y_s(x) = \cos(2\pi\omega_0 x) + \cos(2\pi\omega_1 x)$$
$$y_{ns}(x) = \begin{cases} \cos(2\pi\omega_0 x) & \text{if } x < T \\ \cos(2\pi\omega_1 x) & \text{if } x \geq T \end{cases},$$

where T is a constant and the cosines' frequencies are $\omega_{0,1} = \{2Hz, 4Hz\}$. In both transform domains it can be seen as expected two impulse in correspondence with the original frequencies, however it is equally clear that the locality information is lost since the two spectrums are almost identical, even if the corresponding time-domain signals are quite different. Both of the signals involves the same frequency components, but in the first they occur at different time intervals while in the second one they occur at all times.

The first idea to cope with the non-stationarity problem is to window the signal in time and perform a Fourier transform on the windowed signal and let the window slide along the time axis until the end of the original signal. This is called the



(a) Stationary signal $y_s(x)$ **(b)** Fourier coefficients of $y_s(x)$



(c) Non stationary signal $y_{ns}(x)$ **(d)** Fourier coefficients of $y_{ns}(x)$

Figure 2.7: Fourier transform of a stationary signal $y_s(x)$ and of a non stationary signal $y_{ns}(x)$.

Windowed Fourier Transform, WFT, and it is formally defined as:

$$F(\omega, \tau) = \int_{-\infty}^{\infty} f(x) g(x - \tau) e^{-i\omega x} dx, \tag{2.41}$$

where ω represents the frequency, τ the position of the window and $g(\cdot)$ is the window function. Intuitively the transform $F(\omega, \tau)$ represents the content of the original signal f around time τ and frequency ω. The main limitation of the WFT is that the dimension of the window is fixed and, for this reason it can not reach simultaneously both good time and good frequency resolution, states the Heisenberg Uncertainty Principle. A signal defined over time has a perfect time but zero frequency resolution, while in the opposite side, using the Fourier transform, a perfect frequency resolution is achieved, that is we know exactly what frequencies occur in the signal, but all time information is lost, because we can not place those frequencies in the original time domain.

In WFT a windowing function is used to analyze single portions of the signal, thus lowering the frequency resolution and increasing the time resolution, because

the transform no longer knows the exact frequencies of the whole signal but only those occurring in that specific time interval. Observe that the Fourier transform acts as a WFT with infinite window length. The problem, of course, is the choice of the window function that will be applied, once and for all, during the analysis of the entire signal.

2.2.4 Wavelet Transform

In this section will be introduced the basic concept of the Wavelet transform, WT, a mathematical tool that solves the difficult choice that has to be made between time and frequency resolution in case of non stationary signals.

The wavelet basis is a family of functions based on a well-localized oscillating function $\psi(x)$ of the real variable $x \in \mathbb{R}$. Wavelets are functions generated from one, so-called, mother function by dilation (scaling) and translation (shift) in time (frequency) domain. Let the mother wavelet be $\psi(x)$, then the generated wavelet are defined as:

$$\psi_{s,\tau}(x) = \frac{1}{\sqrt{|s|}}\psi\left(\frac{x-\tau}{s}\right), \tag{2.42}$$

where $s, \tau \in \mathbb{R}$ are two real number representing, respectively, the parameters for dilation and translation in time domain. With this notation the mother wavelet is simply $\psi(x) = \psi_{1,0}(x)$, that is the function with zero translation and a unitary dilation. For example, consider the definition of wavelet called Mexican hat, illustrated in Figure 2.8:

$$\psi(x) = (1-x^2)e^{\frac{1}{2}x^2}. \tag{2.43}$$

A restriction on $\psi(x)$ is that it has to have a zero integral and additionally a high number M of vanishing moments:

$$0 = \int_{-\infty}^{+\infty} \psi(x)dx = \cdots = \int_{-\infty}^{+\infty} t^m\psi(x)dx, \tag{2.44}$$

for all $m < M$. With a mother wavelet $\psi(x)$ satisfying the restriction in Equation (2.44), we can build a set of functions $\{\psi_{s,\tau}(x)\}$ in the form of Equation (2.42), such that any function $f(x)$ in $L^2(\mathbb{R})$ can be reconstructed from the coefficients of the wavelet transform defined by the inner product of $f(x)$ and $\psi_{s,\tau}(x)$. Thus the inner product can be used to define the coefficients of the continuous wavelet transform, CWT, of a signal $f(x)$, formally represented as:

$$W(s,\tau) = \langle f(x), \psi_{s,\tau}(x)\rangle = \int_{-\infty}^{+\infty} \psi_{s,\tau}(x)f(x)dx, \tag{2.45}$$

for $f(x), \psi_{s,\tau}(x) \in L^2(\mathbb{R})$. The CWT is a linear transformation and it is covariant under translation and under dilation, which makes the transform very suitable for analyzing hierarchical structures, like a magnification lens with properties indepen-

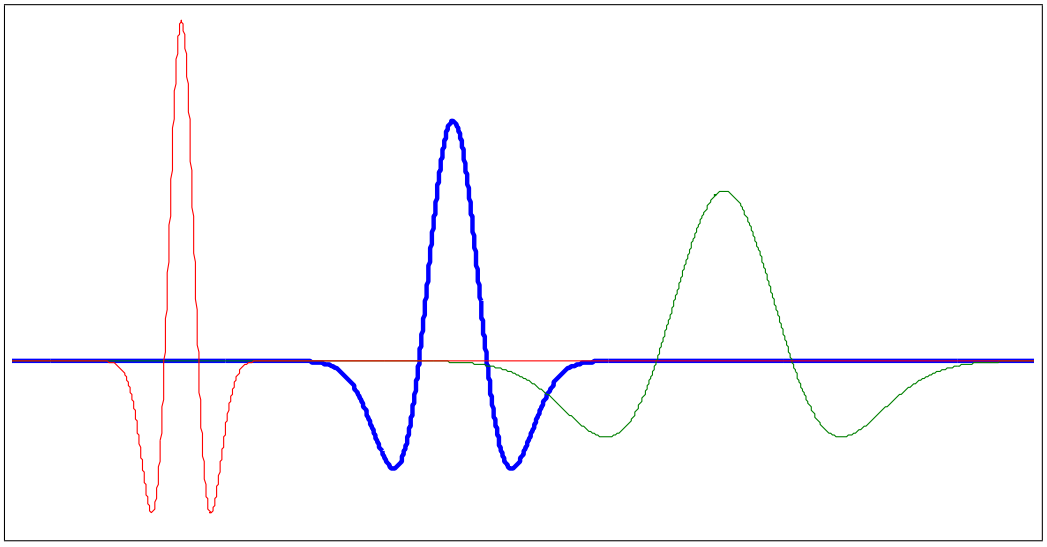


Figure 2.8: Example of translation and dilation of the Mexican hat wavelet. The mother wavelet is illustrated in thick blue line. Dilation parameters $s < 1$ produce narrower and higher-pitched waves as the one illustrated in red line, while $s > 1$ produce wider and lower functions as the one represented in green line.

dent from the chosen scaling parameter.

The transformation defined in Equation (2.45) is called the analysis of the function (or signal) $f(x)$. The synthesis of $f(x)$ from $W(s,\tau)$ is made by a linear combination of the original wavelet using the coefficients of the transform $W(s,\tau)$. Thus the inverse relation, used to reconstruct $f(x)$ from $W(s,\tau)$ is defined as:

$$f(x) = \frac{1}{C}\int_{s=-\infty}^{+\infty}\int_{\tau=-\infty}^{+\infty}\frac{1}{|s|^2}W(s,\tau)\psi_{s,\tau}(x)dsd\tau, \tag{2.46}$$

where

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

and $\Psi(\omega)$ is the Fourier transform of the mother wavelet $\psi(t)$. This is also called the admissibility condition, in fact if $C < \infty$ the wavelet $\psi(x)$ can be used to analyze and reconstruct the signal without loss of information. Note that this condition implies that the Fourier transform $\Psi(\omega)$ vanishes at the zero frequency, or equivalently that the average value of $\psi(x)$ in the time domain must be zero, which is equivalent to the condition of Equation (2.44). Therefore $\psi(x)$ must be a wave. Another important condition for the mother wavelet is the regularity condition, briefly it states that $\psi(x)$ should have some smoothness and concentration in both time and frequency domain.

Hence the continuos wavelet transform maps a one-dimensional function $f(x)$ to a two-dimensional function $W(s,\tau)$ of two continuos real variables s (scaling) and τ (dilation or translation). High scales correspond to a non-detailed global view of the signal and low scales correspond to a detailed view. It is obvious the relation

between high and low scales to low and high frequencies respectively. In Figure 2.8 it is illustrated a few examples of dilation end translation of the mother wavelet ψ informally called Mexican hat.

The CWT, defined in Equation (2.45), is usually impractical. It has one main issue that make it difficult to use and that brings us to the definition of the discrete wavelet transform, DWT, This issue is its redundancy. The CWT is computed continuously shifting a continuously scalable function over a signal $f(x)$. Obviously these scaled function will not form an orthogonal basis and therefore the produced coefficients will be highly redundant while applications almost always seek a sparse signal description with as few components as possible. Thus we will consider a discretization of the continuous wavelet transform in the (s, τ) plane, which allow for numerical solution based on summations instead of continuous integral.

Before introducing the DWT, it is essential to give a formal representation in terms of discrete values of parameters s and τ to define accordingly the discrete wavelets. Many possible discretization of (s, τ) exist, and they typically have a logarithmic nature, however not every such discretization leads to the desired basis functions $\{\psi_{s,\tau}(x)\}$. Among the admissible discretization the most popular approach is the following:

$$\psi_{j,k}(x) = \frac{1}{\sqrt{s_0^j}} \psi \left(\frac{x - k\tau_0 s_0^j}{s_0^j} \right) \tag{2.47}$$

where the indexes $j, k \in \mathbb{Z}$ control the wavelet dilation and translation respectively. A common choice for s_0 and τ_0 is respectively 2 and 1, hence $s = 2^j$ and $\tau = k2^j$. This type of sampling is known as dyadic sampling and the corresponding decomposition of the signal is called the dyadic decomposition, which is perhaps the simplest and most efficient discretization approach. Using these values the discrete wavelet functions become:

$$\begin{aligned} \psi_{j,k}(x) &= s_0^{-\frac{j}{2}} \psi(s_0^{-j} x - k\tau_0) \\ &= 2^{\frac{-j}{2}} \psi(2^{-j} x - k). \end{aligned} \tag{2.48}$$

Discrete dyadic wavelet are commonly chosen to be orthonormal, thus these wavelets are both orthogonal and normalized to have unit energy. In these cases the set of wavelet is an orthonormal basis, therefore the information stored in a wavelet coefficient $c_{j,k}$ is not repeated elsewhere and allows for the complete reconstruction of the original signal without redundancy. In general for dyadic decomposition, the wavelet coefficients (also called detail coefficients) for a continuous function $f(x)$ can be derived by the discrete time wavelet transform:

$$d_{j,k} = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{+\infty} f(x) \psi_{j,k}(x) dx \tag{2.49}$$

This allows us to reconstruct the original signal f from the discrete wavelet coefficients as:

$$f(x) = \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} d_{j,k}\psi_{j,k}(x), \tag{2.50}$$

where the set of expansion coefficients $c_{j,k}$ are called the discrete wavelet transform, DWT, of $f(x)$. Equivalently, recalling Equation (2.49), we can rewrite Equation (2.50) in terms of the inner product as:

$$f(x) = \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} \langle f, \psi_{j,k} \rangle \, \psi_{j,k}(x). \tag{2.51}$$

How close an approximation to the original signal is achieved depends mainly on the resolution of the discretization. In the following section we will see how, given a discrete signal in input, we can compute the forward and inverse wavelet transform discretely, quickly and without loss of signal information using the fast wavelet transform. It can be proven that the necessary and sufficient condition for stable reconstruction of the original signal is that the energy of the wavelet coefficients must satisfy the following:

$$A \, ||f||^2 \leq \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} |\langle f, \psi_{j,k} \rangle|^2 \leq B \, ||f||^2 , \tag{2.52}$$

where $||f(x)||^2$ is the energy of the signal $f(x)$ and the constants $A > 0$ and $B < \infty$ are independent from $f(x)$. When Equation (2.52) is satisfied the family of basis functions $\psi_{j,k}(x)$ is a frame. Remember that when $A = B$ the frame is tight and the discrete wavelets form an orthonormal basis, otherwise when $A \neq B$ an exact reconstruction can still be made with the use of a dual frame.

2.2.4.1 Multiresolution Approximations

Several orthogonal wavelet basis of the form $\psi_{j,k}(x)$ defined in Equation (2.48), have been already discovered, also by using the theory of multiresolution analysis (MRA). The key idea is to approximate a signal $f(x)$ at different levels of resolution. In multiresolution analysis we consider two functions: a scaling function $\phi(x)$ an the well-known wavelet function $\psi(x)$. Moreover the construction of the Fast Wavelet Transform (FWT) is based on the MRA theory [46].

In order to use the idea of multiresolution, we should first define the scaling function and then define the wavelet in terms of it. The scaled and translated version of the scaling function is unsurprisingly defined as:

$$\phi_{j,k}(x) = 2^{-\frac{j}{2}} \phi(2^{-j}x - k). \tag{2.53}$$

Observe that the scaling function is orthogonal to translation of itself, but not to dilations. The scaling function can be convolved to the original signal to produce the so-called approximation coefficients:

$$a_{j,k} = \int_{-\infty}^{+\infty} f(x)\phi_{j,k}(x)dx. \tag{2.54}$$

From the previous equations, we can see that the approximation coefficients are simply weighted averages of the continuous signal factored by $2^{\frac{j}{2}}$. The set of coefficients at a specific scale j are referred to as the discrete approximation at that scale of the given signal $f(x)$, the approximation of the $f(x)$ can be then generated as:

$$f_j(x) = \sum_{k=-\infty}^{+\infty} d_{j,k}\phi_{j,k}(x), \tag{2.55}$$

where $f_j(x)$ is the scaled version of the original function $f(x)$ at scale index j.

We call $\mathcal{V}_j$ the span of the set $\{\phi_{j,k}(x)\}$, for $j > 0$ the span can be larger since the basis $\phi_{j,k}(x)$ are narrower and translated in smaller steps, conversely for $j < 0$ the span is wider and translated in larger steps, that is the wider scaling functions can represent only coarse information. In multiresolution analysis $L^2(\mathbb{R})$ is split into a sequence $(\mathcal{V}_j)_{j\in\mathbb{N}}$ of closed subspaces, each of which is spanned by an orthonormal basis of translates of $\phi_{j,k}(x)$.

$$\ldots \mathcal{V}_{-2} \subset \mathcal{V}_{-1} \subset \mathcal{V}_0 \subset \mathcal{V}_1 \subset \mathcal{V}_2 \subset \ldots \subset L^2(\mathbb{R}). \tag{2.56}$$

Thus the space that contains high resolution signals will contain those of lower resolution also. Moreover the spaces have to satisfy the following condition, which states that the elements in a space are simply scaled version of the ones in the next space:

$$f(x) \in \mathcal{V}_j \Leftrightarrow f(2x) \in \mathcal{V}_{j+1}.$$

Given the previous properties for the basis of $\mathcal{V}_j$, we can define an approximation of a function $f \in L^2(\mathbb{R})$ at the resolution 2^j as the orthogonal projection of $f(x)$ into $\mathcal{V}_j$. In fact each basis functions involved in the approximation of f is orthogonal to every other. A wavelet basis related to $\phi(x)$ represents the additional information of an approximation at a resolution 2^{j+1} compared with the resolution 2^j, which is the projection of the orthogonal component of $\mathcal{V}_j$ in $\mathcal{V}_{j+1}$. Formally this is defined by:

$$\mathcal{V}_{j+1} = \mathcal{V}_j \oplus \mathcal{W}_j, \tag{2.57}$$

where $\mathcal{W}_j$ is the space defined by the projection of the orthogonal component of $\mathcal{V}_j$ in $\mathcal{V}_{j+1}$. In other words $\mathcal{W}_j$ will be spanned by the wavelet orthonormal basis defined in Equation (2.48). We can combine the definition in Equation (2.57) with

the sequence of closed subspaces $\mathcal{V}_j$ in Equation (2.56) as:

$$\mathcal{V}_{j+1} = \mathcal{V}_j \oplus \mathcal{W}_j = \mathcal{V}_0 \oplus \mathcal{W}_0 \oplus \mathcal{W}_1 \oplus \cdots \oplus \mathcal{W}_j. \tag{2.58}$$

The function generated by (2.66) gives the mother wavelet introduced in Equation (2.48). Since we have a set of scaling functions and a set of wavelet functions spanning L^2, then we can express any signal $f(x)$ as a linear combination of $\phi(x)$ and $\psi(x)$. Under certain properties these expansion functions form an orthonormal basis, biorthonormal basis or a frame, which allows the coefficients to be calculated by inner products between the properly related basis functions and the original signal. Orthonormal or biorthonormal bases assure a non-redundant transform, but, on the other hand, they are much harder to find than frames. Thus we can decompose any signal in $L^2(\mathbb{R})$ into a sum of functions starting with lower-resolution approximation followed by a sequence of functions generated by dilations of the wavelet. The two parts represent respectively the coarse approximation of the signal in the lower resolution and the detail information that was lost because of the approximation. We can represent a signal $f(x)$ using a combination of the series expansion using both the approximation and the detail coefficients as follows:

$$f(x) = \sum_{k=-\infty}^{+\infty} a_{j_0,k}\phi_{j_0,k}(x) + \sum_{j=-\infty}^{j_0}\sum_{k=-\infty}^{+\infty} d_{j,k}\psi_{j,k}(t). \tag{2.59}$$

Thus the original signal is expressed as a combination of approximation of itself, at arbitrary scale index j_0, added to a succession of signal details from scales j_0 to negative infinity. If we define the signal detail at scale j as:

$$D_j(x) = \sum_{n=-\infty}^{+\infty} d_{j,k}\psi_{j,k}(x), \tag{2.60}$$

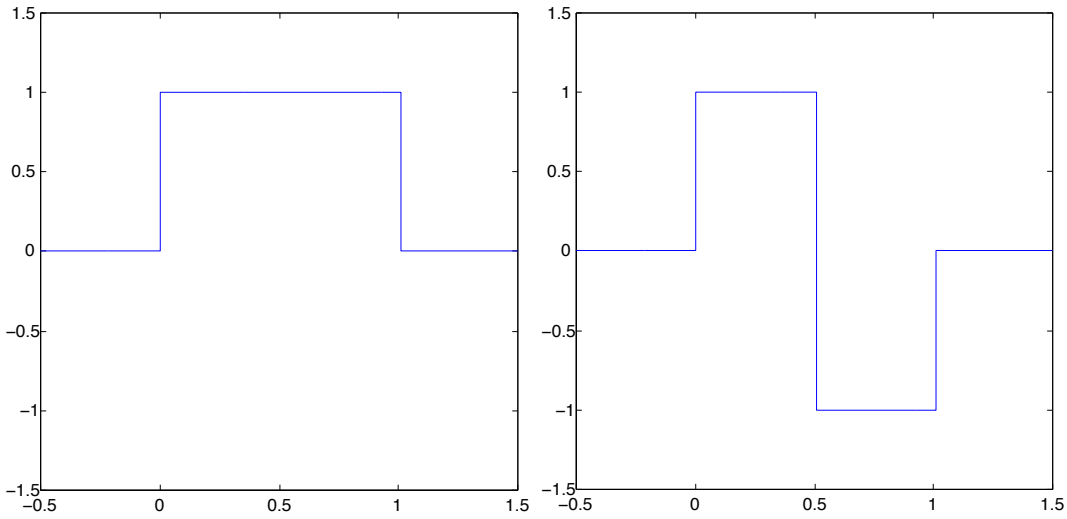
we can rewrite Equation (2.59) as:

$$f(x) = f_{j_0}(x) + \sum_{j=-\infty}^{j_0} D_j(x), \tag{2.61}$$

from which follows that if we add the signal details at an arbitrary scale index j to the approximation at the same scale, we get the signal approximation at an increased resolution (i.e. at the smaller scale $j-1$), this is called the multiresolution approximation:

$$f_{j-1}(x) = f_j(x) + D_j(x). \tag{2.62}$$

To better understand those concepts we can consider the simple Haar basis defined



(a) Haar basis scaling function ϕ. **(b)** Haar basis wavelet function ψ.

Figure 2.9: Illustration of the Haar scaling and wavelet functions.

by:

$$\phi_{Haar}(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{2.63}$$

whose illustration is provided in Figure 2.9(a). Is is easy to see that the convolution of $\phi(x)$ with a signal results in a local weighted averaging of the signal over the non zero portion of the scaling function. Observe that $\phi(x)$ is a solution to the dilation equation, that is can be expressed in terms of weighted sums of shifted scaling functions at the next smaller scales shifted along the time axis by an integer step n, carrying more detailed information:

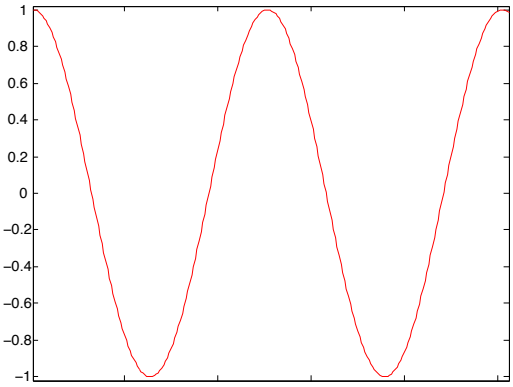
$$\phi(x) = \sum_{n=-\infty}^{+\infty} h_n \phi(2x - n), \tag{2.64}$$

where, in case of wavelet with compact support, the coefficients h_n are a finite sequence of real or complex numbers called the scaling coefficients satisfying the condition $\sum_n h_n = 2$. In addition, in order to create an orthogonal system we require that:

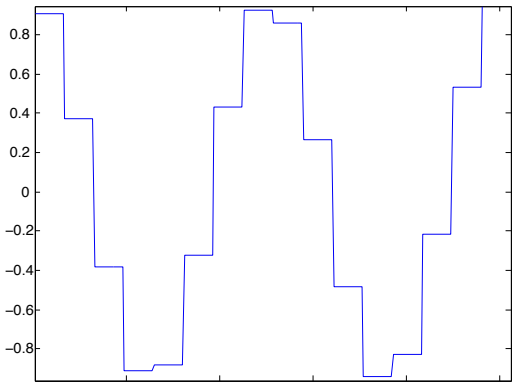
$$\sum_{n=-\infty}^{+\infty} h_n h_{n+m} = \begin{cases} 2 & \text{if } m = 0 \\ 0 & \text{otherwise} \end{cases}. \tag{2.65}$$

Equation (2.64) basically means that a scaling function at a given scale can be constructed using a number of scaling functions at the previous scale. For the Haar basis, we have $h_0 = h_1 = 1$.

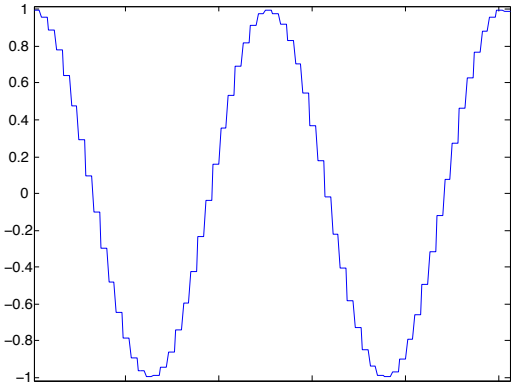
A signal can be better described, not by $\phi_{j,k}(x)$ directly, but by the set of scaling



(a) Original function.



(b) Coarse decomposition in the space $\mathcal{V}_k$.



(c) Fine decomposition in the space $\mathcal{V}_j$.

Figure 2.10: Illustration of the Haar decomposition of a sinusoidal function at different level of resolution, that is the projection of the signal $f(x)$ onto two spaces $\mathcal{V}_j$ and $\mathcal{V}_k$ where $j > k$. Since $\mathcal{V}_k \subset \mathcal{V}_j$ we can say that $\mathcal{V}_j$ is a higher resolution than $\mathcal{V}_k$, thus the projection onto $\mathcal{V}_j$ is much more accurate. Observe that the projection of f onto $\mathcal{V}_k$ can exist in $\mathcal{V}_j$ but the projection onto $\mathcal{V}_j$ does not exist in $\mathcal{V}_k$.

functions $\phi_{j,k}(x)$. Since the wavelet functions resides between the spaces $\mathcal{W}_j \subset \mathcal{V}_{j-1}$ spanned by the various scaling functions $\phi(x)$, they can be represented as a weighted sum of a set of properly shifted $\phi(x)$:

$$\psi(x) = \sqrt{2} \sum_{n=-\infty}^{+\infty} g_{1-n}\phi(2x - n), \tag{2.66}$$

where $g_n = (-1)^n h_{1-n}$. To make $\psi(x)$ orthogonal to $\phi(x)$ we require the set of coefficients $\{g_n\}$ to be orthogonal to the set $\{h_n\}$. For example, the Haar wavelet corresponding to the scaling function defined in Equation (2.63), illustrated in Figure 2.9(b), is $\psi(x) = \phi(2x) - \phi(2x - 1)$, which results explicitly in:

$$\psi_{Haar}(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \leq x \leq 1, . \\ 0 & \text{otherwise} \end{cases} \tag{2.67}$$

However most of the wavelet functions can not be described explicitly, but they can only be defined recursively. It can be derived two conditions on the sets of coefficients $\{h_n\}$ and $\{g_n\}$, the first comes from the normalization of $\phi(x)$:

$$\int_{-\infty}^{+\infty} \phi(x)dx = 1 \Rightarrow \sum_{n=-\infty}^{+\infty} h_n = \sqrt{2}, \tag{2.68}$$

and the second condition is a consequence of zero moment condition expressed in Equation (2.44):

$$\int_{-\infty}^{+\infty} \psi(x)dx = 0 \Rightarrow \sum_{n=-\infty}^{+\infty} g_n = 0. \tag{2.69}$$

From Equation (2.53) and (2.64), we can see that for arbitrary integer values of j the following holds:

$$\begin{aligned}
\phi_{j+1,k}(x) &= 2^{-\frac{j+1}{2}} \phi\left(\frac{x}{2^{j+1} - n}\right) \\
&= 2^{-\frac{j}{2}} 2^{-\frac{1}{2}} \sum_{n=-\infty}^{+\infty} h_n \phi\left(\frac{2x}{2 \times 2^j} - 2k - n\right) \\
&= \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_n \phi_{j,2k+n}(x).
\end{aligned} \tag{2.70}$$

Which means that the scaling function at an arbitrary scale is composed by a sequence of shifted scaling functions at the next smaller scale weighted by their respective scaling coefficients. A similar formulation can be derived for the wavelet function:

$$\psi_{j+1,k}(x) = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} g_n \phi_{j,2k+n}(x). \tag{2.71}$$

Recall from Equation (2.54), that the approximation coefficients at scale index $j+1$ are computed as follows:

$$a_{j+1,k} = \int_{-\infty}^{+\infty} f(x)\phi_{j+1,k}(x)dx,$$

and by using Equation (2.70), we can rewrite the previous equation as:

$$\begin{aligned}
a_{j+1,k} &= \int_{-\infty}^{+\infty} f(x) \left(\frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_n \phi_{j,2k+n}(x)\right) dx \\
&= \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_n \underbrace{\left(\int_{-\infty}^{+\infty} f(x)\phi_{j,2k+n}(x)dx\right)}_{a_{j,2k+n} \forall n},
\end{aligned} \tag{2.72}$$

where the integral within brackets gives the approximation coefficients $a_{j,2k+n}$ for each n, thus:

$$a_{j+1,k} = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_n a_{j,2k+n} = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_{n-2k} a_{j,n}. \tag{2.73}$$

Hence, using this equation, we can generate the approximation coefficients at scale index $j+1$ using the scaling coefficients at the previous scale. Similarly we can obtain the wavelet (detail) coefficients from the approximation coefficients at the previous scale as follows:

$$d_{j+1,k} = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} g_n a_{j,2k+n} = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} g_{n-2k} a_{j,n}. \tag{2.74}$$

Therefore, if we know the approximation coefficients $a_{j_0,k}$ at a specific scale j_0 then, through the repetition of Equation (2.73) and (2.74), we can generate the approximation and detail coefficients at every scale larger than j_0, without even having to know how the specific signal $f(x)$ is made, but only the $a_{j_0,k}$. Equations (2.73) and (2.74) represent the multiresolution decomposition algorithm, which is the first half of the fast wavelet transform. Thus, instead of using the cost-intensive convolution of Equation (2.49), the fast wavelet transform exploit the multiresolution decomposition to compute the wavelet coefficients. If we iterate Equation (2.73) and (2.74), we perform respectively a highpass and lowpass filtering of the input (i.e. the coefficients $a_{j,2k+n}$) to obtain the outputs $a_{j+1,k}$ and $d_{j+1,k}$. The vector $\frac{1}{\sqrt{2}}h_n$ is the lowpass filter which produces a smoothed version of the signal, and $\frac{1}{\sqrt{2}}g_n$ represent the highpass filter which lets through the high frequencies of the signal (i.e. the details).

To reconstruct $a_{j,k}$ from $a_{j+1,k}$ and $d_{j+1,k}$, recalling Equation (2.60), we can expand Equation (2.62) as:

$$f_{j-1}(x) = f_j(x) + D_j(x) = \sum_{k=-\infty}^{+\infty} a_{j,k}\phi_{j,k}(x) + \sum_{k=-\infty}^{+\infty} d_{j,k}\psi_{j,k}(x), \tag{2.75}$$

which can be further rewritten, with respect to Equation (2.70) and (2.71) in terms of the scaling function at the previous scales as follows:

$$\begin{aligned} f_{j-1}(x) = &\sum_{k=-\infty}^{+\infty} a_{j,k} \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_n \phi_{j-1,2k+n}(x) \\ &+ \sum_{k=-\infty}^{+\infty} d_{j,k} \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} g_n \phi_{j-1,2k+n}(x), \end{aligned} \tag{2.76}$$

finally, rearranging the summation indexes, we get:

$$f_{j-1}(x) = \sum_{k=-\infty}^{+\infty} a_{j,k} \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_{n-2k} \phi_{j-1,k}(x) + \sum_{k=-\infty}^{+\infty} d_{j,k} \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} g_{n-2k} \phi_{j-1,k}(x). \tag{2.77}$$

Moreover we can expand $f_{j-1}(x)$ in terms of the approximation coefficients at scale $j-1$:

$$f_{j-1}(x) = \sum_{k=-\infty}^{+\infty} a_{j-1,k} \phi_{j-1,k}(x). \tag{2.78}$$

If we equate the coefficients of Equation (2.77) with Equation (2.78) we note that the index n at scale index j relates to the location index k at scale index $j-1$. Additionally location index k in Equation (2.77) is not equivalent to the location index k in Equation (2.78), because the former corresponds to scale index j with a discretization step of 2^j, and the latter to scale index $j-1$ with a discretization step of 2^{j-1}. Thus the k indexes are twice as dense in Equation (2.78). After swapping the indexes k and n in (2.78), by equating the two equations, we obtain the reconstruction algorithm:

$$a_{j-1,k} = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} h_{k-2n} a_{j,n} + \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{+\infty} g_{k-2n} d_{j,n}, \tag{2.79}$$

where we have reused n as the location index of the transform coefficients at scale index j to differentiate it from k, the location index at scale $j-1$. Thus Equation (2.79) states that, at smaller scale $j-1$, the approximation coefficients can be found in terms of a combination of approximation and detail coefficients at the next scale j. The reconstruction algorithm combined with Equation (2.73) and (2.74) provides the fast wavelet transform (FWT) algorithm.

In many application the data set is structured in the form of a two-dimensional array, and this is the case of image processing, because images are made of a two-dimensional array of size $M \times N$, that is having M rows and N columns, wherein each element carries the intensity value of the image at its position. To perform a discrete wavelet decomposition of two-dimensional data sets, either to compress or to obtain a wavelet-based parametric description of such sets, we must use a two-dimensional discrete wavelet transform. We can simply generate those transforms by means of tensor product of their one-dimensional orthonormal counterpart. If we use the same scaling in the horizontal and vertical directions, we will obtain square transforms. At a given scale, a one-dimensional signal requires two basis functions, but in case

of two-dimensional signals there are four basis functions defined as follows:

$$\begin{aligned}\phi(x_1, x_2) &= \phi(x_1)\phi(x_2)\\ \psi_h(x_1, x_2) &= \phi(x_1)\psi(x_2)\\ \psi_v(x_1, x_2) &= \psi(x_1)\phi(x_2)\\ \psi_d(x_1, x_2) &= \psi(x_1)\psi(x_2),\end{aligned} \tag{2.80}$$

where $\phi(x_1, x_2)$ is the two-dimensional scaling function and $\psi_{h,v,d}(x_1, x_2)$ are the three wavelet functions. For an two-dimensional signal $f(x_1, x_2)$, the transform coefficients are obtained by projecting the input onto the four basis function given in Equation (2.80), which result in four types of transform coefficients located at four different subbands in the decomposition:

$$a_{j+1,(k_1,k_2)} = \frac{1}{2}\sum_{n_1}\sum_{n_2} h_{n_1} h_{n_2} a_{j,(2k_1+n_1,2k_2+n_2)} \tag{2.81}$$

$$d^{(h)}_{j+1,(k_1,k_2)} = \frac{1}{2}\sum_{n_1}\sum_{n_2} g_{n_1} h_{n_2} a_{j,(2k_1+n_1,2k_2+n_2)} \tag{2.82}$$

$$d^{(v)}_{j+1,(k_1,k_2)} = \frac{1}{2}\sum_{n_1}\sum_{n_2} h_{n_1} g_{n_2} a_{j,(2k_1+n_1,2k_2+n_2)} \tag{2.83}$$

$$d^{(d)}_{j+1,(k_1,k_2)} = \frac{1}{2}\sum_{n_1}\sum_{n_2} g_{n_1} g_{n_2} a_{j,(2k_1+n_1,2k_2+n_2)}, \tag{2.84}$$

where n_1, n_2 are the scale coefficient indices and k_1, k_2 are the location indices at scale $j+1$ (compare with the one-dimensional case of Equation (2.73) and (2.74)). The $a_{j+1,(k_1,k_2)}$ is the coarse approximation of the two-dimensional signal $f(x, y)$ and corresponds to the LL band, the $d^{(h,v,d)}_{j+1,(k_1,k_2)}$ coefficients represent the vertical, horizontal and diagonal details respectively and they correspond to the LH, HL and HH subbands.

The simplest approach for two-dimensional implementation of the DWT, as we demonstrated in Section 2.2.2, is to separate the computation by applying one-dimensional DWT row-wise and then perform the same one-dimensional DWT column-wise on the intermediate result. The process is shown in Figure 2.11. After the application of the one-dimensional DWT row wise, each row produce the subbands containing the low (L) frequencies and the high (H) frequencies of the original input signal. Then it is applied another DWT column-wise on these subbands, producing the four subbands defined above. The process can be repeated to achieve higher decorrelation by iteratively decompose the subband LL as shown in Figure 2.12. This results in a pyramid structure for the subbands with the coarsest subband at the top and the finest subband at the bottom. The multiresolution nature of the wavelet decomposition compacts the signal energy into small number of wavelet coefficients and, in case of natural images, the greater part of this energy is

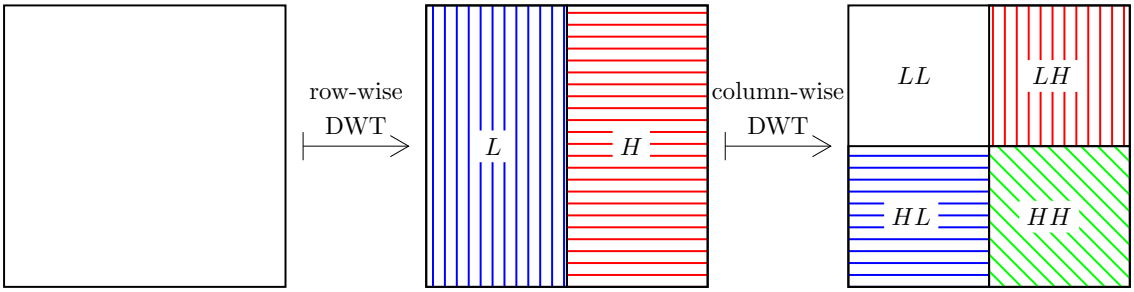


Figure 2.11: Separable computation of a two-dimensional DWT.

