Patch-based methods for variational image processing problems

THÈSE N^{O} 5693 (2013)

PRÉSENTÉE LE 26 AVRIL 2013

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE TRAITEMENT DES SIGNAUX 2
PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Emmanuel D'ANGELO

acceptée sur proposition du jury:

Prof. J.-Ph. Thiran, président du jury
Prof. P. Vandergheynst, directeur de thèse
Prof. J.-F. Aujol, rapporteur
Prof. P. Fua, rapporteur
Prof. L. Jacques, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2013

2013

La tradition ne consiste pas à conserver des cendres,
mais à entretenir la flamme.

–Jean Jaurès–

Abstract

Image Processing problems are notoriously difficult. To name a few of these difficulties, they are usually ill-posed, involve a huge number of unknowns (from one to several per pixel!), and images cannot be considered as the linear superposition of a few physical sources as they contain many different scales and non-linearities. However, if one considers instead of images as a whole small blocks (or patches) inside the pictures, many of these hurdles vanish and problems become much easier to solve, at the cost of increasing again the dimensionality of the data to process.

Following the seminal NL-means algorithm in 2005-2006, methods that consider only the visual correlation between patches and ignore their spatial relationship are called *non-local methods*. While powerful, it is an arduous task to define non-local methods without using heuristic formulations or complex mathematical frameworks. On the other hand, another powerful property has brought global image processing algorithms one step further: it is the *sparsity* of images in well chosen representation basis. However, this property is difficult to embed naturally in non-local methods, yielding algorithms that are usually inefficient or circonvoluted.

In this thesis, we explore alternative approaches to non-locality, with the goals of *i)* developing universal approaches that can handle local and non-local constraints and *ii)* leveraging the qualities of both non-locality and sparsity. For the first point, we will see that embedding the patches of an image into a graph-based framework can yield a simple algorithm that can switch from local to non-local diffusion, which we will apply to the problem of large area image inpainting. For the second point, we will first study a fast patch preselection process that is able to group patches according to their visual content. This preselection operator will then serve as input to a social sparsity enforcing operator that will create sparse groups of jointly sparse patches, thus exploiting all the redundancies present in the data, in a simple mathematical framework.

Finally, we will study the problem of reconstructing plausible patches from a few binarized measurements. We will show that this task can be achieved in the case of popular binarized image keypoints descriptors, thus demonstrating a potential privacy issue in mobile visual recognition applications, but also opening a promising way to the design and the construction of a new generation of smart cameras.

Keywords: Image processing, Inverse problems, Non-local algorithms, Social sparsity, Local binary descriptors, Privacy

Résumé

Malgré la banalisation croissante des images numériques, leur traitement et leur exploitation restent des problèmes compliqués. Ils impliquent généralement de une à quelques inconnues par pixel (soit plusieurs millions avec les dispositifs actuels !) dégradés par l'application d'opérateurs non-inversibles et non-linéaires. De plus, même une image simple contient plusieurs textures d'échelles et de caractéristiques différentes. Toutefois, si l'on considère une image comme l'assemblage de petits blocs de pixels (ou patches), beaucoup de ces difficultés disparaissent au prix d'une augmentation de la dimensionnalité des données à manipuler.

Depuis l'introduction de l'algorithme des moyennes non-locales vers 2005-2006, les méthodes qui exploitent les corrélations visuelles entre les patches d'une image et ignorent leurs relations spatiales sont désignées sous le vocable de *méthodes non-locales*. Bien que conceptuellement puissantes, ces méthodes sont difficiles à formuler sans recourir à des heuristiques ou à des équations mathématiques alambiquées. Parallèlement, l'introduction d'un autre concept a permis aux méthodes locales de réaliser un bond en performance : il s'agit de la *représentation parcimonieuse* des images dans des bases bien choisies. Toutefois, ce concept est délicat à appliquer aux approches non-locales, conduisant le plus souvent à des algorithmes numériquement peu efficaces.

Dans cette thèse, nous explorons des approches non-locales alternatives, afin de *primo* développer des algorithmes universels capables d'embrasser simultanément contraintes locales et non-locales et *secundo* allier les qualités des approches non-locales et parcimonieuses. Pour le premier point, nous verrons que considérer les données à traiter comme les nœuds d'un graphe conduit à un algorithme simple de diffusion locale et non-locale que nous appliquerons à la restauration de larges zones dans une image. Pour le second point, nous étudierons une méthode rapide de tri des patches par leur contenu visuel qui alimentera un processus de création de groupes parcimonieux de patches parcimonieux. Ce double niveau de parcimonie, ou parcimonie sociale, nous permettra d'exploiter toute l'information visuelle contenue dans les patches.

Enfin, nous étudierons le problème de la reconstruction de patches à partir de quelques mesures binaires. Nous montrerons qu'il est possible, pour une certaine famille de descripteurs, de recréer correctement le contenu visuel à leur origine. Si ce résultat révèle une faiblesse dans la protection de la vie privée des utilisateurs d'applications mobiles de reconnaissance visuelle, il permet aussi d'envisager la création d'une nouvelle génération de caméras réellement intelligentes.

Mots-clés : Traitement d'images, Problèmes inverses, Algorithmes non-locaux, Parcimonie sociale, Descripteurs binaires locaux, Vie privée

Acknowledgments

A PhD thesis is quite an adventure! The present work makes no exception, and the list of people who influenced me and helped me along the way is quite long.

I shall start by thanking the people here in EPFL who created the stimulating environment that made this achievement possible. First of all, I would like to express my gratitude towards my advisor, Prof. Pierre Vandergheynst, for trusting in my ability to conduct this research. He gave me the opportunity to start this thesis and consistently helped me find my way when I felt lost. Now that I leave the lab, he will surely miss my daily optimism and my chessboards. Talking about the lab, this thesis was indeed a team work. Luigi, Karin, Anna, those where great moments when we shared coffee and positive thoughts. Gilles, thank you for sharing some time with me, answering my never-ending flow of questions. Yannick, Xiaowen, Benjamin, I am sorry that we did not have more time to work together. I would like to thank also the lab and doctoral school staff, especially Rosy and Corinne, for the time they dedicated in helping me. Last but not least, Alex, we've shared an office, but we've also shared a bit more with all the talking and thinking about the future. I wish you all the best for your new careers.

I would also like to take the opportunity here to thank again my thesis committee, for the time they spent studying my research and for their questions and comments, especially from Prof. Jean-François Aujol and from Prof. Laurent Jacques. Having an external point of view was invaluable to me, especially from such experts in the field. The final version of this manuscript did benefit a lot from their feedback.

This adventure did not exactly start here in Lausanne. Many years ago, there was a seed planted by Jean-Louis Morvant, who directed the curiosity of a young teenager towards photography, cinema, and pictorial arts. This decisive encounter led me later to studying image processing technologies. I am deeply sorry that he is no longer with us to read this manuscript.

Several years later, more people encouraged me in starting a research work and transmitted me their demanding methodology. I am very happy and very proud to have been part of the GIP experience, although it was already reaching its end. I would like to thank Frédéric C., JBT, Bertrand, Véronique, Jérôme, Frédéric D., Sylvain, Jocelyne and all the GIP staff. I learned from your contact, your lessons and your examples. I would also like to thank Prof. Jean-Michel Morel from the ENS Cachan for the work we started together but that I never finished. Some answers to our questions can hopefully be found here at last.

Life is not all work, and work is not all the reasons that made me move to Switzerland. I would like to thank the friends here in Lausanne: Hélène and Momo for their constant support, their

door always open and the goods times together; Matt, for your cocktails and for proofreading this manuscript, making it sound (a bit) more English than French. Finally, I would like to say Annick a heartful “thank you”. Thank you for letting me the chance to do this research work until the end, thank you for the life that we are building here. I know that you are by my side, and I hope I am here for you too.

Contents

List of Figures

1 Introduction

1.1 Motivations

Blocks taken inside images, or patches, are at the very heart of many Image Processing applications. Examples include compression standards such as JPEG decompose images into small blocks of pixels, movie compression algorithms add a step of block motion estimation to this, and block matching is still a competitive approach to motion estimation and structure-from-motion tasks. Patches are indeed handy because they allow us to assume local properties of images that have little chance to be true when looking at the big picture: locally, colors are almost constant, textures almost periodic, or 3D transforms become affine. Hence, considering only *small* image parts does greatly help to simplify and justify the assumptions about the nature of the said data.