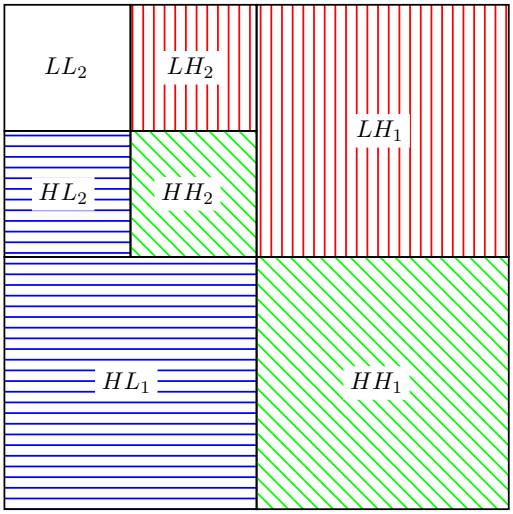


Figure 2.12: Multistage computation of a two-dimensional DWT.

concentrated in the LL band, that is in the coarsest scale.

Note that the LL subband from the first stage has been transformed into four subbands while the three other subbands didn't change. The LL piece comes from low pass filtering in both direction and it is a subsampled approximation of the original image. The remaining three pieces, as said above, are called detail components. The upper-right piece, labeled HL, comes from high pass filtering in the horizontal direction and low pass filtering in the vertical direction. The visible details, such as edges, have an overall vertical orientation, that is perpendicular to the orientation of the high pass filter, consequently they are called the vertical details.

3. DENOISING METHODS

Digital images play a continuously increasing role both in daily life applications, such as photography or satellite television, and in research and technology areas, such as medical or astronomical imaging. The production of digital images, as well as videos, often taken in unfavorable conditions, has massively grown in conjunction with the need of always improved restoration methods, because the data collected by image sensors are generally contaminated by noise.

In this chapter we discuss some of the approaches that can be used to improve the quality of images. Note that even if the focus is mainly on images, similar techniques can be applied to n-dimensional signal as well. These approaches are motivated by the need to improve the quality of the observed signal z to reconstruct a reliable estimate $\hat{y}$ of the original y, which has been corrupted with a random noise component η as described in Chapter 1, by exploiting a priori knowledge about the distribution of the random variable that describes the degradation phenomenon. In fact the model used to describe the noise significantly affects the design of the denoising algorithms.

Even when the digital devices used to acquire the observed signals are capable to produce good results, an image improvement is always a desirable goal. Hence it is necessary to apply an efficient denoising technique to counteract the noise. The main challenge is to remove the noise component without introducing artifacts and/or causing blurring effects in the processed image, however most of the simplest denoising methods, such as those relying on some type of spatial smoothing, degrade or remove the fine details, edges and textures of the original signal.

The following sections presents the specific group of denoising techniques, meaningful to the topics described later in this Thesis. Specifically we start by introducing the basic concept of denoising in case of signal-independent noise and the transform operations to convert a signal-dependent noise into a signal-independent one, then, in the successive sections, we describe the denoising approaches that are going to be the building blocks of the work here presented. These are the parametric, nonlocal and multipoint filtering. Each of those approaches are supported by examples of their applications.

3.1 Homoskedastic Filtering

Recall from Section 1.1 the additive noise model of Equation (1.2):

$$z(x) = y(x) + \eta(x),\ x \in X,$$

where $X \subset \mathbb{Z}^2$, y is the original signal and η is the noise term. The goal is to reconstruct the original signal $y(x)$ from the observed noisy signal $z(x)$ for all $x \in X$. We can treat the function y as a regression of z on x:

$$y(x) = \mathbb{E}\left[z(x)\right],\ x \in X.$$

Formally we can define an operator $\mathsf{D} : \mathbb{R}^{|X|} \to \mathbb{R}^{|X|}$ to express a generic, ideal, denoising operator that produces an estimate of the expectation of the noisy input as:

$$\mathsf{D}(z(x)) = \mathbb{E}\left[z(x)\right] = y(x),\ x \in X. \tag{3.1}$$

As said before, the most common assumption that we can make on the noise is that $\eta(\cdot)$ is an independent and identically distributed normal random variable with zero-mean and constant variance σ^2, i.e. $\eta(\cdot) \sim \mathcal{N}(0, \sigma^2)$. This assumption leads to a noise which model has been broadly used in the field of signal processing mainly due to its simplicity and to its convenient mathematical properties. As a matter of fact, besides some specific algorithms designed to cope with peculiar situation, most of the denoising methods in the literature are built upon the assumption of additive white Gaussian noise. However, as presented in Section 1.3 and Section 1.4, different assumptions are required to better explain the behavior of noise in digital images. In order to handle signal-dependent noise, we have two possibilities when designing a denoising operator D:

- Use an operator D_{he} specifically designed to handle and work with heteroskedastic noise [9];
- Exploit a variance stabilizing homomorphic transformation to make the noise in z signal-independent, and then apply any D_{ho} for common homoskedastic noise on the transformed noisy signal.

In this section will be mainly discussed the latter approach, in particular presenting the Anscombe transform [12] along with it unbiased inverse [45], and finally the homomorphic transformations for clipped signals [25, 10, 24]. The goal is the same for all the above mentioned transformations, that is transform the heteroskedastic signal-dependent noise to apply state-of-the-art denoising algorithms meant for homoskedastic Gaussian noise. In the following sections will be presented the transformations used to stabilize the variance in case of signals corrupted by Poissonian noise and in case of clipped noisy signals, whose models have been presented in

Section 1.3 and 1.4 respectively.

3.1.1 Poissonian Noisy Signals

Even if several algorithms have been designed to deal directly with Poissonian noise, as PH-HMT [57] or MS-VST [14], it is not unusual to apply denoising algorithm not originally meant for this kind of noise to data previously processed with the transformation originally proposed by F.J. Anscombe and thus called the Anscombe transformation. The procedure of denoising a signal corrupted by Poissonian noise is composed by the following three steps:

i. Stabilize the noise variance applying a function f called the Anscombe transformation [12] to the observed noisy signal z:

$$f(z(x)) = 2\sqrt{z(x) + \frac{3}{8}},\ x \in X \tag{3.2}$$

In the resulting signal $f(z)$, the signal-dependence of the noise variance is removed, in a way that the variance itself becomes constant at any $x \in X$ of the signal. In particular, the noise corrupting $f(z)$ can be asymptotically approximated as additive Gaussian with unitary variance;

ii. The noise left by the transformation f is removed using any conventional homoskedastic denoising algorithm D_{ho} designed for additive white Gaussian noise to produce the denoised signal:

$$\mathsf{D}_{\mathsf{ho}}(f(z)) = \mathbb{E}\left[f(z)\right] \tag{3.3}$$

iii. Eventually an inverse transformation $\mathcal{I} = f^{-1}$ is applied to the denoised signal of Equation (3.3) inherited from the previous step, obtaining the final estimate of the original signal y:

$$\hat{y} = \mathcal{I}\left(\mathsf{D}_{\mathsf{ho}}(f(z))\right) \tag{3.4}$$

The above procedure suffers from an imperfection that has restricted its use in practical application because it is not as performing as the ones provided by algorithms specifically designed for Poissonian noise. The criticality can be identified not in the forward (stabilization) transform, but in the inverse transformation of step *iii*. Since the forward transformation is a non-linear function a bias error is introduced after the application of f, for this reason we have to properly choose an adequate inverse transformation in order to minimize that error.

As a first, intuitive attempt, we could define the inverse transformation $\mathcal{I}$ as the direct algebraic inverse of Equation (3.2):

$$\mathcal{I}_A\left(\mathsf{D}_{\mathsf{ho}}(f(z))\right) = f^{-1}\left(\mathsf{D}_{\mathsf{ho}}(f(z))\right) = \left(\frac{\mathsf{D}_{\mathsf{ho}}(f(z))}{2}\right)^2 - \frac{3}{8}, \tag{3.5}$$

but the resulting estimate of the original signal y is biased. In fact, because the non-linearity of the forward transformation f, we can not state that the estimate of the transformed signal is equal to the transform of the estimated signal. Formally:

$$\mathbb{E}\left[f(z)\right] \neq f\left(\mathbb{E}\left[z\right]\right), \tag{3.6}$$

and therefore:

$$f^{-1}\left(\mathbb{E}\left[f(z)\right]\right) \neq \mathbb{E}\left[z\right]. \tag{3.7}$$

Note that we avoid to explicitly write the conditioning on y of the various statistics $\mathbb{E}\left[\cdot|y\right]$, $\mathbb{V}\mathrm{ar}\left[\cdot|y\right]$ and $\mathrm{std}\left[\cdot|y\right]$ to lighten the notation. Anscombe in [12] proposed to adjust the direct algebraic inverse defined in Equation (3.5) using an asymptotical unbiased inverse for large photon counts, i.e. when $\mathsf{D_{ho}}(f(z)) \gg 0$:

$$\mathcal{I}_B\left(\mathsf{D_{ho}}(f(z))\right) = \left(\frac{\mathsf{D_{ho}}(f(z))}{2}\right)^2 - \frac{1}{8}, \tag{3.8}$$

However, even if the unbiased inverse of Equation (3.8) provides good results for high-count data, it does not cope well with low-count data wherein this inverse transformation produces again a biased estimate. Conversely the algebraic inverse is affected by a dual behavior, because the effects of the bias become significant in large-count data while it guarantee good performance in low-count data.

Assuming that $\mathsf{D_{ho}}$ performs an ideal, perfect, denoising process, that is when the produced denoised signal $\mathsf{D_{ho}}(f(z))$ is treated as $\mathbb{E}\left[f(z)\right]$, the focus must be set on the research of an exact unbiased inverse of the Anscombe transformation f. In other words we should find an inverse transformation $\mathcal{I}_C$ that maps the values $\mathbb{E}\left[f(z)\right]$ to the desired values $\mathbb{E}\left[z\right]$ [45], formally:

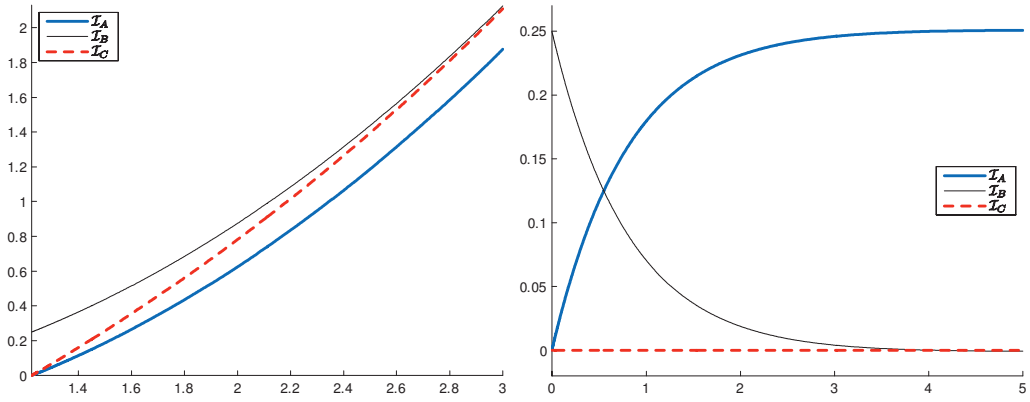
$$\mathcal{I}_C : \mathbb{E}\left[f(z)\right] \mapsto \mathbb{E}\left[z\right]. \tag{3.9}$$

Recall that for any given y, we have that $\mathbb{E}\left[z\right] = y$, thus if we compute the values of $\mathbb{E}\left[f(z)\right]$ we will find the exact unbiased inverse $\mathcal{I}_C$. From the definition of the expectation operator $\mathbb{E}\left[\cdot\right]$ we obtain:

$$\mathbb{E}\left[f(z)\right] = \int_{-\infty}^{+\infty} f(z)p(z|y)dz, \tag{3.10}$$

where $p(z|y)$ is the generalized probability density function of z conditioned on y. Specifically we are considering a Poisson distribution, so we can replace the generalized p.d.f. and the integral of Equation (3.10) with the discrete Poisson probabilities $P(z|y)$ and a summation:

$$\mathbb{E}\left[f(z)\right] = \sum_{z=0}^{+\infty} f(z)P(z|y). \tag{3.11}$$



(a) Inverse transformation. **(b)** Ansolute value of the bias.

Figure 3.1: Comparative plots of the inverse Anscombe transformations $\mathcal{I}_A$ (algebraic), $\mathcal{I}_B$ (asymptotically unbiased) and $\mathcal{I}_C$ (exact unbiased).

From the definition of the Poisson distribution, the probability of observing z given y is formally defined as:

$$P(z|y) = \frac{y^z e^{-y}}{z!}. \tag{3.12}$$

Substituting the Anscombe transform of Equation (3.2) in Equation (3.10), we can rewrite Equation (3.11) as:

$$\mathbb{E}\left[f(z)\right] = 2\sum_{z=0}^{+\infty}\left(\sqrt{z+\frac{3}{8}}\cdot\frac{y^z e^{-y}}{z!}\right). \tag{3.13}$$

In practice, it is not necessary to evaluate Equation (3.13) for every possible y, if we compute $\mathbb{E}\left[f(z)\right]$ for a limited set of values we can derive the result of Equation (3.13) for any arbitrary y by linearly interpolating the computed already known values. Moreover, for large values of y we can approximate $\mathcal{I}_C$ by $\mathcal{I}_B$.

In Figure 3.1(a), it is illustrated the plot of the three inverse Anscombe transformation $\mathcal{I}_A$, $\mathcal{I}_B$ and $\mathcal{I}_C$ described in this section. Note how at low counts the asymptotically unbiased inverse generates to a bigger bias error than the algebraic inverse, while at high counts the bigger bias is produced by $\mathcal{I}_A$ and $\mathcal{I}_B$ has a better behavior. The plot of the bias error is also explicitly illustrated in Figure 3.1(b).

3.1.2 Clipped Noisy Signals

When we apply the generic denoising operator $\mathsf{D} : \mathbb{R}^{|X|} \to \mathbb{R}^{|X|}$ of Equation (3.1) to a clipped signal $\tilde{z}$ we obtain:

$$\mathsf{D}(\tilde{z}) = \widehat{\mathbb{E}\left[\tilde{z}\right]} \approx \mathbb{E}\left[\tilde{z}\right] = \tilde{y}, \tag{3.14}$$

where, in general, $\tilde{y} \neq y$. In other words a denoising algorithm applied to a clipped signal $\tilde{z}$ results in an estimate of $\tilde{y}$ instead of an estimate of the original signal y. Thus the operator $\mathsf{D}(\tilde{z})$ is a biased estimator of y, with a bias error $\tilde{y} - y$:

$$\mathbb{E}\left[\mathsf{D}(\tilde{z})\right] \approx \tilde{y} \neq y.$$

For simplicity, as we did in the previous section, we can assume that D is an ideal denoising operator capable to perfectly recover the expectation of a non-clipped signal z corrupted by i.i.d. Gaussian noise, i.e. $\mathsf{D}(z) = y$.

The procedure is the same of the previous section, we first build a variance-stabilizing homomorphic transformation f, then we apply an homoskedastic denoising algorithms D_{ho} designed for additive white Gaussian noise to the transformed clipped noisy images $f(\tilde{z})$, whose model is defined in Section 1.4, and finally we perform an inverse transformation to the denoised transformed signal to obtain the desired result. Examples of such procedure are shown in [24, 25]. Thus, to accomplish this goal, we can delineate a precise procedure that begins with a preprocessing step. In fact it is needed an estimate the parameters of clipped signal-dependent noise models to identify the functions $\tilde{\sigma}$ and σ. As showed in [10] this can be done automatically using a single noisy image.

At this point formally we need to devise the homomorphic transformation $f : [0, 1] \to \mathbb{R}$ in order to stabilize the variance in the clipped signal $\tilde{z}$ to a desired constant $c \in \mathbb{R}^+$. Let ν be a random variable with known mean and variance:

$$\begin{aligned} \mathbb{E}\left[\nu\right] &= \mu \\ \mathbb{V}\mathrm{ar}\left[\nu\right] &= \mathbb{E}\left[(\nu - \mu)^2\right] = \sigma^2, \end{aligned}$$

and consider a transformation f which admits first and second derivatives applied to ν: $f(\nu)$. If the exact calculations of $\mathbb{E}\left[f(\nu)\right]$ and $\mathbb{V}\mathrm{ar}\left[f(\nu)\right]$ are difficult, we can expand $f(\nu)$ in a Taylor series about the mean μ and use this series representation to approximate $\mathbb{E}\left[f(\nu)\right]$ and $\mathbb{V}\mathrm{ar}\left[f(\nu)\right]$ [21, 51]. The first three terms of the Taylor expansion are:

$$f(\nu) \approx f(\mu) + f'(\mu)(\nu - \mu) + \frac{1}{2} f''(\mu)(\nu - \mu)^2.$$

The approximation for the expected value of $f(\nu)$ is:

$$\begin{aligned} \mathbb{E}\left[f(\nu)\right] &\approx \mathbb{E}\left[f(\mu) + f'(\mu)(\nu - \mu) + \frac{1}{2} f''(\mu)(\nu - \mu)^2\right] \\ &= f(\mu) + \frac{1}{2} f''(\mu)\mathbb{V}\mathrm{ar}\left[\nu\right], \end{aligned} \tag{3.15}$$

which becomes, neglecting the second-order term, $\mathbb{E}\left[f(\nu)\right] \approx f(\mu)$. Similarly, the

approximation of the variance is:

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[f(\nu)\right] &= \mathbb{E}\left[(f(\nu) - \mathbb{E}\left[f(\nu)\right])^2\right] \\ &\approx \mathbb{E}\left[(f(\mu) + f'(\mu)(\nu-\mu) - f(\mu))^2\right] \\ &= \mathbb{E}\left[(f'(\mu)(\nu-\mu))^2\right] \\ &= (f'(\mu))^2\mathbb{E}\left[(\nu-\mu)^2\right] \\ &= (f'(\mu))^2\mathbb{V}\mathrm{ar}\left[\nu\right]. \end{aligned} \tag{3.16}$$

Therefore the standard-deviation is:

$$\begin{aligned}\mathrm{std}\left[f(\nu)\right] &= \sqrt{\mathbb{V}\mathrm{ar}\left[f(\nu)\right]} \\ &= f'(\mu)\sqrt{\mathbb{V}\mathrm{ar}\left[\nu\right]} \\ &= f'(\mu)\mathrm{std}\left[\nu\right]. \end{aligned} \tag{3.17}$$

We obtain an approximated formulation of the standard deviation of $\tilde{z}$ by substituting the generic random variable ν of Equation (3.17) with the transformed clipped signal $\tilde{z}$ and the mean μ with the expected value $\mathbb{E}\left[\tilde{z}\right]$, as follows:

$$\mathrm{std}\left[f(\tilde{z})\right] \approx f'(\mathbb{E}\left[\tilde{z}\right])\mathrm{std}\left[\tilde{z}\right] = f'(\tilde{y})\tilde{\sigma}(\tilde{y}). \tag{3.18}$$

Hence the final formulation of the transform f can be derived by imposing that the standard-deviation of the transformed signal is always equal to an arbitrary constant c, formally:

$$\mathrm{std}\left[f(\tilde{z})\right] \equiv c,$$

or, explicitly, since $\mathrm{std}\left[f(\tilde{z})\right] = f'(\tilde{y})\tilde{\sigma}(\tilde{y})$:

$$f(t) = \int_{t_0}^{t} \frac{c}{\tilde{\sigma}(s)} ds \tag{3.19}$$

where $t, t_0 \in [0,1]$. It can be proved that the indefinite integral in Equation (3.19) is convergent when the following condition is satisfied [24, 25]:

$$\tilde{\sigma}(\tilde{y}) \to 0 \text{ as } \tilde{y} \to 0^+ \text{ or } \tilde{y} \to 1^-, \tag{3.20}$$

which implies that f is a bounded function, and, additionally, we assume that it is also strictly increasing, hence invertible. Observe that the conditions of Equation (3.20) are often satisfied in practice.

After the transformation f has been applied to the clipped data, the signal $f(\tilde{z})$ can be treated as a clipped normal random variable with variance equal to c^2. Thus we can safely utilize an (ideal) homoskedastic denoising algorithm D_{ho} to the trans-

formed data as follows:

$$\mathsf{D}_{\mathsf{ho}}(f(\tilde{z})) \approx \mathbb{E}\,[f(\tilde{z})]$$

However, since f is not a linear function, we have that $\mathbb{E}\,[f(\tilde{z})] \neq f(\mathbb{E}\,[\tilde{z}])$, which is not acceptable because an error in the expectation generates a systematic estimation bias. This problem must be compensated before applying the inverse transformation of f by an additional invertible operator h [24, 25] such that $h(f(\mathbb{E}\,[(]\,\tilde{z}))) = \mathbb{E}\,[f(\tilde{z})]$. The final form of the approximation for homoskedastic filtering then becomes:

$$f^{-1}(h^{-1}(\mathsf{D}_{\mathsf{ho}}(f(\tilde{z})))) \approx \mathbb{E}\,[\tilde{z}] = \tilde{y}. \tag{3.21}$$

The final step is to recover the original signal y from the estimate of the clipped noisy observation $\mathbb{E}\,[\tilde{z}] = \tilde{y}$, and this can be done by exploiting the inverse transformation $\mathcal{C}$ defined in Equation (1.20):

$$\hat{y} = \mathcal{C}(\mathsf{D}_{\mathsf{ho}}(\tilde{z}), \tilde{\sigma}\mathsf{D}_{\mathsf{ho}}(\tilde{z})). \tag{3.22}$$

3.2 Nonlocal Filtering

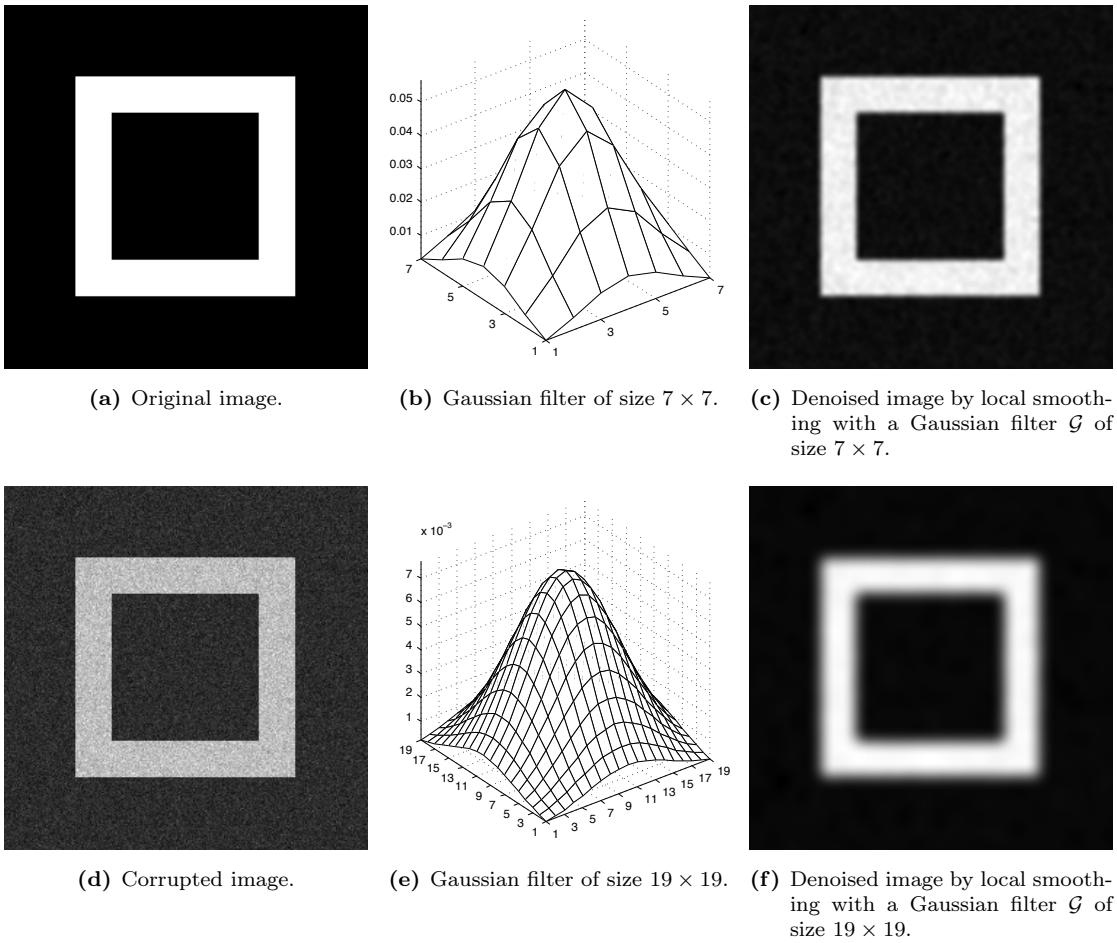
A denoising algorithm D is called local if an estimate of the noise-free signal in a reference pixel x_R is obtained by a set of weights that depend on some distance metric $d(x_R, x_s)$ between the estimation point x_R and the observation points x_s. Typically the weights become smaller as the distance of (x_R, x_s) increases, regardless of the actual signal content. In other words distant points have smaller weights and thus, they give a small contribute to the final estimate of x_R. In practice the support of local filtering is restricted to an area where the metric d is significative.

In Figure 3.2 it is illustrated an example of application of a local Gaussian smoothing, using two kernels of different size, over a synthetic digital image corrupted with white Gaussian noise with $\sigma = 0.1$. Note that the filter correctly reduces the noise in homogeneous areas, while, at the same time, it blurs areas where the image content is rapidly changing, like the sharp edges of the white squares. The trade-off between the blurring in edges and smoothing in constant areas is ruled by the dimension of the filter. The output of the convolution between the noisy image z presented in Figure 3.2(d) and the discrete Gaussian filters $\mathcal{G}$ illustrated in Figure 3.2(b) and 3.2(e) is shown in Figure 3.2(c) and 3.2(f), and it is formally defined as follows:

$$\hat{y}(x) = \mathcal{G}(x) \circledast z(x),\ x \in X, \tag{3.23}$$

which means that each pixel x in the smoothed image $\hat{y}$ is the results of a Gaussian weighted average of a neighborhood of the noisy image z having the same size of the Gaussian kernel $\mathcal{G}$.

On the other hand, an algorithm is said to be nonlocal if the set of weights used in the estimation of the target depend on the similarity of the signal values at the



(a) Original image.

(b) Gaussian filter of size 7×7.

(c) Denoised image by local smoothing with a Gaussian filter $\mathcal{G}$ of size 7×7.

(d) Corrupted image.

(e) Gaussian filter of size 19×19.

(f) Denoised image by local smoothing with a Gaussian filter $\mathcal{G}$ of size 19×19.

Figure 3.2: Gaussian smoothing of a synthetic image.

reference point $z(x_R)$ and at the observation points $z(x_s)$. For this reason, even distant points from the target can have a large contribution to the final estimate, when they exhibit a significant similarity with the target. Note that the similarity of the intensity values is typically computed through the ℓ^2-norm:

$$||z(x_R) - z(x_s)||_2 \,.$$

In this section, to clarify the concept of nonlocal denoising, we present the well-known Non Local Means algorithm [3], from now on simply referred to as NL-means.

The NL-means algorithm is a nonlocal filter that aims at removing the noise, without deteriorating the meaningful information of the original image, by exploiting the redundancy and self-similarity present in any natural image. In fact this method evaluates the similarity between two pixels not only through the intensity value of every pixel encompassed in a predefined neighborhood centered around the processed pixel. More specifically the NL-means algorithm replaces the values in a noisy observation z at a given position x_R by a weighted average of all the pixels of

the image. These weights are function of the similarity between the window of a reference pixel x_R and the window associated to every other pixel x_s in the image.

$$NL(x_R) = \sum_{x_s \in X} w(x_R, x_s) z(x_s),\ x_R \in X, \tag{3.24}$$

where the set of weights $\{w(x_R, x_s)\}_{x_s}$ depend on the similarity between the pixels x_R and x_s. Moreover the weights are normalized as follows:

$$0 \leq w(x_R, x_s) \leq 1 \tag{3.25}$$

$$\sum_{x_s \in X} w(x_R, x_s) = 1. \tag{3.26}$$

In order to define the concept of similarity between any pair of pixels (x_R, x_s) of a noisy image z defined over a grid $X \subset \mathbb{Z}^2$, we have to build a neighborhood system on X.

Definition 3.2.1. *A neighborhood system on X is a family $\mathcal{N} = \{\mathcal{N}_x\}_{x \in X}$ of subsets of X such that for all $x \in X$ the following conditions hold:*

i. $x \in \mathcal{N}_x$;

ii. $x_0 \in \mathcal{N}_{x_1} \Rightarrow x_1 \in \mathcal{N}_{x_0}$.

The set $\mathcal{N}_x \subset X$ is called the neighborhood, or the similarity window, of x. Additionally we call $\tilde{\mathcal{N}}_x = \mathcal{N} \setminus x$

For simplicity we consider only square neighborhoods $\mathcal{N}_x$ of fixed size. The restriction of z to a neighborhood $\mathcal{N}_x$, denoted by $z(\mathcal{N}_x)$, is:

$$z(\mathcal{N}_x) = \Big\{ z(x), x \in \mathcal{N}_x \Big\}. \tag{3.27}$$

The similarity of two pixels (x_R, x_s) is a function of the similarity of the intensity grey-level vectors identified by their neighborhoods $z(\mathcal{N}_{x_R})$ and $z(\mathcal{N}_{x_s})$. The larger similarity of the neighborhoods, the higher the weight assigned at the involved pixels.

Thus, the problem of computing the similarity of two pixels is equivalent to the problem of computing the similarity between two intensity grey-level vectors. One possible solution is the Gaussian weighted Euclidean distance, that consists in taking the sum of squares differences between the two neighborhoods weighted with a Gaussian kernel $\mathcal{G}_\alpha$ having a predefined standard deviation α:

$$d(x_R, x_s) = ||z(\mathcal{N}_{x_R}) - z(\mathcal{N}_{x_s})||^2_{2,\mathcal{G}_\alpha}, \tag{3.28}$$

which is a reliable estimate of the (unknown) distance that the same neighborhoods $y(\mathcal{N}_{x_R})$ and $y(\mathcal{N}_{x_s})$ would have in the original signal y, because this measure is

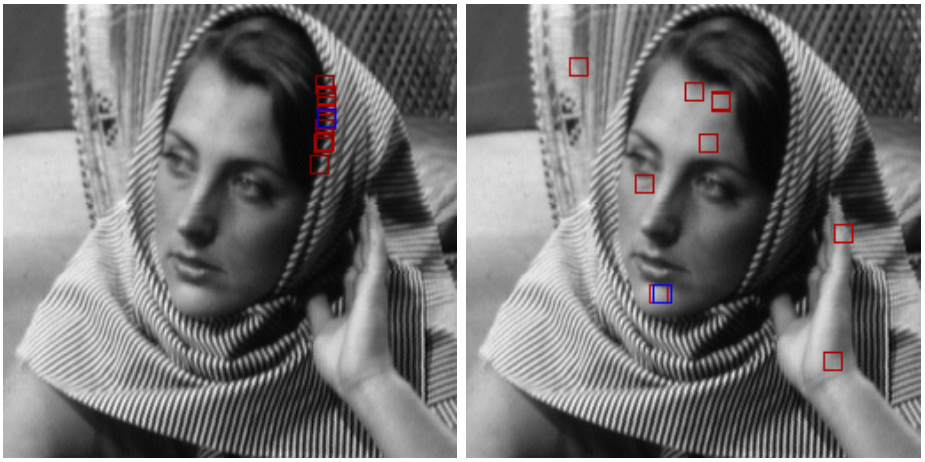


Figure 3.3: Similarity of neighborhoods in NL-means. The blue square is the reference block and the red squares are some of the blocks similar to the reference one.

altered uniformly by the (white Gaussian) noise in z [3, 4]:

$$\mathbb{E}\left[||z\left(\mathcal{N}_{x_R}\right) - z\left(\mathcal{N}_{x_s}\right)||^2_{2,\mathcal{G}_\alpha}\right] = ||y\left(\mathcal{N}_{x_R}\right) - y\left(\mathcal{N}_{x_s}\right)||^2_{2,\mathcal{G}_\alpha} + 2\sigma^2, \tag{3.29}$$

where σ^2 is the variance of the noise η corrupting the original signal y.

Thus the Euclidean distance preserves, in expectation, the order of similarity between pixels. In other words similar pixels in z are expected to be similar also in y. At this point we can formally define the function $w(\cdot,\cdot)$:

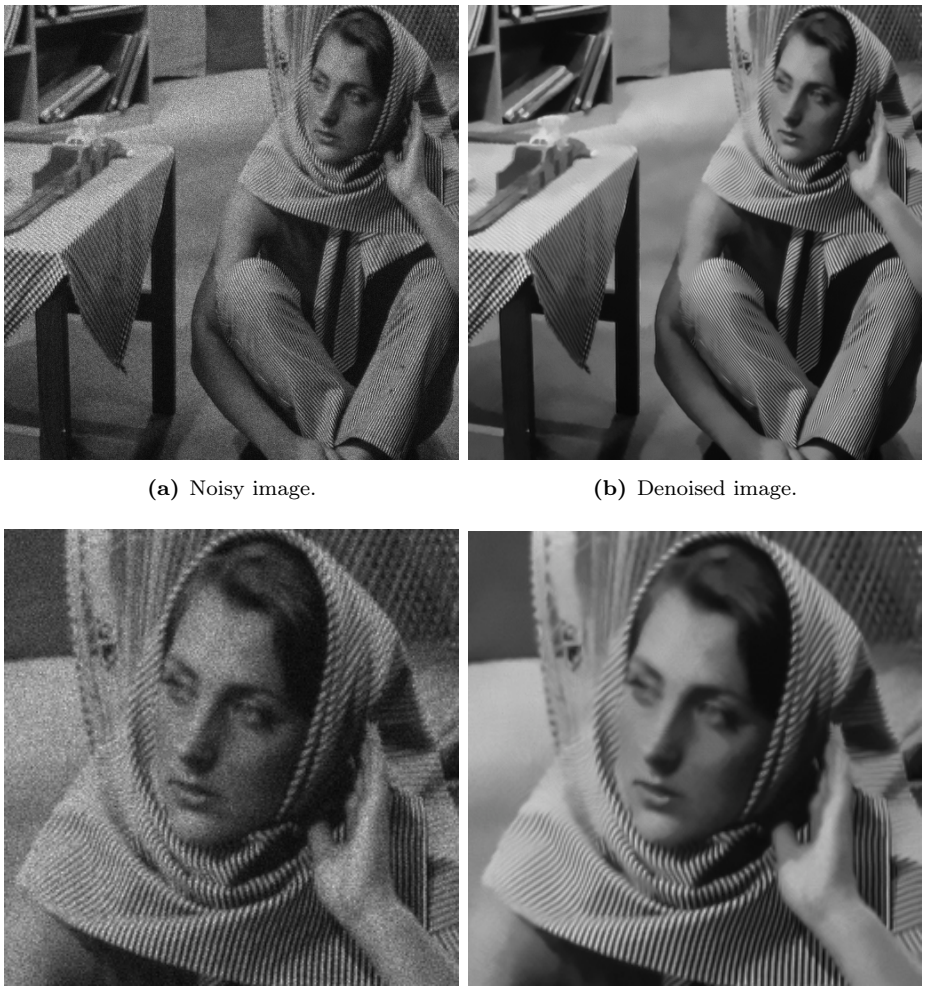
$$w(x_R, x_s) = \frac{1}{Z(x_R)} e^{-\frac{\left|\left|z\left(\mathcal{N}_{x_R}\right) - z\left(\mathcal{N}_{x_s}\right)\right|\right|^2_{2,\mathcal{G}_\alpha}}{h^2}}, \tag{3.30}$$

where the parameter h controls the decay[1] of the weights, and the term $Z(x_R)$ is the normalizing factor, introduced to guarantee that the weights w will satisfy the conditions expressed in Equation (3.25) and (3.26):

$$Z(x_R) = \sum_{x_s \in X} e^{-\frac{\left|\left|z\left(\mathcal{N}_{x_R}\right) - z\left(\mathcal{N}_{x_s}\right)\right|\right|^2_{2,\mathcal{G}_\alpha}}{h^2}}. \tag{3.31}$$

In Figure 3.3 there are illustrated some of the most similar neighborhoods to the reference neighborhood colored in red. As expected they are spread potentially everywhere in the image and in Figure 3.4 it is shown the result of the NL-means applied to the test image *Barbara* corrupted with Poissonian noise. Observe that the image has been transformed with the Anscombe transform described in Section 3.1.1, since NL-means deals naturally with data corrupted by white Gaussian noise.

[1] Usually h is set between 10σ and 15σ, where σ is the standard deviation of the noise in the image. When h is too small, noise removal may not be effective, conversely if h is too large the image will become oversmoothed.



(a) Noisy image. **(b)** Denoised image.

(c) Noisy image (detail). **(d)** Denoised image (detail).

Figure 3.4: Example of the application of the NL-means on an image corrupted with Poissonian noise.

Proceeding top-bottom and left-to-right in Figure 3.4, there are represented the noisy image z, the Poissonian noise η, the denoised image $\hat{y}$ and finally an outlook of the noise removed by NL-means simply computed as the difference between the denoised and the noisy image $|\hat{y} - z|$.

3.3 Parametric Filtering

The algorithms introduced previously in this chapter, the Gaussian local smoothing and NL-means, are both elements of a more general family of denoising methods that encompasses the so-called nonparametric approaches. In general a nonparametric filtering performs a nonparametric regression of the noisy signal z to recover the best possible estimate of the underlying original signal y exploiting the information of one or several predictors $\{x_i\}$:

$$\hat{y}(x) = \mathbb{E}\left[z|x\right] = f(\{x_i\}). \tag{3.32}$$

Nonparametric methods rely on the data itself to determine the structure of the implicit model, often referred to as a regression function. In nonparametric regression, the aim is to estimate the regression function directly that best fits the data without significant a priori knowledge about the form of the true function which is being estimated. Common examples of the regression function f are kernel or nearest-neighbors estimation, local-polynomial regression and local smoothing.

Conversely, parametric denoising methods, which are the main subject of this section, rely on a specific model of the signal, and try to compute the parameters of this model in the presence of noise. Transform domain filtering is an example of parametric approach, refer to Chapter 2 for a brief overview of some of the most used transform operators. The foundational assumption of such methods states that the underlying original signal y admits a sparse representation in a suitable transformed domain. A generative model based upon the estimated parameters is then produced as the best estimate of the underlying signal. Specifically signal models can be used to design a dictionary, i.e. a basis or a frame, that exploits the known features of the signal to build a sparse representation.

Signals that admit a sparse representation can be entirely described and identified using a small set of coefficients, in a way where the signal is disassembled with respect to elementary basis function chosen within a specific family, called dictionary. The Fourier and Wavelet transform decompose a signal y over oscillatory waveforms that can uncover some interesting features of y, which eventually are capable to provide a sparse and more compact representation of y. For example Fourier basis can approximate effectively a uniformly regular signal y with a small number of low-frequency components, while Wavelet basis are a powerful tool to sparsely represent localized and/or transient phenomena of y.

The sparsity can be exploited by thresholding the coefficients of the suitably transformed signal. In fact a method that performs coefficients shrinkage is actually an efficient nonlinear estimator for y. Generally, when the transform is performed with an orthonormal basis, these estimators selects the coefficients having the largest amplitude, because of the assumption that the significant information of signals is localized in such components of the transform. In the remaining part of this section, will be presented a method based upon this concept, the Wavelet Shrinkage denoising algorithm [18, 46, 17]. Wavelet denoising attempts to remove the noise present in the signal while preserving the signal characteristics, regardless of its frequency content, in opposition to smoothing whose effect is to remove high frequency terms only.

Recall from Chapter 1 the standard noise model for digital images:

$$z(x) = y(x) + \eta(x),\ x \in X \subset \mathbb{Z}^2, \tag{3.33}$$

where the component $\eta \sim \mathcal{N}(0, \sigma^2)$. Equation (3.33) is the starting point of the following dissertation: note that in case of signal-dependent noise, the observations

are still valid by doing transformations, like those described in Section 3.1.

The Wavelet Shrinkage method recovers y from the noisy observation z, by thresholding suitable coefficients in the wavelet domain. The algorithm is composed of threes steps:

i. A forward wavelet transform that decompose the signal up to a chosen level N;

ii. A nonlinear thresholding of the detail coefficients from level 1 to N;

iii. A inverse wavelet transform using the altered coefficients.

Observe that the nonlinearity of the thresholding operators implies the nonlinearity of the Wavelet Shrinkage algorithm itself. It has been shown that the tree-steps procedure outlined above is a powerful tool that can provide a near-optimal separation between the meaningful information and the noise in Gaussian corrupted signals [42]. Formally the Wavelet Shrinkage algorithm can be expressed by the following formula:

$$\hat{y} = \mathcal{W}^{-1}\left(\Upsilon_\lambda\left(\mathcal{W}\left(z\right)\right)\right), \tag{3.34}$$

where the operators $\mathcal{W}$ and $\mathcal{W}^{-1}$ are the forward and inverse discrete wavelet transform respectively, and Υ_λ stands for wavelet-domain pointwise thresholding operator having λ as threshold value.

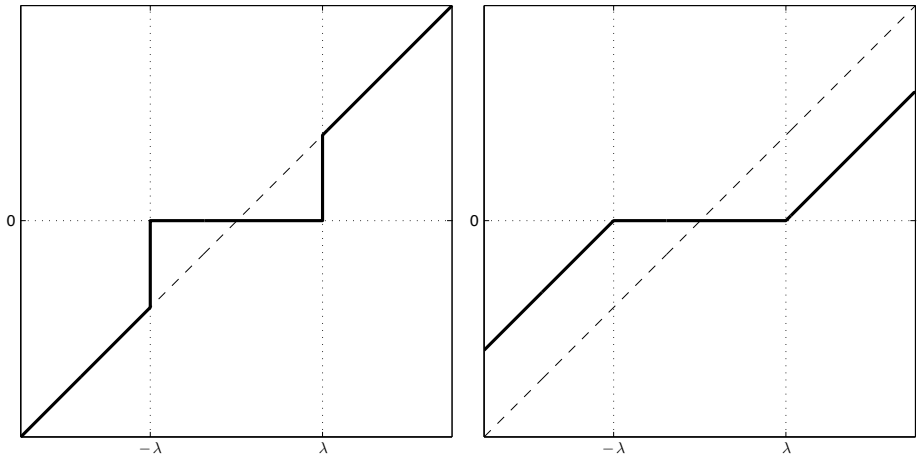
This method is motivated by the idea that the discrete wavelet transform, DWT, produces a representation of a noisy signal z wherein the meaningful information (the original signal y) are noticeably separated from the noise η. Moreover the orthogonal property of the transform assures that the noise in the transform domain is also additive and of Gaussian nature. The DWT compacts the energy of the signal in a small number of coefficients having large magnitude, while it spreads the energy of the noise over a large number of small valued DWT coefficients. Hence a threshold that removes only coefficients below the threshold, and that does not change the large ones, seems to be an effective method to attenuate the effects of the noise η on y.

The most utilized thresholding rules use the so-called hard and soft-thresholding operator. Given a function $f : X \to \mathbb{R}$ with $X \subset \mathbb{R}^n$ the hard thresholding operator is defined as:

$$\Upsilon_\lambda^{\mathsf{ht}}(f(x)) = \begin{cases} f(x) & \text{if } |f(x)| > \lambda \\ 0 & \text{otherwise} \end{cases}, \tag{3.35}$$

while the soft threshold one is:

$$\Upsilon_\lambda^{\mathsf{st}}(f(x)) = \begin{cases} f(x) - \lambda & \text{if } f(x) > \lambda \\ f(x) + \lambda & \text{if } f(x) < -\lambda \\ 0 & \text{otherwise} \end{cases}, \forall x \in X. \tag{3.36}$$



(a) Hard threshold operator $\Upsilon_\lambda^{\mathsf{ht}}$. **(b)** Soft threshold operator $\Upsilon_\lambda^{\mathsf{st}}$.

Figure 3.5: Thresholding operators. The input is represented in black dashed line and the result of threshold is illustrated in thick black line. The vertical dashed grey lines identify the threshold interval $[-\lambda, \lambda]$.

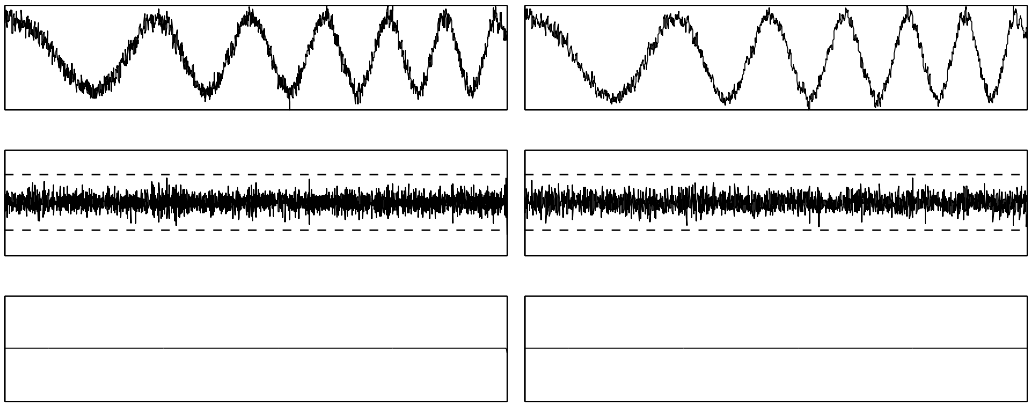
In Equation (3.35) and (3.36) $\lambda \in \mathbb{R}$ is the thresholding parameter[2]. In Figure 3.5 it is illustrated an example of application of both thresholding operators. Beside these two possibilities there are many others (semi-soft shrinkage, firm shrinkage, ...) and, as long as the thresholding rule preserves the sign and shrinks the amplitude of each coefficients, we can expect a denoising effect. Formally these conditions are defined as:

$$\begin{aligned} \operatorname{sign}(\Upsilon_\lambda(f(x))) &= \operatorname{sign}(f(x)) \\ |\Upsilon_\lambda(f(x))| &\leq |f(x)| \end{aligned}$$

Generally, an hard thresholding may create abrupt artifacts because of its discontinuous nature: thus using a soft thresholding is alter preferable. Moreover the soft thresholding yields visually more pleasing images and has been shown to achieve better error rates [22].

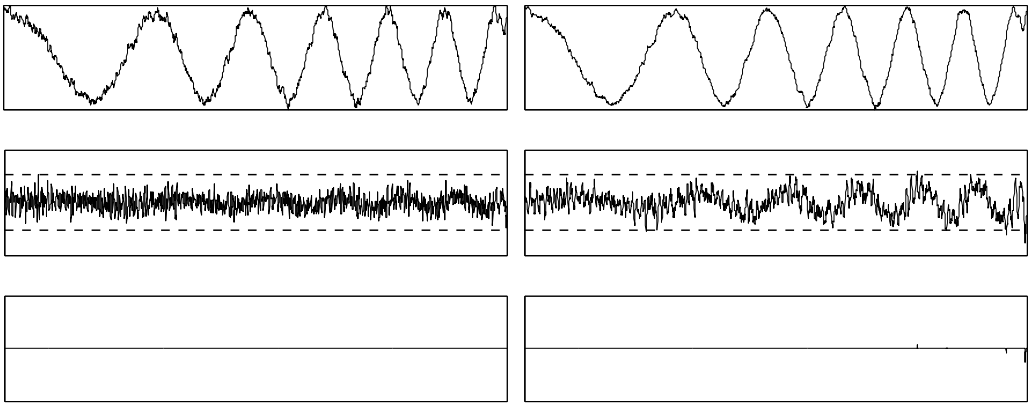
Several approaches have been proposed to select a proper value for λ, i.e. *VisuShrink* and *SureShrink* [18], mainly based on the minimization of the averaged squared error. The choice of the threshold value has a significant impact in the efficacy on the denoising procedure, and when the exact form of either the underlying signal y or the statistics of the noise η are unknown, it becomes a non-trivial task. We can select either a level-dependent threshold or a global constant threshold value, fixed for all levels of the wavelet decomposition. On of the most popular, and the simplest threshold, is the so-called universal threshold, utilized for example by

[2]When the parameter λ is too large, the thresholding operator removes a significant amount of information, i.e. it causes oversmoothing. Conversely when λ is too small, not enough noise energy will be suppressed in the noisy signal z.



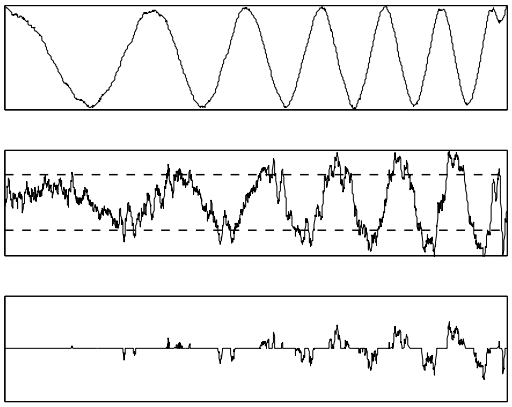
(a) Level 1.

(b) Level 2.



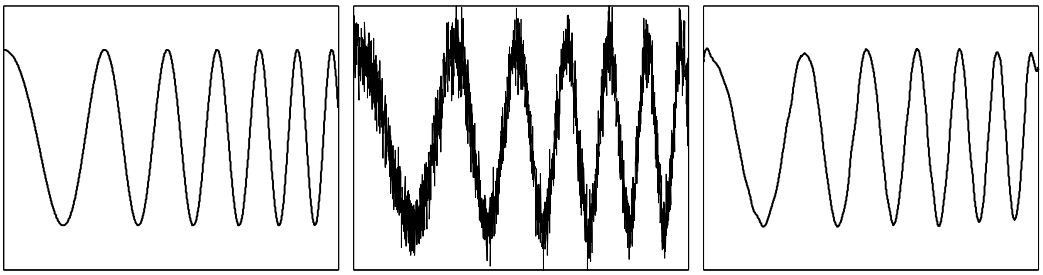
(c) Level 3.

(d) Level 4.



(e) Level 5.

Figure 3.6: Details of the Wavelet Shrinkage on a chirp signal corrupted by white Gaussian noise having standard-deviation $\sigma = 0.2$. The signal is decomposed using the first Daubechies wavelet basis. Each subfigure represents a different decomposition level and it is composed, from top to bottom, by the approximation coefficients, the detail coefficients and the thresholded coefficients.



(a) Original signal. **(b)** Noisy signal. **(c)** Denoised signal.

Figure 3.7: Example of Wavelet Shrinkage on a chirp signal corrupted by white Gaussian noise with standard deviation $\sigma = 0.2$ using the first Daubechies wavelet basis.

(a) Noisy image. **(b)** Denoised image.

(c) Noisy image (detail). **(d)** Denoised image (detail).

Figure 3.8: Example of Wavelet Shrinkage on the image *Barbara* corrupted with white Gaussian noise with standard deviation $\sigma = 0.2$. In the transform the first Daubechies wavelet has been utilized for the 5 levels decomposition of the image.

VisuShrink, and defined as follows [18, 49]:

$$\lambda_U = \sigma\sqrt{2\ln(L)}, \tag{3.37}$$

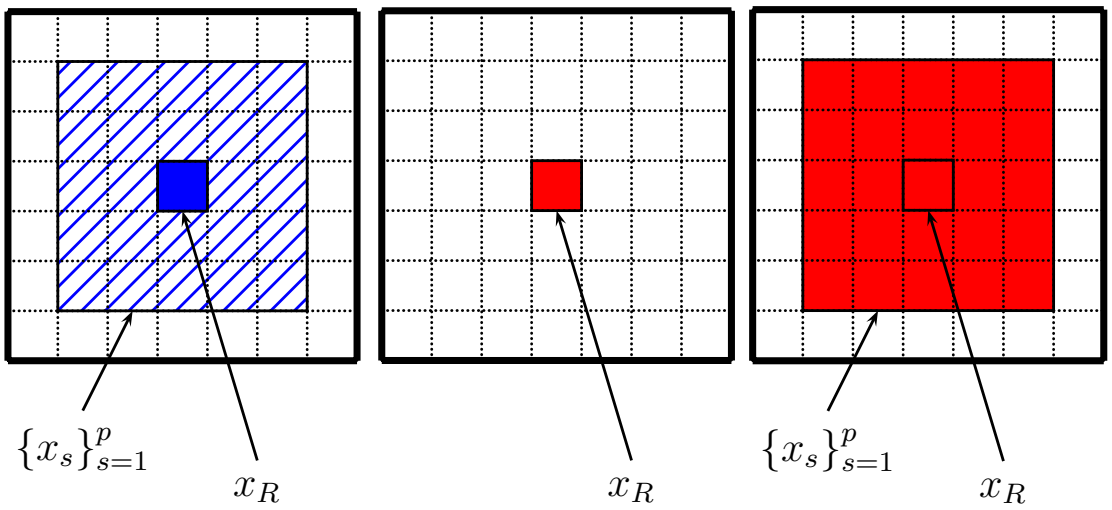
where σ is the standard deviation of the Gaussian noise η, L is the number of samples (pixels in case of digital images) of the observation z and the term $\sqrt{2\ln(L)}$ is the expected maximum value of a white noise sequence of length L with unitary variance. For large samples λ_U will remove, with high probability, all the noise in the estimate $\hat{y}$. In other words, with high probability, a pure noise signal is estimated as being identically zero. However part of the underlying signal will might be lost because in practice λ_U tends to oversmooth the signal as it is derived under the assumption that the estimate $\hat{y}$ is at least as smooth as the signal y. Generally, the standard deviation of the noise σ is not known, thus a robust estimator of σ is needed [20]. As another example, *SureShrink* uses a local threshold λ_i estimated adaptively for each level i of the wavelet decomposition.

In Figure 3.7 it is shown an example of application of the Wavelet Shrinkage, we use a soft thresholding, on a non stationary signal, in particular a chirp signal, corrupted by white Gaussian noise with standard-deviation $\sigma = 0.2$. The signal has been decomposed in 5 levels using the first Daubechies wavelet basis. In Figure 3.6 it is reported the result of the decomposition at every given level $l \in \{1, \ldots, 5\}$ with the corresponding shrinkage. In each subfigure it is shown from top to bottom the approximation coefficients, the details coefficients and the soft-thresholded details coefficients. We select the value of the threshold as in Equation (3.37). Observe that the threshold drops all the coefficients in the first levels of the decomposition, i.e. where the transform reveals the fine details of the signal. Conversely, in the last levels, i.e. where the scale gets coarser, a bigger number of coefficients are preserved from the shrinkage. In Figure 3.7(c) it it presented the result of the algorithm.

Figure 3.8 shows an example of Wavelet Shrinkage applied to two-dimensional signals. The algorithm is tuned with basically the same parameters and characteristics used in the denoising of the chirp signal. The signal (image) has been corrupted by white Gaussian noise haviwithng $\sigma = 0.2$ and the transform consists in 5 levels of decomposition using the first Daubechies wavelet basis. The only difference is on the choice of the λ, in fact, since the universal threshold value λ_U produced an oversmooth estimate, we have manually tuned the threshold bound to improve the result.

3.4 Multipoint Filtering

The last distinguishing feature of a denoising approach, somehow relevant for the purpose of this Thesis, regards the pointwise and multipoint estimation. Observe that every dyadic taxonomy reviewed in this chapter, that is local/nonlocal, parametric/nonparametric and now pointwise/multipoint, turn attention only to the



(a) Initial condition of a generic local denoising algorithm.

(b) The pointwise approach uses the estimate for the reference point x_R only.

(c) The multipoint approach estimates all the points of the image involved in the denoising process.

Figure 3.9: Example of pointwise and multipoint estimation. The solid blue pixel x_R is the reference point and the area denoted by diagonal blue lines contains the set of the observation points $\{x_s\}_{s=1}^p$ used by the denoising algorithm. The solid red areas contains the pixel involved in the estimation process.

fundamental ideas and basic principles of the algorithms. As a matter of fact, most of the implementations of the algorithms presented in the literature, combine denoising techniques that belong to different approaches.

Let D a denoising algorithm and let x_R be a reference pixel for which we want to obtain an estimate. Assuming that D uses a set of p observation points $\{x_s\}_{s=1}^p$, than D is called a pointwise estimator if D returns an estimate for the pixel x_R only. Conversely, a multipoint estimator returns an estimate for all the pixels involved in the denoising process, i.e. $\{x_s\}_{s=1}^p$. In other words a multipoint methods returns a set of estimate, one for each pixel involved in the filtering, while a pointwise approach returns only the estimate of the reference point x_R [64].

As a reference, we can refer to Figure 3.9. In the Figure 3.9(a) it is shown a general situation wherein a local denoising algorithm processes a reference pixel x_R, illustrated in solid blue, using a neighborhood of p pixels $\{x_s\}_{s=1}^p$, denoted by diagonal blue lines. Figure 3.9(b) and Figure 3.9(c) display the pointwise approach and the multipoint approaches, respectively. The solid red areas show the pixel for which the denoising algorithm returns an estimate. As we discussed before, the pointwise method returns an estimate of x_R and the multipoint gives an estimate for all pixels in the neighborhood.

In general, a multipoint approach produces several different estimates in pixel of the image as the same pixel may appear in more than one observation set $\{x_s\}_{s=1}^p$, or, equivalently, it belongs to many overlapping neighborhoods (generally referred to as blocks). As a favorable consequence of this redundancy, the multipoint approaches

typically yield final estimates that have a better accuracy than the ones produced by pointwise methods. However, in multipoint approaches there is the need to aggregate the results of such overcomplete set of estimates, and give a single final estimate for each pixel of the image. The overall estimation process can be informally split up in the three following steps:

i. Data windowing (blocking);

ii. Multipoint estimation for each block;

iii. Calculation of the final estimates by aggregating the overcomplete transform set provided by the multipoint algorithm.

As an example of application of the multipoint approach we describe a denoising algorithm, based on the discrete cosine transform, DCT, introduced in Section 2.2.2. The algorithm is commonly referred to as Block DCT [56, 50, 54, 28], wherein the *Block* term will be clear later in this section.

Signal processing in transform domain rather than in spatial domain has a series of advantages such as the possibility to exploit a-priori knowledge on signals into the design of the algorithms. The effectiveness of filtering can be improved if the transform operator, the DCT in this case, is applied locally rather than globally. Specifically the algorithm at first applies a local orthogonal transform to a sliding window that covers a block of data., and then it performs a nonlinear threshold the transform coefficients to obtain an estimate of the original signal. The main difference with the Wavelet Shrinkage is the locality feature of the DCT. An orthogonal transform like the DCT, as well as the wavelet transform, is an excellent candidate for denoising algorithm in transform domain because of its decorrelating properties.

As always, our aim is to recover an unknown signal y from a noisy observation z corrupted with additive white Gaussian noise:

$$z(x) = y(x) + \eta(x),\ x \in X \subset \mathbb{Z}^2.$$

If the noise is signal-dependent we can exploit the homomorphic transformation described in Section 3.1. A straightforward implementation of a Block DCT denoising algorithm for a noisy signal z is exemplified by the general multipoint procedure outlined in the previous paragraphs. Firstly the noisy image is split in a series of blocks having size equal to $N \times N$ pixels, then each block is transformed using the DCT and the resulting coefficients are thresholded using a proper threshold value λ. Finally an inverse DCT is applied to the thresholded coefficients obtaining the final estimate $\hat{y}$ of the original signal y.

A more powerful approach consists in finding an estimate of a local neighborhood for each pixel belonging to the noisy image, and subsequently producing a final estimate of the every pixel by aggregating the overlapping estimates of the

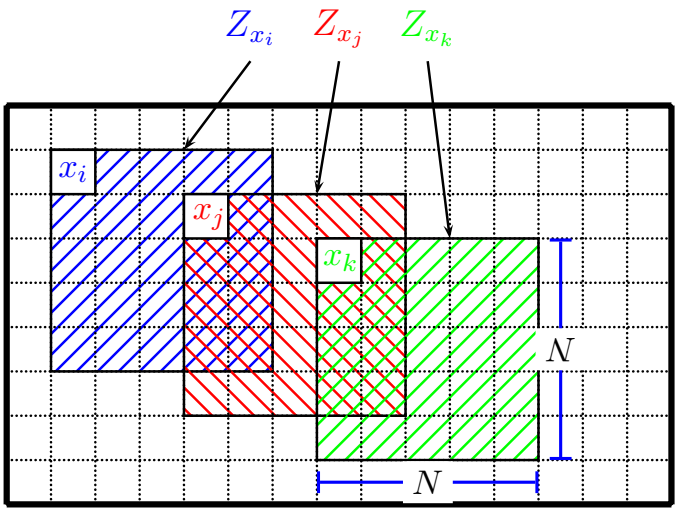


Figure 3.10: Example of overlapping blocks in a digital image. Each block Z_x is a square window of size $N \times N$ and is identified by the top-left pixel $x \in \{x_i, x_j, x_k\}$.

neighborhoods wherein the pixel is present. The critical point of this algorithm is the choice of the aggregation function, several approaches use uniform averaging, however weighted averaging is proven to yield better-quality results [28, 59] in the final estimates.

Suppose to slide a squared window (block) of fixed size $N \times N$ over the noisy signal z, then a set of M overlapping blocks $\mathbf{Z} = \{Z_{x_i}\}_{i=1}^{M}$ is produced, where each block Z_x is a subset of size $N \times N$ of the noisy signal z identified by the coordinate of the top-left corner of the block. In Figure 3.10 it is illustrated an example of three overlapping squared blocks with fixed size $N \times N$ over a generalized grid. After the windowing of the noisy image z, we apply the following denoising procedure:

i. Perform a separate two-dimensional DCT transform for each block Z_x in the set $\mathbf{Z}$;

ii. Apply a thresholding rule with threshold value λ to the transformed coefficients of the blocks;

iii. Invert the two-dimensional DCT to the thresholded coefficients to obtain a local estimate $\hat{Y}_x$ of the block.

Formally the procedure above can be defined as:

$$\hat{Y}_x = \mathcal{T}_{2D}^{-1}\left(\Upsilon_\lambda\left(\mathcal{T}_{2D}\left(Z_x\right)\right)\right), \; Z_x \in \mathbf{Z}, \tag{3.38}$$

where $\mathcal{T}_{2D}$ and $\mathcal{T}_{2D}^{-1}$ denote the forward and inverse two-dimensional DCT transform respectively, and Υ_λ is one of the threshold operator defined in Equation (3.35) or (3.36), having threshold value equal to λ.

At this point we may have multiple estimate on several pixels of z. For example, in Figure 3.10, the pixel x_j belongs to every element in the overcomplete set $\{Z_{x_j}, Z_{x_i}\}$,

thus, after the execution of the algorithm defined in Equation (3.38) we obtain two different estimates $\hat{Y}_{x_j}(x_j)$ and $\hat{Y}_{x_i}(x_j)$ that must be somehow aggregated to yield a final result $\hat{y}(x_j)$.

The simplest aggregation function for the final estimate of a given pixel x_R is the uniform average-based combination of the single denoised estimates $\hat{Y}_x(x_R)$, and it is formally defined as:

$$\hat{y}(x_R) = \frac{1}{|S_{x_R}|} \sum_{x \in S_{x_R}} \hat{Y}_x(x_R),\ x_R \in X, \tag{3.39}$$

where S_{x_R} is the set of all the coordinates of the blocks containing the pixel x_R and $|S_{x_R}|$ is its cardinality:

$$S_{x_R} = \{x \in X : x_R \in Z_x\}. \tag{3.40}$$

From Equation (3.39) it is clear that each estimate $\hat{Y}_x(x_R)$ is given an equal weight which only depends on the cardinality of the set S_{x_R}. A much more general version of Equation (3.39) gives the following weights:

$$\hat{y}(x_R) = \sum_{x \in S_{x_R}} w_x \hat{Y}_x(x_R),\ x_R \in X, \tag{3.41}$$

where w_x is the weight associated to the estimate of x_R produced by the block $\hat{Y}_x$. Obviously in Equation (3.39) the weights w_x associated to the estimates of a reference pixel x_R are all equal to the following:

$$\bar{w} = \frac{1}{|S_{x_R}|},\ \forall x \in S_{x_R}.$$

Intuitively larger weights should be given to blocks that correspond to smoother parts of the image. In fact when the transformed signal does not include singularities, or equivalently when it is regular enough, the DCT yield a sparse transform coefficients. In such cases the denoising process is simpler because the meaningful information of the transformed signal is concentrated in a small number of well-localized coefficients and thus a more accurate estimate of the signal can be provided. For this reason, when a given pixel x_R belongs to two blocks, covering an uniform area and an edge respectively, it is reasonable to weight more the first one when aggregating the two estimates. Moreover, even if all the blocks involved in the aggregation comprise of a regular area in the image, the preferable weight set is still not the uniform average combination of Equation (3.39). Hereby we review some of the possibilities that have been proposed to define adaptive weights w_x to every given block $\hat{Y}_x$ for all $x \in S_{x_R}$, such that a more reliable estimate for x_R is produced by Block DCT [59, 39, 28].

Once we obtain a local estimate $\hat{Y}_x$ of every block in a given set S_{x_R}, we should give the weights w_x a value that reflects the significance of the block $\hat{Y}_x$ evaluated

from the local DCT spectrum properties. For example, we can assign to w_x a value inversely proportional to the number of remaining non-zero coefficients of the transformed block after the application of the threshold operator Υ_λ. Formally, we can define the significant-only weight set for the final estimate of a given pixel x_R as:

$$w_x = \frac{c_x}{\mathsf{N}\left(\Upsilon_\lambda\left(\mathcal{T}_{2D}\left(Z_x\right)\right)\right)}, \forall x \in S_{x_R}, \tag{3.42}$$

where $\mathsf{N} : \mathbb{Z}^2 \rightarrow \mathbb{N}$, returns the number of remaining non-zero coefficients of the transformed block Z_x after the thresholding Υ_λ, and c_x is a constant used by the algorithm to normalize the weights:

$$\sum_{x \in S_{x_R}} w_x = 1, \forall x_R \in X.$$

As a last comment we remark that the Block DCT algorithm defined in this section still has three degrees of freedom. The first regards the choice of the threshold operator Υ_λ, and whether the threshold value λ should be constant or defined adaptively over the transformed blocks.

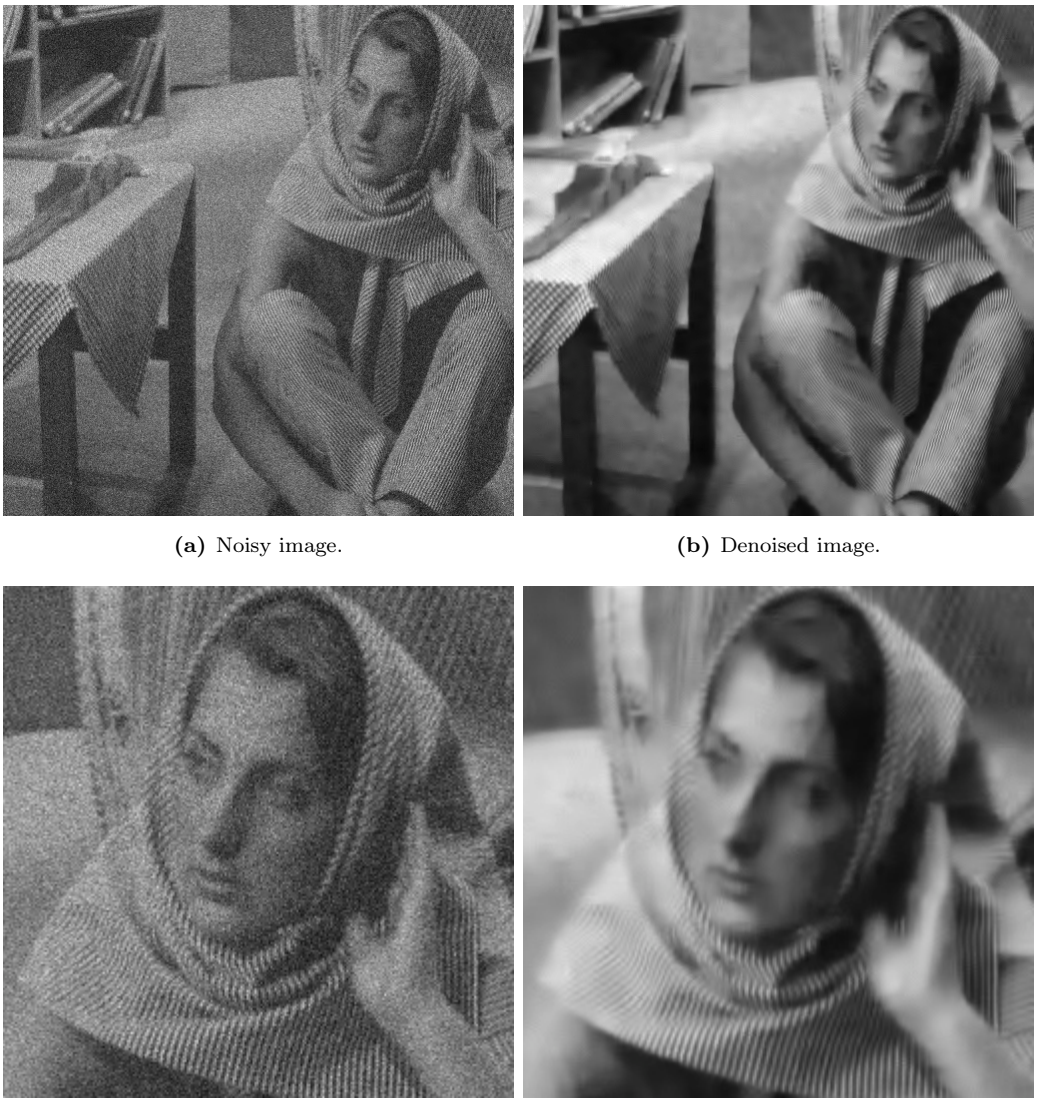
The second degree of freedom is concerned to the windowing of the signal z through the blocks Z_x. Several approaches can be used to define the behavior of the windowing, one can simply skim the window Z_x across z sliding it with constant horizontal and vertical translation Δ_H and Δ_W, until all the image has been covered. For example if the image z has a size of $H \times W$ pixels and the window size is $N_H \times N_W$, then z would be split in the following number of blocks:

$$\left\lceil \frac{H - N_H}{\Delta_H} + 1 \right\rceil \cdot \left\lceil \frac{W - N_W}{\Delta_W} + 1 \right\rceil. \tag{3.43}$$

Others, more sophisticated approaches, are oriented to the construction of overcomplete sets S_{x_R} [39] for every given block Z_{x_R} identified by the reference pixel x_R. These sets contain a series of blocks somewhat similar to the reference block Z_{x_R}, such that a denoising procedures, being advantaged by the high correlation among the blocks, produces better estimates because of the sparsity of the transform. A more extensive description of this approach used by the BM3D algorithm [39] will be delineated in Chapter 4.

Finally, the third degree of freedom considers the size and the shape of the block Z_{x_R}. Commonly the blocks are square-shaped window of dimension 8×8, however some more advanced algorithms, such as the SA-DCT [8], build a shape adaptive neighborhood for each reference pixel x_R to produce the pointwise estimate of x_R.

In Figure 3.11 is shown an example of application of the Block DCT to denoise the test image *Barbara* corrupted with additive white Gaussian noise having a standard-deviation $\sigma = 0.1$. In the implementation of the Block DCT a window of size 8×8



(a) Noisy image. **(b)** Denoised image.

(c) Noisy image (detail). **(d)** Denoised image (detail).

Figure 3.11: Denoising of the test image *Barbara* corrupted with additive white Gaussian noise with $\sigma = 0.1$ using the Block DCT algorithm.

has been slid over the image with constant $\Delta_H = \Delta_W = 1$, and the aggregation has been made using the weighted average described in Equation (3.41) using the weights defined in Equation (3.42).

4. BLOCK-MATCHING AND 3D FILTERING FOR IMAGES AND VIDEOS

This chapter provides an in-depth discussion of a denoising algorithm for digital images, called BM3D [37, 39, 64] and its extension to video denoising V-BM3D [36]. because these are the inspirational algorithms of the work proposed in this Thesis. The main idea behind both these algorithms is to group similar two-dimensional fragments extracted from an image (or frame in case of videos) into three-dimensional arrays and then transform these arrays using a separable linear three-dimensional transform. Since the noise-free signals are assumed sparse in transform domain, the algorithms can perform an effective shrinkage of the transformed spectrum and, consequently a reliable estimates of the unknown original signal. Such strategy has been experimentally proven to achieve state-of-the-art performance in terms of PSNR and subjective visual quality [39, 36] for images and videos corrupted by white Gaussian noise.