Furthermore, images have an important property: their content is *highly redundant*, and for small enough blocks the chances are that similar patches can be found all over the *entire image* or group of video frames to process and even inside other *non-related* images. This is of course even stronger with movies, and we experience it almost daily since this redundancy is leveraged by highly effective compression schemes such as those described in the H.264 standard and that allow for the transmission of small sized video files over wireless networks such as 3G mobile phone networks.

This phenomenon is emphasized by the current trend in the imaging sensor industry, which is packing more and more pixels into the same light-sensitive area. For example, in 1999 the Nikon Corporation released to market a digital camera, the Nikon D1, embedding 2.7 Millions of Pixels (MP) on a sensor of 23.7-by-15.6 mm. Its 2012 distant sibling, the D3200 model, still has a sensor of the exact same physical size but filled with 24.2 MP. This higher density of photosensitive elements leads to a better sampling (in the Shannon-Nyquist sense) of the captured scenes, but many of the additional pixels are actually used to capture slowly-varying quantities such as the color of the sky, hence bringing ever more redundancy into the acquired data.

Besides *redundancy*, there is another important property of image patches that can be (and is more and more often) exploited: it is their *sparsity*. In some well chosen basis, patches from natural images will have *sparse* representations, *i.e.*, few non-zero coefficients are required to correctly describe their content. This fact is now widely accepted for basis like the harmonic functions (DCT) or wavelets and has been successfully applied to various problems such as image denoising [1], MRI reconstruction or people detection [2]. In the case of Image Processing problems, the sparsity property is usually enforced on the image as a whole, with the significant exception of compression. If you think however of an image depicting a group of people, then intuitively the decomposition in a DCT basis of small patches is going to be much more sparse (e.g. due to periodic patterns on clothes or the bricks of a wall) than the coefficients computed on the entire image (superposition of several textures separated by sharp edges).

In this thesis, our main goal is to explore the use of patches as the principal primitives of interest. By exploiting their redundancy and imposing different forms of sparsity, we will revisit some classical Image Processing problems with a patch-centric viewpoint in a mathematical framework that allows leveraging of both local patch-wise and global constraints. Finally, we will take our reflection even further by considering a case where only some binarized measures, and not pixel data, is available. We will demonstrate that, under certain conditions and with the proper regularized inverse problem approach, it is actually feasible to reconstruct plausible image patches from these few measurements, which not only has practical implications but should also help the community in assessing how much information is actually contained inside these binary descriptors.

1.2 The rise of non-locality

Before we describe in more detail in chapter 2 the seminal non-local means (NL-means) algorithm [3], we briefly introduce here the intuition behind non-local image processing. Patch-based methods were introduced first for texture synthesis, because they allow to bypass the important difficulty of accurately and realistically modeling image textures. They were then adapted to the problem of filling-in image regions (also called inpainting) because they don't require a prior correct identification of the different textures.

1.2.1 The problem of textures

Surprisingly, the intrinsic redundancy in the image data was ignored by a large part of the researchers involved in Image Processing (except of course for the compression) for a long time period. People focused instead on the development of global mathematical models of images that became more and more complex over the years, possibly involving layers of hidden dependancies or additional functional spaces. This complexity race comes from the fact that these works faced a major difficulty, which is creating and applying a mathematical model for image textures. They are indeed very hard to consider in theoretical models: empirically, a texture is made of elementary elements (or texton) that are reproduced *almost* identically, following an *almost* regular pattern and at *approximately* the same scale.

The task of correctly handling all these *almost true* properties inside mathematical equations is made even harder by the implicit scale selection that the human brain applies, picking up or discarding frequencies depending on the task at hand. If we have a look at the scene depicted in

Fig. 1.1, we can identify three main phenomenons with different spatial scales:

- the gravel on the grounds makes a first, high frequency, texture;
- the sake barrels make a coarse scale texture of almost identical patterns since the drawing on the side of each barrel varies. Furthermore, the different groups of barrels can also be considered as a coarser scale texture themselves;
- the fence is also an intermediate texture with a privileged direction of oscillation.

It is indeed an arduous task to find a unique mathematical framework (be it a filter bank, an association of functional spaces...) to consider all these image areas at once. Consequently, texture synthesis algorithms using such approaches were hardly successful beyond stationary stochastic textures with very few geometric structures.

Figure 1.1: *In this image, a viewer can identify at least three textures of very different scales: the gravels on the ground, the fence and the sake barrels. It is however a typical real image, not an image that was specifically created for this purpose. Hence, Image Processing algorithms need the ability to adapt themselves to such varied image content.*

1.2.2 Prior patch-based methods

This situation gradually changed around the year 2000 with the appearance of the so-called *example based* algorithms, that demonstrated that a single image could contain enough information to address the problems of super-resolution [4], texture synthesis [5] and even large region inpainting [6]. We briefly describe here these early works because they can help in intuitively understanding the power of non-locality.

When applied to texture synthesis [5, 6], the main idea of example-based methods is to bypass the difficult mathematical modeling of textures by considering the reference image content as an example source. Instead of learning or training a local appearance model, the appearance of the reference is duplicated by literaly copy-and-paste operations: smaller blocks (examples) are chosen inside the reference data, then copied and merged in the area where the new texture should be synthesized under the constraint of avoiding the introduction of visual discrepancies. For the super-resolution task [4] [1], Freeman's approach consists in a first training step where the correspondences between the blurred low resolution patches and their sharp high resolution counterparts are learned. Then, a global optimization process is applied in order to produce a zoomed image that is plausible. This global step is required because several high resolution patches can yield the same low resolution patch. Hence, one needs a way to find the correct one among these, *i.e.*, , the one that will not introduce further discrepancies when considering the whole image at once.

From this short description, it is easy to understand why these method are coined « example based » : instead of explicitly modeling a complex mathematical process (usually a texture or its degradation through an imaging system), they rely on learning examples either from the input image itself or from a prior training step. The complex problem then becomes simpler as expressed as a problem over the examples.

Starting with NL-means, an algorithm is said to be *non-local* when the considered primitives are patches, and examples of patches are searched in the entire input data, without considerations of spatial relationship. The so called *non-local algorithms* will exploit the same ideas as example-based methods: exploring the whole image in order to detect redundant patches (the examples) that will be processed jointly, thus preserving the textures (because of their redundancy) while ignoring the noise (that is itself too random to be captured as a redundancy).

Relations to Machine Learning. As a side note, please note that example-based algorithms usually do not belong to the Machine Learning world. While the input examples could be considered at first glance as members of a training set, there is no patch or image model being trained here. An objective function on the whole image is optimized instead, using only the input examples or some smaller parts therein. In the works presented above, only the super-resolution algorithm of Freeman relies on some Machine Learning background. It is however bounded to the low resolution - high resolution correspondence process and to the global image formation smoothness. Hence, there is no implicit or explicit texture content model training: everything is contained in the appearance of the input examples.

1.3 Contributions of the thesis

In this thesis, we explore the possibilities offered by the patch-based approach for the processing of digital images and movies. Our goal is to move the non-local algorithms from a heuristic to a mathematical framework in order to allow their generalization to various image processing tasks. Our main contributions are:

– a variational approach to non-local inpainting that can unify the previous diffusion based and example based methods;

1. Or more accurately, the image zooming problem.

– a fast and intuitive algorithm for similar patch queries, thus allowing real non-local algorithms. We further apply this speed up method to a super-resolution problem where patch queries happen in an image sequence;
– using the proposed patch retrieval algorithm as a clustering tool, we propose an efficient jointly sparse non-local denoising algorithm that avoids learning patch dictionaries beforehand and that can be applied indifferently to image and video data;
– considering then single patches, we demonstrate that plausible patches can be reconstructed from a family of binary descriptors that are becoming more and more popular in Computer Vision applications. This result can have many further implications, from the design of smart cameras recording quantized descriptors to the development of better patch descriptors.

1.4 Thesis structure

This thesis is organized as follows: chapter 2 introduces non-locality as an example-based approach by studying the seminal NL-means algorithm [3] and showing how example-based approaches can be embedded in a variational framework through the example of inpainting. Then, chapter 3 presents the strategies that were proposed to reduce the computational burden of the naive non-local algorithms. By identifying the desirable properties of such a strategy and the weaknesses of the previous ones, we naturally arrive at the proposal of two algorithms for fast non-local patch queries: the first one based on spectral hashing, which is the most efficient and versatile, and the second one which is best fitted for hardwares with limited capacity. In chapter 4, we start by introducing the notion of *non-local sparsity* through the example of the pioneering BM3D algorithm [7]. Then, we propose a simple jointly sparse non-local denoising algorithm that does not require any additional dictionary training by interpreting the output of the spectral hashing as patch clusters.