4.1 BM3D

In the previous chapter we reviewed some transform-domain denoising methods, which rely on the assumption that the original signal admits a sparse representation in a proper domain. Multiresolution methods, i.e. the Wavelet shrinkage described in Section 3.3, are particularly suited for spatially localized details, such as edges and singularities, but their performances are not as good in case of smooth transitions. Conversely, other more common orthogonal transform methods, such as the two-dimensional DCT described in Section 3.4, are not able to sparsely represent sharp transitions. Even if most of the meaningful information in natural images is mainly brought by fine details and singularities, it is still not possible to assume that a particular transform performs always better with respect to another one. In other words every two-dimensional transform can not achieve a good sparsity for all types of input signal.

The BM3D algorithm [37, 39] is based on an enhanced sparse representation in transform domain. The key ideas behind BM3D are grouping, collaborative filtering and aggregation. We assume that the noisy observation is corrupted by white Gaussian noise, thus the observation model is the same described in Section 1.2:

$$z(x) = y(x) + \eta(x),\ x \in X \subset \mathbb{Z}^2,$$

where x is a two-dimensional spatial coordinate, y is the true, unknown, image and η is i.i.d. white Gaussian noise with variance σ^2.

4.1.1 Grouping

By grouping we mean building $(d+1)$-dimensional data structures (groups) that contains similar d-dimensional fragments (blocks) of the image. When these fragments have the same shape, the formed group is called a generalized cylinder. In natural images the grouped fragments are characterized by the following types of correlations:

- intra-fragment correlations between the pixel of each fragments, which is a peculiar characteristic of natural images;
- inter-fragments correlations between the corresponding pixels of different fragments belonging to the same group, which is a consequence of the similarity between fragments.

These groups can be realized by using several approaches. Partitioning-oriented methods, such as K-means, are not adequate because of their high computational complexity, they always form disjoint clusters (groups), and the elements distant from the centroid are not well-represented by the cluster they belong to. A more effective grouping can be achieved by matching, wherein the formed groups are not necessarily disjoint. With a nonlocal approach, see Section 3.2 for details, matching finds fragments similar to a given reference one pairwise testing the similarity between the reference block and any other block located anywhere in the image. Any signal fragments is considered as a reference block, and the similarity between two blocks is computed using some distance measure, i.e. based on the ℓ^2-norm of the fragments difference.

4.1.2 Collaborative Filtering

The main motivation behind grouping is the possibility to exploit the sparsity of the group in transform domain which is due to the high similarity among the grouped fragments and, thus, a more effective coefficients shrinkage to removes the noise. In other words the denoising reveals fine details shared by the grouped fragments while preserving their individual features. This approach is referred to as collaborative filtering.

Given a group of fragments, each fragment collaborates for the filtering for all others, therefore the estimates of the single fragments can be different. The first and simplest collaborative filtering is the element-wise averaging, that is averaging between pixels at the same (relative) position with respect to the grouped fragments. This filtering is a satisfactory estimator only in case of perfectly identical grouped blocks: it does not matter how complex the patterns are or how strong the variance

of the noise is, whenever a sufficient number of grouped fragments is available, we can still obtain reasonably good estimates because the residual error is only due to the noise variance in the output which is inversely proportional to the number of averaged blocks.

Unfortunately, since natural images seldom contain perfectly identical fragments, this simple collaborative filtering based on element-wise average would be a biased estimator for grouped fragments. It is needed then an estimator that is able to produce individual estimates of similar, but not identical, grouped fragments exploiting the correlation between them. This can be done by performing a coefficients shrinkage in transform domain. In analogy to what we have seen in Section 3.3 and Section 3.4, collaborative filtering is performed with a transform domain shrinkage on each $(d+1)$-dimensional group, and thus:

i. Perform a linear $(d+1)$-dimensional transform of the group;

ii. Shrink the transformed coefficients, for example by thresholding or Wiener filtering;

iii. Invert the linear transform to obtain an estimate of all the grouped fragments.

In case of image denoising, this procedure takes advantage of both the inter- and intra-fragments correlation of the groups yielding a sparse representation, a better shrinkage, and eventually a more accurate denoising.

4.1.3 Aggregation

Each collection of d-dimensional estimates is an overcomplete representation of the original signal because, in general the estimates can overlap. Additionally we can have multiple estimates located at exactly the same coordinate, but obtained from the collaborative filtering of different $(d+1)$-dimensional group. Eventually we can expect to have overcomplete representation of the signal where a d-dimensional fragments is similar to many others. For this reason the redundancy depends both on grouping and on the particular processed signal.

To compute a final estimate of the original signal, we have to aggregate the possibly overlapping different estimates provided by every $(d+1)$-dimensional group. This aggregation is performed by a weighted averaging with adaptive weights.

4.1.4 Algorithm

As we already did in Section 3.4 let Z_x denote a square block of fixed size $N \times N$ extracted from the noisy observation z, where x is the coordinate of the top-left corner of the block. A group of blocks, that is a three-dimensional array, is denoted by $\mathbf{Z}_S$, where $S \subseteq X$ is the set of the coordinates identifying the blocks Z_x grouped in $\mathbf{Z}_S$:

$$\mathbf{Z}_S = \{Z_x : x \in S \subseteq X\}. \tag{4.1}$$

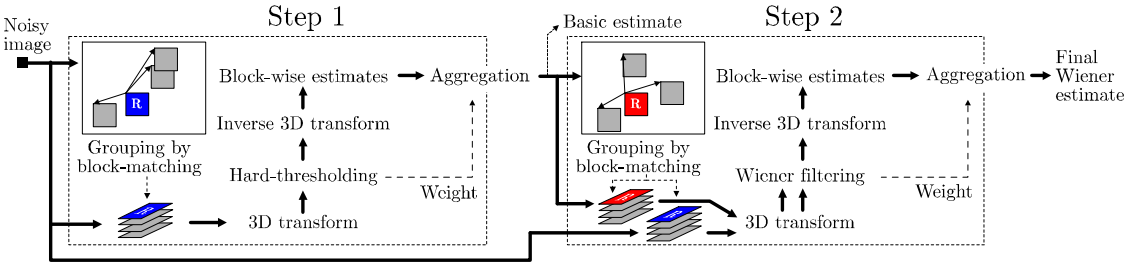


Figure 4.1: Flowchart of the BM3D denoising algorithm.

The most general description of the BM3D procedure is:

i. For each given reference block Z_{x_R}, find the blocks similar to Z_{x_R} an stack them in a three-dimensional array $\mathbf{Z}_{S_{x_R}}$, that is the group;

ii. Perform collaborative filtering on the transformed group and return the two-dimensional estimates of al the grouped blocks to their original position;

iii. If for any given pixel x, there exists multiple estimates originating from overlapping blocks, then these estimates are aggregated to produce a final result $\hat{y}(x)$.

Actually BM3D repeats the above procedure in two different steps, that differs on the implementation of the collaborative filtering. The first step uses a collaborative hard-thresholding and produces a basic estimate of the original signal y denoted $\hat{y}^{\text{basic}}$. The second step uses $\hat{y}^{\text{basic}}$ to perform a more reliable block matching and a better collaborative filtering by means of the Wiener filter[1]. In Figure 4.1 it is shown the global flowchart [39] of the two steps composing the BM3D denoising algorithm.

The first (basic) step begins with grouping by block-matching within the noisy image z. The image is processed in a sliding-window fashion, we denote the current processed block as Z_{x_R} where $x_R \in X$ and we call it as reference block. As we already said, blocks are grouped when their distance (dissimilarity) with respect to the reference block Z_{x_R} is smaller than a fixed threshold value. Distance is calculated as the ℓ^2-norm of the difference of the block. Note that ideally if we knew the original image y, the distance between the reference block Z_{x_R} and any other one Z_x would be calculated as:

$$d^{\text{ideal}}(Z_{x_R}, Z_x) = \frac{||Y_{x_R} - Y_x||_2^2}{(N^{\text{ht}})^2}, \tag{4.2}$$

where $||\cdot||_2$ is the ℓ^2-norm, Y_{x_R} and Y_x are the blocks within the image y located at x_R and $x \in X$ respectively, and the constant N^{ht} is the size of the blocks $Z_{(\cdot)}$. Observe that the superscripts "ht" stands for hard-thresholding and it is used to denote the constants employed in the first step of the algorithm. However the original image y

[1]The basic estimate $\hat{y}^{basic}$ is used in the Wiener filter as the true (pilot) energy spectrum.

is not known, thus the distance can only be calculated from the noisy blocks as:

$$d^{\text{noisy}}\left(Z_{x_R}, Z_x\right) = \frac{||Z_{x_R} - Z_x||_2^2}{\left(N^{\text{ht}}\right)^2}. \tag{4.3}$$

Note this metric depends only on the intensity similarity of the blocks, while their relative position is not considered at all, which is a fundamental aspect of nonlocal search. The counter-indication of this approach is that, being those blocks random quantities, the distance $d^{\text{noisy}}(Z_{x_R}, Z_x)$ is also a random variable following a non-central chi-squared distribution [39], and it is proved that the variance of that distance grows asymptotically with $\mathcal{O}(\sigma^4)$. Therefore, when the noise is particularly heavy or the size of the blocks is relatively small, it is very likely to obtain an erroneous grouping, dissimilar blocks might be matched as similar and similar blocks might not be grouped together.

Reliability in block matching can be improved by performing a coarse prefiltering of the blocks by applying a normalized two-dimensional linear transform followed by hard-thresholding of the resulting coefficients, formally:

$$d^{\text{ht}}\left(Z_{x_R}, Z_x\right) = \frac{\left|\left|\Upsilon'\left(\mathcal{T}_{2D}^{\text{ht}}\left(Z_{x_R}\right)\right) - \Upsilon'\left(\mathcal{T}_{2D}^{\text{ht}}\left(Z_x\right)\right)\right|\right|_2^2}{\left(N^{\text{ht}}\right)^2}, \tag{4.4}$$

where Υ' is a hard-threshold operator with threshold value λ_{2D}, refer to Section 3.3 for more details, and $\mathcal{T}_{2D}^{\text{ht}}$ denotes the two-dimensional linear transform employed in the basic step. Observe that if the transform $\mathcal{T}_{2D}^{\text{ht}}$ is orthogonal, the distance coincides with the ℓ^2-distance of the denoised block-estimates in space domain, thus there is no need to apply an inverse transformation after the threshold and the similarity can be computed directly from the spectral coefficients.

Using the metric of Equation (4.4), the result of block-matching is a set $S_{x_R}^{\text{ht}}$ containing the coordinates of the blocks having a satisfying similarity to the reference block Z_{x_R}:

$$S_{x_R}^{\text{ht}} = \left\{x \in X : d^{\text{ht}}\left(Z_{x_R}, Z_x\right) \leq \tau_{\text{match}}^{\text{ht}}\right\}, \tag{4.5}$$

where the constant $\tau_{\text{match}}^{\text{ht}}$ expresses the maximum distance accepted between two blocks. To reduce the complexity of block-matching the search for candidate matching is performed in a local neighborhood $\mathcal{N}_{x_R}^{N_S}$ of restricted size $N_S \times N_S$ centered about the currently processed coordinate $x_R \in X$ instead of the whole domain X:

$$S_{x_R}^{\text{ht}} = \left\{x \in \mathcal{N}_{x_R}^{N_{\text{S}}} : d(Z_{x_R}, Z_x) < \tau_{\text{match}}^{\text{ht}}\right\} \tag{4.6}$$

To further speed up the block-matching process, it can be used the predictive search, i.e. search windows are non-rectangular and are adaptively shaped depending on the previously matched blocks. Such windows are the union of $N_{PR} \times N_{PR}$ neighborhoods ($N_{PR} \ll N_S$) centered at the previously matched coordinates shifted by N_{step} in

the direction of processing the image. Exhaustive search in the larger $N_S \times N_S$ window is nevertheless performed every N_{FS} blocks, and if $N_{FS} = 1$ implies that only exhaustive search is used.

The set $S_{x_R}^{\text{ht}}$ is used to build the group of the (potentially overlapping) blocks similar to Z_{x_R} as:

$$\mathbf{Z}_{S_{x_R}^{\text{ht}}} = \left\{ Z_x : x \in S_{x_R}^{\text{ht}} \right\}, \tag{4.7}$$

having size $N^{\text{ht}} \times N^{\text{ht}} \times \left|S_{x_R}^{\text{ht}}\right|$, where $\left|S_{x_R}^{\text{ht}}\right|$ is the cardinality of $S_{x_R}^{\text{ht}}$. Grouping can be accelerated if its maximum cardinality $\left|S_{x_R}^{\text{ht}}\right|$ is restricted to an upper bound N_B.

At this point we need to perform the collaborative filtering of $\mathbf{Z}_{S_{x_R}^{\text{ht}}}$. In step one it is realized by hard-thresholding in three-dimensional transform domain, followed by an inverse transformation that produces a three-dimensional structure of block-wise estimates:

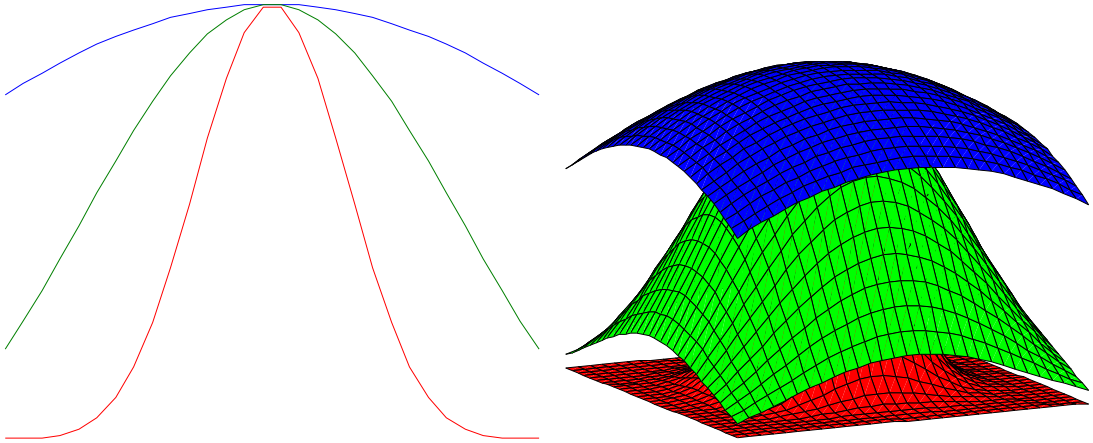
$$\hat{\mathbf{Y}}_{S_{x_R}^{\text{ht}}} = \mathcal{T}_{3D}^{\text{ht}^{-1}} \left(\Upsilon \left(\mathcal{T}_{3D}^{\text{ht}} \left(\mathbf{Z}_{S_{x_R}^{\text{ht}}} \right) \right) \right), \tag{4.8}$$

where Υ is a hard-threshold operator with threshold value $\sigma\lambda_{3D}$ and $\mathcal{T}_{3D}^{\text{ht}}$ is the adopted normalized three-dimensional linear transform. The underlying assumption is that we can obtain a reliable estimate of $\hat{\mathbf{Y}}_{S_{x_R}^{\text{ht}}}$ by hard-thresholding because the transformation $\mathcal{T}_{3D}^{\text{ht}}$ yields a sparse representation of the true signal group $\mathbf{Y}_{S_{x_R}^{\text{ht}}}$, and thus only a few elements in the transform domain carry most of the meaningful information of the original signal. Two-dimensional transformations can not achieve the same degree of sparsity reached by $\mathcal{T}_{3D}^{\text{ht}}$, for this reason the hard-thresholding (shrinkage) of the three dimensional spectrum is much more effective. Of course $\hat{\mathbf{Y}}_{S_{x_R}^{\text{ht}}}$ is composed by $\left|S_{x_R}^{\text{ht}}\right|$ block-wise estimates of the blocks contained in $\mathbf{Z}_{S_{x_R}^{\text{ht}}}$ with regard to the reference block of coordinate $x_R \in X$:

$$\hat{\mathbf{Y}}_{S_{x_R}^{\text{ht}}} = \left\{ \hat{Y}_x^{\text{ht},x_R} : x \in S_{x_R}^{\text{ht}} \right\}. \tag{4.9}$$

Recall that each filtered group $\hat{\mathbf{Y}}_{S_{x_R}^{\text{ht}}}$ for $x_R \in X$ in general may contain overlapping block-wise estimates, and additionally several sets can contain an estimate of the same block. Thus, the collection of the groups $\hat{\mathbf{Y}}_{S_{x_R}^{\text{ht}}}$ for all $x_R \in X$ is an overcomplete representation of the true image, and we need a method to aggregate the individual results to obtain a single final estimate of y. This aggregation is performed by weighted average, where the aggregation weights $w_{x_R}^{\text{ht}}$ are chosen to be inversely proportional to the total sample variance of the corresponding block-wise estimate. In such a way the noisier the estimate, the smaller the relative weight in the aggregation process. Assuming for simplicity the independence of the noise[2] in the groups $\mathbf{Z}_{S_{x_R}^{\text{ht}}}$, the total sample variance is equal to $\sigma^2 N_{\text{har}}^{x_R}$, where $N_{\text{har}}^{x_R}$ is the number of non-zero coefficients after hard-thresholding. Therefore, the weights relative

[2]The independence is satisfied only in groups that do not contain overlapping blocks. When there are overlapping blocks in the same group, the weights defined in Equation (4.10) are only loosely proportional to the corresponding total sample variance.



(a) One-dimensional Kaiser windows. **(b)** Two-dimensional Kaiser windows.

Figure 4.2: Example of Kaiser windows with increasing parameter $\beta_1 > \beta_2 > \beta_3$.

to the group of estimates $\hat{\mathbf{Y}}_{S^{\text{ht}}_{x_R}}$ are defined as:

$$w^{\text{ht}}_{x_R} = W_{2D} \cdot \begin{cases} \frac{1}{\sigma^2 N^{x_R}_{\text{har}}} & \text{if } N^{x_R}_{\text{har}} \geq 1 \\ 1 & \text{otherwise} \end{cases}, \forall x_R \in X, \tag{4.10}$$

where W_{2D} is a two-dimensional Kaiser window of size $N^{\text{ht}} \times N^{\text{ht}}$ used to reduce border effects that can appear when certain $2D$ transforms, i.e. DCT, DFT or periodized wavelet, are used.

Formally the family of one-dimensional Kaiser window is defined as:

$$W_{1D}(n) = \begin{cases} \frac{I_0\left(\beta\sqrt{1-\left(\frac{2n}{M}-1\right)^2}\right)}{I_0(\beta)}, & -\frac{M-1}{2} \leq n \leq \frac{M-1}{2} \\ 0 & \text{otherwise} \end{cases}, \tag{4.11}$$

where I_0 is the 0^{th} order modified Bessel function. In Figure 4.2 there are illustrated three Kaiser windows characterized by different parameters β, the smaller the parameter, the narrower is the resulting "bell". The β parameter of the Kaiser window allows to control the trade-off between side-lobe level and main-lobe width, or, equivalently how quickly the window approaches zero at the edges. Thus sparser decompositions of $\mathbf{Z}_{S^{\text{ht}}_{x_R}}$ result in less noisy estimates and for this reason have higher weights by Equation (4.10).

The actual aggregation is computed averaging the block-wise estimates contained

in $\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{ht}}}$, using the weights defined in Equation (4.10):

$$\hat{y}^{\mathrm{basic}} = \frac{\displaystyle\sum_{x_R \in X} \sum_{x_m \in S_{x_R}^{\mathrm{ht}}} w_{x_R}^{\mathrm{ht}} \hat{Y}_{x_m}^{\mathrm{ht},x_R}(x)}{\displaystyle\sum_{x_R \in X} \sum_{x_m \in S_{x_R}^{\mathrm{ht}}} w_{x_R}^{\mathrm{ht}} \chi_{x_m}(x)}, \forall x \in X, \tag{4.12}$$

where $\chi_{x_m} : X \to [0,1]$ is the characteristic function of the square support of a block located at $x_m \in X$, that is returns 1 if the current position x lies within the block $\hat{Y}_{x_m}^{\mathrm{ht},x_R}$ with respect to the whole image, and the estimates $\hat{Y}_{x_m}^{\mathrm{ht},x_R}$ are zero-padded outside of their support to simplify the formulation. Step one ends with the computation of the basic estimate $\hat{y}^{\mathrm{basic}}$.

Step two performs an improved grouping and collaborative filtering using the basic estimate $\hat{y}^{\mathrm{basic}}$ obtained from step one, which allows also to improve the accuracy of the block-matching. Assuming that the noise in $\hat{y}^{\mathrm{basic}}$ is relatively small, we can replace the d-distance of Equation (4.4), with a normalized ℓ^2-distance based on blocks extracted from $\hat{y}^{\mathrm{basic}}$:

$$d^{\mathrm{wie}}\left(Z_{x_R}, Z_x\right) = \frac{\left|\left|\hat{Y}_{x_R}^{\mathrm{basic}} - \hat{Y}_x^{\mathrm{basic}}\right|\right|_2^2}{\left(N^{\mathrm{wie}}\right)^2}, \tag{4.13}$$

where the superscript "wie" stands for "Wiener" and it is used to distinguish the constants used in step two. Again, a blocks similar to a given reference block located at x_R are determined as:

$$S_{x_R}^{\mathrm{wie}} = \left\{x \in X : d^{\mathrm{wie}}(Z_{x_R}, Z_x) < \tau_{\mathrm{match}}^{\mathrm{wie}}\right\}. \tag{4.14}$$

The set $S_{x_R}^{\mathrm{wie}}$ is used to form a group from the basic estimate and another one from the noisy observation, defined respectively as:

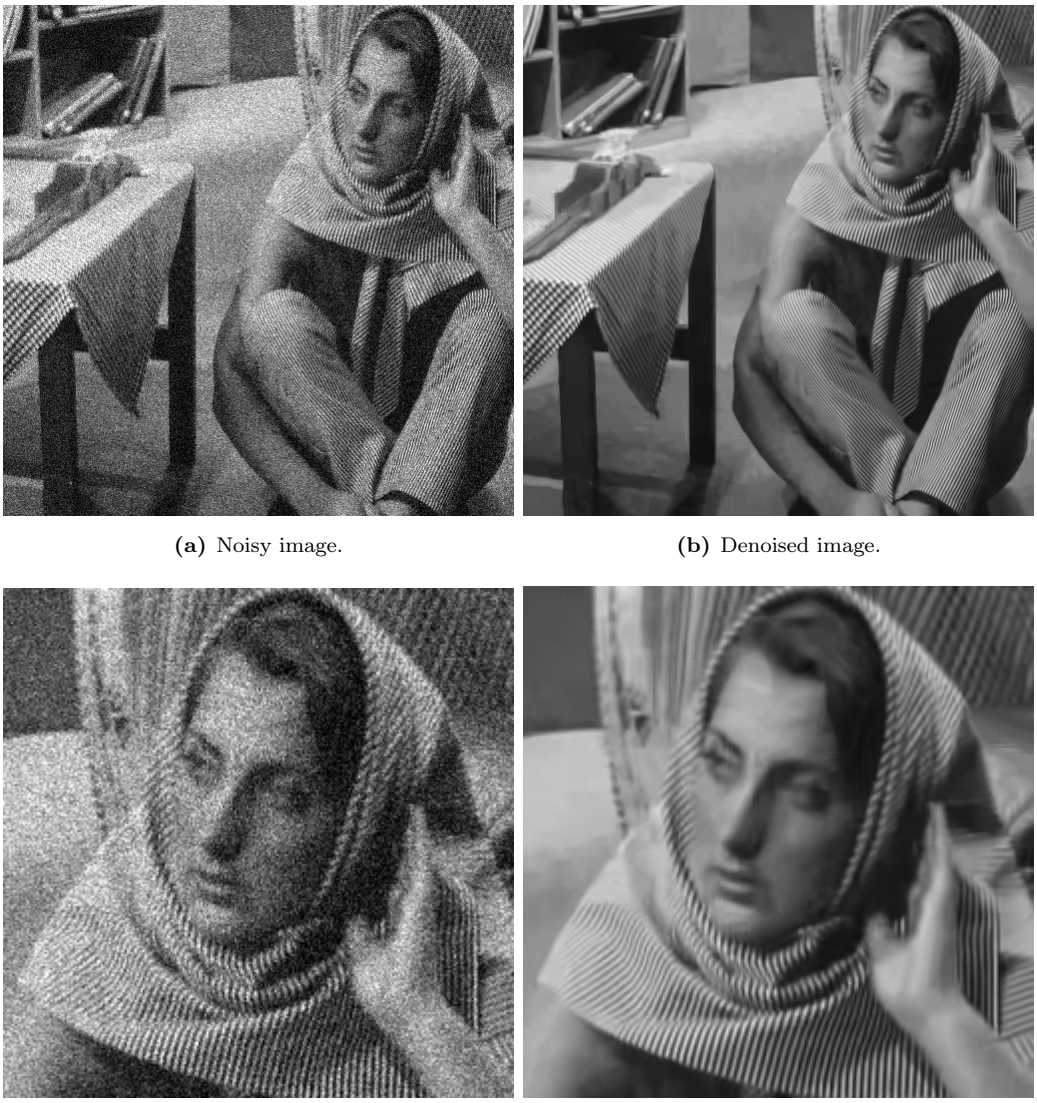
$$\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{wie}}}^{\mathrm{basic}} = \left\{\hat{Y}_x^{\mathrm{basic}} : x \in S_{x_R}^{\mathrm{wie}}\right\}, \tag{4.15}$$

$$\mathbf{Z}_{S_{x_R}^{\mathrm{wie}}} = \left\{Z_x : x \in S_{x_R}^{\mathrm{wie}}\right\}. \tag{4.16}$$

Collaborative filtering is performed using an empirical Wiener filter, whose shrinkage coefficients are computed from the energy of the three-dimensional spectrum of the basic estimate group:

$$\mathbf{W}_{S_{x_R}^{\mathrm{wie}}} = \frac{\left|\mathcal{T}_{3D}^{\mathrm{wie}}\left(\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{wie}}}^{\mathrm{basic}}\right)\right|^2}{\left|\mathcal{T}_{3D}^{\mathrm{wie}}\left(\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{wie}}}^{\mathrm{basic}}\right)\right|^2 + \sigma^2}. \tag{4.17}$$

The Wiener shrinkage coefficients obtained from Equation (4.17) are multiplied element-wisely by the three-dimensional transformed coefficients. Subsequently a



(a) Noisy image. **(b)** Denoised image.

(c) Noisy image (detail). **(d)** Denoised image (detail).

Figure 4.3: Results of the BM3D algorithm on the test image *Barbara* corrupted by white Gaussian noise with $\sigma = 20/255$.

three-dimensional inverse transform is applied to the shrunk coefficients:

$$\hat{\mathbf{Y}}_{S^{\text{wie}}_{x_R}} = \mathcal{T}^{\text{wie}^{-1}}_{3D}\left(\mathbf{W}_{S^{\text{wie}}_{x_R}} \cdot \mathcal{T}^{\text{wie}}_{3D}\left(\mathbf{Z}_{S^{\text{wie}}_{x_R}}\right)\right), \tag{4.18}$$

and the filtered group is:

$$\hat{\mathbf{Y}}_{S^{\text{wie}}_{x_R}} = \left\{\hat{Y}^{\text{wie},x_R}_{x} : x \in S^{\text{wie}}_{x_R}\right\}. \tag{4.19}$$

Using the basic estimate $\hat{y}^{\text{basic}}$ in the empirical Wiener filtering is much more effective and accurate than the simple hard-thresholding of the three-dimensional spectrum of the noisy data performed in Equation (4.8) during step one.

Overlapping block-wise estimates of a given group $\hat{\mathbf{Y}}_{S^{\text{wie}}_{x_R}}$ are aggregated using a

set of weight defined as follows:

$$w_{x_R}^{\text{wie}} = \sigma^{-2} \left|\left|\mathbf{W}_{S_{x_R}}\right|\right|_2^{-2} W_{2D},\ \forall x_R \in X, \tag{4.20}$$

where W_{2D} is a two-dimensional Kaiser window of size $N^{\text{wie}} \times N^{\text{wie}}$, $\mathbf{W}_{S_{x_R}}$ are the Wiener coefficients of Equation (4.17). The final estimate $\hat{y}^{\text{final}}$ is obtained replacing in Equation (4.12) the weights $w_{x_R}^{\text{ht}}$, the blocks $\hat{Y}_{x_m}^{\text{ht},x_R}$, and the set $S_{x_R}^{\text{ht}}$ with $w_{x_R}^{\text{wie}}$, $\hat{Y}_{x_m}^{\text{wie},x_R}$, and $S_{x_R}^{\text{wie}}$ respectively:

$$\hat{y}^{\text{final}} = \frac{\displaystyle\sum_{x_R \in X} \sum_{x_m \in S_{x_R}^{\text{wie}}} w_{x_R}^{\text{wie}} \hat{Y}_{x_m}^{\text{wie},x_R}(x)}{\displaystyle\sum_{x_R \in X} \sum_{x_m \in S_{x_R}^{\text{wie}}} w_{x_R}^{\text{wie}} \chi_{x_m}(x)},\ \forall x \in X. \tag{4.21}$$

Figure 4.3 shows an example of final estimate produced by BM3D from the noisy image *Barbara* corrupted by white Gaussian noise having $\sigma = 0.1$.

4.1.5 Three-dimensional Transforms

The three-dimensional linear transforms employed by the collaborative filtering in Equation (4.8) and (4.18) are formed by a separable decomposition of a two-dimensional linear transform $\mathcal{T}_{2D}$ with a mono-dimensional transform $\mathcal{T}_{1D}$, that is:

$$\mathcal{T}_{3D} = \mathcal{T}_{2D} \circ \mathcal{T}_{1D}. \tag{4.22}$$

Note that the transforms $\mathcal{T}_{2D}$ are composed by two mono-dimensional linear transforms as well and that the composition holds also for the inverse transformations:

$$\mathcal{T}_{3D}^{-1} = \mathcal{T}_{2D}^{-1} \circ \mathcal{T}_{1D}^{-1}. \tag{4.23}$$

In [39] it is reported the denoising performances obtained using various combination of transforms, such as DCT, DWT or biorthogonal DWT, along with the specification of the parameters λ_{2D} and N. The significant result is that the performance is mainly influenced by the choice of the $\mathcal{T}_{1D}$, that is the transform that operates in the third dimension of the grouped blocks (exploiting the inter-fragments correlation), as long as $\mathcal{T}_{1D}$ includes a constant basis function, i.e. the so-called DC-term.

In Figure 4.4 it is shown an illustrative example of the collaborative filtering [64] performed in both steps of the BM3D algorithm, formally defined in Equation (4.8) and (4.18). Every stack in the figure represents the non-zero elements of the corresponding spectrum of the transformed group. The top-left group is the result of the three-dimensional transformation applied to the groups $\mathbf{Z}_{S_{x_R}}$.

After shrinking the coefficients, either by hard-thresholding or by Wiener filtering,

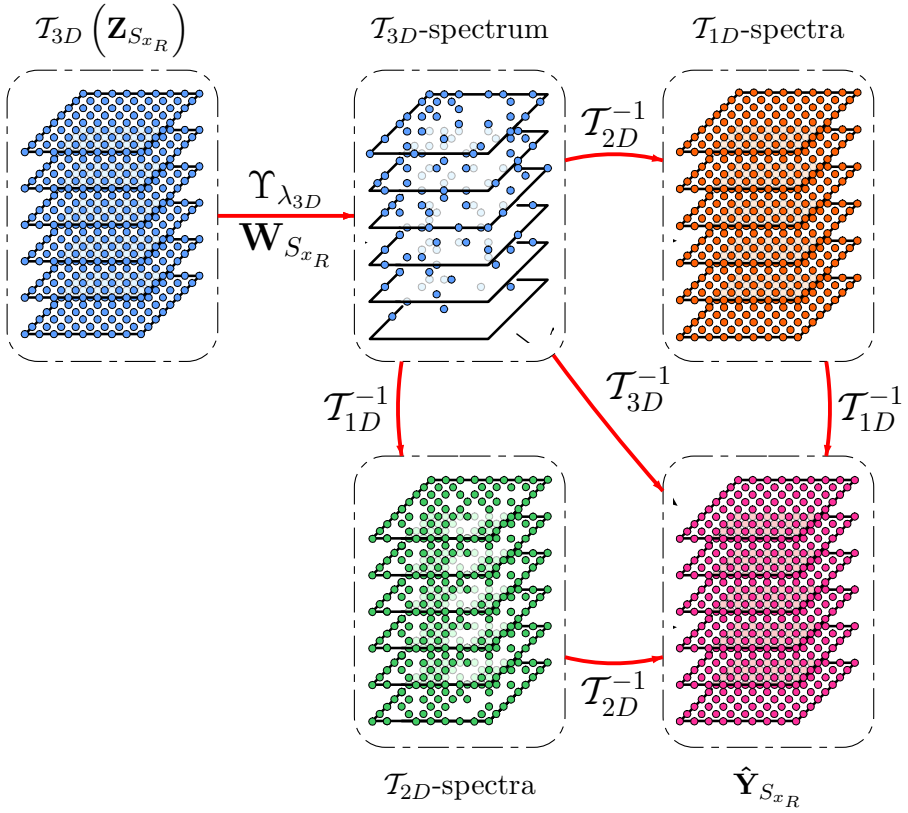


Figure 4.4: Sparsity in collaborative filtering using three-dimensional transforms. The stacks represent a collection of transformed blocks, and within each grouped block the solid circular dots illustrate the non-retained transform coefficients.

there remain only a small number of coefficients in the filtered $\mathcal{T}_{3D}$-spectrum, most of which are concentrated around the DC-term, that is at the upper-left corner of the spectrum in Figure 4.4. We can obtain the estimates in spatial domain either by inverting $\mathcal{T}_{2D}$ and then applying the $\mathcal{T}_{1D}^{-1}$ on the intermediate block estimates or vice versa. Each block in $\hat{\mathbf{Y}}_{S_{x_R}}$ is the result of a linear combination of the transformed blocks in the $\mathcal{T}_{1D}$-spectra, while each block in the $\mathcal{T}_{1D}$-spectra is $\mathcal{T}_{2D}$-sparse because it is obtained from the few coefficients of the $\mathcal{T}_{3D}$-spectrum. The coefficients of such combination are the $\mathcal{T}_{1D}$ basis elements. Observe that the $\mathcal{T}_{1D}$-spectra are seldom identically zero, that is only when the corresponding layer in the shrunk $\mathcal{T}_{3D}$-spectrum has no non-zero coefficients.

4.2 V-BM3D

In this section is presented the extension to video denoising of BM3D. The algorithm, called V-BM3D [36], shares most of the fundamental ideas of BM3D. It is also based on the assumption of highly sparse signal representation in a suitable three-dimensional transform domain. The main difference lies in the groping strategy, in fact V-BM3D employs a spatio-temporal predictive-search block-matching, similar to the techniques employed in motion estimation, in order to reduce significantly

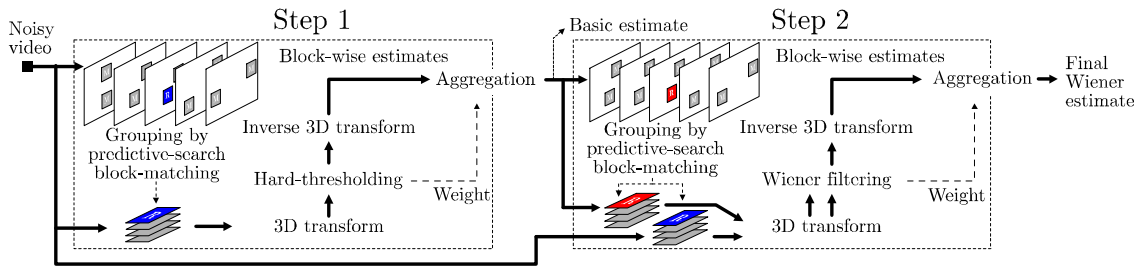


Figure 4.5: Flowchart of the V-BM3D denoising algorithm.

the complexity of the search for similar blocks.

In analogy with BM3D, once the groups of similar blocks are collected, the algorithm performs a collaborative filtering using a three-dimensional transform-domain shrinkage to obtain an estimate of the grouped blocks. If such blocks overlap each others, i.e. there is an overcomplete representation of the signal, V-BM3D apply a weighted average of the individual estimates to obtain a final estimate of the original video. The complete algorithm is composed by two steps wherein the grouping through predictive-search block-matching is followed by collaborative hard-thresholding and empirical Wiener filtering respectively.

Formally a noisy video can be defined as:

$$z(x) = y(x) + \eta(x),\ x \in X \subset \mathbb{Z}^3, \tag{4.24}$$

where y is the true video signal, $\eta(\cdot) \sim \mathcal{N}(0, \sigma^2)$ is the i.i.d. Gaussian noise corrupting the signal, as already defined in Section 1.2, and the independent variable x belongs to the three-dimensional spatio-temporal domain $X \subset \mathbb{Z}^3$, and it is composed by the following three coordinates:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ t \end{bmatrix}. \tag{4.25}$$

The coordinates occupying the first and second position, that is x_1 and x_2, are the usual two-dimensional spatial coordinates while the third component, $t \in \mathbb{Z}$, is the time index. In other words the index t of the coordinate x identifies which frame of the video the corresponding spatial coordinates (x_1, x_2) of x belongs to.

4.2.1 Predictive-Search by Block-Matching

Since V-BM3D is heavily inspired by BM3D, the analysis carried out in Section 4.1 is still well-founded, and hereafter we focus only on the peculiar concept of the video denoising algorithm. By looking at the flowchart of V-BM3D illustrated in Figure 4.5, one can note that the algorithm still consists of two steps. Analogously

to BM3D, grouping is realized by a block-matching procedure that relies on some distance measure, in particular similarity is computed using the ℓ^2-norm of the difference between two blocks, and then collaborative filtering and averaging are performed on the grouped blocks. The innovative part of V-BM3D is the localization of similar blocks, in fact given the three-dimensional nature of videos the searching for blocks to be grouped can not be restricted to the frame containing the reference block only, but it should also encompass the time dimension, to exploit inter-frames redundancy. This technique, called predictive-search block-matching [36], adopts a data-adaptive spatio-temporal three-dimensional search neighborhoods, because exhaustive search would be computationally not feasible.

The goal of predictive-search block-matching is to efficiently find similar blocks to a reference block Z_{x_R} within a spatio-temporal subdomain of the video sequence which is adaptively defined depending on the data, and limited by a temporal window of $2N_{\mathrm{FR}}+1$ frames. Note that the adaptivity of the search subdomain dimension is necessary because of the trade-off between the computational cost and the quality of grouping, which eventually affects the quality of denoising. In fact that both the cost and the quality increase with the dimension of the search subdomain.

Given a reference block Z_{x_R}, where $x_R = [\ x_1 \quad x_2 \quad t_0\ ]^T$, the adaptive predictive-search by block-matching [36] at first performs a non-adaptive search in frame t_0 in a square neighborhood $\mathcal{N}_{x_R}^{t_0,N_{\mathrm{S}}}$ of size $N_{\mathrm{S}} \times N_{\mathrm{S}}$ centered about the spatial coordinate $[\ x_1 \quad x_2\ ]^T$ of x_R. The coordinates of the blocks within this square neighborhood exhibiting a satisfying similarity, or equivalently having a distance from Z_{x_R} smaller than τ_{match} are collected in a set $S_{x_R}^{t_0} \subset \mathbb{Z}^3$ bounded in cardinality as $\left|S_{x_R}^{t_0}\right| < N_B$:

$$S_{x_R}^{t_0} = \left\{x \in \mathcal{N}_{x_R}^{t_0,N_{\mathrm{S}}} : d(Z_{x_R}, Z_x) < \tau_{\mathrm{match}}\right\}. \tag{4.26}$$

Then in each frame t_0+k, where $0 < |k| \leq N_{\mathrm{FR}}$, it is performed a predictive search procedure, which is inductively defined from the matching results of the previously processed frame $t_0 + k - \mathrm{sign}(k)$, that is the preceding frame when $k > 0$ or the subsequent frame when $k < 0$. For every matched block Z_{x_m} found in the previous frame $t_0 + k - \mathrm{sign}(k)$ located at x_m, the search window in the current frame is a neighborhood $\mathcal{N}_{x_m}^{t_0+k,N_{\mathrm{PR}}}$ of size $N_{\mathrm{PR}} \times N_{\mathrm{PR}}$ centered at the spatial coordinates of x_m. Eventually for each processed frame $t_0 + k$ the coordinates of the matched blocks are collected in a set $S_{x_R}^{t_0+k}$ defined as:

$$S_{x_R}^{t_0+k} = \left\{x \in \mathcal{N}_{x_m}^{t_0+k,N_{\mathrm{PR}}} : d(Z_{x_R}, Z_x) < \tau_{\mathrm{match}}\right\} \tag{4.27}$$

$$x_m \in S_{x_R}^{t_0+k-\mathrm{sign}(k)}. \tag{4.28}$$

Once all frames $t_0 + k$ with $k = -N_{\mathrm{FR}}, \ldots, N_{\mathrm{FR}}$ have been processed, we form a

single set $S_{x_R} \subset \mathbb{Z}^3$ as:

$$S_{x_R} = \bigcup_{k=-N_{\mathrm{FR}}}^{N_{\mathrm{FR}}} S_{x_R}^{t_0+k}, \tag{4.29}$$

that contains a bounded number, N_2, of blocks having the highest similarity with respect to the reference block Z_{x_R}. The actual order of such blocks in S_{x_R} is not relevant for the collaborative filtering [39].

Observe that the predictive-search in any frame $t_0 + k$ is performed only where it is more likely to find similar blocks, because the adaptive window predicts the position of the matched blocks in the current frame $t_0 + k$ given the positions of the same blocks in the previous frame $t_0 + k - \mathrm{sign}(k)$. This adaptivity can be interpreted as motion estimation of objects moving across frames. Therefore we can regulate the dimension of the predictive window such that $N_{\mathrm{PR}} < N_{\mathrm{S}}$.

4.2.2 Algorithm

In the first step, for each reference block Z_{x_R} with $x_R \in X$ of size $N^{\mathrm{ht}} \times N^{\mathrm{ht}}$, V-BM3D builds the set of its similar blocks as defined in Equation (4.29) by using the predictive-search block-matching (PS-BM) procedure:

$$S_{x_R}^{\mathrm{ht}} = \text{PS-BM}\big(Z_{x_R}\big), \tag{4.30}$$

and such set is subsequently used to form the three-dimensional group as:

$$\mathbf{Z}_{S_{x_R}^{\mathrm{ht}}} = \left\{ Z_x : x \in S_{x_R}^{\mathrm{ht}} \right\}, \tag{4.31}$$

then it produces the estimates of the blocks grouped in $\mathbf{Z}_{S_{x_R}}$ by collaborative hard-thresholding with threshold value $\sigma\lambda_{3D}$:

$$\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{ht}}} = \mathcal{T}_{3D}^{-1}\left(\Upsilon_{\lambda_{3D}}\left(\mathcal{T}_{3D}\left(\mathbf{Z}_{S_{x_R}^{\mathrm{ht}}}\right)\right)\right), \tag{4.32}$$

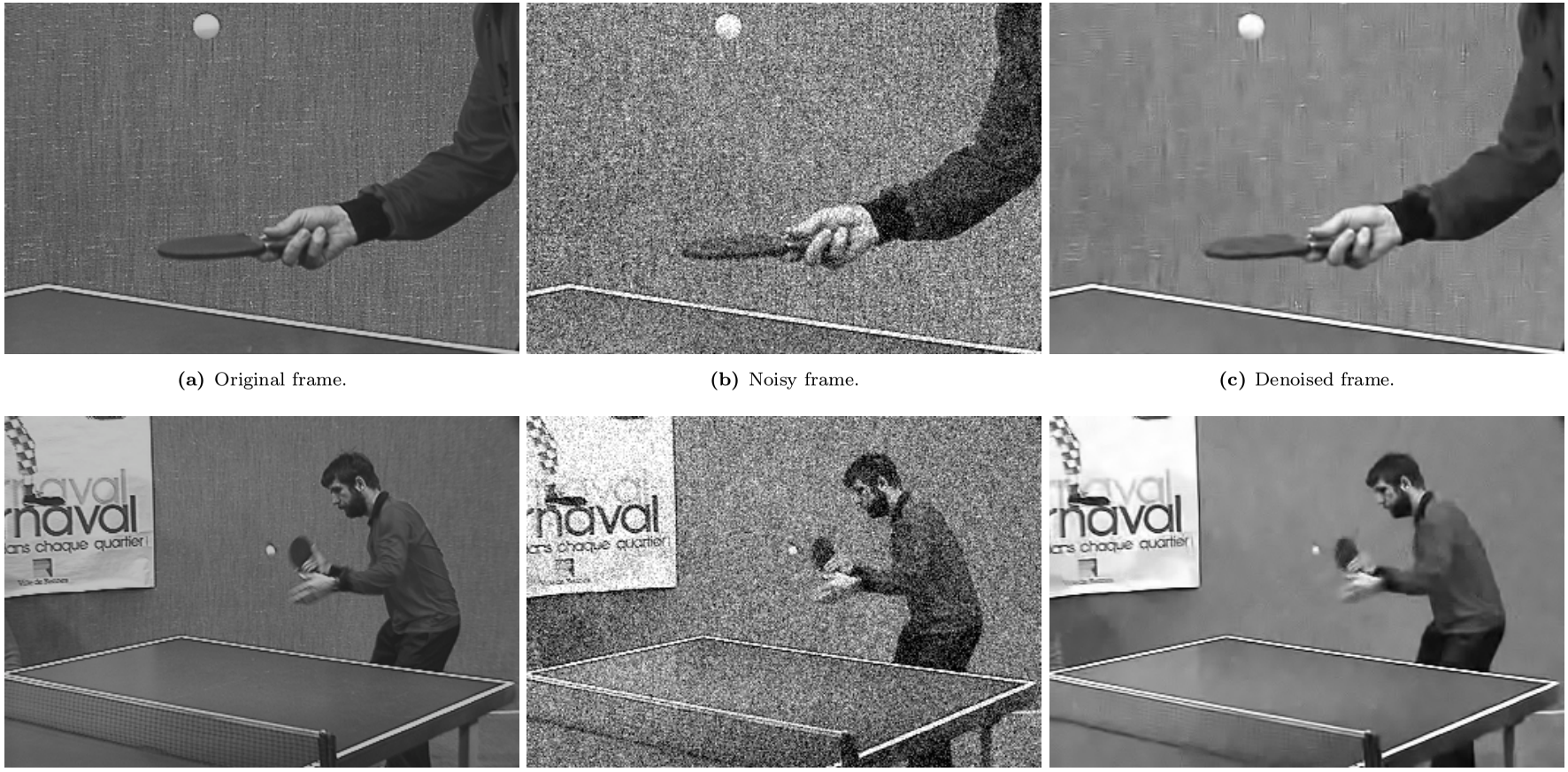
where the obtained group $\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{ht}}}$ contains the block-wise estimates:

$$\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{ht}}} = \left\{ \hat{Y}_x^{x_R} : x \in S_{x_R}^{\mathrm{ht}} \right\} \tag{4.33}$$

Therefore the basic estimate $\hat{y}^{\mathrm{basic}}$ is obtained by the aggregation formula of Equation (4.12), using the block-wise estimates in $\hat{\mathbf{Y}}_{S_{x_R}^{\mathrm{ht}}}$ for all $x_R \in X$ weighted by:

$$w_{x_R} = \frac{1}{\sigma^2 N_{\mathrm{har}}^{x_R}} W_{2D}, \tag{4.34}$$

where W_{2D} is a two-dimensional Kaiser window having the same size $N^{\mathrm{ht}} \times N^{\mathrm{ht}}$ of the blocks and it is used to reduce border effects that certain two-dimensional transforms, as DCT or DFT, may produce, and $N_{\mathrm{har}}^{x_R}$ denotes the number of retained



(a) Original frame. **(b)** Noisy frame. **(c)** Denoised frame.

(d) Original frame. **(e)** Noisy frame. **(f)** Denoised frame.

Figure 4.6: Results of the V-BM3D algorithm on two frames of the test video *Tennis* corrupted by white Gaussian noise with $\sigma = 40/255$.

coefficients after hard-thresholding $\mathcal{T}_{3D}\left(\mathbf{Z}_{S^{\text{ht}}_{x_R}}\right)$.

The second step compute the final estimate by grouping within the basic estimate $\hat{y}^{\text{basic}}$ obtained from step one as follows:

$$S^{\text{wie}}_{x_R} = \text{PS-BM}\left(\hat{Y}^{\text{basic}}_{x_R}\right), x_R \in X, \tag{4.35}$$

where $\hat{Y}^{\text{basic}}_{x_R}$ denotes the block of size $N^{\text{wie}} \times N^{\text{wie}}$ extracted from $\hat{y}^{\text{basic}}$. From the set of coordinates $S^{\text{wie}}_{x_R}$ V-BVM3D builds the following two three-dimensional arrays:

$$\hat{\mathbf{Y}}^{\text{basic}}_{S^{\text{wie}}_{x_R}} = \left\{\hat{Y}^{\text{basic}}_{x} : x \in S^{\text{wie}}_{x_R}\right\}, \tag{4.36}$$

$$\mathbf{Z}_{S^{\text{wie}}_{x_R}} = \left\{Z_x : x \in S^{\text{wie}}_{x_R}\right\}. \tag{4.37}$$

The collaborative filtering is performed with an empirical Wiener filter with shrinkage coefficients defined as in Equation (4.17):

$$\mathbf{W}_{S^{\text{wie}}_{x_R}} = \frac{\left|\mathcal{T}^{\text{wie}}_{3D}\left(\hat{\mathbf{Y}}^{\text{basic}}_{S^{\text{wie}}_{x_R}}\right)\right|^2}{\left|\mathcal{T}^{\text{wie}}_{3D}\left(\hat{\mathbf{Y}}^{\text{basic}}_{S^{\text{wie}}_{x_R}}\right)\right|^2 + \sigma^2}. \tag{4.38}$$

Thus the estimate of a noisy group $\mathbf{Z}_{S_{x_R}}$ is:

$$\hat{\mathbf{Y}}_{S^{\text{wie}}_{x_R}} = \mathcal{T}^{-1}_{3D}\left(\mathbf{W}_{S^{\text{wie}}_{x_R}}\mathcal{T}_{3D}\left(\mathbf{Z}_{S^{\text{wie}}_{x_R}}\right)\right). \tag{4.39}$$

The final estimation $\hat{y}^{\text{final}}$ is produced by aggregation of the overlapping estimates within every group $\hat{\mathbf{Y}}_{S_{x_R}}$ for each $x_R \in X$, using Equation (4.21) with the following weights:

$$w_{x_R} = \sigma^{-2}\left|\left|\mathbf{W}_{S^{\text{wie}}_{x_R}}\right|\right|^{-2}_2 W_{2D}, \forall x_R \in X, \tag{4.40}$$

W_{2D} is a two-dimensional Kaiser window of size $N^{\text{wie}} \times N^{\text{wie}}$.

5. BLOCK-MATCHING AND 4D FILTERING FOR VIDEOS

This chapter provides a formal description and an in-depth discussion of the proposed video denoising algorithm, named V-BM4D, which is constructed upon the fundamental principles of the BM3D [37, 39, 64] algorithm, described in Section 4.1.

The chapter is structured as follows. Section 5.1 gives a brief introduction on video denoising, then, in Section 5.2, we present the key ideas behind V-BM4D, the main differences from V-BM3D and a formal definition of the algorithm. Subsequently, Sections 5.3 reviews the implementation aspects of V-BM4D, with a particular emphasis to the motion estimation algorithm. Finally, supported by the experimental results reported in Section 5.5, Sections 5.7 presents the concluding remarks and discusses the future research directions.

5.1 Introduction

Digital video sequences even with the advancing of the underlying sensors technology, may suffer from grave degradations due, for instance, to noise, blur, blocking, ringing, and other acquisition or compression artifacts. Even if this mixture of disturbances is usually nonlinear, we model the aggregated effect of the noise as a i.i.d. Gaussian random variable with zero mean independent from the original underlying signal.

Video enhancement and restoration has a critical impact in applications where it is important to improve the perceived quality of images in order to facilitate the subsequent processing tasks such as video coding, analysis or interpretation. In medical imaging, for instance, image sequence restoration is needed for vascular imaging and quantification of heart dynamics [31]. The goal of a restoration algorithm is to reconstruct the original spatial and temporal correlation structure of digital image sequences by removing every irrelevant information, such as noise, while preserving the significant structural elements of the sequence.

Over the last several decades an enormous amount of research has focused on the problem of enhancing and restoring noisy images, some of which has been reviewed in Chapter 3 and Chapter 4. Clearly, these spatial methods can be seamlessly reapplied to image sequences, with the simplistic assumption that the individual frames composing the sequence are temporally independent. However without leveraging the temporal correlation between frames, we can only obtain suboptimal results,

and moreover, spatial intra-frame filters tend to introduce temporal artifacts in the restored image sequence [34]. These artifacts generally appear as oversmoothing or abrupt changes in the intensity values, as a result of different estimates of the same feature in different frames.

Thus denoising algorithm applied to image sequences should exploit both spatial and temporal (inter-frame) correlation. These filters, commonly referred to as spatiotemporal or three dimensional filters, tend to be less sensitive to non-stationarities in both spatial and temporal direction by using data-adaptive models and motion compensation. Temporal filters avoid the spatial artifacts by modeling the image sequence as a series of one-dimensional pixel trajectories that traverse the temporal axis, consequently temporal variations in the same pixel are avoided. The literature contains a plethora of video restoration or enhancement algorithms, some examples are linear temporal filters, order-statistic filters and multiresolution filters [34, 13, 17, 50, 44].

Due to the scene dynamics, a temporal signal is generally non-stationary and a filter that do not address this problem may cause oversmoothing in the produced estimate. To overcome this problem motion compensation techniques can be used, however the estimation of motion is not an easy problem and the presence of noise makes it even harder. In fact, while noise in the video disturbs motion estimators, at the same time a correct motion estimation is often used as a preprocessing step in many denoising algorithms. The ideal filter should estimate both motion and intensity values at the same time, but normally motion estimation is done before filtering. In Section 5.3.1 it is described the approach used in V-BM4D to estimate motion through concatenation of motion vectors. Motion is important in video processing because it can be used, for example, to track and identify single features moving in the scene [33, 17].

The main drawback of simple temporal filters, beside the complexity cost required by the motion estimation, is that they do not use any spatial correlation of the signal. If compared to a single image, a video is a considerably richer source of visual information, primarily due to spatiotemporal relationships between frames, that can be exploited in the interest of an effective denoising algorithm. While an image provides a single picture of a given scene, a sequence of images represents the dynamics of objects in the scene moving along time.

5.2 V-BM4D

At the moment, the most effective approach in restoring images or video sequences exploits the redundancy given by the *nonlocal* similarity between patches at different locations within the data [64]. Algorithms based on this approach have been proposed for various signal processing problems, and mainly for image denoising [44, 64, 4, 5, 43, 39, 36, 16, 40, 6, 7]. Specifically, in [4] was introduced an adaptive

pointwise image filtering strategy, called *non-local means*, where the estimate of each pixel $\mathbf{x}_i$ is obtained as a weighted average of, in principle, all the pixels $\mathbf{x}_j$ of the noisy image, using a family of weights proportional to the similarity between the regions centered at $\mathbf{x}_i$ and $\mathbf{x}_j$. So far, the most effective image denoising algorithm is BM3D [39, 64], which relies on the so-called grouping and collaborative filtering paradigm: the observation is processed in a blockwise manner and mutually similar two-dimensional image blocks are stacked into a three-dimensional group (grouping), which is then filtered through a transform-domain shrinkage (collaborative filtering), simultaneously providing different estimates for each grouped block. These estimates are then returned to their respective locations and eventually aggregated into the estimate of the image. In doing so, BM3D leverages the spatial correlation of natural images both at the nonlocal and local level, due to the abundance of mutually similar patches and to the high correlation of image data within each patch, respectively. The BM3D filtering scheme has been applied successfully to video denoising in our previous work, V-BM3D [36], as well as to several other applications including image and video super-resolution [6, 7], image sharpening [40], and image deblurring [41].

The proposed video denoising algorithm V-BM4D aims to exploit both spatial and temporal correlation in the video sequence, by grouping properly shaped nonlocal structures from the video, then performing a two-steps collaborative filtering in transform domain and finally aggregating the obtained estimates. The fundamental principle of BM3D that originally inspired V-BM4D, exploits the d-dimensionality of the original signal to form $(d+1)$-dimensional groups of similar d-dimensional elements. Thus in case of two-dimensional signals ($d = 2$), i.e. images, the algorithm groups similar two-dimensional fragments (blocks) of the image to forge three-dimensional arrays that will be consequently collaboratively filtered. The algorithm relies both on local and nonlocal characteristic in natural images, as the presence of similar patches and the high local correlation of data within the patches themselves, to forge groups that benefit from the correlation in the $(d+1)$-dimension to obtain a sparse representation of the group. The sparsity of the representation implies a more effective coefficients shrinkage and, consequently, a better final estimate. Recapitulating, given a d-dimensional noisy signal, in principle BM3D performs the following processing chain:

i. Group similar d-dimensional elements into $(d+1)$-dimensional groups;

ii. Perform a linear separable $(d+1)$-dimensional transform to each group;

iii. Shrink the transformed coefficients;

iv. Invert the linear transform to obtain an estimate of the each grouped elements;

v. Adaptively aggregate the possibly overlapping estimate of the grouped elements to their original position in the signal.

In case of videos, the principle of BM3D has not been completely fulfilled by its video extension V-BM3D, because even if the processed signal has gained a dimension, the time ($d = 3$), the algorithm is still based on two-dimensional blocks and three-dimensional groups. The temporal dimension is only exploited to extend the search window wherein block-matching is performed in the time domain, but the groups are still three-dimensional array of mutually similar blocks. Therefore, since the grouping strategy is still block-based, V-BM3D is not capable to distinguish temporal to spatial correlation within the data.

In V-BM3D, groups are three-dimensional arrays of mutually similar blocks extracted from a set of consecutive frames of the video sequence. A group may include multiple blocks from the same frame, naturally exploiting in this way the nonlocal similarity. However, it is typically along the temporal dimension that most mutually similar blocks can be found. It is well known that motion-compensated videos [30] are extremely smooth along the temporal axis and this fact is exploited by nearly all modern video-coding techniques. As shown by the experimental analysis in [16], even when motion is present, the similarity along the motion trajectories is much stronger than the nonlocal similarity existing within an individual frame. In spite of this, in V-BM3D the blocks are grouped regardless of whether their similarity is due to the tracking of motion along time or to the nonlocal spatial self-similarity within each frame. In other words, the filtering in V-BM3D is not able to distinguish between temporal versus spatial nonlocal similarity. We recognize it as a conceptual as well as practical weakness of the algorithm: as simple experiments reported in Section 5.5 demonstrate, increasing the number of spatially self-similar blocks in a group does not lead to an improvement in the final result of V-BM3D and instead it most often leads to a systematic degradation.

The core elements of V-BM4D are three-dimensional structures, called spatiotemporal volumes [2, 66, 47], formed by a sequence of blocks extracted from the noisy video following a specific trajectory, given for example by a concatenation of motion vectors along time. Once, these procedure is performed for each trajectory in the video, V-BM4D groups mutually similar volumes in four-dimensional stacks and applies a separable linear four-dimensional spatiotemporal transform. The decorellating transformation takes advantage of the intra-frame nonlocal spatial redundancy of groups and the inter-frame temporal redundancy of volumes, to produce a sparse representation of the groups in transform domain and, consequently, a more effective coefficient shrinkage leveraged by:

- local spatial correlations between the pixel of each patch (block) in the volume, which is a characteristic of almost every natural image;
- temporal correlations between (the corresponding pixels of) blocks in the volume, because of the Lambertian assumption stating that a pixel belonging to a certain object conserves the same intensity value along its trajectory [5];

- nonlocal spatial correlation between corresponding blocks of grouped volumes, because of the abundance of similar volumes in videos.

Note that, since we are dealing with a three-dimensional signal ($d = 3$), we actually construct and then collaboratively filter four-dimensional ($d + 1$) groups. The four-dimensional group spectrum is thus highly sparse, which makes the shrinkage more effective than in V-BM3D and results in the superior performance of V-BM4D in terms of noise reduction.

We introduce V-BM4D in its basic implementation as a denoising filter. Additionally, we discuss and analyze its applications to the deblocking and deringing of compressed video.

Given the high degree of redundancy, there might exists multiple estimates for each pixel of the video, that has to be aggregated in order to produce a final estimate. In general these contributions are not equal because they are derived by a different set of data (group), thus, as well as BM3D and V-BM3D, the filtering step is always followed by an adaptive aggregation function.

5.2.1 Observation Model

An image sequence (video) can be defined as a function $z : X \times T \to \mathbb{R}$:

$$z(\mathbf{x}, t) = y(\mathbf{x}, t) + \eta(\mathbf{x}, t), \tag{5.1}$$

where z is the noisy signal given by the addition of the original signal y with i.i.d. additive white Gaussian noise $\eta(\cdot, \cdot) \sim \mathcal{N}(0, \sigma^2)$. In case of signal-dependent noise, we can still use the transformation described in Section 3.1. The independent variable is a voxel $(\mathbf{x}, t)$ whose first component $\mathbf{x} \in X \subset \mathbb{Z}^2$ is the two element vector:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{5.2}$$

where the integers (x_1, x_2) identify the spatial location of the corresponding voxel. The second independent variable, $t \in T \subset \mathbb{Z}$, indicates the time index t of the frame where the spatial coordinate $\mathbf{x}$ lies on. Supported by Equation (5.1), we denote a frame, or a picture in the image sequence z, located at a given time index $t \in T$ as:

$$z(X, t) = z(\mathbf{x}, t), \ \forall \mathbf{x} \in X. \tag{5.3}$$

The aim of the proposed algorithm is to provide an estimate $\hat{y}$ of the original video y from the observed data z. The V-BM4D algorithm comprises three fundamental steps inherited from the BM3D paradigm, specifically grouping, collaborative filtering and aggregation. These steps are performed for every spatiotemporal volume of the video.

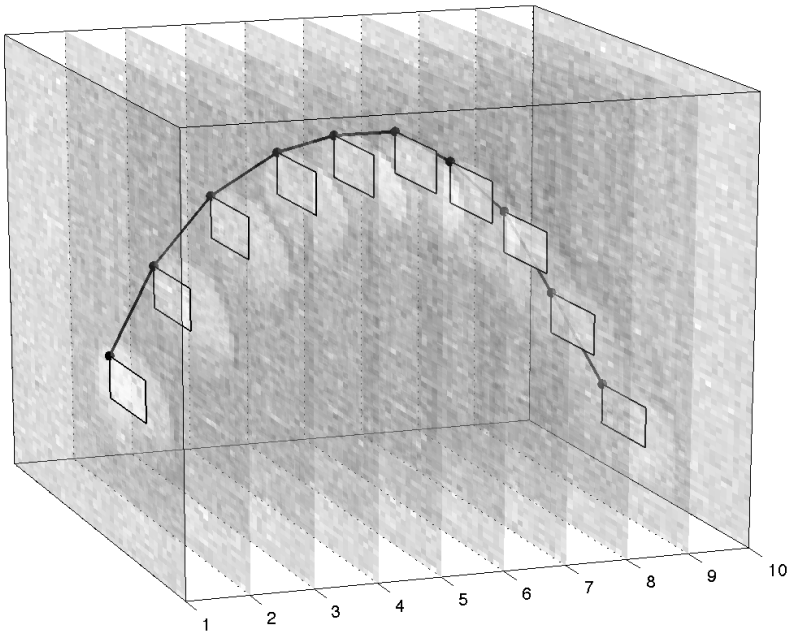


Figure 5.1: Example of trajectory built concatenating motion vectors in the noisy video z *Tennis*. The black line is a linear interpolation of the position of the ball with respect to the camera, and the interpolated points are the coordinates of the top-left corner of the blocks in the volume composing the set $\mathrm{Traj}_z(\mathbf{x}_0, t_0)$ where $(\mathbf{x}_0, t_0)$ is the spatiotemporal coordinate of the ball in the reference frame $z(X, t_0)$.

5.2.2 Spatiotemporal Volumes

Similarly to what we have done in Chapter 4, let $B_z(\mathbf{x}_0, t_0)$ denote a square block of fixed size $N \times N$ extracted, as shown by the subscript, from the noisy video z, and $(\mathbf{x}_0, t_0)$ is the spatiotemporal coordinate of a voxel identifying the top-left corner of the block within the frame $z(X, t_0)$. A three-dimensional spatiotemporal volume $V_z(\mathbf{x}_0, t_0)$ is the sequence of blocks built following a specific trajectory along time, which is supposed to adhere to the motion of $B_z(\mathbf{x}_0, t_0)$ in the scene. We denote these trajectories, independently from the technique used to calculate them, as a set of time-consecutive indices:

$$\mathrm{Traj}_z(\mathbf{x}_0, t_0) = \Big\{(\mathbf{x}_j, t_0 + j)\Big\}_{j=-h^-}^{h^+}, \tag{5.4}$$

where the elements $(\mathbf{x}_j, t_0 + j)$ are time-consecutive coordinates defining the position of the reference block $B_z(\mathbf{x}_0, t_0)$ within the neighboring frames $z(X, t_0 + j)$ as j varies in $j = -h^-, \ldots, h^+$. For the sake of notation simplicity, in this section the extents h^- and h^+ are assumed constant and equal to each other as $h^- = h^+ = h$, while the consideration on the general case are postponed to Section 5.3. In Section 5.3.1 we present an adaptive motion-vector search that can be used to retrieve the trajectories of Equation (5.4) from a noisy video z.

A volume $V_z(\mathbf{x}_0, t_0)$ is the stack of blocks belonging to the relative trajectory $\mathrm{Traj}_z(\mathbf{x}_0, t_0)$ of the reference block $B_z(\mathbf{x}_0, t_0)$. The trajectories can be either com-

puted from the noisy video (as reported in Section 5.3.1), or, when given a coded video, they can be obtained by concatenating motion vectors. In what follows we assume that, for each $(\mathbf{x}_0, t_0) \in X \times T$, a trajectory $\mathrm{Traj}(\mathbf{x}_0, t_0)$ is always given and thus the three-dimensional spatiotemporal volume in $(\mathbf{x}_0, t_0)$ can be determined as:

$$V_z(\mathbf{x}_0, t_0) = \big\{B_z(\mathbf{x}_i, t_i) : (\mathbf{x}_i, t_i) \in \mathrm{Traj}_z(\mathbf{x}_0, t_0)\big\}, \tag{5.5}$$

where the subscript z specifies that the volumes are extracted from the noisy video. Observe that the volumes $V_z(\mathbf{x}_0, t_0)$ contain exactly one block for each frame $z(X, t)$ having time index t in:

$$t \in \{t_0 - h^-, \ldots, t_0 - 1, t_0, t_0 + 1, \ldots, t_0 + h^+\},$$

and for this reason we can state that the length (cardinality) L_0 of the volume is:

$$L_0 = h^+ + h^- + 1 = 2h + 1, \tag{5.6}$$

where the second equality holds when $h^+ = h^- = h$.

5.2.3 Grouping

From Equation (5.5) the four-dimensional group can be defined as a stack $G_z(\mathbf{x}_0, t_0)$ of mutually similar volumes and constitute the nonlocal element of V-BM4D. Similarity between three-dimensional volumes is measured using a generic distance operator δ^{v}:

$$\delta^{\mathrm{v}} : \mathbb{Z}^3 \times \mathbb{Z}^3 \to \mathbb{R}^+. \tag{5.7}$$

Mutually similar volumes are determined with a nonlocal search procedure as in [39]. Specifically, let $\mathrm{Ind}(\mathbf{x}_0, t_0)$ be the set of indexes identifying volumes that, according to the distance operator δ^{v}, are similar to the reference volume $V_z(\mathbf{x}_0, t_0)$:

$$\mathrm{Ind}_z(\mathbf{x}, t) = \big\{(\mathbf{x}_i, t_i) : \delta^{\mathrm{v}}\left(V_z(\mathbf{x}, t), V_z(\mathbf{x}_i, t_i)\right) < \tau_{\mathrm{match}}\big\}, \tag{5.8}$$

where the predefined parameter $\tau_{\mathrm{match}} > 0$ controls the minimum degree of similarity among volumes with the distance δ^{v} typically being the ℓ^2-norm of the difference between two volumes.

The group associated to the reference volume $V_z(\mathbf{x}_0, t_0)$ is consequently defined as the set of volumes having coordinate in the set $\mathrm{Ind}_z(\mathbf{x}_0, t_0)$ as:

$$G_z(\mathbf{x}_0, t_0) = \big\{V_z(\mathbf{x}_i, t_i) : (\mathbf{x}_i, t_i) \in \mathrm{Ind}_z(\mathbf{x}_0, t_0)\big\}. \tag{5.9}$$

In other words $G_z(\mathbf{x}_0, t_0)$ is the group containing the volumes having distance to the reference volume $V_z(\mathbf{x}_0, t_0)$ smaller than τ_{match}. In Equation (5.9) we implicitly assume that the three-dimensional volumes are stacked along a fourth dimension,

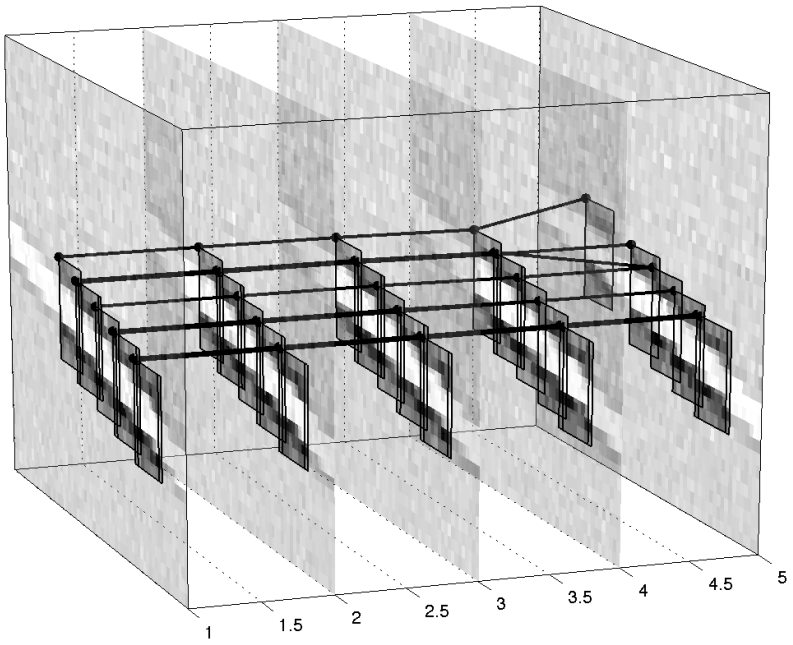


Figure 5.2: Illustration of similar spatiotemporal volumes belonging to the same group, spanning the first five frames of the noisy video *Tennis* corrupted with white Gaussian noise with $\sigma = 20/255$. Each volume is identified by one of the trajectories drawn in solid black.

hence the groups are four-dimensional data structures. Recalling Equation (5.6), the size of a group is the following:

$$|G_z(\mathbf{x}_0, t_0)| = N \times N \times L_0 \times |\mathrm{Ind}_z(\mathbf{x}_0, t_0)|\,, \tag{5.10}$$

where N is the size of the blocks, L_0 is the length of the volumes, and $|\mathrm{Ind}_z(\mathbf{x}_0, t_0)|$ is the cardinality of the set $\mathrm{Ind}_z(\mathbf{x}_0, t_0)$, that is the number of grouped volumes. In Figure 5.2 it is illustrated an example of mutually similar spatiotemporal volumes forming a typical four-dimensional group. Observe that if the distance operator δ^{v} satisfies the following condition:

$$\delta^{\mathrm{v}}(V_z(\mathbf{x}_0, t_0), V_z(\mathbf{x}_0, t_0)) = 0,\ \forall (\mathbf{x}_0, t_0)) \in X \times T,$$

it is implied that:

$$|\mathrm{Ind}_z(\mathbf{x}_0, t_0)| \geq 1. \tag{5.11}$$

In fact, in such a way, every group $G_z(\mathbf{x}_0, t_0)$ contains at least one element, which is the reference volume $V_z(\mathbf{x}_0, t_0)$, because:

$$\delta^{\mathrm{v}}(V_z(\mathbf{x}_0, t_0), V_z(\mathbf{x}_0, t_0)) = 0 < \tau_{\mathrm{match}}. \tag{5.12}$$

5.2.4 Collaborative Filtering

Collaborative filtering is realized as a shrinkage in transform domain. In the general formulation of the grouping and collaborative-filtering approach for a d-dimensional signal [39], groups are $(d+1)$-dimensional structures of similar d-dimensional elements, which are then jointly filtered. In particular, each of the grouped elements influences the filtered output of all the other elements of the group: this is the basic idea of collaborative filtering. It is typically realized with the following steps:

- Apply a $(d+1)$-dimensional separable linear transform to the group;
- Shrink the transformed coefficients in transform domain, for example by hard-thresholding or by Wiener filtering;
- Invert the $(d+1)$-dimensional transform to obtain an estimate for each grouped element.