Finally, chapter 5 moves away from non-local relationship to consider single patches instead. We address the question of how much visual information some binarized patch descriptors contain by reconstructing patches from them. We formulate this problem as an inverse problem in both the binarized and non-binarized cases and propose iterative algorithms to solve them.

We conclude in chapter 6 with a summary of our contributions and proposals of future work and investigations.

2 Exemplar-based non-locality

In this chapter, we present the genuine Non-Local means (NL-means) algorithm. This algorithm is important not only for the quality of its results but also because of its elegant and simple formulation. Hence, it is a good entry point to non-local image processing and the intuitions behind it.

While NL-means was presented as an exemplar-based method, we review several subsequent variational interpretations that were proposed after its introduction. Finally, we build on one of these variational frameworks to design a non-local image inpainting algorithm expressed as an inverse problem.

2.1 Non-Local means

2.1.1 The intuition behind non-local denoising

Denoising as averaging. Let us start with a simple example. We suppose that we want to estimate the value of some constant phenomenon (a pressure, a temperature…). We have taken n independent measures $\{x_i\}_{i=1}^n$ that are corrupted by some additive random noise. How can we obtain a good estimate of the underlying value ? A simple answer can be found in any introductory Probability textbook: since the noise is random, the *empirical mean* $M(x)$ defined by

$$M(x) = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.1}$$

is a good estimator of the true value, and the variance of the noisy measurement is reduced by a factor $\sqrt{n}$. Directly translated to images by assuming that connected pixels were likely to represent the same object, this simple method has led to local averaging schemes, that effectively remove

some noise but at the cost of introducing some undesirable blur on textures and at the edges of objects (Fig. 2.1).

Why is it that this simple denoising approach is so destructive regarding the image content ? It is because the images do not meet previously made fundamental assumptions: all the pixels acquired do not represent the same physical phenomenon (color, amount of light) in the observed scene. Hence, blindly averaging pixels just because they are spatially close leads to mixing unrelated colors or light intensities and eventually introduces blur.

Beyond averaging: complexity. In order to tackle this problem, one might be tempted to use some *anisotropic* diffusion, (by opposition to the isotropic diffusion taking place in the case of spatial averaging), that is, the diffusion process is not allowed to jump above object boundaries anymore. A famous example of this approach is the Rudin-Osher-Fatemi functional [8] that aims at minimizing the Total Variation (TV) of an image, *i.e.*, the sum of the magnitude of its gradient. However, this approach ignores fine textures, hence also yielding undesirable artifacts (Fig. 2.1). Again one can tackle this new problem by designing algorithms that separate the image structure and texture parts [9, 10], then the image structure, texture and noisy parts [11], in a more and more complicated process.

This everlasting process comes from a simple fact: it is mathematically very difficult to define what a texture is. As pointed out in the introduction (see chapter 1), even putting in words a correct and accurate statement of what a texture is is a challenging task. Hence, separating an image into its structural, textural and noisy components is an arduous task, thus justifying the need for alternative approaches.

Redundancy and simplicity. Let us suppose that we want to come back to a simple denoising-by-averaging procedure. To avoid the introduction of unwanted blur, we need to avoid mixing together pixels that are unrelated, that is pixels that belong to different textures or have different colors. Since we want to avoid the chicken-and-egg problem of segmenting the image into objects and textures first, we will build a simple pixel descriptor by considering not only the pixel value but also a small square around it in order to capture the particular texture element it belongs to. This small square is called a *patch*.

As we saw at the beginning of this section that n measures yield an improvement of $\sqrt{n}$ in the quality of the mean estimator, we need to gather as many examples of our texture element as possible. One can remark that the image content is usually highly redundant (especially at the patch level) and there may be several occurrences of an object, thus we explore the entire image in order to maximize our chances of finding similar patches.

Finally, we need a criterion to tell us that the currently explored patch depicts the same texture element as the one to denoise. A simple and classical visual similarity measure is to take the squared error between two patches. This squared error can be divided by a decay parameter to take into account different levels of noise, and filtered by a decreasing exponential to obtain a similarity score between 0 (completely unrelated patches) and 1 (perfectly identical patches). Then, given these similarity weights, we can replace the uniform averaging of Eq. (2.1) by a weighted mean, or more accurately by a center of mass in order to preserve the range of the values.

This is exactly the intuition behind the NL-means algorithm, that we are going to describe

(a) Original image

(b) Noisy image

(c) Isotropic diffusion

(d) Anisotropic diffusion

Figure 2.1: *Top row: original image and with Gaussian additive noise. Bottom row: denoising with isotropic diffusion or heat flow (left) and with anisotropic diffusion (right). The isotropic diffusion has removed some noise, at the cost of having a blurred output. The anisotropic diffusion (obtained by minimization of the Rudin-Osher-Fatemi functional) is less destructive but creates flat areas that damage the textures and the shadings (on the skin and the hair for example).*

formally now.

2.1.2 Mathematical formulation of NL-means

From now on, we suppose that we are given an image I defined on a domain $\Omega \subset \mathbb{R}^2$, usually a rectangle, and with values in $\mathbb{R}$ (for grayscale images) or $\mathbb{R}^3$ (for color images). We write $\boldsymbol{x}, \boldsymbol{y} \in \Omega \times \Omega$ two pixel locations. We define a patch extraction operator R that returns the pixels inside a squared $\sqrt{d} \times \sqrt{d}$ neighbourhood around $\boldsymbol{x}$ and stacks them in a vector using lexicographic ordering:

$$R : \boldsymbol{x} \mapsto R(\boldsymbol{x}) = (I(x_1) \dots I(x_d))^T \in \mathbb{R}^d. \tag{2.2}$$

The lexicographic ordering process is illustrated in Fig. 2.2.

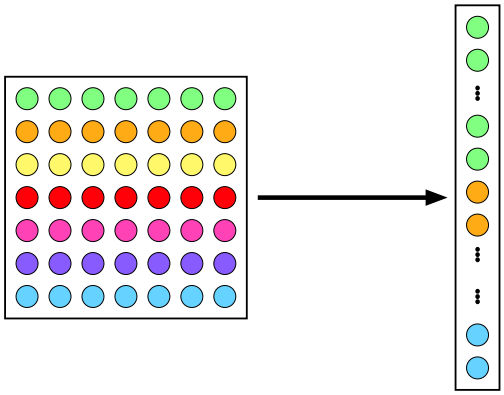


Figure 2.2: *Patch vectorization by lexicographic ordering. The original patch is on the left. Each colored circle denotes an original pixel value.*

Given two patches, we define the distance between them either as the usual squared Euclidean distance:

$$d(R(\boldsymbol{x}), R(\boldsymbol{y})) = \|R(\boldsymbol{x}) - R(\boldsymbol{y})\|_2^2 = \sum_{i=1}^{d} (R(\boldsymbol{x})_i - R(\boldsymbol{y})_i)^2 , \tag{2.3}$$

or as the Euclidean distance after multiplying the patches by a 2D Gaussian window $\boldsymbol{\mathcal{G}}^a$ of width a:

$$\|R(\boldsymbol{x}) - R(\boldsymbol{y})\|_{2,a}^2 = \sum_{i=1}^{d} \left((R(\boldsymbol{x})_i - R(\boldsymbol{y})_i) \times \boldsymbol{\mathcal{G}}_i^a \right)^2 \tag{2.4}$$

Given the distance between two patches, on can compute their visual similarity, normalized into the interval $[0, 1]$ (ranging from no similarity to identical) by a simple exponential filtering:

$$w(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{d(\boldsymbol{x},\boldsymbol{y})}{h^2}} = e^{-\frac{\|R(\boldsymbol{x}) - R(\boldsymbol{y})\|_2^2}{h^2}} . \tag{2.5}$$

The parameter h controls the *decay* of the exponential function: for small h, only very similar patches will have a significant similarity score and the filter will be very selective. On the other hand, a large h will attribute a more uniform importance to all the patches of the input image, yielding a stronger denoising but also blurring more aggressively the image structures.

Finally, the denoised pixel value is obtained by computing the weighted average over an area centered on $\boldsymbol{x}$ and with radius ρ, that can extend to the whole image:

$$NL(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \sum_{\boldsymbol{y}:\|\boldsymbol{x}-\boldsymbol{y}\|\leq\rho} w(\boldsymbol{x},\boldsymbol{y})I(\boldsymbol{y}). \tag{2.6}$$

The term $Z(\boldsymbol{x})$ is the sum of all the contributions of the pixels from the circle of radius ρ centered on $\boldsymbol{x}$:

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}:\|\boldsymbol{x}-\boldsymbol{y}\|\leq\rho} w(\boldsymbol{x},\boldsymbol{y}). \tag{2.7}$$

It acts as a normalization factor that enforces the stability of the dynamic range of the image: the value $NL(\boldsymbol{x})$ can be interpreted as the center of mass of the values of the pixels in the circular domain centered on $\boldsymbol{x}$, with the weights given by Eq. (2.5).

Remark 1. *In their seminal paper [3], Buades et al. do not define a patch extraction operator R but use the Gaussian $\mathcal{G}_a$ instead. Indeed, it suffices to define the spatial extent of a patch, and in practice implementations will actually cut the image at a given number of times of the variance. However, having an explicit notion of finite patch is interesting in several ways: for consistency with subsequent non-local algorithms, to emphasize the exemplar-based nature of NL-means, and to make the high dimensionality of the space of patches more obvious.*

2.1.3 Original interpretation of NL-means

While the authors of NL-means did claim that they found their inspiration in exemplar-based texture synthesis algorithms such as [5], they have readily given a mathematical interpretation of NL-means as a generalization of the neighbourhood filters introduced by Yaroslavsky [12] and popularized by Lee and his sigma filter [13].

For a given pixel location $\boldsymbol{x}$, the output of a neighbourhood filter is computed as:

$$YNF_{h,\rho}(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \sum_{\boldsymbol{y}:\|\boldsymbol{y}-\boldsymbol{x}\|\leq\rho} I(\boldsymbol{y}) \exp\left(-\frac{|I(\boldsymbol{x}) - I(\boldsymbol{y})|^2}{h^2}\right), \tag{2.8}$$

where the neighbourhood of the pixel $\boldsymbol{x}$ is again given by a circle of radius ρ. As before with NL-means, $Z(\boldsymbol{x})$ is a normalization constant that ensures the stability of the dynamic range of the image. The links between Yaroslavsky's filters and popular filters are clear given Eq. (2.8):

- Lee's sigma filter is an instance of Eq. (2.8) but with an additional *preselection threshold*, in order to take into account in the integral only the pixels whose value is close (up to a fraction of the noise variance) to the value of $\boldsymbol{x}$;
- the bilateral filter [14] retains the term in Eq. (2.8), but modulates it with a similarly shaped term that depends on the distance between the pixels, and not on their value;
- NL-means extends it by considering patches instead of pixel values.

Remark 2. *Note that by commodity most NL-means implementations compute the similarity weights in a* learning window *(or search window) smaller than the entire image, as Eq.* (2.8) *also suggests. In that sense, they can be considered as semi-non-local, while full non-locality is achieved when the radius ρ is big enough to extend the neighbourhood to the whole image. Although we have introduced this learning neighbourhood centered on $\boldsymbol{x}$, the filter defined in Eq.* (2.6) *is still non-local*

in the sense that the spatial distance between the pixels $\boldsymbol{x}$ and $\boldsymbol{y}$ does not play any role in the computation of the output value $NL(\boldsymbol{x})$. Originally, using learning window smaller than the entire image allowed to control the computation times. We will also show in chapter 4 that smaller learning windows yield better performance, which is kind of paradoxal for a non-local algorithm.

Bayesian interpretation. In order to analyze NL-means and extend this non-local approach to various imaging problems, different authors have re-interpreted this algorithm in several mathematical frameworks. The first authors to propose a substantially different interpretation of NL-means were Kervrann *et al.*in [15]. In this work, they introduced ideas that were proposed at the same time by the more famous BM3D algorithm [16], namely similar patch grouping, iterative application of the filter and joint denoising, but using a Bayesian estimation framework. BM3D and these three properties are described in more detail in chapter 4. For now, it suffices to emphasize the differences with NL-means:

- inside the learning window, only the patches that are very similar to the patch of interest are retained, and the other ones are discarded;
- the observed noisy patches are considered as members of a dictionary from which optimal clean patches can be inferred at the cost of an important approximation to make the computations tractable (the dictionary members should be clean patches without noise instead of noisy ones);
- by applying iteratively the denoising procedure, this approximation becomes more and more legitimate since the noise is progressively annihilated.

2.2 Non-local inverse problems

2.2.1 Introducing inverse problems in imaging

In its original formulation, NL-means works only as a clean patch estimator from noisy observations. In order to generalize it to more image processing tasks, one needs a way to introduce some operators that will represent the transformations that are applied to an image during its acquisition process, such as the convolution by a blur kernel, a downsampling or a Fourier transform (in the case of Fourier imaging).

Of course, this requirement appeared well before the introduction of NL-means, and several solutions were proposed. Among these, regularized inverse problems are a convenient way to separate a task between finding a good estimate that correctly explains the observations (including the knowledge of the physical measurement process) and some a priori constraints that describe the qualities that a good solution should fulfill (the regularizer). While many imaging problems are actually ill-posed, the search for a solution is often not a completely blind search. Prior knowledge is usually available about the shape of a good solution. This prior allows us to pick the best one among the candidates that successfully explain the observation.

In its generic form, an inverse problem approach consists in recovering about an object of interest from some observations or measurements. In the case of Image Processing, the observations will often be one or several images. Furthermore, since information is lost in the image acquisition process (via aliasing, blur, 3D-to-2D projection...), imaging problems are usually ill-posed in the sense of Hadamard. To overcome this difficulty, a classical choice is to add a regularization con-

straint that will stabilize the solution and embed additional a priori domain knowledge about the shape and the properties of a good solution to the problem at hand.

Such a regularized inverse problem can be cast as finding a solution $\boldsymbol{x}^{\text{opt}}$ of the following minimization program:

$$\boldsymbol{x}^{\text{opt}} = \arg\min_{\boldsymbol{x}} \; f(\boldsymbol{x}, \boldsymbol{y}) + \lambda g(\boldsymbol{x}), \tag{2.9}$$

where $\boldsymbol{y}$ is the observed signal or image. The term $f(\boldsymbol{x}, \boldsymbol{y})$ binds the current solution to the observation: if $\boldsymbol{x}$ is very unlikely to produce the observation $\boldsymbol{y}$, its value will be high. It incorporates our prior knowledge on the physical model of creation of the observations $\boldsymbol{y}$. The second term $g(\boldsymbol{x})$ is the regularization constraint and will penalize solutions $\boldsymbol{x}$ that do not have the desired shape. If one (or both) of the functions f and g has some nice mathematical properties such as being convex, smooth, or Lipschitz-differentiable, then it is very likely that a converging optimization scheme solving Eq. (2.9) will exist [17]. Furthermore, via the use of proximal operations [18], practical implementations of the chosen scheme are usually easy to write and computationally efficient.

The first term $f(\boldsymbol{x}, \boldsymbol{y})$ is called the *data term* because it will measure the spread between the current estimate $\boldsymbol{x}$ and the observation $\boldsymbol{y}$. In a Bayesian framework, it will naturally be linked with the *likelihood* of a tentative solution given the observation. Calling A a linear degradation operator, classical choices for $f(\boldsymbol{x}, \boldsymbol{y})$ include:

- $f(\boldsymbol{x}, \boldsymbol{y}) = \|A\boldsymbol{x} - \boldsymbol{y}\|_2^2$, *i.e.*, the squared error between the tentative solution $\boldsymbol{x}$ (after applying the operator A) and the observation $\boldsymbol{y}$. This data term corresponds to an observation $\boldsymbol{y}$ that was contaminated by some white noise;
- $f(\boldsymbol{x}, \boldsymbol{y}) = \|A\boldsymbol{x} - \boldsymbol{y}\|_1$, *i.e.*, the ℓ_1-norm of the error. This data term is robust to pointwise strong noise values, such as defects in the pixel grid (meaning there is no observation), and more generally to phenomena that create large but spatially sparse errors.

Naturally, many different data terms were proposed in the literature in order to deal with specific noise distributions: see [19] for an example of different data terms applied to the same task with the same prior and [20, 21] for examples of non-white noise handling.