As we said before, this mechanism can be highly effective because it exploits both the spatial and temporal correlation naturally present in image sequences. The core elements of V-BM4D are the spatiotemporal volumes ($d = 3$), and thus the collaborative filtering performs a four-dimensional separable linear transform $\mathcal{T}_{4D}$ on each four-dimensional group $G_z(\mathbf{x}_0, t_0)$, and provides an estimate for each grouped volume V_z as:

$$\hat{G}_y(\mathbf{x}_0, t_0) = \mathcal{T}_{4D}^{-1}\big(\Upsilon\left(\mathcal{T}_{4D}\left(G_z(\mathbf{x}_0, t_0)\right)\right)\big), \tag{5.13}$$

where Υ denotes a generic shrinkage operator. The filtered four-dimensional group $\hat{G}_y(\mathbf{x}_0, t_0)$ is composed of volumes $\hat{V}_y(\mathbf{x}, t)$ as:

$$\hat{G}_y(\mathbf{x}_0, t_0) = \big\{\hat{V}_y(\mathbf{x}_i, t_i) : (\mathbf{x}_i, t_i) \in \mathrm{Ind}_z(\mathbf{x}_0, t_0)\big\}, \tag{5.14}$$

with each $\hat{V}_y$ being an estimate of the corresponding volume V_y extracted from the original -unknown- video y.

5.2.5 Aggregation

The groups $\hat{G}_y$ constitute a very redundant representation of the video, because in general the volumes $\hat{V}_y$ overlap and, within the overlapping parts, the collaborative filtering provides multiple estimates at the same coordinates $(\mathbf{x}, t)$. For this reason, the estimates are aggregated through a convex combination with adaptive weights.

In particular, the estimate $\hat{y}$ of the original video is computed as follows:

$$\hat{y} = \frac{\displaystyle\sum_{(\mathbf{x}_0,t_0)\in X\times T} \left(\sum_{(\mathbf{x}_i,t_i)\in \mathrm{Ind}_z(\mathbf{x}_0,t_0)} w_{(\mathbf{x}_0,t_0)} \hat{V}_y(\mathbf{x}_i,t_i) \right)}{\displaystyle\sum_{(\mathbf{x}_0,t_0)\in X\times T} \left(\sum_{(\mathbf{x}_i,t_i)\in \mathrm{Ind}_z(\mathbf{x}_0,t_0)} w_{(\mathbf{x}_0,t_0)} \chi_{(\mathbf{x}_i,t_i)} \right)}, \tag{5.15}$$

where we assume $\hat{V}_y(\mathbf{x}_i, t_i)$ to be zero-padded outside its domain, the function $\chi_{(\mathbf{x}_i,t_i)} : X \times T \to \{0,1\}$ is the characteristic function (indicator) of the support of the volume $\hat{V}_y(\mathbf{x}_i, t_i)$, and the aggregation weights $w_{(\mathbf{x}_0,t_0)}$ are different for different groups.

The particular choice of the aggregation weights depends on the result of shrinkage in the collaborative filtering: typically the weights are defined so that the sparser is the shrunk four-dimensional spectrum $\hat{G}_y(\mathbf{x}_0, t_0)$, the larger is the weight $w_{(\mathbf{x}_0,t_0)}$ associated to it. In any case the specific weight of a group is inherited by every grouped volume in Equation (5.15).

5.3 Implementation

5.3.1 Motion Vector and Trajectory Estimation

The fundamental assumption of almost every tracking mechanism states that the properties of single features, such as intensity value or position in the scene, do not significantly change between consecutive frames, or, equivalently, that such properties have an high degree of correlation in the direction of motion. For example if we track an hand located at some position $(\mathbf{x}_i, t_i)$ in the frame $z(X, t_i)$, we expect to find a similar feature in frames $z(X, t_j)$ with $t_j = t_i + 1$ at roughly the same spatial location $\mathbf{x}_i$, unless a scene change occurs. This fact is largely exploited in video compression, but it can be also used for temporal video filtering, as it is actually one of the building blocks of V-BM4D. Even if we may assume to have the motion vectors beforehand, because for instance they are already coded in the video, as happens with the MPEG codec [30], in this section we present the motion estimation algorithm employed by V-BM4D to explicitly calculate the trajectories in case motion vectors are not given.

In our implementation, we construct trajectories by concatenation of motion vectors. The aim of the predictive spatiotemporal motion vector search performed by V-BM4D in the preprocessing step of the algorithm is to build the trajectory set $\mathrm{Traj}_z(\mathbf{x}, t)$ for each voxel $(\mathbf{x}, t)$ of the noisy video z. Observe that the computation is repeated independently for voxel $(\mathbf{x}, t)$ of the video z, which allows for an easy parallelization of the motion estimation algorithm.

5.3.1.1 Similarity Criterion

Motion of a block is generally tracked by identifying the most similar block in the subsequent (and precedent) frame, and a motion vector can be defined as the displacement of a given block between two consecutive frames. Let $z(X, t_i)$ be the frame of the image sequence at time t_i, for each pixel $\mathbf{x}_i \in z(X, t_i)$ V-BM4D extracts the corresponding square block $B_z(\mathbf{x}_i, t_i)$ of size $N \times N$. The motion estimation then it searches for the block $B_z(\mathbf{x}_j, t_j)$ in the subsequent frame $z(X, t_j = t_i + 1)$ that minimizes a given δ^{b}-distance from $B_z(\mathbf{x}_i, t_i)$. Similarity between $B_z(\mathbf{x}_i, t_i)$ and $B_z(\mathbf{x}_j, t_j)$ is measured using the normalized ℓ^2-norm of the difference between the intensity values of the two blocks:

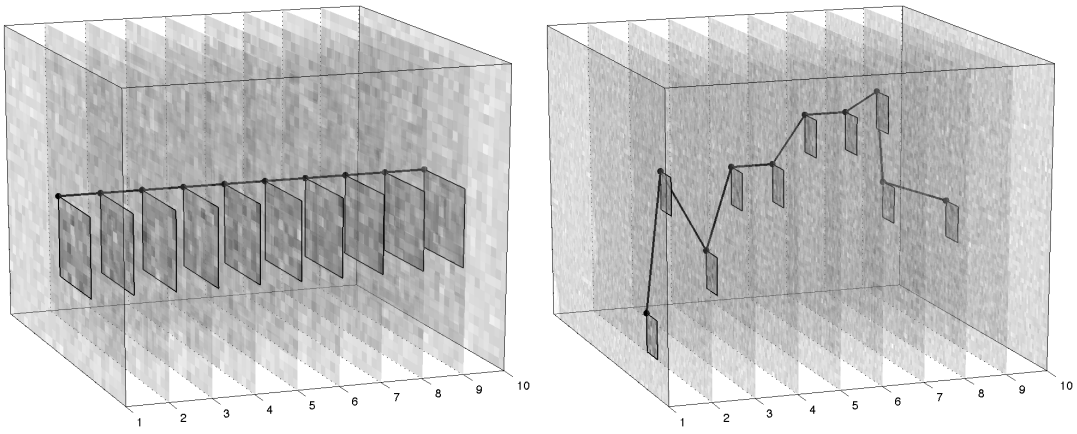
$$\delta^{\mathrm{b}}\big(B_z(\mathbf{x}_i, t_i), B_z(\mathbf{x}_j, t_j)\big) = \frac{||B_z(\mathbf{x}_i, t_i) - B_z(\mathbf{x}_j, t_j)||_2^2}{N^2}. \tag{5.16}$$

However, since we deal with noisy signals, prior information about motion smoothness can be exploited to improve the tracking. Since we suppose that motion between consecutive frames is smooth and consistent, we modify the δ^{b}-distance, to make it biased toward blocks having similar spatial coordinates. This is easily accomplished by adding to Equation (5.16) the ℓ^2-norm of the difference of the spatial coordinates of the input blocks. In particular, provided that a rough guess $\hat{\mathbf{x}}_i(t_j)$ of the future (or past) location of the block $B_z(\mathbf{x}_i, t_i)$ at the time $t_j = t_i + 1$ ($t_j = t_i - 1$) is available, we define the similarity between $B_z(\mathbf{x}_i, t_i)$ and $B_z(\mathbf{x}_j, t_j)$, through a penalized quadratic difference defined as follows:

$$\delta^{\mathrm{b}}\big(B_z(\mathbf{x}_i, t_i), B_z(\mathbf{x}_j, t_j)\big) = \frac{||B_z(\mathbf{x}_i, t_i) - B_z(\mathbf{x}_j, t_j)||_2^2}{N^2} + \gamma_d \, ||\hat{\mathbf{x}}_i(t_j) - \mathbf{x}_j||_2 \,, \tag{5.17}$$

where $\hat{\mathbf{x}}_i(t_j)$ is the predicted position of $B_z(\mathbf{x}_i, t_i)$ in the frame $z\,(X, t_j)$, and $\gamma_d \in \mathbb{R}^+$ is the penalization parameter. Whenever a rough guess $\hat{\mathbf{x}}_i(t_j)$ is not available, we consider lack of motion as the most likely condition and take $\hat{\mathbf{x}}_i(t_j) = \mathbf{x}_i$.

This spatially awareness is particularly useful in the calculation of motion for blocks lying in noisy areas having uniform intensities or repeated textures. In such cases, if we did not have any restriction on the coordinates, the similarity criteria would focus only on the intensity value, and therefore the resulting trajectory would more likely follow the random patterns of the noise instead of the true transition of the tracked feature. This is clearly illustrated in Figure 5.3, where it is shown two different volumes, whose extents are the first ten frames of the image sequence *Tennis*. Note that the scene is almost static during this time period and the initial blocks, extracted from the background wall, are the same in both experiments. In the right-most figure, we set $\gamma_d = 0$ to inhibit the coordinate norm of Equation (5.17), thus the resulting trajectory flips unpredictably within a vast portion of image. Conversely, by setting a proper γ_d, as shown in the left-most figure, the trajectory is



(a) Trajectory with $\gamma_d = 0.025$.

(b) Trajectory with $\gamma_d = 0$.

Figure 5.3: Effect of different γ_d penalties during the computation of the trajectory of a block extracted from the background texture of the sequence *Tennis* corrupted by Gaussian noise with zero mean and $\sigma = 20/255$. The scene remains fixed throughout all the experiments, thus the most genuine result would be a linear trajectory, as the one illustrated on the left.

forced to have smoother transitions between consecutive frames. However, when γ_d is too high the real motion of blocks might not be captured, because the δ^{b}-distance would prefer blocks having similar coordinates ignoring their intensity similarity.

The motion vector of $B_z(\mathbf{x}_i, t_i)$ from time t_i to $t_j = t_i + 1$ is determined by the pair $(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_j$ is the spatial coordinate of the block at time t_j that minimizes Equation (5.17) as:

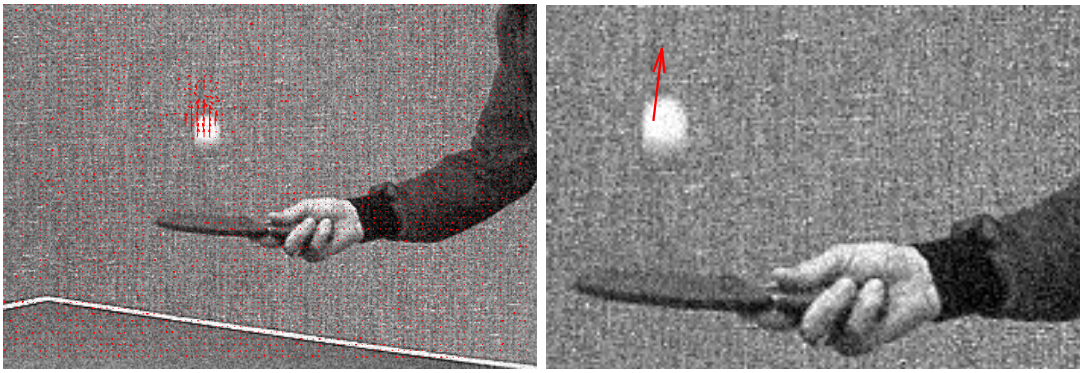
$$\mathbf{x}_j = \underset{\mathbf{x}_k \in z^{|\mathcal{N}(\hat{\mathbf{x}}_i(t_j), N_S)}}{\arg\min} \left\{ \delta^{\mathrm{b}}\big(B_z(\mathbf{x}_i, t_i), B_z(\mathbf{x}_k, t_j)\big) \right\}, \tag{5.18}$$

where $z^{|\mathcal{N}(\hat{\mathbf{x}}_i(t_j), N_S)}$ is the restriction of z to a spatial search window $\mathcal{N}$ of size $N_S \times N_S$ centered at $\hat{\mathbf{x}}_i(t_j)$:

$$z^{|\mathcal{N}(\hat{\mathbf{x}}_i(t_j), N_S)} = \big\{ \mathbf{x}_k \in z(X, t_j) : \mathbf{x}_k \in \mathcal{N} \big\}. \tag{5.19}$$

In doing so, we expect that the block $B_z(\mathbf{x}_j, t_j)$ represents the same feature contained in $B_z(\mathbf{x}_i, t_i)$ at the time $t_j = t_i + 1$. V-BM4D repeatedly applies Equation (5.18) in a chosen temporal interval $[t_0 - h^-, t_0 + h^+]$ to construct the trajectory of Equation (5.4).

The restriction applied by the search window $\mathcal{N}$ is motivated mainly by two facts, the former regards the complexity cost which would explode if we did not limit the search space and the latter regards the assumptions of smooth transition between frames. In fact, since the block $B_z(\mathbf{x}_j, t_j)$ that minimize (5.17) is likely to have similar coordinate to $B_z(\mathbf{x}_i, t_i)$, using a large search window to find $B_z(\mathbf{x}_j, t_j)$ would be pointless. Because of the penalty term $\gamma_d \left|\left|\hat{\mathbf{x}}_i(t_j) - \mathbf{x}_j\right|\right|_2$, the minimizer of Equation (5.17) cannot be far from $\hat{\mathbf{x}}_i(t_j)$. We therefore restrict the minimization



(a) Motion field. **(b)** Motion vector.

Figure 5.4: Example of motion field and motion vector of the frame $z(X, 14)$ of the image sequence *Tennis* corrupted by Gaussian noise with zero mean and standard deviation $\sigma = 20/255$. The red arrows are the motion vectors of the corresponding blocks, with lengths proportional to their magnitude.

of Equation (5.17) to a spatial search neighborhood $\mathcal{N}$ centered at $\hat{\mathbf{x}}_i(t_j)$. However is not unusual to have features, such as the ball in the test video *Tennis*, whose positions change fast and unpredictably within the scene, that would require a bigger search window. Nevertheless, we usually set the dimension N_S of the search window $\mathcal{N}$ to a relatively small value, and in case $\mathcal{N}$ is not wide enough to capture the motion of a block the trajectory is stopped. In fact, even though a minimizer for (5.18) can always be found, we interrupt the trajectory whenever the corresponding minimum distance δ^{b} exceeds the fixed parameter $\tau_{\text{traj}} \in \mathbb{R}^+$, which determines the minimum accepted similarity along spatiotemporal volumes. In doing so, V-BM4D does not build trajectories made of blocks that would potentially correlate in an unsatisfactory way.

5.3.1.2 Location Prediction

The motion vector (velocity) of a block $B_z(\mathbf{x}_0, t_0)$ in a discrete image sequence can be described by a vector $\mathbf{v}$ defined as:

$$\mathbf{v}(\mathbf{x}_0, t_0) = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \tag{5.20}$$

where the two elements $v_1, v_2 \in \mathbb{Z}$ are the horizontal and vertical spatial displacements of the tracked block between two consecutive frames. If we do not specify the spatial coordinate, we refer as $\mathbf{v}(X, t_0)$ to the motion field (or velocity filed), that is the set of all motion vectors associated to every pixel $\mathbf{x}_0$ in the frame $Z(X, t_0)$. There is an easy way to compute such motion vectors $\mathbf{v}$, if the positions of the block representing the same feature in at least two consecutive frames are known. In such cases, we can compute the motion vector by differentiating the spatial coordinate $\mathbf{x}_i$ of the block in the current frame $z(X, t_i)$ and the spatial coordinate $\mathbf{x}_{i-1}$

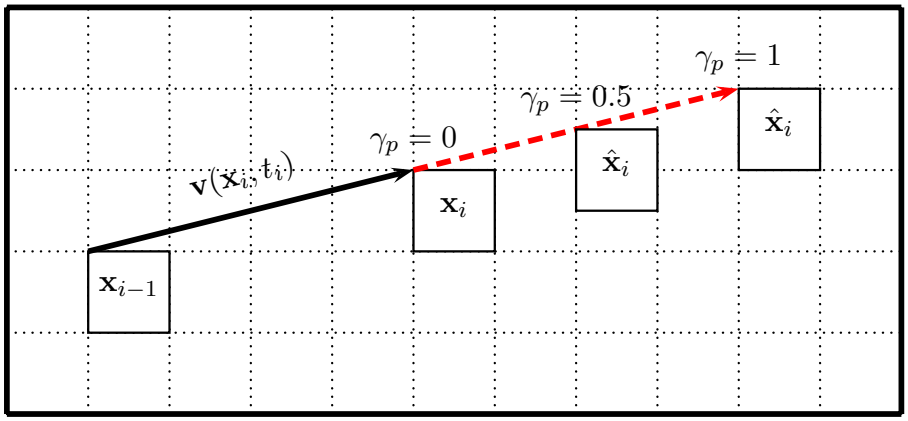


Figure 5.5: Position prediction of $(\mathbf{x}_j, t_i + 1)$ given two point $(\mathbf{x}_{i-1}, t_i - 1)$, $(\mathbf{x}_i, t_i)$ and the relative motion vector $\mathbf{v}(\mathbf{x}_i, t_i)$. As $\lambda_p \in [0, 1]$, the locus of points representing the predicted position varies is the red dashed line. In figure are provided three examples corresponding to $\gamma_p = \{0, 0.5, 1\}$, observe that when $\gamma_p = 0$, the predicted position $\hat{\mathbf{x}}_i(t_i + 1)$ is equal to $\mathbf{x}_i$ itslef.

in the previously processed frame $z(X, t_i - 1)$. Thus, as the trajectory is repeatedly constructed and the motion of the block has already been tracked at consecutive spatiotemporal locations $(\mathbf{x}_{i-1}, t_i - 1)$ and $(\mathbf{x}_i, t_i)$, we can formally define the motion vector as follows:

$$\mathbf{v}(\mathbf{x}_i, t_i) = \mathbf{x}_i - \mathbf{x}_{i-1}. \tag{5.21}$$

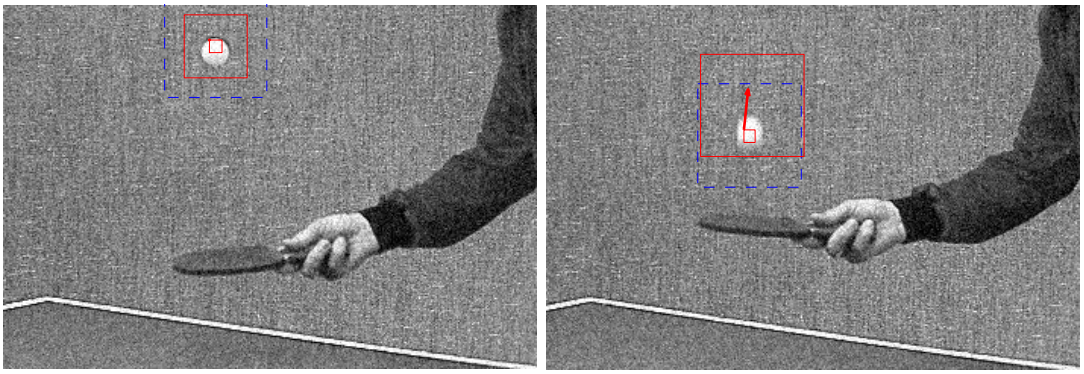
Under the assumption of smooth motion between frames, we can suppose that the motion vector $\mathbf{v}(\mathbf{x}_i, t_i)$ at any time t_i does not significantly change at the subsequent time $t_i + 1$, thus we can use Equation 5.21 to predict the position of the tracked feature in frame $z(X, t_i + 1)$. Formally, we define the guess $\hat{\mathbf{x}}_i(t_i + 1)$ as:

$$\hat{\mathbf{x}}_i(t_i + 1) = \mathbf{x}_i + \gamma_p \cdot \mathbf{v}(\mathbf{x}_i, t_i), \tag{5.22}$$

where $\gamma_p \in [0, 1]$ is the weighting factor of the prediction and it is usually a constant. Analogous prediction can be made for $\hat{\mathbf{x}}_{i-1}(t_i - 1)$, when we go backwards in time. Of course if $\gamma_p = 0$, we would have no position prediction derived from the velocity vector because the position of the tracked block at time $t_i + 1$ would be considered equal to the last known position $\mathbf{x}_i$, conversely if $\gamma_p = 1$, the position at time $t_i + 1$ would be the same as considering constant speed motion for the tracked block. Figure 5.22 provides an illustration of both such cases.

5.3.1.3 Search Neighborhood

The motion vector allows not only to predict the position of the block in the subsequent frame, but also to adapt the size of the search window $\mathcal{N}$ used in Equation (5.18). Because of the penalty term $\gamma_d \, ||\hat{\mathbf{x}}_i(t_j) - \mathbf{x}_j||_2$, the minimizer is likely close to $\hat{\mathbf{x}}_i(t_j)$. We therefore restrict the minimization of (5.18) to a spatial search neighborhood $\mathcal{N}$ centered at $\hat{\mathbf{x}}_i(t_j)$. The size $N_{PR} \times N_{PR}$ of this neighborhood can be



(a) Size adaptivity. Since the velocity vector of the tracked block has magnitude close to zero, the search window shrinks to the minimum allowed value and it is centered around the last tracked position.

(b) Position prediction. Since the motion vector has a relevant magnitude, the search window is translated accordingly to the direction of the motion vector and its size is close to the maximum value.

Figure 5.6: Illustration of the size adaptivity and position prediction of the search window in frame $z(X, 6)$ and $z(X, 14)$ of the image sequence *Tennis* corrupted by Gaussian noise with $\sigma = 20/255$. The dashed blue square represents the position of the non-adaptive search-window while the solid red square represents the adaptive one.

adapted based on the velocity (magnitude of motion vector) of the tracked block by setting:

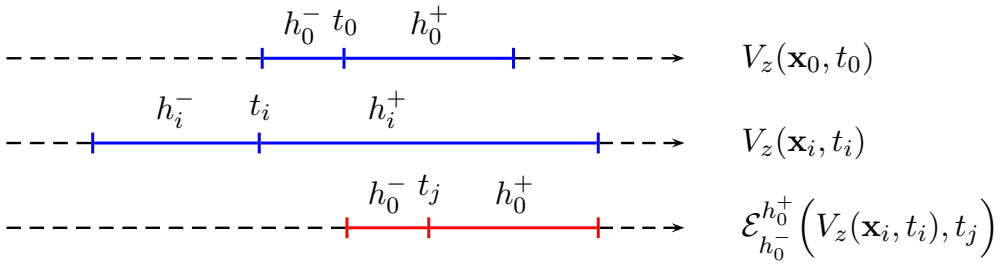
$$N_{PR} = N_S \cdot \left(1 - \gamma_w \cdot e^{-\frac{||\mathbf{v}(\mathbf{x}_i, t_i)||_2^2}{2 \cdot \sigma_w^2}}\right), \tag{5.23}$$

where N_S is the maximum size of $\mathcal{N}$, $\gamma_w \in [0, 1]$ is the maximum scaling factor of the window size and $\sigma_w > 0$ is a tuning parameter.

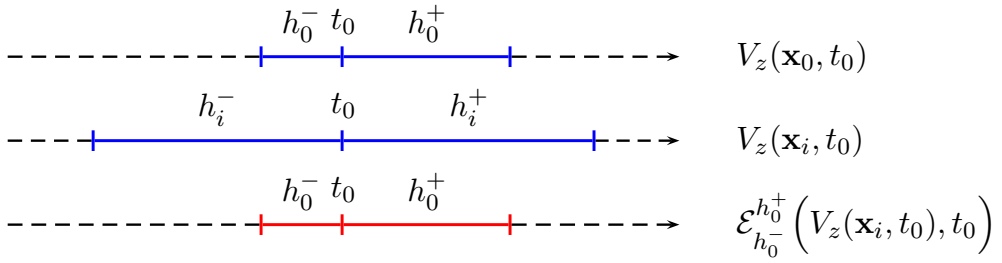
The exponential term is a Gaussian function with variance σ_w^2, therefore as the magnitude of the motion vector $\mathbf{v}(\mathbf{x}_i, t_i)$ gets higher, its norm $||\mathbf{v}(\mathbf{x}_i, t_i)||_2^2$ increases and the exponential term approaches zero accordingly to the standard deviation σ_w, and the window size converges to the maximum allowed value N_S. Conversely when the velocity is to zero, that is when the feature represented by a given block is not moving, the weight is unitary and the window size shrinks to the minimum available value $N_S(1 - \gamma_w)$.

By setting a proper value of σ_w we can control how fast the Gaussian term approaches zero, or, in other words, how permissive is the window shrinkage with respect of the velocity of the tracked block. For instance, considering the same velocity $\mathbf{v}$ for a given spatiotemporal position $(\mathbf{x}, t)$ and using two different standard deviations σ_{w_1} and σ_{w_2} in Equation (5.23), if σ_{w_1} is greater than σ_{w_2} in the former case we would obtain a smaller window than the latter, because the decay of the Gaussian function is slower.

As a final comment on the predictive spatiotemporal motion vector search, observe that we can perform the motion vector search at a given frame $z(X, t_i)$ either by matching the reference block $B_z(\mathbf{x}_0, t_0)$ or by matching the block $B_z(\mathbf{x}_{i-1}, t_i - 1)$



(a) General case. From top to bottom: the first segment represents the reference volume, the second is a volume $V_z(\mathbf{x}_i, t_i)$ centered around t_i with temporal extent (h_i^-, h_i^+) and the third is the sub-volume of $V_z(\mathbf{x}_i, t_i)$ having the same temporal extent (h_0^-, h_0^+) of the reference block centered around t_j.



(b) Synchronized case. From top to bottom: the first segment represents the reference volume, the second is a volume $V_z(\mathbf{x}_i, t_0)$ centered around t_0 with temporal extent (h_i^-, h_i^+) and the third is the sub-volume of $V_z(\mathbf{x}_i, t_0)$ synchronous in time and temporal extent to the reference volume.

Figure 5.7: Example of an application of the extraction operator $\mathcal{E}$. Each line represents a volume in the time dimension. The up-most segment corresponds to the reference volume $V_z(\mathbf{x}_0, t_0)$ centered in t_0 with temporal extent (h_0^-, h_0^+).

found in the previously processed frame at the time $t_i - 1$. The latter approach, referred to as telescopic, is adopted by default in V-BM4D.

5.3.2 Sub-volumes Extraction

The temporal extent (h^-, h^+) of the trajectories were assumed to be fixed and equal to each other for every $(\mathbf{x}, t) \in X \times T$. However because of noise, occlusions or scene changes any trajectory $\mathrm{Traj}_z(\mathbf{x}_i, t_i)$ can be interrupted at any time, as determined by the parameter τ_{traj}. Thus, if $\left[t_i - h_i^-, t_i + h_i^+\right]$ is the specific temporal extent of the trajectory $\mathrm{Traj}_z(\mathbf{x}_i, t_i)$, we can have:

$$0 \leq h_i^- \leq h, \quad 0 \leq h_i^+ \leq h, \tag{5.24}$$

where h denotes the maximum forward and backward extent of trajectories and volumes allowed in the algorithm. Note that when $h_i^- + h_i^+ = 0$ the resulting volume has unitary length because it only contains the reference block.

In the course of grouping V-BM4D faces the contemporary presence of volumes having different lengths. However, due to the separability of the transform $\mathcal{T}_{4D}$, every group $G_z(\mathbf{x}_i, t_i)$ has to be composed of volumes having equal lengths. Thus in our current implementation, every group $G_z(\mathbf{x}_0, t_0)$ is composed of volumes having

length equal to the length L_0 of the reference volume $V_z(\mathbf{x}_0, t_0)$. When a processed volume $V_z(\mathbf{x}_i, t_i)$ satisfies the following condition:

$$t_i = t_0, \tag{5.25}$$

$$h_i^- \geq h_0^-, \tag{5.26}$$

$$h_i^+ \geq h_0^+, \tag{5.27}$$

V-BM4D extracts from $V_z(\mathbf{x}_i, t_i)$ the sub-volume having temporal extent $[t_0 - h_0^-, t_0 + h_0^+]$ denoted as:

$$\mathcal{E}_{h_0^-}^{h_0^+}\Big(V_z(\mathbf{x}_i, t_i), t_0\Big), \tag{5.28}$$

where $\mathcal{E}$ is the extraction operator, h_0^- and h_0^+ are the dimension of the reference volume and t_0 is the central time index of the extracted sub-volume.

By means of the operator $\mathcal{E}$ we can extract $L_i - L_0 + 1$ different sub-volumes of $V_z(\mathbf{x}_i, t_i)$ having length L_0 but with time indices differing at most by $L_i - L_0$. Thus there are obviously many other, less restrictive, possibilities for extracting sub-volumes of length L_0 from longer volumes, however, by the above restriction we aim at limiting the complexity while maintaining a high correlation within the grouped volumes.

The synchronization constraint frees V-BM4D from the burden of which sub-volumes of $V_z(\mathbf{x}_i, t_i)$ among the $L_i - L_0 + 1$ possible ones should be grouped in $G_z(\mathbf{x}_0, t_0)$. In fact, observe that this choice at least would have required the maximization of some objective function (i.e. the similarity) with a cost of $\mathcal{O}(L_i - L_0 + 1)$ for each volume $V_z(\mathbf{x}_i, t_i)$ having $L_i > L_0$.

Figure 5.7(a) illustrates an example of application of the extraction operator in a general case where the reference volume, the volume from which we want to extract a sub-volume, and the extracted sub-volume itself have different central time indices. The synchronized case is illustrated in Figure 5.7(b).

Generally, in the grouping, volume-matching is performed through the ℓ^2-norm of the difference between time-synchronous volumes normalized with respect to their lengths:

$$\delta^{\mathrm{v}}\big(V_z(\mathbf{x}_0, t_0), V_z(\mathbf{x}_i, t_i)\big) = \frac{\Big|\Big|V_z(\mathbf{x}_0, t_0) - \mathcal{E}_{h_0^-}^{h_0^+}\big(V_z(\mathbf{x}_i, t_i), t_0\big)\Big|\Big|_2^2}{L_0}, \tag{5.29}$$

where (h_0^-, h_0^+) is the temporal extent of the reference volume $V_z(\mathbf{x}_0, t_0)$ and L_0 is the length of both the reference volume and the sub-volume extracted from $V_z(\mathbf{x}_i, t_i)$ using $\mathcal{E}$. Observe that, as shown in Equation 5.25, in practice t_i is always equal to t_0, because volume matching is performed only among volumes having the same central time index t_0, that is the situation illustrated by Figure 5.7(b).

5.3.3 Algorithm

The general procedure described in the previous section is implemented in two cascading stages, each composed of the grouping, collaborative filtering (coefficients shrinkage in transform domain) and aggregation steps.

5.3.3.1 Hard-thresholding Stage

In the first (hard-thresholding) stage, volumes are extracted from the noisy video z following the trajectories Traj_z, then groups are formed using the similarity measure δ^{v}-operator of Equation (5.29), and the predefined threshold $\tau^{\text{ht}}_{\text{match}}$. At first, the coordinates of the spatiotemporal volumes similar to $V^{\text{ht}}_z(\mathbf{x}_0, t_0)$ are collected in:

$$\text{Ind}^{\text{ht}}_z(\mathbf{x}_0, t_0) = \left\{(\mathbf{x_i}, t_i) : \delta^{\text{v}}\left(V_z(\mathbf{x}_0, t_0), V_z(\mathbf{x}_i, t_i)\right) < \tau^{\text{ht}}_{\text{match}}\right\}, \tag{5.30}$$

where δ^{v} is the distance operator defined in Equation (5.29), and $\tau^{\text{ht}}_{\text{match}}$ is the predefined threshold that controls the maximum distance among grouped volumes.

Then, after stacking the mutually similar volumes in $\text{Ind}^{\text{ht}}_z(\mathbf{x}_0, t_0)$ as shown in Equation (5.9) into a four-dimensional group $G^{\text{ht}}_z(\mathbf{x}_0, t_0)$, collaborative filtering is realized by hard-thresholding each group $G^{\text{ht}}_z(\mathbf{x}, t)$ in four-dimensional transform domain:

$$\hat{G}^{\text{ht}}_y(\mathbf{x}_0, t_0) = \mathcal{T}^{\text{ht}^{-1}}_{4D}\left(\Upsilon^{\text{ht}}\left(\mathcal{T}^{\text{ht}}_{4D}\left(G^{\text{ht}}_z(\mathbf{x}_0, t_0)\right)\right)\right), \tag{5.31}$$

where $\mathcal{T}^{\text{ht}}_{4D}$ is the four-dimensional transform and Υ^{ht} is the hard-threshold operator with threshold $\sigma\lambda_{4D}$.

The outcome of hard-thresholding stage, the basic estimate $\hat{y}^{\text{ht}}$ of the original video y, is obtained by aggregation of all the estimated groups $\hat{G}^{\text{ht}}_y$. The weights $w^{\text{ht}}_{(\mathbf{x}_0,t_0)}$ used in the convex combination (5.15) are inversely proportional to the number $N^{\text{ht}}_{(\mathbf{x}_0,t_0)}$ of non-zero coefficients in the corresponding hard-thresholded group $\hat{G}^{\text{ht}}_y(\mathbf{x}_0, t_0)$:

$$w^{\text{ht}}_{(\mathbf{x}_0,t_0)} = \begin{cases} \frac{1}{\sigma^2 N^{\text{ht}}_{(\mathbf{x}_0,t_0)}}, & \text{if } N^{\text{ht}}_{(\mathbf{x}_0,t_0)} \geq 1, \\ \frac{1}{\sigma^2}, & \text{otherwise} \end{cases}, \tag{5.32}$$

where $\sigma > 0$ is the standard deviation of the noise.

Consequently, from the general aggregation formula of Equation (5.15), the basic estimate is calculated as follows:

$$\hat{y}^{\text{ht}} = \frac{\displaystyle\sum_{(\mathbf{x}_0,t_0)\in X\times T}\left(\sum_{(\mathbf{x}_i,t_i)\in \text{Ind}^{\text{ht}}_z(\mathbf{x}_0,t_0)} w^{\text{ht}}_{(\mathbf{x}_0,t_0)}\hat{V}^{\text{ht}}_y(\mathbf{x}_i,t_i)\right)}{\displaystyle\sum_{(\mathbf{x}_0,t_0)\in X\times T}\left(\sum_{(\mathbf{x}_i,t_i)\in \text{Ind}^{\text{ht}}_z(\mathbf{x}_0,t_0)} w^{\text{ht}}_{(\mathbf{x}_0,t_0)}\chi_{(\mathbf{x}_i,t_i)}\right)}. \tag{5.33}$$

5.3.3.2 Wiener Filtering Stage

In the second (Wiener filtering) stage, new trajectories $\text{Traj}_{\hat{y}^{\text{ht}}}$ are extracted from the basic estimate $\hat{y}^{\text{ht}}$, and the grouping is performed among the volumes $V_{\hat{y}^{\text{ht}}}$ derived from the new trajectories. Volume-matching is still performed through the δ^{v}-distance, but using a different threshold parameter $\tau^{\text{wie}}_{\text{match}}$ as:

$$\text{Ind}^{\text{wie}}_{\hat{y}^{\text{ht}}}(\mathbf{x}_0, t_0) = \Big\{(\mathbf{x_i}, t_i) : \delta^{\text{v}}\left(V_{\hat{y}^{\text{ht}}}(\mathbf{x}_0, t_0), V_{\hat{y}^{\text{ht}}}(\mathbf{x}_i, t_i)\right) < \tau^{\text{wie}}_{\text{match}}\Big\}, \tag{5.34}$$

where $\tau^{\text{wie}}_{\text{match}}$ is the maximum accepted distance value of the Wiener step. Then the same volume indices $\text{Ind}^{\text{wie}}_{\hat{y}^{\text{ht}}}$ is used to construct two groups G^{wie}_{z} and $G^{\text{wie}}_{\hat{y}^{\text{ht}}}$ composed by volumes extracted from the noisy video z and from the basic estimate y^{ht}, respectively.