The second term is the prior that will stabilize the optimization process in the presence of noise, and that will help propagating information into the image parts where the data term lacks information. In Bayesian frameworks, it is linked to the probability that a given solution exists under the prior model. Classical priors include:

- $g(\boldsymbol{x}) = \|\nabla \boldsymbol{x}\|_2^2$, also called the Tikhonov regularization. Penalizing the gradient of $\boldsymbol{x}$ allows to avoid oscillating solutions, which are often undesirable (ringing artifacts, Gibbs effect, etc.). Since the squared norm of the gradient is penalized, it produces smooth solutions without discontinuities. One can prove that it corresponds to a Partial Derivative Equation (PDE) whose solution is the heat flow, *i.e.*, isotropic diffusion, which explains why it tends to produce blurry outputs;
- $g(\boldsymbol{x}) = \|\nabla \boldsymbol{x}\|_1$, also known as the Total Variation [8]. Unlike the Tikhonov regularization, it corresponds to anisotropic diffusion and will respect large object boundaries, at the cost of producing solutions that may be too flat or piecewise constant. Note that it is also popular because fast and stable minimization algorithms are available, see [19, 22, 23];
- $g(\boldsymbol{x}) = \|W\boldsymbol{x}\|_1$, where W is a wavelet analysis operator. This prior promotes the sparsity of $\boldsymbol{x}$ on a wavelet basis.

These priors tend to favor non-oscillating solutions which is usually the desired behavior. Of course, different applications such as texture extraction can require the opposite: see for example [24] and [11] for priors that are defined on Sobolev spaces.

Since inverse problems are a convenient approach in Image Processing, non-local inverse problems were proposed to interpret and extend the original NL-means patch-wise estimation algorithm.

2.2.2 Variational interpretations of NL-means

Working on the extension of NL-means to the super-resolution task [1], Protter *et al.* proposed in [25] a variational formulation, classically made of two terms:

$$E_{Protter}(\boldsymbol{x}) = \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{1}{4}\sum_{u\in\boldsymbol{x}}\sum_{v\in\boldsymbol{y}} w(\boldsymbol{u},\boldsymbol{v})\|R_{\boldsymbol{x}}(\boldsymbol{u}) - R_{\boldsymbol{x}}(\boldsymbol{v})\|_2^2, \tag{2.10}$$

where $\boldsymbol{x}$ is the unknown image to estimate, $\boldsymbol{y}$ is the observed noisy image, and $R_{\boldsymbol{x}}(\boldsymbol{u})$ extracts a patch centered on u in the image $\boldsymbol{x}$. The weights $w(u, v)$ are learned by applying to Eq. (2.5) on the observed image $\boldsymbol{y}$.

The first part of Eq. (2.10) is simply a data term that binds the estimate to look like the input image, and the second part imposes that the patch similarities in the estimate $\boldsymbol{x}$ have the same shape as in the input image $\boldsymbol{y}$. As Kervrann *et al.*, they rely on a Bayesian framework to interpret the energy function defined by Eq. (2.10). The data term is indeed the log-likelihood of an image $\boldsymbol{x}$ given an observation $\boldsymbol{y}$ contaminated with white noise. The second term is a penalty on badly shaped estimates $\boldsymbol{x}$. Therefore it is a prior on the resulting image and it is interpreted as minus the log of the probability of $\boldsymbol{x}$ to exist.

This formulation as an inverse problem is interesting because it allows the introduction in the equations of some additional blurring and downsampling operators which are physically involved in the super-resolution process. However, the definition of the weights $w(\boldsymbol{u},\boldsymbol{v})$ as priors is not satisfying, because in practice it is computationally too costly to update these weights and only very small learning windows are used to avoid the creation of very large matrices. Hence, we have proposed in [26] to move the non-local weights to the data term, and to use a generic prior such as the Total Variation (TV) [8] instead. The functional to minimize reads then:

$$E_{NL\text{-}TV}(\boldsymbol{x}) = \sum_{\boldsymbol{u}}\sum_{\boldsymbol{v}} w(\boldsymbol{u},\boldsymbol{v})\left(\boldsymbol{x}(\boldsymbol{u}) - \boldsymbol{y}(\boldsymbol{v})\right)^2 + 2\lambda TV(\boldsymbol{x}), \tag{2.11}$$

where $w(\boldsymbol{u},\boldsymbol{v})$ is computed with the patches of the input image $\boldsymbol{y}$. The data term is now a *non-local* reweighted squared error instead of the usual square error. While this may not look like a big difference, in this case the non-local constraint is used to enforce that the patch relationship in the output of the optimization process is the same as in the input image. Thus, it is now exact to not update the weights $w(\boldsymbol{u},\boldsymbol{v})$ and there is no approximation as before. For fixed weights $w(\boldsymbol{u},\boldsymbol{v})$, one can compute the gradient of this non-local data term and a solution to the proximal mapping [17, 18] of TV can be obtained with the efficient FISTA algorithm of [23]. The whole functional can thus be minimized using a Forward-Backward scheme [17].

2.2.3 Non-local inverse problems on graphs

How to represent non-local interactions ? When trying to embed the non-local inter-patches interactions in an usual variational framework, the difficulty of correctly representing these high

1. The super-resolution problem consists of creating an high resolution estimate from a sequence of low resolution frames. For example, it can be applied to obtain an high quality still image from a short movie.

dimensional phenomenon arises quickly. Indeed, patches are already high dimensional vectors. For example, a 7-by-7 pixels patch (a typical value according to Buades in [3]) is already a vector of $\mathbb{R}^{49}$. Also, even when limiting the search window to 21-by-21 pixels, this already makes 441 weights to store *for only one pixel*.

Keeping only the best contributors inside the search window is tempting. However, this leads to a variable number of meaningful neighbouring patches per pixel. Finally, the real question is which kind of representation is best adapted to high dimensional, irregularly sampled data points? Exploring the literature, one can see that *graphs* are a popular tool in this case. Furthermore, they have been extensively studied in semi-supervised Machine Learning [27]. The main interest of graphs in this case is their ability to handle data of arbitrary dimension and sampling. Indeed, building a graph only requires to be able to compute the distances between a data point and its closest neighbours. The exact form of the underlying function or the exact number of these neighbours does not come into play.

Differential calculus on graphs. Formally, a weighted graph $\mathcal{G} = (V, E, w)$ is made of a finite set of *vertices* $V = \{v_1, \dots v_N\}$ linked by *edges* taken in $E \subset V \times V$ with associated *weights* computed using a similarity function $w_{\mathcal{G}} : V \times V \to \mathbb{R}_+$. The vertices correspond to the actual data points. Two vertices u and v are connected by an edge $e = (u, v)$ if $w_{\mathcal{G}}(u, v) > 0$, and we write $u \sim v$. If the weights have the additional property of symmetry $w(u, v) = w(v, u)$, then the graph is said to be *undirected* and $u \sim v$ is equivalent to $v \sim u$. Otherwise, the graph is said to be directed and the order of the vertices forming an edge matters.

Let us write Φ a function defined on the set of vertices V with values in $\mathbb{R}^d$. We recall here the definitions of various differential calculus operators from [28].

Given an undirected weighted graph $(\mathcal{G}, V, w)$, one can define the *directional derivative* of the i-th component Φ_i at the vertex v in the direction given by the edge (u, v) by:

$$\partial_u \Phi_i(v) = \sqrt{w(u, v)}\,(\Phi_i(v) - \Phi_i(u))\,. \tag{2.12}$$

Therefore, the directional derivative is a function defined on the set of edges (it depends on the couple (u, v)), and not on the vertices. The *weighted gradient operator* of Φ_i at a given vertex v is then defined as the vector of all the directional derivatives measured with the vertex v for origin:

$$\nabla_w \Phi_i(v) = [\partial_u \Phi_i(v) : u \sim v]^T. \tag{2.13}$$

The norm of the gradient defines the *local variation* of Φ_i at v, and measures the regularity of Φ_i with respect to the graph connections. It is computed in the usual way:

$$|\nabla_w \Phi_i(v)| = \sqrt{\sum_{u \sim v} (\Phi_i(u) - \Phi_i(v))^2}. \tag{2.14}$$

Since we have defined a gradient operator, it will be possible to define gradient descent schemes to solve various inverse problems defined on graphs.

The divergence operator is defined qualitatively as minus the adjoint of the edge derivative. Let Ψ be a function defined on the edges. By definition, Ψ is then a vector field whose adjoint was computed by Elmoataz *et al.* [28] and is given by $(d^*\Psi)(v) = \sum_{u \sim v} \sqrt{w(u, v)}\,(\Psi(u, v) - \Psi(v, u))$ in

the case of an undirected graph. Applying this result to the edge derivative, we obtain the divergence of Φ_i:

$$\operatorname{div}(\nabla_w \Phi_i)(v) = \sum_{v \sim u} \sqrt{w(u,v)} \Big(\sqrt{w(u,v)}\,(\Phi(u) - \Phi(v)) - \sqrt{w(u,v)}\,(\Phi(v) - \Phi(u)) \Big) \tag{2.15}$$

$$= 2 \sum_{u \sim v} w(u,v)\,(\Phi_i(v) - \Phi_i(u))\,. \tag{2.16}$$

Finally, applying the standard relation $(\operatorname{div} \nabla) = \Delta$, we can compute the Laplacian of Φ_i:

$$\Delta \Phi_i(v) = \sum_{u \sim v} w(u,v)(\Phi_i(v) - \Phi_i(u)). \tag{2.17}$$

Note that if the graph is computed from spatial 4-connectivity relationship with uniform weights equal to 1/4, then these definitions match the usual discretization of the differential operators. As an addendum, one can also remark that the availability of a divergence operator is very interesting because it allows one to adapt modern fast minimization algorithms such as Chambolle's dual approach for TV denoising [22] on graph-defined functions (see for example [29] and [30]) although we use here a more classical gradient descent for simplicity.

Back to imaging: non-local graphs. As stated in the introduction to this section, our goal is to use a graph in order to model the relationship between the patches extracted from an image. Similarly to NL-means, each pixel is first represented by a patch, which is a vector in $\mathbb{R}^d$. Then, the nearest neighbours of a patch centered on a pixel $\boldsymbol{x}$ are given by the patches of the image that are the most similar to itself. There is an immediate one-to-one mapping between image pixels and graph vertices: for each pixel $\boldsymbol{x}$, there is a vertex u whose coordinates in the non-local space are given by the patch $R(\boldsymbol{x})$. The nearest neighbours of $R(\boldsymbol{x})$ correspond to vertices v that should be connected to u by an edge in the graph. The weight of each edge is obtained by computing the similarity function $w(\boldsymbol{x}, \boldsymbol{y}) := w(u, v)$ defined in Eq. (2.5). This weight is positive and tends to 0 when patches are highly dissimilar. Hence, it can be used without modifications to compute the graph weights ($w = w_{\mathcal{G}}$). The construction of the non-local graph is illustrated in Fig. 2.3

Remark 3. *Since the graphs obtained via the process here encode the non-local relationship between the patches of an image, we will designate them in the sequel with the term* non-local graph.

Since the symmetry property $w(u, v) = w(v, u)$ holds, $\mathcal{G}$ is an undirected graph. Consequently, one can immediately turn the NL-means algorithm into an equivalent variational problem on a graph, as remarked in [31]. In this work, Jacobi iterations are then computed in order to solve inverse problems of the form:

$$\mathcal{E}_i^p \left(\Phi_i, \Phi_i^0, \lambda \right) = \frac{1}{p} \sum_{v \in V} |\nabla_w \Phi_i(v)|^p + \frac{\lambda}{2} \|\Phi_i - \Phi_i^0\|_2^2, \tag{2.18}$$

where the parameter p is strictly positive. Note that the case $p = 2$ corresponds to the Tikhonov regularization (on the graph-defined function of course) and $p = 1$ is analogous to the Total Variation.

Remark 4. *The differential operators on graphs introduced in this chapter were defined on symmetric graphs only. Hence, the non-local graphs should be built accordingly. In order to enforce*

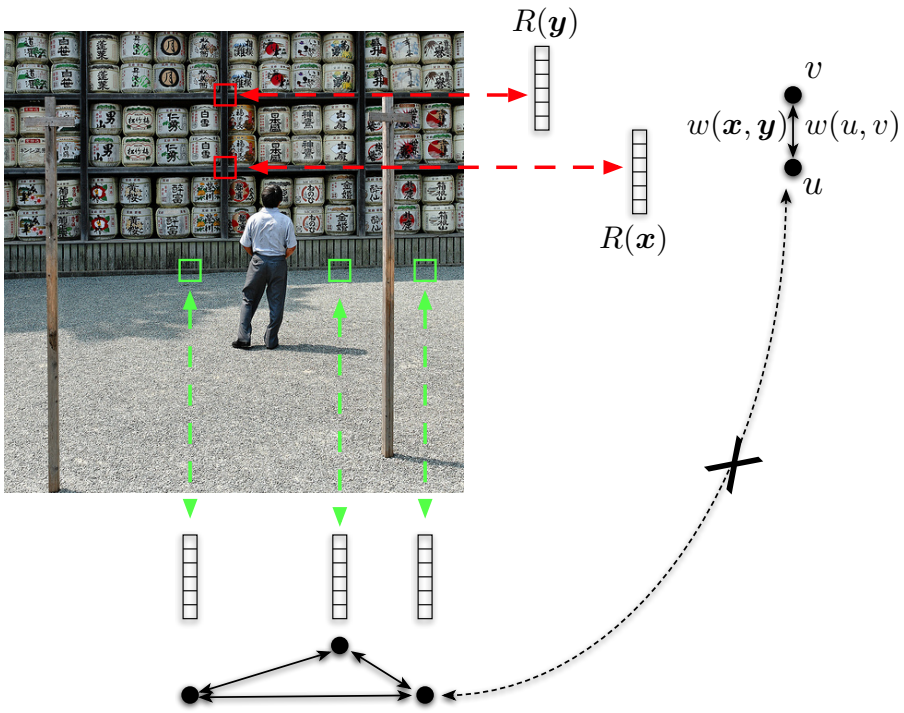


Figure 2.3: *Construction of a non-local graph. Each pixel, e.g., $\boldsymbol{x}$ and $\boldsymbol{y}$, is mapped to a vertex, e.g., u and v. The patches centered on the different pixels are the coordinates of u and v in a high dimensional appearance space. The visual similarity between the patches $w(\boldsymbol{x},\boldsymbol{y})$ is used as the weight of the undirected edge joining u and v. If we limit the connectivity of the graph to the k nearest neighbours of a patch with moderate values of k (typically less than 50 or 100), then the patches form disconnected clusters in the graph.*

this property, while building the graph, each time a vertex v is added as a neighbour of u, then u is also added to the list of the neighbours of v. Hence, one cannot control the number of neighbours of a given vertex in the final graph.

While Elmoataz *et al.* proposed in [31] a general diffusion framework, two alternative solvers were proposed afterwards tailored to the non-local Total Variation on graphs (case $p = 1$ in Eq. (2.18)) in [32] and [29]. These works both report results for different Image Processing problems, including deconvolution and inpainting. The size of the areas to be inpainted is however kept small in both cases. While [32] seems to have a slight advantage in the visual quality of the output and can be adapted to more problems (such as structure + texture decomposition), [29] adapts Chambolle's dual approach to TV minimization for a faster and more robust minimization scheme.

In order to assess the interest of non-local TV minimization, we have proceeded to a simple experiment. We have built a non-local graph, implemented the Jacobi iterations from [31] and ran them in the two cases $p = 1$ and $p = 2$ in Eq. (2.18). The denoising results can be seen in Figure 2.4. The results are almost identical, and textures were preserved in both cases. Hence, non-local TV does not seem to bring any major improvement, at the cost of breaking the differentiability of the regularizer.

Interpretation of the TV model on non-local graphs As shown by our experiments, there is very little difference between the application of a smooth non-local manifold model (Tikhonov regularization) or a piecewise smooth non-local manifold model (TV constraint). Our interpretation of this observation is that the patches form very tight clusters inside the graph, thus making the concept of anisotropic diffusion irrelevant.

The patches that correspond to the same texture element will aggregate in a very compact sub-volume of the non-local space. Hence, there is no real transition (either smooth or modeled by a surface in the non-local space) between the patches corresponding to two different texture elements: they form almost disjoint clusters and one patch does only see for nearest neighbours some patches that correspond to the same texture.

An extreme case of this phenomenon can be obtained with a chessboard image, where there are only two disjoint cycles in the graph. Hence, the actual norm used to penalize the oriented gradient is not important, because two patches are either very similar (and consequently produce a very low gradient for any choice of p) or so far from each other that the edge connecting them has a negligible weight.