Collaborative filtering is hence performed using an empirical Wiener filter in $\mathcal{T}^{\text{wie}}_{4D}$ transform domain, whose shrinkage coefficients are computed from the energy of the four-dimensional spectrum of the basic estimate group $G^{\text{wie}}_{\hat{y}^{\text{ht}}}$ as follows:

$$\mathbf{W}(\mathbf{x}_0, t_0) = \frac{\left|\mathcal{T}^{\text{wie}}_{4D}\left(G^{\text{wie}}_{\hat{y}}(\mathbf{x}_0, t_0)\right)\right|^2}{\left|\mathcal{T}^{\text{wie}}_{4D}\left(G^{\text{wie}}_{\hat{y}}(\mathbf{x}_0, t_0)\right)\right|^2 + \sigma^2}, \tag{5.35}$$

Shrinkage is realized as element-by-element multiplication between the four-dimensional transform coefficients of the group $G^{\text{wie}}_{z}(\mathbf{x}_0, t_0)$ extracted from the noisy video z and the Wiener coefficients $\mathbf{W}(\mathbf{x}_0, t_0)$. Overall, we obtain the group of volumes estimates by inverting the four-dimensional transform as:

$$\hat{G}^{\text{wie}}_{y}(\mathbf{x}_0, t_0) = \mathcal{T}^{\text{wie}^{-1}}_{4D}\left(\mathbf{W}(\mathbf{x}_0, t_0) \cdot \mathcal{T}^{\text{wie}}_{4D}\left(G^{\text{wie}}_{z}(\mathbf{x}_0, t_0)\right)\right). \tag{5.36}$$

The global final estimate $\hat{y}^{\text{wie}}$ is computed using the following weights:

$$w^{\text{wie}}_{(\mathbf{x}_0, t_0)} = \frac{1}{||\mathbf{W}(\mathbf{x}_0, t_0)||^2_2\, \sigma^2}. \tag{5.37}$$

in the aggregation Equation (5.15) as:

$$\hat{y}^{\text{wie}} = \frac{\displaystyle\sum_{(\mathbf{x}_0, t_0) \in X \times T} \left(\sum_{(\mathbf{x}_i, t_i) \in \text{Ind}^{\text{wie}}_{z}(\mathbf{x}_0, t_0)} w^{\text{wie}}_{(\mathbf{x}_0, t_0)} \hat{V}^{\text{wie}}_{y}(\mathbf{x}_i, t_i) \right)}{\displaystyle\sum_{(\mathbf{x}_0, t_0) \in X \times T} \left(\sum_{(\mathbf{x}_i, t_i) \in \text{Ind}^{\text{wie}}_{z}(\mathbf{x}_0, t_0)} w^{\text{wie}}_{(\mathbf{x}_0, t_0)} \chi_{(\mathbf{x}_i, t_i)} \right)}. \tag{5.38}$$

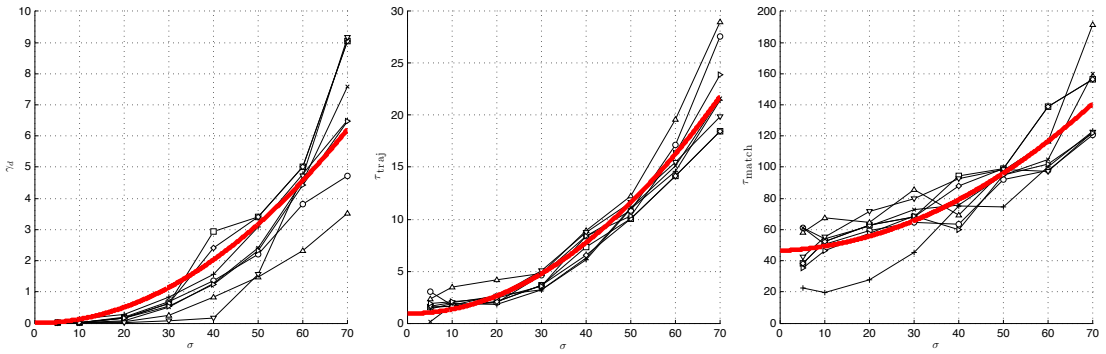


Figure 5.8: Parameters depending on σ in the hard-thresholding stage. The functions showed in red are the least squares polynomial regression on the optimum parameters obtained from the Nelder-Mead simplex direct search algorithm applied on a set of test sequences corrupted by white Gaussian noise having different values of σ. Each curve in the above plots represents the optimum value of a specific variable for a given test video as a function of σ.

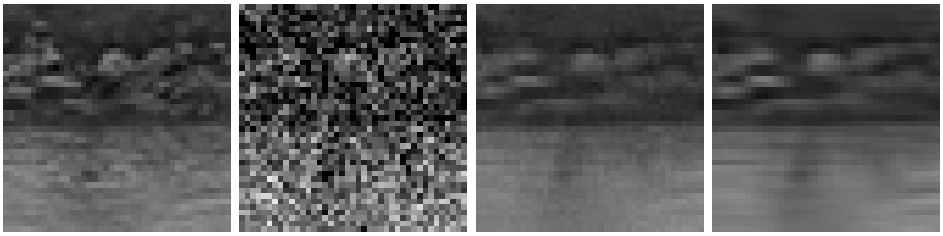


Figure 5.9: V-BM4D two stage denoising of the sequence *Coastguard*. From left to right: original video, noisy video ($\sigma = 40$), result after first stage (frame PSNR 28.58) and final estimate (frame PSNR 29.38).

5.3.4 Settings

The parameters involved in the motion estimation and in the grouping, that is γ_d, τ_{traj} and τ_{match}, vary with σ. Intuitively, the larger is σ, the bigger the thresholds controlling blocks and volumes matching become, in order to compensate the effects of the noise. Thus, in order to approximate the behavior of such non-constant parameters with respect to σ, we used the Nelder-Mead simplex direct search algorithm [48, 35] on V-BM4D in a multivariate space to find the optimum value of the triplet (γ_d, τ_{traj}, τ_{match}) for eight test video corrupted by i.i.d. white Gaussian noise having eight different value of σ, ranging from 5 to 70. Subsequently, we approximate the behavior of the three parameters as a function of σ using a least-squares quadratic polynomial regression. After the terms having negligible coefficients have been dropped, the resulting fit is

$$\gamma_d(\sigma) = 0.0013 \cdot \sigma^2, \tag{5.39}$$

$$\tau_{\text{traj}}(\sigma) = 0.0043 \cdot \sigma^2 + 0.91, \tag{5.40}$$

$$\tau_{\text{match}}(\sigma) = 0.017 \cdot \sigma^2 + 0.12 \cdot \sigma + 46.39. \tag{5.41}$$

The above functions, as well as the points used in the regression, are shown

in Figure 5.8: experimentally they were found to be a good approximation of the optimum (γ_d, τ_{traj}, τ_{match}). Note that during the second stage such parameters can be considered constants independent to σ, because in the processed sequence $\hat{y}^{\text{ht}}$ the noise is considerably lower than in the observation z; this is evident looking at the second and third image of Figure 5.9.

5.4 Deblocking

The ever increasing demand for the storage and transmission of digital videos motivated the research towards compression techniques in order to reduce the amount of information required to represent the video content by removing its spatial, temporal and visual redundancies. Transform-based data compression is undoubtedly the most used technique in both image and video coding application: specifically, popular video compression techniques, such as MPEG-4 [1] or H.264 [63] make use of the block-based discrete cosine transform, because of its near-optimal energy compaction property. The typical DCT compression scheme consists in the division of the input image into small block of size $N \times N$ (typically $N = 8$), then each block is transformed in DCT domain to apply a scalar quantization on the DCT coefficients. Usually, because of the local correlation, the energy of the original signal is concentrated in few coefficients, thus the DCT block tend to be quite sparse after quantization.

Even though transform-based methods can in general compress data almost without introducing any significant artifacts, at low bitrates they may suffer from severe compression artifacts such as blocking, ringing, mosquito and flickering. Blocking artifacts are mainly introduced by the coarse quantization of the block-based DCT error coding and by the motion compensation prediction, and they manifest themselves as staircase noise along oblique or curved edges, grid noise in monotone regions (which mainly attracts human attention [69]) checkerboard effect in textures, and mosquito temporal noise across different frames of the video. The primary source of blocking artifacts is the fact that each block is coded independently by its neighbors, thus discontinuities can occur in the decoded video because the correlation between pixels at the border of neighboring blocks is lost during the compression, as can be seen in the blocky frames in Figure 5.13.

A deblocking filter can be used either as post filter or as a loop filter: the former operates on the display buffer outside the coding loop, and thus would force the decoder to perform identical filtering in order to stay in synchronization with the encoder, making extremely hard its integration with the existing standards. On the other hand, post filters aim to reduce blocking artifacts at the decoder side: they only require the decoded image/video, hence they are independent from the used coding standard.

A large number of deblocking filter have been recently proposed. Global low-pass

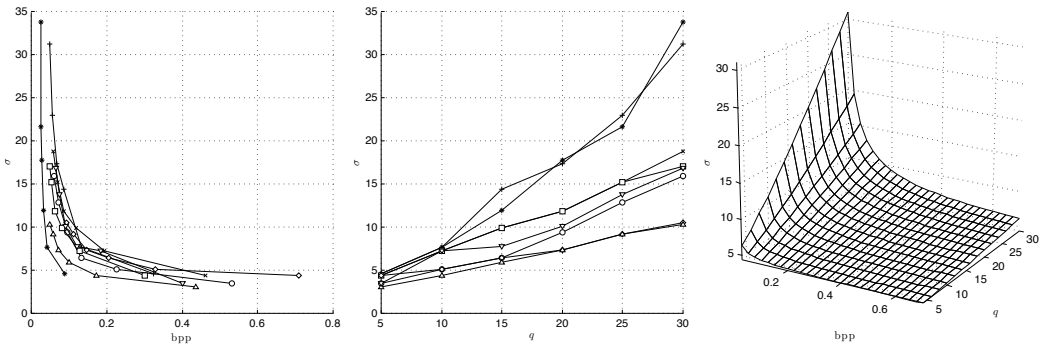


Figure 5.10: From left to right: optimum σ values for deblocking a test set of compressed videos plotted against the bit-per-pixel rate and quantization parameter q; adaptive value of σ as the function $\sigma(\text{bpp}, q)$ given in Equation (5.42).

filtering is the most intuitive and simple approach, but it has the obvious shortcoming of blurring textures and fine details. To reduce blocking artifacts without significantly degrading valid high-frequency information, several space-variant adaptive filtering algorithms have been proposed. For example techniques based on the reapplication of JPEG [55], which however tend to excessively blur small edges and textured regions. A similar approach uses an adaptive weighted averaging, taking into account the masking effect of the human visual system (HVS) [62], but, even if detailed and edge regions are well restored, it does not filter effectively flat regions. Fuzzy filter based deblocking methods [67] classify local image regions into four classes based on activity (strong edge, weak edge, texture and smooth), then apply an adaptive linear filter with variable support to the image locally: only flat regions, but not the detailed ones, are satisfyingly restored. Projection onto convex sets (POCS) methods [29, 15] assume that the original image is highly correlated, that is the local correlation among neighboring blocks are similar to the local characteristics of the single blocks: thus deblocking is performed dropping the global high-frequency components of the decoded image, significantly different from the local ones. One of the best-performing method, the shape-adaptive DCT (SA-DCT) [8] filters the image using patches of arbitrary shape extracted through the Local Polynomial Approximation - Intersection of Confidence Intervals (LPA-ICI) in a pointwise adaptive manner. Other methods are spatial block boundary filter [32], statistical modeling methods [61] or adaptive shifted thresholding [65, 26].

Our approach treats the blocking artifacts as additive Gaussian noise [8], therefore we model the deblocking problem similarly as in (5.1), where y is the original video, z the compressed video and η represents the compression artifacts. In what follows, we focus our attention on MPEG-4 compressed videos.

In order to use V-BM4D as a deblocking filter, we need to provide an appropriate value for σ given any compressed video. To accomplish this, we have first identified the optimum value of σ for each video sequence from a standard test-set at various compression rates. Figure 5.10 shows these optimum values plotted

against the average bit-per-pixel (bpp) rate of the compressed video (Figure 5.10 left) and the parameter q that controls the quantization of the block-transform coefficients [1] (Figure 5.10 center). These plots suggest that a power law can explain the dependency of the optimum value of σ on both the bpp rate and quantization parameter. Hence, we fit such bivariate function to the optimum values via least-squares regression, obtaining the adaptive value of σ for the V-BM4D deblocking filter as

$$\sigma(\text{bpp}, q) = 0.02 \cdot q^{1.25} \cdot \text{bpp}^{-0.79} + 4.29. \tag{5.42}$$

The function $\sigma(\text{bpp}, q)$ is shown in Figure 5.10 (right). The constant term 4.29 has been set to be different than zero because, in this way, the resulting equation fits the data with a lower average error. Let us observe that both the bpp and q parameters are easily accessible from an MPEG-4 encoded video. Note that in MPEG-4 the parameter q ranges from 2 to 31, where higher values correspond to a coarser quantization and consequently lower bitrates. As a matter of fact, when q increases and/or bpp decreases, both the optimum and the adaptive σ increase, in order to effectively cope with stronger blocking artifacts. Clearly, a much larger σ could result in oversmoothing, while much smaller values may not suffice for effectively reducing the compression artifacts. While in this paper we mostly deal with short test sequences and thus compute the bpp as the average rate over the whole sequence, we argue that in practice this rate should be computed as the average over a limited set of frames, namely the so-called group of pictures (GOP) built around each intra-coded frame.

Figure 5.10 illustrates the final form of (5.42), as well as the two sets of points used in the approximation, separately represented each in its own domain. As can be seen, q is loosely linear and it has a quite large variability at the highest compression level, while bpp is inversely proportional to σ and exhibits a consistent behavior among the whole set of test videos. As a matter of fact, when q increases and/or bpp decreases, the estimated σ enlarges in order to effectively cope with stronger blocking artifacts. Intuitively, larger σ would result in an oversmoothing filter, while lower values would not significantly reduce the compression artifacts. We chose to estimate σ using both parameters because, depending on the content, different videos can results in different bpp for any fixed q failing to effectively capturing the actual impact of the artifacts on the decoded video.

Observe, the V-BM4D filter can be applied to video compressed by encoder other than MPEG-4, because the q parameter can be also provided either as a subjective quality metric on the input video or as an objective measurement [70] on the impairing artifacts to be filtered out.

5.5 Experimental Results

In this section we present the experimental results obtained with a C/MATLAB implementation of the V-BM4D algorithm, and we compare it against V-BM3D, as it represents the state of the art in video denoising. Observations z are obtained by synthetically adding Gaussian noise to greyscale image sequences, according to (5.1). The denoising performance is measured using the PSNR as a global measure for the whole processed video:

$$\mathrm{PSNR}(\hat{y}) = 10 \log_{10} \left(\frac{255^2}{|X|^{-1}|T|^{-1} \sum_{(\mathbf{x},t)\in X\times T} \big(y((\mathbf{x},t)) - \hat{y}(\mathbf{x},t)\big)^2} \right), \qquad (5.43)$$

where $|X|$ and $|T|$ stand for the cardinality of X and T, respectively. Additionally, in order to provide a better experimental validation, we compare the performance of V-BM4D and V-BM3D also by means of the MOVIE[58] index. Briefly, MOVIE is a recently introduced video quality assessment (VQA) metric which is found to be more reliable than PSNR because it evaluates space, time and jointly space-time video quality, closely mimicking the human visual perception judgement.

The transforms employed in the collaborative filtering are similar to those in [39, 36]: in the hard-thresholding stage $\mathcal{T}_{4D}^{\mathrm{ht}}$ is a four-dimensional separable composition of one-dimensional biorthogonal wavelet in both spatial dimensions, one-dimensional DCT in the temporal dimension, and one-dimensional Haar wavelet in the fourth (grouping) dimension while, in the Wiener filtering stage, $\mathcal{T}_{4D}^{\mathrm{wie}}$ uses a two-dimensional DCT for the spatial dimension. Note that, because of the Haar transform, the cardinality M of each group must be a power of 2, thus we restrict M to be the largest power of 2 smaller than or equal to the minimum value between the original cardinality of the groups and M itself. In order to reduce the complexity of the grouping phase, we restrict the search of similar volumes within a $N_G \times N_G$ neighborhood centered around the coordinates of the reference volume, moreover, to lighten the computational complexity of the grouping, a step of $N_{\mathrm{step}} \in \mathbb{N}$ pixels in both horizontal and vertical directions separates each processed volume. Notwithstanding the use of a $N_{step} > 1$, the trajectory of every possible volume in the video must be computed beforehand, because any volume is a potential candidate element of every group. Additionally, similarly to what described in Chapter 4, a Kaiser window of parameter β is used in the aggregation step in order to alleviate the boundary effect of the weighted block-wise estimates.

Table 5.1 provides a complete overview of the parameters setting in V-BM4D; the table is organized in two sets, corresponding to the standard and the high-quality profile. The high-quality profile is mainly characterized by the increase of h and M, which control the temporal extent and the groups cardinality, respectively, and by the reduction of the processing step N_{step}.

Table 5.1: Parameter settings of V-BM4D under the standard and high-quality profile for the first (hard-thresholding) and the second (Wiener-filtering) stage. Where reported $f(\sigma)$, the corresponding parameter varies according to the noise. The apex S, T and G on the transforms $\mathcal{T}$ stands for spatial, temporal and grouping dimension, respectively.

Parameter	Standard Profile		High-quality Profile	
	Hard thr.	Wiener filt.	Hard thr.	Wiener filt.
N	8	7	8	
N_S	11		17	
N_G	19	27	34	
h	4		7	
M	32	8	32	
β	2	1.3	2	1.3
λ_{4D}	2.7			
γ_p	0.3			
γ_w	0.5		0.45	
σ_w	1			
N_{step}	6	4	3	
γ_g	$f(\sigma)$	0.005	$f(\sigma)$	0.005
τ_{traj}	$f(\sigma)$	1	$f(\sigma)$	1
τ_{match}	$f(\sigma)$	13.5	$f(\sigma)$	13.5
$\mathcal{T}_{2D}^{\text{S}}$	2-D Bior1.5		2-D DCT	
$\mathcal{T}_{1D}^{\text{T}}$	1-D DCT			
$\mathcal{T}_{1D}^{\text{G}}$	1-D Haar			

The comparison against V-BM3D[1] is carried out using the two different set of parameters reported in Table 5.1. First, in the standard profile we compare against the implementation with default parameters as described in [36]. Second, in the high-quality profile, we modify V-BM3D by reducing its N_{step} to 3 and by increasing its temporal extent N_{FR} from 9 to 15 frames and its group dimension N_2 to 16. Further details on the mentioned parameters N_B, N_{FR} and N_2 can be found in [36]. We decorate the names of the V-BM4D and V-BM3D algorithm by postponing the subscript hq to distinguish the high quality profile. Table 5.2, 5.3 and Table 5.4, 5.5 compare the denoising performance of V-BM3D and V-BM4D under the standard and high quality profile in terms of PSNR and MOVIE, respectively. In our experiments V-BM4D and V-BM3D are applied to a set of standard video sequences corrupted by white Gaussian noise with increasing standard deviation σ, which is assumed known. Further details concerning the original sequences, such as the resolution and number of frames, are shown in the header of the table.

As one can see, V-BM4D outperforms V-BM3D in nearly all the experiments, with PSNR improvement of almost 1 dB, under both the standard and high quality profile. However, it is more interesting to observe that the spatio-temporal correlation in

[1]MATLAB code at `http://www.cs.tut.fi/~foi/GCF-BM3D/`.

Table 5.2: Comparison between the PSNR (dB) outputs obtained from the proposed V-BM4D algorithm and the V-BM3D algorithm under both the standard and the high quality (hq) profiles. The test sequences are corrupted by i.i.d. Gaussian noise with zero mean and different standard deviations σ. The table reports the experiments having $\sigma \leq 20$.

σ	Video:	*Salesm.*	*Tennis*	*Fl. Gard.*	*Miss Am.*	*Coastg.*	*Foreman*	*Bus*	*Bicycle*
	Res.:	288×352	240×352	240×352	288×360	144×176	288×352	288×352	576×720
	Frames:	50	150	150	150	300	300	150	30
5	V-BM4D	41.03	38.87	37.13	42.11	39.12	**40.24**	38.27	**40.97**
	V-BM3D	40.44	38.47	36.46	41.58	38.25	39.77	37.55	40.89
	V-BM4D$_{\text{hq}}$	**41.43**	**38.93**	**37.18**	**42.18**	**39.23**	40.22	**38.48**	40.88
	V-BM3D$_{\text{hq}}$	41.09	38.63	36.43	41.80	38.44	40.09	37.68	40.94
10	V-BM4D	37.49	35.00	32.64	40.18	35.18	36.71	34.01	37.54
	V-BM3D	37.21	34.68	32.11	39.61	34.78	36.46	33.32	37.62
	V-BM4D$_{\text{hq}}$	**37.99**	**35.14**	**32.71**	**40.42**	**35.34**	36.71	**34.24**	37.48
	V-BM3D$_{\text{hq}}$	37.78	34.90	32.07	39.83	34.93	**36.88**	33.47	**37.76**
15	V-BM4D	35.46	32.92	30.11	38.92	33.06	34.78	31.62	35.44
	V-BM3D	35.44	32.63	29.81	38.64	33.00	34.64	31.05	35.67
	V-BM4D$_{\text{hq}}$	35.99	**33.08**	**30.19**	**39.25**	**33.24**	34.82	**31.82**	35.42
	V-BM3D$_{\text{hq}}$	**36.06**	32.84	29.86	38.93	33.14	**35.13**	31.19	**35.93**
20	V-BM4D	34.04	31.52	28.30	38.01	31.63	33.43	30.00	33.82
	V-BM3D	34.04	31.20	28.24	37.85	31.71	33.30	29.57	34.18
	V-BM4D$_{\text{hq}}$	34.58	**31.69**	**28.39**	**38.35**	31.82	33.50	**30.21**	33.86
	V-BM3D$_{\text{hq}}$	**34.78**	31.42	**28.39**	38.28	**31.93**	**33.84**	29.71	**34.54**

Table 5.3: Comparison between the PSNR (dB) outputs obtained from the proposed V-BM4D algorithm and the V-BM3D algorithm under both the standard and the high quality (hq) profiles. The test sequences are corrupted by i.i.d. Gaussian noise with zero mean and different standard deviations σ. The table reports the experiments having $\sigma \geq 25$.

σ	Video:	*Salesm.*	*Tennis*	*Fl. Gard.*	*Miss Am.*	*Coastg.*	*Foreman*	*Bus*	*Bicycle*
	Res.:	288×352	240×352	240×352	288×360	144×176	288×352	288×352	576×720
	Frames:	50	150	150	150	300	300	150	30
25	V-BM4D	32.96	30.51	26.83	37.26	30.57	32.45	28.88	32.49
	V-BM3D	32.79	30.11	27.00	37.10	30.62	32.19	28.48	32.90
	V-BM4D$_{\text{hq}}$	33.51	**30.72**	26.94	**37.67**	30.75	32.53	**29.07**	32.58
	V-BM3D$_{\text{hq}}$	**33.62**	30.36	**27.25**	37.65	**30.93**	**32.78**	28.63	**33.34**
30	V-BM4D	32.07	29.68	25.62	36.61	29.74	31.70	27.99	31.41
	V-BM3D	31.68	29.22	25.89	36.41	29.68	31.27	27.59	31.77
	V-BM4D$_{\text{hq}}$	**32.64**	**29.91**	25.73	**37.08**	29.91	31.80	**28.18**	31.53
	V-BM3D$_{\text{hq}}$	32.61	29.51	**26.26**	37.07	**30.06**	**31.87**	27.78	**32.28**
35	V-BM4D	31.32	28.99	24.64	36.04	29.07	31.07	27.26	30.52
	V-BM3D	30.72	28.56	25.16	35.87	28.92	30.56	26.91	30.85
	V-BM4D$_{\text{hq}}$	**31.91**	**29.24**	24.74	**36.57**	**29.23**	**31.19**	**27.45**	30.65
	V-BM3D$_{\text{hq}}$	31.53	28.76	**25.30**	36.46	**29.23**	31.05	26.96	**31.17**
40	V-BM4D	30.68	28.43	23.78	35.54	28.50	30.52	26.65	29.74
	V-BM3D	29.93	27.99	24.33	35.45	28.27	29.97	26.28	30.02
	V-BM4D$_{\text{hq}}$	**31.28**	**28.68**	23.86	**36.12**	**28.66**	**30.66**	**26.83**	29.88
	V-BM3D$_{\text{hq}}$	30.71	28.20	**24.48**	35.94	28.56	30.41	26.34	**30.33**

Table 5.4: Denoising MOVIE [58] score (the lower the better). In order to enhance the readability of the results, every value has been multiplied by 10^3. The table reports the experiments having $\sigma \leq 20$.

σ	Video:	*Salesm.*	*Tennis*	*Fl. Gard.*	*Miss Am.*	*Coastg.*	*Foreman*	*Bus*	*Bicycle*
	Res.:	288×352	240×352	240×352	288×360	144×176	288×352	288×352	576×720
	Frames:	50	150	150	150	300	300	150	30
5	V-BM4D	0.0216	**0.0260**	0.0192	**0.0267**	0.0260	0.0318	0.0374	0.0245
	V-BM3D	0.0226	0.0300	0.0249	0.0323	0.0335	0.0367	0.0465	**0.0241**
	V-BM4D$_{\text{hq}}$	0.0227	0.0261	**0.0189**	**0.0267**	**0.0247**	**0.0315**	**0.0349**	0.0263
	V-BM3D$_{\text{hq}}$	**0.0199**	0.0295	0.0260	0.0312	0.0314	0.0375	0.0442	0.6737
10	V-BM4D	**0.0817**	0.1207	0.0736	**0.0872**	0.1034	**0.1175**	0.1558	0.0939
	V-BM3D	0.0907	0.1473	0.0914	0.1072	0.1296	0.1303	0.2047	**0.0878**
	V-BM4D$_{\text{hq}}$	0.0880	**0.1144**	**0.0720**	0.0890	**0.0975**	0.1193	**0.1447**	0.0985
	V-BM3D$_{\text{hq}}$	0.0822	0.1430	0.0952	0.1058	0.1333	0.1446	0.2019	0.0879
15	V-BM4D	0.1855	0.2913	0.1565	**0.1717**	0.2133	**0.2294**	0.3338	0.1933
	V-BM3D	0.2135	0.3684	0.1769	0.2001	0.2491	0.2440	0.4461	0.1711
	V-BM4D$_{\text{hq}}$	0.1915	**0.2683**	**0.1526**	0.1750	**0.2001**	0.2356	**0.3163**	0.2032
	V-BM3D$_{\text{hq}}$	**0.1851**	0.3532	0.1787	0.1961	0.2702	0.2696	0.4514	**0.1710**
20	V-BM4D	0.3382	0.5107	0.2680	**0.2671**	0.3454	**0.3534**	0.5372	0.3253
	V-BM3D	0.4636	0.7267	0.2763	0.3096	0.4142	0.3792	0.7175	0.2689
	V-BM4D$_{\text{hq}}$	**0.3337**	**0.4797**	**0.2612**	0.2743	**0.3273**	0.3649	**0.5199**	0.3451
	V-BM3D$_{\text{hq}}$	0.3679	0.6874	0.2691	0.2936	0.4283	0.4014	0.7303	**0.2666**

Table 5.5: Denoising MOVIE [58] score (the lower the better). In order to enhance the readability of the results, every value has been multiplied by 10^3. The table reports the experiments having $\sigma \geq 25$.

σ	Video:	*Salesm.*	*Tennis*	*Fl. Gard.*	*Miss Am.*	*Coastg.*	*Foreman*	*Bus*	*Bicycle*
	Res.:	288×352	240×352	240×352	288×360	144×176	288×352	288×352	576×720
	Frames:	50	150	150	150	300	300	150	30
25	V-BM4D	0.5405	0.7429	0.4079	**0.3687**	0.4958	**0.4846**	0.7467	0.4795
	V-BM3D	0.9270	1.1016	0.3947	0.4392	0.6455	0.5479	0.9999	0.3907
	V-BM4D$_{\text{hq}}$	**0.5151**	**0.7197**	0.3996	0.3802	**0.4781**	0.5023	**0.7294**	0.5094
	V-BM3D$_{\text{hq}}$	0.7006	1.0662	**0.3707**	0.4071	0.6303	0.5644	1.0261	**0.3823**
30	V-BM4D	0.7839	0.9738	0.5788	**0.4724**	0.6667	**0.6195**	0.9582	0.6428
	V-BM3D	1.5636	1.4609	0.5500	0.5782	0.9646	0.7476	1.3028	0.5372
	V-BM4D$_{\text{hq}}$	**0.7347**	**0.9678**	0.5701	0.4898	**0.6562**	0.6442	**0.9405**	0.6848
	V-BM3D$_{\text{hq}}$	1.1624	1.4148	**0.4988**	0.5343	0.8863	0.7534	1.3342	**0.5224**
35	V-BM4D	1.0608	**1.1985**	0.7699	**0.5747**	**0.8590**	**0.7542**	1.1720	0.8100
	V-BM3D	2.3636	1.8539	0.7026	0.7418	1.3640	0.9792	1.6110	0.7320
	V-BM4D$_{\text{hq}}$	**0.9836**	1.2108	0.7628	0.6004	0.8607	0.7879	**1.1520**	0.8684
	V-BM3D$_{\text{hq}}$	1.8120	1.8050	**0.6626**	0.6899	1.2353	0.9719	1.6585	**0.7256**
40	V-BM4D	1.3667	**1.4102**	0.9734	**0.6759**	**1.0660**	**0.8885**	1.3831	0.9721
	V-BM3D	3.0905	2.1700	0.9207	0.8921	1.8648	1.2147	1.9279	0.9403
	V-BM4D$_{\text{hq}}$	**1.2603**	1.4455	0.9763	0.7109	1.0915	0.9315	**1.3653**	1.0581
	V-BM3D$_{\text{hq}}$	2.4669	2.1165	**0.8637**	0.8353	1.6745	1.1910	1.9937	**0.9225**

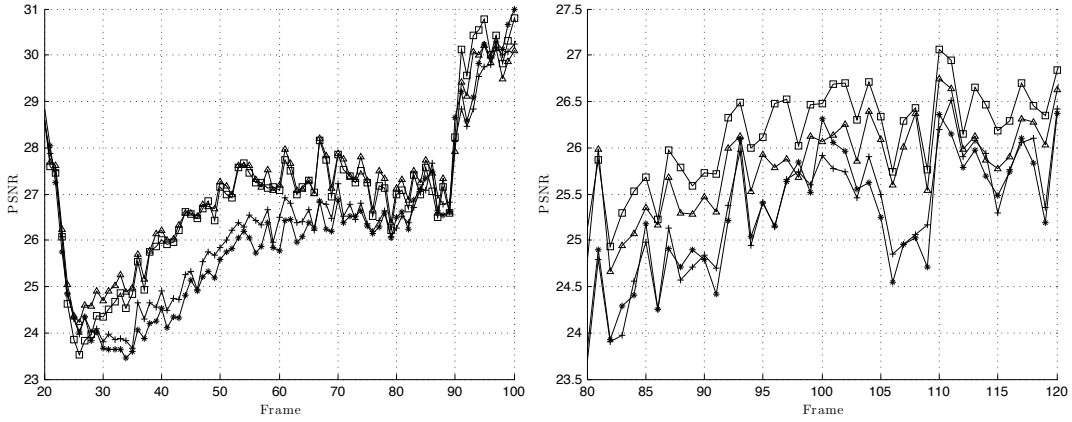


Figure 5.11: PSNR output frame-by-frame of the sequences *Tennis* (left) and *Bus* (right) denoised by V-BM4D$_{\text{hq}}$ (□), V-BM4D (△), V-BM3D$_{\text{hq}}$ (∗) and V-BM3D (+).

V-BM4D handles effectively the sequences characterized by high dynamics, frequent scene changes and heavy noise, as *Tennis*, *Coastguard* and *Bus*. In particular, Figure 5.11 shows that as soon as the sequence presents a significant change in the scene, the denoising performance decreases significantly for the three algorithms, but, in these situations, V-BM4D requires much less frames to recover high PSNR values, as shown by the lower peaks at frame 90 of *Tennis*. Furthermore, V-BM4D significantly outperforms V-BM3D in scene characterized by high dynamism, as shown in the plot of *Bus*, where even the standard profile of V-BM4D constantly overcomes the high quality profile of V-BM3D. Nonetheless, in the sequence *Bicycle*, V-BM4D exhibits a systematic loss of performance if compared to V-BM3D. In fact, since *Bicycle* represents a fixed scene with a large rotating wheel with a few objects fastened over it, the large uniform background and the frequent object occlusions, can challenge the motion estimation favoring a self-similar block-based grouping strategy. In fact, observe that only in the experiments of *Bycicle* with $\sigma \leq 15$ V-BM4D$_{\text{hq}}$ performs worse than V-BM4D.

As anticipated in the introduction, the main issue about V-BM3D lies in the grouping step, which indiscriminately stacks together spatial and temporal similar blocks in the final three-dimensional group through an adaptive three-dimensional spatio-temporal window. However, as a matter of fact, Table 5.6 shows that a grouping based on motion always guarantees better performances than what can be achieved by grouping even a larger number of blocks in the spatial domain. In particular, every pair (M,h) forces the algorithm to build volumes of temporal extent $2h+1$ and to stack M of such volumes in the grouping step. Thus, the two extreme pairs (1,7) and (16,0) means groups of single volumes of extent 15, and groups of 16 volumes of extent 1, respectively. In other words, we are comparing a fully temporal-oriented grouping strategy against a fully spatial-oriented one. The PSNR outputs of V-BM4D applied in the two above mentioned cases are a perfect example of the importance of accounting motion in the grouping and collaborative filtering

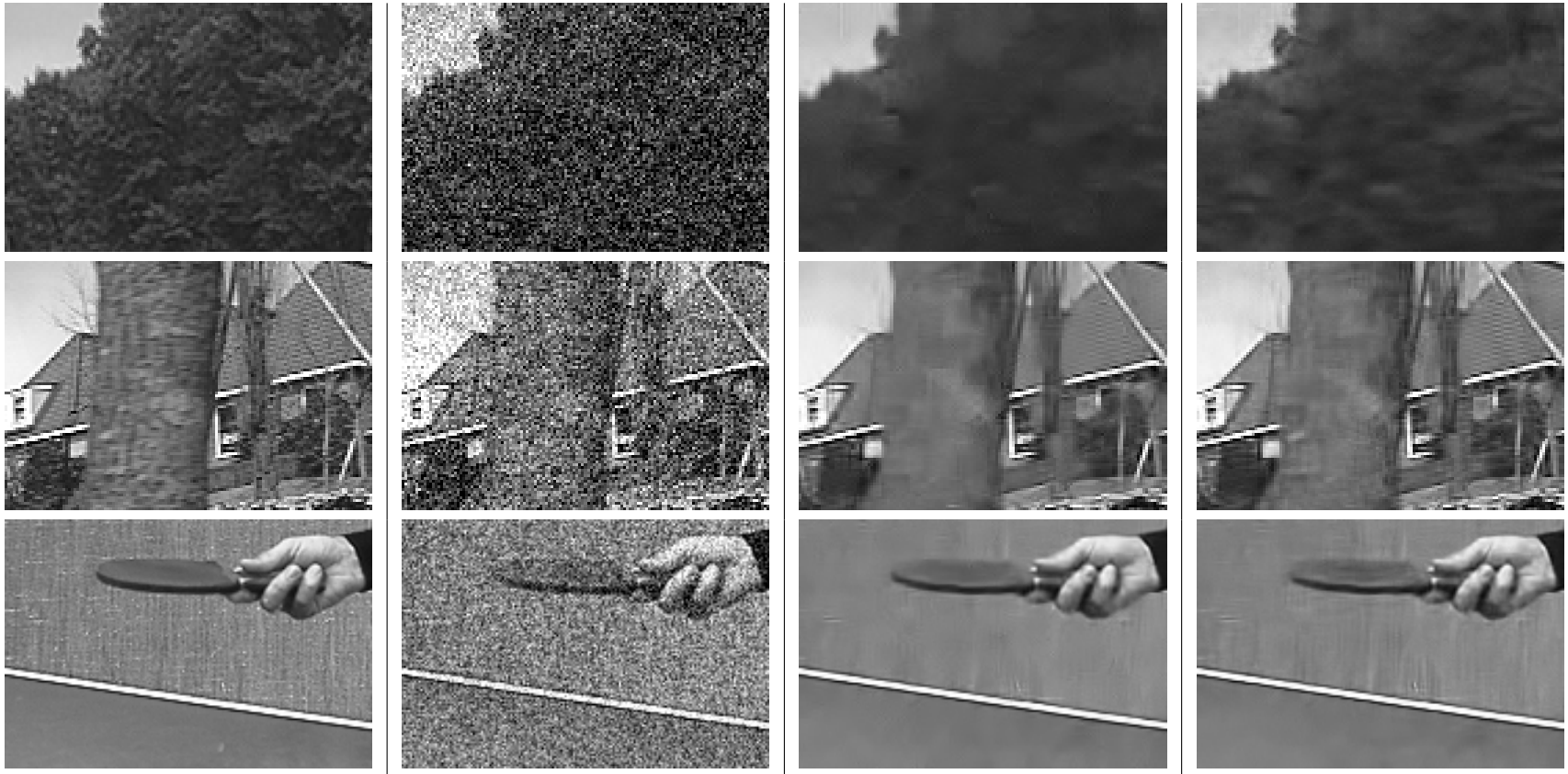


Figure 5.12: Visual comparison of the sequences, from top to bottom, *Bus* and *Tennis* corrupted by white Gaussian noise with standard deviation $\sigma = 40/255$, denoised by the proposed V-BM4D and the V-BM3D algorithm.