(a) Non-local TV (b) Non-local heat flow

Figure 2.4: *Non-local graph-based denoising: TV vs. Tikhonov. Apart from some local contrast change, both results are almost identical. In particular, both cases correctly preserved the textures.*

2.3 Application of non-local graphs to image inpainting

In this section, we will study an extension of [31] to the problem of image inpainting, that we originally proposed in [33]. We will start by briefly describing the inpainting problem and the various attempts at solving it, that can be roughly divided in two groups: exemplar-based or diffusion-based. Since graphs can model any kind of inter-data relationship, depending only on the way connections are made, our initial intuition was that they are good candidates to derive both

local and non-local regularization frameworks. Then, we detail the Jacobi iterations used to solve the inverse problem in this case. Finally, we comment on some results on both synthetic and real images with an emphasis on large area inpainting, which is currently the most challenging task since relatively simple algorithms can correctly reconstruct images with numerous missing random pixels.

2.3.1 Introduction to the image inpainting problem

Image inpainting, also known as image completion, is the problem of finding missing parts of an image using only the available content and some regularization constraints. The disoccluded areas should cause few, if any, visual artifacts, which makes inpainting a difficult image processing problem involving knowledge about image models and regularization techniques. While it has been a long time problem for painting restoration, it has gained in importance with the growth of the digital photography market, since it allows users to remove disturbing elements from their pictures or repair damages such as visible dust on the sensor of the camera or scratches in an old digitized photograph.

Traditional approaches to inpainting try to find a good continuation of the surroundings of the holes, effectively propagating lines inside. Hence, these techniques are also known as *geometry-driven* algorithms. Isophote propagation is obtained by solving partial differential equations such as the heat flow of the Navier-Stokes equation as in [34]. Since this tend to produce blurred estimates, edge-preserving techniques such as Total Variation minimization [35] or curvature motion [36] were used as an alternative. However, these models still lack a support for texture information and are consequently unable to reproduce it. Geometry-based methods can thus hardly be applied to large area inpainting without noticeable visual disturbance.

Since textures, and especially structured ones like brick walls, are hard to model due to their high yet constrained variability, people working in the texture synthesis field successfully applied alternative exemplar-based techniques, which tackle the lack an explicit mathematical texture model. Starting with the work of Efros and Leung [37], exemplar-based techniques take as input a source image containing the desired texture. Then, random parts of the source are extracted and used to fill the larger target picture, with a special treatment to avoid visual discontinuities between neighbouring overlapping patches (see [5, 38] for instance). These techniques were quickly applied to texture inpainting [39]. They are however sensitive to the order in which gaps are filled, which can create disturbing subjective contours. Hence, the authors of [6] defined careful heuristics for choosing which pixels to process first, and managed to successfully inpaint large areas. This type of approach can even be extended to the case of video inpainting [40].

Few attempts were made however to conciliate geometry-driven and exemplar-based techniques in an unified framework. Simultaneous geometry and texture inpainting algorithms that have so far been proposed, such as the work in [41], separate inpainting in two distinct steps dedicated to geometry diffusion and texture synthesis respectively. This approach requires first the decomposition of the image into a geometric part, or sketch, and a textural part, which is still an arduous task [9]. Specifically, the texture function spaces used contain only very regular oscillating patterns, which is seldom the case for real life textures such as the aforementioned bricks.

While the work in [42] is also based on the non-local approach, the constraints it provides (based on the estimation of texture probability density functions) cannot be adapted to handle both

geometric and textural inpainting in a single framework.

2.3.2 An inverse problem approach to non-local inpainting

As proposed in the prequel, we will leverage the non-local graph diffusion from [31] to the case of inpainting. We begin by introducing the objective function to minimize. From our experiments (see for example Fig. 2.4), we conclude that it is not useful to adopt a TV penalized algorithm and we stick with the graph-based heat flow.

We build a non-local graph of all the available image data using an image self-similarity measure. Qualitatively, vertices originating inside the hole will exhibit different connections than others because of their visual dissimilarity, and hence will cause a non-local singularity on the underlying non-local appearance-based manifold. We simply apply an iteration of heat flow on this non-local graph to reduce this singularity and smooth the manifold. Then, patches are back-projected onto the image domain to obtain a novel image where the information inside the hole is now closer to the reference image content. This procedure is iterated several times after updating the graph connectivity to diffuse the non-local information into the gaps. Note that unlike [29] we do not update the graph by solving an optimization subproblem but instead by looking for the most similar patches, as during the graph creation.

Remark 5. *This interpretation of non-local processing as a smoothing action on an appearance manifold was explored independently by Peyré in [43]. However, Peyré relied in this work on explicit parameterizations of the appearance manifold, and limited his study to image with simple local shapes (e.g., cartoon images or fingerprint images where small patch can be assumed to contain at most one straight edge defined with two parameters). Hence, he he limited his study mostly to special categories of images. Instead, we use an implicit parameterization of the manifold obtain from image patches, which is closer to both the non-local intuition and the way graphs are used in semi-supervised Machine Learning. Note that we will se in chapters 3 and 5 that* i) *the patches of real images have a low dimensionality and* ii) *most of the information inside a patch is about a local edge orientation. Hence, it is probably possible to extend Peyré's approach to more realistic cases.*

Problem statement. Let I be an image defined on a domain $\Omega \subset \mathbb{R}^2$. $R(\boldsymbol{x})$ and $R(\boldsymbol{y})$ are square patches of area d centered on pixels $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, and also denote the corresponding vectors obtained by stacking the values inside the patch in lexicographic order [1]. We want to infer missing information in a subdomain $\omega \subset \Omega$ in a plausible way. Note that we did not make any assumptions about the shape or topology of ω, which is of arbitrary shape and can contain holes or several disconnected parts.

There is a natural one-to-one correspondence between each patch, its central image pixel and a vertex of the non-local graph (Fig. 2.3). To distinguish between the image space and the non-local manifold, we call $\Phi : \Omega \to \mathbb{R}^d$ the embedding of the image data in $\mathbb{R}^d$ Φ transforms the image into a d-dimensional space by replacing each pixel he $\sqrt{d} \times \sqrt{d}$ patch $R(\boldsymbol{x})$ centered on it. Hence, we will write indifferently $\Phi(\boldsymbol{x})$ or $R(\boldsymbol{x})$ for the patch centered on $\boldsymbol{x}$. It has a reverse operation Φ^{-1} which is the *back-projection* of the patches: the pixels in a patch are sent back to their original locations

1. For color images, we stack the RGB or LAB values the same way, hence multiplying the patch size by 3.

in the image; all the pixels are projected and not only the central one (see Fig. 2.5). Since patches corresponding to neighbouring pixels in the image do overlap, we merge the corresponding patches by averaging after the back-projection step. This allows to enforce some spatial regularity in the output image.

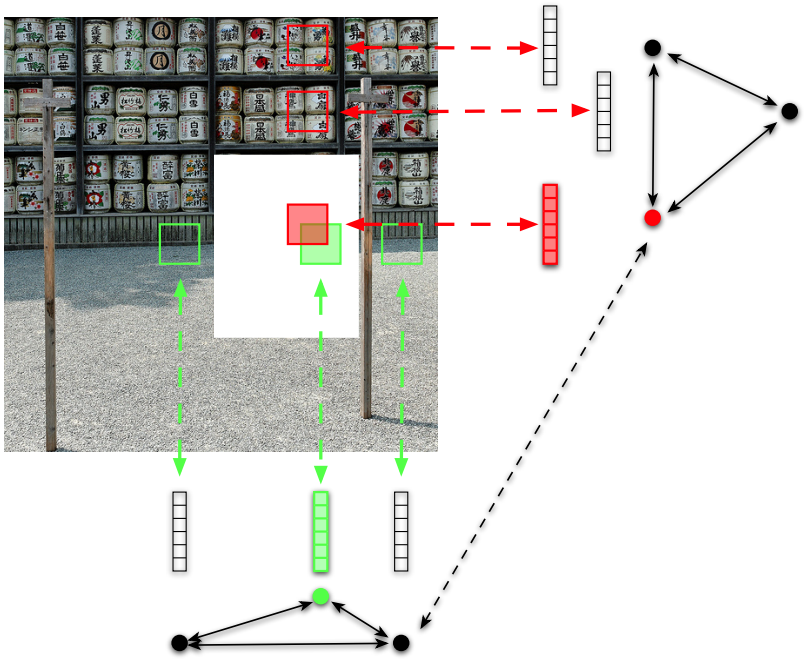


Figure 2.5: *Back-projection of the patches into the image. We back-project all the pixels of a patch to their original locations., as illustrated by the red and green squares. When patches overlap, their values are averaged to obtain the final image value. The vertices corresponding to the green and red patches may be connected in the non-local graph. This hypothetical situation is represented by the dashed line joining them.*

In this embedding Φ, the distance between two points is obtained by the usual patch similarity measure:

$$w(\boldsymbol{x},\boldsymbol{y}) = w(\boldsymbol{y},\boldsymbol{x}) = \exp\left(-\frac{\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{y})\|_2^2}{h}\right), \tag{2.19}$$

where $\|\cdot\|_2^2$ is the sum of squared differences between the two patches:

$$\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{y})\|_2^2 = \left(\sum_{i=1}^{d} (\Phi_i(\boldsymbol{x}) - \Phi_i(\boldsymbol{y}))^2\right). \tag{2.20}$$

and is identical to the standard non-local weight between two patches. The function w defined in Eq. (2.19) is symmetric positive, hence, we cam use it to build an undirected graph.