Table 5.6: PSNR outputs of V-BM4D tuned with different space (M) and time (h) parameters combinations. Recall that the dimension of the temporal extent is defined as $2h+1$. The test sequence *Salesman* and *Tennis* have been corrupted by i.i.d. white Gaussian noise with $\sigma = 20/255$.

M	Video	h							
		0	1	2	3	4	5	6	7
1	*Salesm.*	29.26	32.19	33.07	33.51	33.76	33.92	34.02	34.10
	Tennis	27.99	30.12	30.74	31.04	31.20	31.29	31.34	31.37
2	*Salesm.*	29.70	32.28	33.08	33.46	33.69	33.83	33.92	33.99
	Tennis	28.37	30.36	30.98	31.27	31.42	31.50	31.55	31.57
4	*Salesm.*	30.07	32.37	33.08	33.43	33.63	33.76	33.84	33.89
	Tennis	28.58	30.47	31.05	31.32	31.46	31.53	31.57	31.59
8	*Salesm.*	30.34	32.53	33.23	33.57	33.77	33.89	33.98	34.03
	Tennis	28.70	30.52	31.09	31.35	31.48	31.55	31.58	31.60
16	*Salesm.*	30.48	32.69	33.41	33.77	33.98	34.12	34.21	34.27
	Tennis	28.75	30.55	31.11	31.36	31.49	31.56	31.60	31.61

paradigm. Even if the temporal-oriented groups have a slightly smaller size than the spatial ones, in the former case V-BM4D obtains an increase in performance of about 16% in *Salesman* and 12% in *Tennis* with respect to the basic configuration (1,0), while in the latter case the PSNR augments about the 4% and 3% only. Moreover, it is not unusual to observe a drop of performances in the spatial direction, as shown in particular by the sequence *Salesman* for every $h \geq 3$.

Finally, Figure 5.12 offers a visual comparison of the performance of the two algorithms. As a subjective quality assessment, V-BM4D better preserves the textures, without introducing significant artifacts in the restored video: this is clearly visible in the tree leaves of the *Bus* sequence or in the background texture of *Tennis*.

Table 5.7 and 5.8 compare the performance of V-BM4D deblocking filter with the *MPlayer* deblocking filter[2], in terms of PSNR. Eight sequences compressed by an MPEG-4 encoder with different values of q and bit-per-pixel rates have been considered: additional details concerning these sequences are reported in the table header. Numerical results show that the V-BM4D filter outperforms MPlayer, providing restored videos having PSNR about 1 dB higher. Figure 5.13 shows the results of V-BM4D deblocking on the *Foreman*, *Tennis* and *Coastguard* sequences, encoded at high compression level ($q = 25$). The visual quality of the filtered videos has been significantly improved, since the compression artifacts, such as blocking or ghosting, have been successfully filtered without losing fine image details. In particular, we can note how the face of *Foreman*, the player and the white poster of *Tennis* and the horizontal stone-wall of *Coastguard* quite sharply emerge from their blocky counterparts, while almost-uniform areas, such as the white striped building

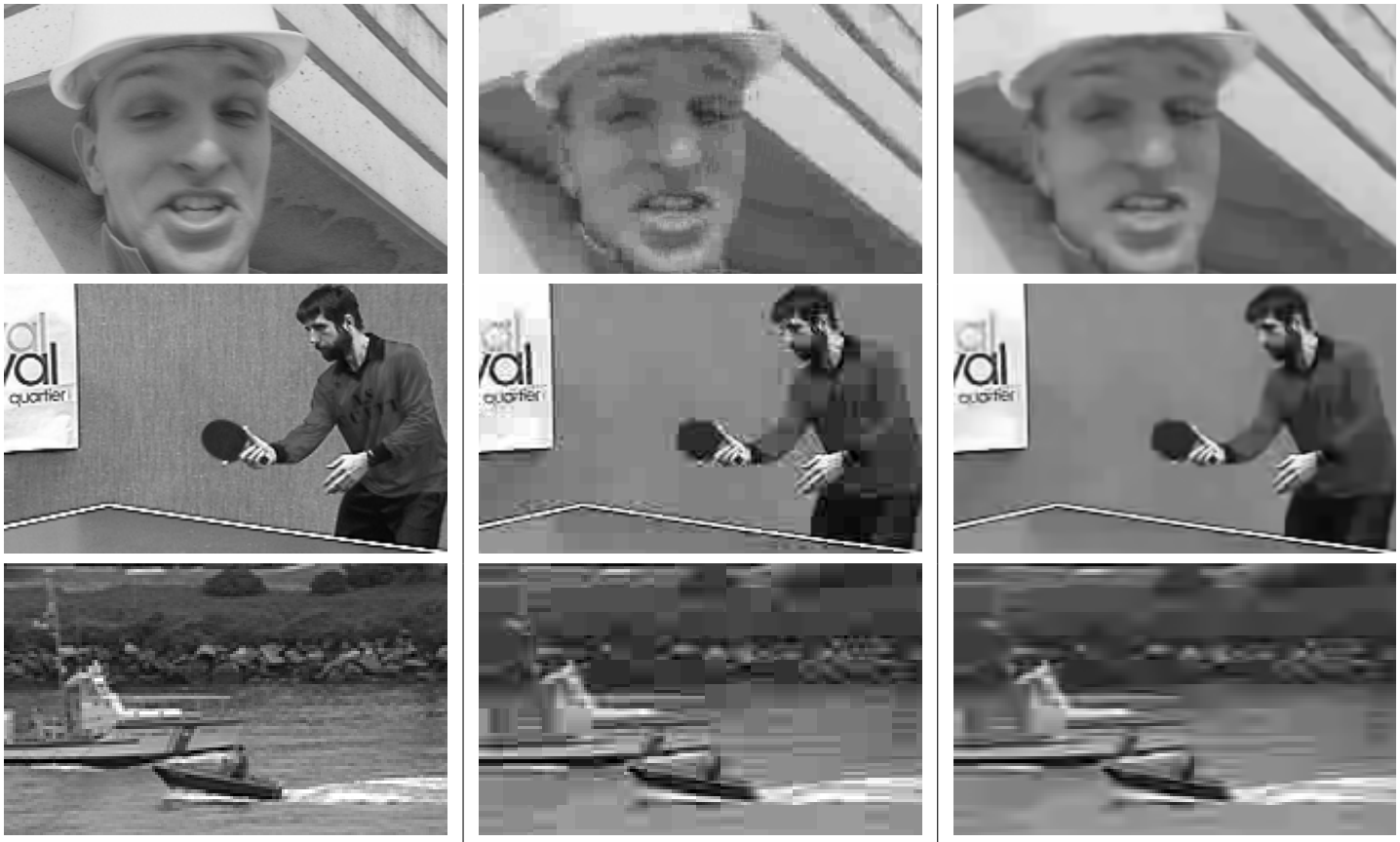
[2]Source code and documentation can be found at `http://sourceforge.net/projects/ffdshow-tryout/` and `http://www.mplayerhq.hu/`

Table 5.7: Deblocking performance of V-BM4D in terms of PSNR: q is the scale parameter of the quantization matrix of the MPEG-4 encoder and bpp denotes the average bit-per-pixel rate of the compressed video. As a reference, the PSNR of both the MPlayer accurate deblocking filter and the unfiltered compressed (compr.) video are also provided for each value q of the MPEG-4 quantizer. The table reports the experiments having $q \leq 15$.

q	Video:	*Salesm.*	*Tennis*	*Fl. Gard.*	*Miss Am.*	*Coastg.*	*Foreman*	*Bus*	*Bicycle*
	Res.:	288×352	240×352	240×352	288×360	144×176	288×352	288×352	576×720
	Frames:	50	150	150	150	300	300	150	30
5	bpp	0.3232	0.5323	1.4824	0.0884	0.4609	0.3005	0.7089	0.4315
	V-BM4D	**35.96**	**34.35**	**33.55**	**39.53**	**34.71**	**36.42**	**34.98**	**38.04**
	MPlayer	35.14	33.79	32.73	38.58	34.00	35.60	34.36	36.53
	Compr.	35.28	33.87	32.81	39.03	34.12	35.70	34.45	36.71
10	bpp	0.1319	0.2249	0.7288	0.0399	0.1926	0.1276	0.3285	0.2076
	V-BM4D	**32.11**	**30.37**	**27.94**	**37.31**	**30.72**	**32.90**	**30.68**	**33.47**
	MPlayer	31.66	29.87	27.40	36.61	30.23	32.16	30.11	32.45
	Compr.	31.54	29.84	27.41	36.66	30.19	32.09	30.07	32.37
15	bpp	0.0865	0.1326	0.4470	0.0318	0.1184	0.0812	0.2039	0.1333
	V-BM4D	**30.03**	**28.46**	**25.11**	**36.14**	**28.70**	**31.08**	**28.45**	**31.03**
	MPlayer	29.65	28.03	24.68	35.59	28.30	30.36	27.89	30.12
	Compr.	29.48	27.97	24.67	35.41	28.18	30.27	27.83	30.00

Table 5.8: Deblocking performance of V-BM4D in terms of PSNR: q is the scale parameter of the quantization matrix of the MPEG-4 encoder and bpp denotes the average bit-per-pixel rate of the compressed video. As a reference, the PSNR of both the MPlayer accurate deblocking filter and the unfiltered compressed (compr.) video are also provided for each value q of the MPEG-4 quantizer. The table reports the experiments having $q \geq 20$.

q	Video:	*Salesm.*	*Tennis*	*Fl. Gard.*	*Miss Am.*	*Coastg.*	*Foreman*	*Bus*	*Bicycle*
	Res.:	288×352	240×352	240×352	288×360	144×176	288×352	288×352	576×720
	Frames:	50	150	150	150	300	300	150	30
20	bpp	0.0661	0.0943	0.3058	0.0280	0.0852	0.0625	0.1453	0.0985
	V-BM4D	**28.64**	**27.23**	**23.27**	**35.03**	**27.39**	**29.85**	**26.92**	**29.42**
	MPlayer	28.31	26.82	22.90	32.93	27.04	29.12	26.42	28.60
	Compr.	28.11	26.76	22.88	34.21	26.90	29.03	26.35	28.43
25	bpp	0.0546	0.0710	0.2225	0.0257	0.0679	0.0523	0.1121	0.0846
	V-BM4D	**27.63**	**26.34**	**22.00**	**34.34**	**26.46**	**29.02**	**25.89**	**28.23**
	MPlayer	27.30	25.96	21.63	33.66	26.11	28.25	25.38	27.35
	Compr.	27.07	25.85	21.62	33.45	25.98	28.10	25.27	27.22
30	bpp	0.0477	0.0604	0.1697	0.0244	0.0584	0.0480	0.0921	0.0676
	V-BM4D	**26.85**	**25.59**	**21.01**	**33.29**	**25.71**	**28.33**	**25.03**	**27.36**
	MPlayer	26.51	25.26	20.65	32.80	25.38	27.57	24.55	26.54
	Compr.	26.28	25.11	20.64	32.39	25.25	27.37	24.41	26.35



Original frame | Blocky frame | V-BM4D

Figure 5.13: Visual comparison of the sequences, from top to bottom, *Foreman*, *Tennis* and *Coastguard* compressed with the MPEG-4 encoder with quantization parameter $q = 25$, deblocked by the proposed V-BM4D algorithm.

behind *Foreman* or the table and background texture in *Tennis* have been pleasingly smoothed.

5.6 Complexity

One run of V-BM4D requires the execution of the hard-thresholding stage (whose complexity is $\mathcal{C}^{\text{ht}}_{\text{V-BM4D}}$), of the Wiener filtering stage (whose complexity is $\mathcal{C}^{\text{wie}}_{\text{V-BM4D}}$), and two runs of the motion estimation algorithm (whose complexity is $\mathcal{C}_{\text{CT}}$). Hence, the V-BM4D overall complexity is

$$\mathcal{C}_{\text{V-BM4D}} = 2\mathcal{C}_{\text{ME}} + \mathcal{C}^{\text{ht}}_{\text{V-BM4D}} + \mathcal{C}^{\text{wie}}_{\text{V-BM4D}}, \tag{5.44}$$

where the superscripts *ht* and *wie* denote hard-thresholding and Wiener filtering, respectively. Differently, V-BM3D does not require motion estimation, thus its complexity is simply given by the sum of the cost of its hard-thresholding $\mathcal{C}^{\text{ht}}_{\text{V-BM3D}}$ and Wiener-filtering stage $\mathcal{C}^{\text{wie}}_{\text{V-BM3D}}$ as follows

$$\mathcal{C}_{\text{V-BM3D}} = \mathcal{C}^{\text{ht}}_{\text{V-BM3D}} + \mathcal{C}^{\text{wie}}_{\text{V-BM3D}}. \tag{5.45}$$

In this analysis we compute the complexity as the number of arithmetic operations required, without considering any other external factor, such as the memory requirements or I/O access. Table 5.9 provides a comprehensive summary of the parameters involved in this analysis, as well as a brief comment about the role they play in the algorithm. To provide an easy comparison among V-BM3D and V-BM4D, we assume that the number of blocks in a spatiotemporal volume is $\bar{h}$, and that this corresponds to the size of temporal window in V-BM3D; similarly, we assume that the number of grouped volumes in V-BM4D (referred as M) corresponds to the number of grouped blocks in V-BM3D.

5.6.1 Motion Estimation

The computation of the trajectory requires searching for a most similar block within an adaptive search window of size $N_S \times N_S$ once for each of the preceding and following frames, i.e. $\bar{h} - 1$ times. The computation of the ℓ^2 distance between a pair of blocks requires $3N^2$ operations, because requires two additions and one multiplication for each corresponding pixel. Since a trajectory is constructed for each pixel in every frame of the video, the total cost is

$$\mathcal{C}_{\text{ME}} = nT(\bar{h} - 1)N_S^2\left(3N^2\right) \tag{5.46}$$

5.6.2 Hard-thresholding Stage

In the hard-thresholding stage, for each processed block according to N_{step}, at most M similar volume extracted from a search window of size $N_G \times N_G$ are stacked to-

Table 5.9: Summary of the parameters involved in the complexity analysis.

Parameter	Notes
T	Total number of frames in the video.
n	Number of pixel per frame.
N	Dimension of the two-dimensional square blocks.
$\bar{h}$	Temporal extent of the volumes in V-BM4D, size of the temporal window in V-BM3D, referred to as N_{FR} in [36].
N_S	Size of the motion estimation window.
M	Size of the groups, that is the number of grouped volumes in V-BM4D or the number of grouped blocks in V-BM3D.
N_G	Size of the window used in the grouping.
N_{step}	Processing step (refer to Section 5.5 for further details).
$\mathcal{C}_{(m,p,n)}$	Numeric operations required by a multiplication between matrices of size $m \times p$ and $p \times n$. Although more efficient algorithms exist, we assume this complexity to be mpn.

gether. Once the group is formed, a separable four-dimensional transform is applied, and the hard-thresholding is then performed via element-wise multiplication. Eventually, the basic estimate is obtained by aggregating the inverse four-dimensional transform of the filtered groups:

$$\mathcal{C}^{\text{ht}}_{\text{V-BM4D}} = \frac{n}{N^2_{\text{step}}} T\Big(\mathcal{C}^{\text{G}}_{\text{V-BM4D}} + 2\mathcal{C}^{\text{T}}_{\text{V-BM4D}} + \mathcal{C}^{\text{F}}_{\text{V-BM4D}}\Big) + \mathcal{C}^{\text{A}}_{\text{V-BM4D}}, \tag{5.47}$$

where G, T, F and A stands for Grouping, Transformation, Filtering and Aggregation. More specifically their costs are:

$$\mathcal{C}^{\text{G}}_{\text{V-BM4D}} = N^2_G 3\bar{h}N^2, \tag{5.48}$$
$$\mathcal{C}^{\text{T}}_{\text{V-BM4D}} = 2M\bar{h}\mathcal{C}_{(N,N,N)} + M\mathcal{C}_{(\bar{h},\bar{h},N^2)} + \mathcal{C}_{(M,M,\bar{h}N^2)}, \tag{5.49}$$
$$\mathcal{C}^{\text{F}}_{\text{V-BM4D}} = M\bar{h}N^2, \tag{5.50}$$
$$\mathcal{C}^{\text{A}}_{\text{V-BM4D}} = Tn(\bar{h}+1), \tag{5.51}$$

where the symbol $\mathcal{C}_{(\cdot,\cdot,\cdot)}$ stands for matrix multiplication cost as explained in Table 5.9, and the factor 3 in the grouping complexity is due to the ℓ^2 distance computed between two three-dimensional volumes of size $N \times N \times \bar{h}$. Note that, since the four-dimensional transform is separable, its cost is the sum of four different matrix multiplications, one for each dimension of the group.

In V-BM3D, the fourth dimension is missing, hence the algorithm only transforms the M grouped blocks and the group itself:

$$\mathcal{C}^{\text{ht}}_{\text{V-BM3D}} = \frac{n}{N^2_{\text{step}}} T\Big(\mathcal{C}^{\text{G}}_{\text{V-BM3D}} + 2\mathcal{C}^{\text{T}}_{\text{V-BM3D}} + \mathcal{C}^{\text{F}}_{\text{V-BM3D}}\Big) + \mathcal{C}^{\text{A}}_{\text{V-BM3D}}, \tag{5.52}$$

where, as before:

$$\mathcal{C}^{\mathrm{G}}_{\text{V-BM3D}} = \left(N_G^2 + N_B \bar{h} N_{PR}^2\right) 3N^2, \tag{5.53}$$

$$\mathcal{C}^{\mathrm{T}}_{\text{V-BM3D}} = 2M\mathcal{C}_{(N,N,N)} + \mathcal{C}_{(M,M,N^2)}, \tag{5.54}$$

$$\mathcal{C}^{\mathrm{F}}_{\text{V-BM3D}} = MN^2, \tag{5.55}$$

$$\mathcal{C}^{\mathrm{A}}_{\text{V-BM3D}} = Tn(\bar{h} + 1). \tag{5.56}$$

Observe in Equation (5.53) that the grouping is accomplished through a technique called predictive-search block-matching [36] described in Section 4.2.1: briefly it performs a full-search inside a $N_G \times N_G$ window in the first frame to extract the N_B best-matching blocks, then, in the following $\bar{h}$ frames it inductively searches the N_B best-matching blocks inside windows of size $N_{PR} \times N_{PR}$, with $N_{PR} \ll N_G$, centered at the position of the previous N_B blocks.

5.6.3 Wiener-filtering Stage

The complexity of the Wiener-filtering stage can be expressed as that of hard-thresholding stage in (5.47), with the exception that transforms have to be applied to two groups having equal size, and that the filtering, which is still performed via element-wise multiplication, requires also the pre-computation of a family of weights, which takes 6 arithmetic operations per pixel:

$$\mathcal{C}^{\text{wie}}_{\text{V-BM4D}} = \frac{n}{N_{\text{step}}^2} T\left(\mathcal{C}^{\mathrm{G}}_{\text{V-BM4D}} + 4\mathcal{C}^{\mathrm{T}}_{\text{V-BM4D}} + \mathcal{C}^{\mathrm{F}}_{\text{V-BM4D}}\right) + 6\mathcal{C}^{\mathrm{A}}_{\text{V-BM4D}}, \tag{5.57}$$

Analogously to (5.52), in V-BM3D the complexity is

$$\mathcal{C}^{\text{wie}}_{\text{V-BM3D}} = \frac{n}{N_{\text{step}}^2} T\left(\mathcal{C}^{\mathrm{G}}_{\text{V-BM3D}} + 4\mathcal{C}^{\mathrm{T}}_{\text{V-BM3D}} + \mathcal{C}^{\mathrm{F}}_{\text{V-BM3D}}\right) + 6\mathcal{C}^{\mathrm{A}}_{\text{V-BM3D}}, \tag{5.58}$$

5.6.4 Comparative Analysis

Provided that every other parameters remain unchanged, the complexity of the proposed V-BM4D algorithm, as well as the one of V-BM3D, scales linearly $\mathcal{O}(n)$ with the number of processed pixels. However it is worth carrying out a deeper analysis, since the different multiplying constants factors have a remarkable impact on the final cost of the two algorithms.

Analyzing separately the single components of the cost expansion equations (5.47) and (5.52), observe that we can neglect both the cost of the grouping and the aggregation since they are similar, or even equal, in both algorithms. The filtering in V-BM4D requires exactly $\bar{h}$ times more operations than in V-BM3D, as well as the transformation applied to the blocks and to the groups. Thus the overall cost due to the transformation step in V-BM4D is $\bar{h}$ times the cost of V-BM3D in addition

to the operations due to to transform the M grouped volumes:

$$\mathcal{C}^{\mathrm{G}}_{\text{V-BM4D}} \approx \mathcal{C}^{\mathrm{G}}_{\text{V-BM3D}}, \tag{5.59}$$

$$\mathcal{C}^{\mathrm{T}}_{\text{V-BM4D}} = \bar{h}\mathcal{C}^{\mathrm{T}}_{\text{V-BM3D}} + M\mathcal{C}_{(\bar{h},\bar{h},N^2)}, \tag{5.60}$$

$$\mathcal{C}^{\mathrm{F}}_{\text{V-BM4D}} = \bar{h}\mathcal{C}^{\mathrm{F}}_{\text{V-BM3D}}, \tag{5.61}$$

$$\mathcal{C}^{\mathrm{A}}_{\text{V-BM4D}} = \mathcal{C}^{\mathrm{A}}_{\text{V-BM3D}}. \tag{5.62}$$

An analogous inference can be made also for the Wiener filtering stage given in (5.57) and (5.58).

In conclusion, we can state that V-BM4D is at least $\bar{h}$ times computationally more demanding than V-BM3D in both the hard-thresholding and the Wiener-filtering stage, but the former is also burdened with the cost $\mathcal{C}_{\mathrm{ME}}$ required by the motion estimation step. However, observe that $\mathcal{C}_{\mathrm{ME}}$ can be entirely eliminated if the input video is encoded with a motion-compensated algorithm, such as MPEG-4 or H.264, because the motion vectors required to build the spatio-temporal volumes can be extracted directly from the video byte stream.

5.7 Conclusions

In this Thesis, we have presented a novel methodology for filtering noisy video in four-dimensional transform domain, called V-BM4D. The proposed algorithm exploits both the nonlocal and spatiotemporal correlation within the data to obtain a sparse representation of the noisy signal and achieve an effective denoising by coefficients shrinkage in transform domain. Video denoising constitute an important research field because digital videos are widely used in a large number of applications. Possible employments of V-BM4D, encompass medical and astronomical imaging, multimedia services, teleconferencing, surveillance, object tracking or video compression.

Experiments show that V-BM4D outperforms V-BM3D in terms of measured performance (PSNR, MOVIE), and of visual appearance (Figure 5.12), thus achieving state-of-the-art results in video denoising. In particular, V-BM4D can restore much better than V-BM3D fine image details, even in sequences corrupted by heavy noise ($\sigma = 40$): this difference is clearly visible in the processed frames shown in Figure 5.12. Moreover, the comparison between V-BM3D and V-BM4D highlights that the temporal correlation is a key element in video denoising, and that it has to be adequately handled when designing nonlocal video restoration algorithms.

We wish to remark that V-BM4D has obviously a higher computational complexity when compared to V-BM3D, mainly due to the processing of higher-dimensional arrays and to the calculation of the trajectories by concatenation of motion vectors. Ongoing work addresses the parallelization of V-BM4D, inasmuch as most of its computing task are well-localized in terms of space-time complexity and, more

importantly, they are independent from each other. Specifically, each trajectory can be calculated separately in the preprocessing stage, and, additionally, both the grouping and the collaborative filtering step can be performed independently with ease. For these reasons, we aim to implement V-BM4D on GPUs, leveraging the inherent parallel architecture of such devices.

BIBLIOGRAPHY

[1] The MPEG-4 video standard verification model. pages 142–154, 2001.

[2] Y. Zhai A. Basharat and M. Shah. Content based video matching using spatiotemporal volumes. *Computer Vision and Image Understanding*, 110(3):360–377, 2008.

[3] B. Coll A. Buades and J-M. Morel. A non-local algorithm for image denoising. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:60–65, June 2005.

[4] B. Coll A. Buades and J-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling Simulation*, 4(2):490–530, 2005.

[5] B. Coll A. Buades and J-M. Morel. Nonlocal image and movie denoising. *International Journal of Computer Vision*, 76(2):123–139, 2008.

[6] V. Katkovnik A. Danielyan, A. Foi and K. Egiazarian. Image and video super-resolution via spatially adaptive block-matching filtering. August 2008.

[7] V. Katkovnik A. Danielyan, A. Foi and K. Egiazarian. Image upsampling via spatially adaptive block-matching filtering. August 2008.

[8] V. Katkovnik A. Foi and K. Egiazarian. Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. *IEEE Transactions on Image Processing*, 16(5):1395–1411, May 2007.

[9] V. Katkovnik A. Foi and K. Egiazarian. Signal-dependent noise removal in pointwise shape-adaptive DCT domain with locally adaptive variance. *European Signal Processing Conference (EUSIPCO), Poznan, Poland*, September 2007.

[10] V. Katkovnik K.Egiazarian A. Foi, M. Trimeche. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754, October 2008.

[11] V. Zlokolica A. Pizurica, R. Pizurica and W. Philips. Combined wavelet domain and temporal video denoising. *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 334–341, 2003.

[12] F.J. Anscombe. The transformation of poisson, binomial and negative binomial data. *Biometrika*, 35(3/4):246–254, December 1948.

[13] G.C. Arce. Multistage order statistic filters for image sequence processing. *IEEE Transaction of Signal Processing*, 39:1147–1163, 1991.

[14] J.M. Fadili B. Zhang and J.L. Starck. Wavelets, ridgelets, and curvelets for poisson noise removal. *IEEE Transactions on Image Processing*, 17(7):1093–1108, July 2008.

[15] Y. Altunbasak B.K. Gunturk and R.M. Mersereau. Multiframe blocking-artifact reduction for transform-coded video. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4):276–282, April 2002.

[16] G. Boracchi and A. Foi. Multiframe raw-data denoising based on block-matching and 3-D filtering for low-light imaging and stabilization. August 2008.

[17] A. Bovik. *Handbook of Image and Video Processing*. Academic Press, 2000.

[18] I. Johnstone D. Donoho and I.M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1993.

[19] D. Larson D. Han, K. Kornerlson and E. Weber. *Frames for Undergraduates*, volume 40. American Mathematical Society, Providence, Rhode Island, 2007.

[20] A. Danielyan and A. Foi. Noise variance estimation in nonlocal transform domain. pages 41–45, August 2009.

[21] C.S. Davis. *Statistical Methods for the Analysis of Repeated Measurements*. Springer, 2003.

[22] D. Donoho. De-noising by soft thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, May 1995.

[23] S. Carrato F. Cocchia and G. Ramponi. Design and real-time implementation of a 3-D rational filter for edge preserving smoothing. *IEEE Transactions on Consumer Electronics*, 43:1291–1300, 1997.

[24] A. Foi. Practical denoising of clipped or overexposed noisy images. *Proceedings of the European Signal Processing Conference (EUSIPCO), Lausanne, Switzerland*, August 2008.

[25] A. Foi. Clipped noisy images: Heteroskedastic modeling and practical denoising. *Signal Processing*, 89(12):2609–2629, 2009.

[26] X. Yang W. Lin G. Zhai, W. Zhang and Y. Xu. Efficient deblocking with coefficient regularization, shape-adaptive filtering, and quantization constraint. *IEEE Transactions on Multimedia*, 10(5):735–745, August 2008.

[27] O.G. Guleryuz. Weighted overcomplete denoising, 2003.

[28] O.G. Guleryuz. Weighted averaging for denoising with overcomplete dictionaries. *IEEE Transactions on Image Processing*, 16(12):3020–3034, December 2007.

[29] R.C. Kim H. Paek and S.U. Lee. On the pocs-based postprocessing technique to reduce the blocking artifacts in transform coded images. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(3):358–367, June 1998.

[30] H. Hang, Y. Chou, and S. Cheng. Motion estimation for video coding standards. *Journal of VLSI Signal Processing Systems*, 17(2/3):113–136, 1997.

[31] K.H. Höhne and M. Böhm. Processing and analysis of radiographic image sequences. *Image Sequence Processing and Dynumic Scene Analysis*, 1983.

[32] M. Crouse J. Chou and K. Ramchandran. A simple algorithm for removing blocking artifacts in block-transform coded images. *Signal Processing Letters, IEEE*, 5(2):33–35, February 1998.

[33] B. Jähne. *Spatio-Temporal Image Processing: Theory and Scientific Applications*. Springer, 1 edition, 1993.

[34] S. Efstratiadis K. Katsaggelos J.C. Brailean, R.P. Kleihorst and R.L. Lagendijk. Noise reduction filters for dynamic image sequences: A review. *Proceedings of the IEEE*, 83(9):1272–1292, 1995.

[35] M.H. Wright J.C. Lagarias, J.A. Reeds and P.E. Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[36] A. Foi K. Dabov and K. Egiazarian. Video denoising by sparse 3D transform-domain collaborative filtering. *European Signal Processing Conference (EUSIPCO), Poznan, Poland*, September 2007.

[37] V. Katkovnik K. Dabov, A. Foi and K. Egiazarian. Image denoising with block-matching and 3D filtering. *Proceedings of the International Society for Optical Engineering (SPIE)*, 6064(30), 2006.

[38] V. Katkovnik K. Dabov, A. Foi and K. Egiazarian. Image denoising with block-matching and 3D filtering. *Proceedings in Electronic Imaging, San Jose, CA, USA*, 6064:606414.1–606414.12, January 2006.

[39] V. Katkovnik K. Dabov, A. Foi and K. Egiazarian. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[40] V. Katkovnik K. Dabov, A. Foi and K. Egiazarian. Joint image sharpening and denoising by 3D transform-domain collaborative filtering. September 2007.

[41] V. Katkovnik K. Dabov, A. Foi and K. Egiazarian. Image restoration by sparse 3D transform-domain collaborative filtering. 6812(1D), January 2008.

[42] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, 2 edition, 1993.

[43] X. Li and Y. Zheng. Patch-based video processing: a variational bayesian approach. *IEEE Transactions on Circuits and Systems for Video Technology*, 29:27–40, January 2009.

[44] Y. Chahir M. Ghoniem and A. Elmoataz. Nonlocal video denoising, simplification and inpainting using discrete regularization on graphs. *Signal Processing*, 90(8):2445–2455, 2010.

[45] M. Mäkitalo and A. Foi. On the inversion of the anscombe transformation in low-count poisson image denoising. *Proceedings of the International Workshop on Local and Non-Local Approximation in Image Processing (LNLA)*, pages 26–32, August 2009.

[46] S. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, 3 edition, December 2008.

[47] R. Megret and D. DeMenthon. A survey of spatio-temporal grouping techniques. Technical report, Institut National Des Sciences Appliquèes de Lyon, University of Maryland, August 2002.

[48] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[49] T. Nguyen and G. Strang. *Wavelets and Filter Banks*. Wellesley College, 2 edition, 1996.

[50] M. Protter and M. Elad. Image sequence denoising via sparse and redundant representations. *IEEE Transactions on Image Processing*, 18(1):27–35, January 2009.

[51] P.R. Prucnal and B.E.A. Saleh. Transformation of image-signal-dependent noise into image-signal-independent noise. *Optics Letters*, 6:316–318, July 1981.

[52] K. Egiazarian R. Öktem, L. Yaroslavsky and J. Astola. Transform based denoising algorithms: Comparative study, 1999.

[53] L. Yaroslavsky R. Öktem and K. Egiazarian. Signal and image denoising in transform domain and wavelet shrinkage: a comparative study. *European Signal Processing Conference (EUSIPCO), Island of Rhodes, Greece*, pages 2269–2272, September 1998.

[54] V.V. Lukin N.N. Ponomarenko R. Öktem, K. Egiazarian and O.V. Tsymbal. Locally adaptive DCT filtering for signal-dependent noise removal. *EURASIP Journal on Advances in Signal Processing*, 2007:1–10, 2007.

[55] A. Sundararajan R. Samadani and A. Said. Deringing and deblocking DCT compression artifacts with efficient shifted transforms. *International Conference on Image Processing. (ICIP).*, 3:1799–1802, October 2004.

[56] D. Rusanovskyy and K. Egiazarian. Video denoising algorithm in sliding 3D DCT domain. *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 618–625, 2005.

[57] P. Maragos S. Lefkimmiatis and G. Papandreou. Bayesian inference on multiscale models for poisson intensity estimation: Applications to photon-limited image denoising. *IEEE Transactions on Image Processing*, 18(8):1724–1741, August 2009.

[58] K. Seshadrinathan and A.C. Bovik. Motion tuned spatio-temporal quality assessment of natural videos. *IEEE Transactions on Image Processing*, 19(2):335–350, 2010.

[59] O.G. Sezer and Y. Altunbasak. Weighted average denoising with sparse orthonormal transforms. *proceedings of the IEEE International Conference on Image Processing (ICIP), Cairo, Egypt*, 2009.

[60] S.W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing.* California Technical Pub., 1997.

[61] Y.H. Chan S.W. Hong and W.C. Siu. Subband adaptive regularization method for removing blocking effect. 2:2523, October 1995.

[62] H.R. Wu T. Chen and B. Qiu. Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(5):594–602, May 2001.

[63] G. Bjontegaard T. Wiegand, G.J. Sullivan and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[64] K. Egiazarian V. Katkovnik, A. Foi and J. Astola. From local kernel to nonlocal multiple-model image denoising. *International Journal of Computer Vision*, July 2009.

[65] A. Wong and W. Bishop. Deblocking of block-transform compressed images using phase-adaptive shifted thresholding. *IEEE International Symposium on Multimedia (ISM)*, pages 97–103, December 2008.

[66] R. Sukthankar Y. Ke and M. Hebert. Spatio-temporal shape and flow correlation for action recognition. pages 1–8, 2007.

[67] A. Vetro H. Sun Y. Nie, H.S. Kong and K.E. Barner. Fast adaptive fuzzy post-filtering for coding artifacts removal in interlaced video. *IEEE International Conference on Acoustics, Speech, and Signal Processing. (ICASSP)*, 2:993–996, March 2005.

[68] Y. Zhang Y. Wang and J. Ostermann. *Video Processing and Communications*. Prentice Hall, Upper Saddle River, NJ, USA, 1 edition, 2001.

[69] M. Yuen. Coding artifacts and visual distortions. *Digital Video Image Quality and Perceptual Coding*, pages 87–112, 2006.

[70] A.C. Bovik Z. Wang and B.L. Evan. Blind measurement of blocking artifacts in images. *Proceedings of the International Conference on Image Processing (ICIP)*, 3:981–984, 2000.

[71] D. Zhong and S. Chang. Spatio-temporal video search using the object based video representation. 3:21, 1997.