The parameter h in Eq. (2.19) acts here as a selectivity parameter instead of a pure denoising parameter as in Eq. (2.5). Large values of h will gather dissimilar looking patches, while smaller values will be more selective. This parameter thus provides a way to infer the missing data despite

the presence of noise, which may not be desirable since it leads to blurry estimates. Finally, note that w does not take into account the positions of x and y in the image, and hence is non-local.

We build a non-local graph $\mathcal{G}$ as described above to obtain the final version of the embedding $\Phi : V \to \mathbb{R}^d$ that we want to regularize. Φ is d-dimensional: there is one dimension per component of a patch. The vertices coming from ω disrupt the non-local smoothness of Φ measured using $\mathcal{G}$, since the corresponding image patches differ from the patches of $\Omega \backslash \omega$. The non-local inpainting formulation can be restated to enforce the smoothness of Φ, given the immutable data outside ω:

$$\hat{\Phi}_i \in \underset{\Phi: V \to \mathbb{R}^d}{\arg\min} \frac{1}{2} \sum_{i=1}^{d} \sum_{v \in V} |\nabla_w \Phi_i(v)|^2 + \frac{\Lambda}{2} \sum_{i=1}^{d} \|\Phi_i - \Phi_i^0\|_2^2, \tag{2.21}$$

with Λ a scalar field defined on the set of vertices.

The first term $|\nabla_w \Phi_i(v)|^2$ ensures that we are not introducing too many visual innovations by controlling the smoothness of Φ: if we were to inpaint the holes using random patches or patches that have no close counterpart in the image, then large gradients would be created. The data term can be used to enforce some proximity with the initial image, typically along the borders of the hole. In the absence of noise, we will set $\Lambda(v)$ to infinity for vertices outside the holes to be inpainted (thus yielding immutable image content) and 0 inside to allow the full replacement of their content. This little trick allows us to avoid explicitly introducing a masking operator inside the functional.

2.3.3 Inpainting algorithm

Eq. (2.21) can then be solved component-wise using an iterative procedure of gradient descent after defining differential calculus operators on a graph. In the sequel, we denote iteration indices by superscripts, and the initial condition (given by the input image) is written Φ_i^0 for each component i.

Solving Eq. (2.21) is equivalent to minimizing the energy:

$$\mathcal{E}_i^w \left(\Phi_i, \Phi_i^0, \Lambda \right) = \frac{1}{2} \sum_{v \in V} |\nabla_w \Phi_i(v)|^2 + \frac{\Lambda}{2} \|\Phi_i - \Phi_i^0\|_2^2, \tag{2.22}$$

for each component $i \in [1, d]$. Since $\mathcal{E}_i^w$ is convex, we obtain a global minimum by solving the system of equations:

$$\frac{\partial \mathcal{E}_i^w(\Phi_i, \Phi_i^0, \Lambda)}{\partial \Phi_i} = 0, \ \forall v \in V. \tag{2.23}$$

Differentiating Eq. (2.23) yields:

$$\Delta \Phi_i(v) + Llambda(\Phi_i(v) - \Phi_i^0(v) = 0, \ \forall v \in V, \tag{2.24}$$

which is the graph equivalent of the heat flow. One can then derive easily the Gauss-Jacobi iterations solving this problem:

$$\begin{cases} \Phi_i^{(0)} = \Phi_i^0 \\ \Phi_i^{t+1}(v) = \frac{\Lambda \Phi_i^0(v) + \sum_{u \sim v} 2w(u,v) \Phi_i^t(u)}{\Lambda + \sum_{u \sim v} 2w(u,v)}. \end{cases} \tag{2.25}$$

Note that the update step does normalize the output value by the sum of the weights that arrive at a given vertex v. This plays the same role as the normalization constant called $Z(\boldsymbol{x})$ in NL-means and defined in Eq. (2.7). This implicit normalization is welcome: an alternative solution such as

pre-dividing the weights arriving at a given vertex by $Z(\boldsymbol{x})$ would have broken the symmetry of the weights and removed the undirected property of the graph.

This leads to the following iterative algorithm, where the superscript depicts the index of the current iteration (recall that superscripts depict the iteration index, with 0 for the initial condition):

Initialization initialize Φ^0 with the patches computed from the input image I;

Main loop at step t:

1. construct the non-local graph $\mathcal{G}_{nl}^t$ and the current embedding Φ^t from the image I^t. For the first iteration ($t = 0$) the connections of the patches that are inside the inpainting area are initialized with their spatial neighbours ;
2. compute the regularized embedding $\hat{\Phi}^t$ with respect to $\mathcal{G}_{nl}^t$ by applying Eq. (2.25);
3. back project the patches according to $\hat{\Phi}^t$ to form the image $\hat{I}^{t+1}$;
4. compute the updated solution I^{t+1} as I^0 in $\Omega\backslash\omega$ (outside the hole) and $\hat{I}^{t+1}$ inside ω;
5. go back to 1 until done.

Note that, by replacing the non-local weights of $\mathcal{G}_{nl}$ by fixed values depending on pixel locations, we fall back to a geometry-based diffusion process. Furthermore, although the solver is applied component-wise, the weights used in the diffusion of each component Φ_i are the same (given by the similarity between patches). Hence, we are actually diffusing patches and not individual pixel values.

Implementation details. The update iterations described in Eq. (2.25) are straightforward to implement. The back-projection step at the end of each iteration leads to overlapping patches. In this case, we simply average the corresponding values weighted by the position of the current pixel in each of the overlapping patch. The weight of each contribution is given by the Gaussian G_σ that defines the spatial extension of a patch.

Using a full non-local graph requires to compute and store the weights between all the patches of an image. To overcome this computational burden, we did not build the full non-local graph, but instead a k-nearest neighbour version of it, i.e. a vertex can have at most k neighbours that are the nearest according to the function w. In all the experiments, k is kept small (between 1 and 10), and the neighbours are searched in a restricted (yet large) area to avoid the exploration of the full image. This restriction is discussed in the following section.

The choice of the parameter h is of interest. From the definition of the weights in Eq. (2.5), it is clear that smaller values of h are of better choice since higher selectivity produces less averaging between patches. In the limit, h equals to 0 corresponds to copying only identical patches. Hence, this case can be approximated in the implementation by looking for the nearest neighbour of a given patch, and assigning to it a weight of 1 (see also [42] for a similar argument).

All operations are done pixel or patch-wise. Furthermore, a practical advantage of using Jacobi iterations (instead of Gauss-Seidel for example) is that the results of the iteration Φ^{t+1} is stored in a different array of the input Φ^t. Hence, it is immediate and trivial to derive a parallel implementation of the iterations in Eq. (2.25) in order to reduce the computation time.

2.3.4 Inpainting results

2.3.5 Validation

First, we validate that a perfect non-local graph contains enough information for the inpainting. In this experiment (Fig. 2.6), we build the graph before removing some image parts and let the non-local diffusion iterate. As can be seen, an isotropic diffusion defined on a non-local graph can correctly inpaint textures. This experiment again emphasizes that anisotropic non-local diffusion is not required to obtain correct results.

(a) Input image with holes (b) Reconstruction detail

Figure 2.6: *Example reconstruction assuming perfect knowledge of the non-local graph. Left: input image (with holes). Right: reconstruction detail after inpainting using non-local heat flow. The graph was built from the original (non corrupted) image. Note that the missing texture can be correctly inferred for any shape and size of the target area. Hence, knowing a correct non-local graph allows to inpaint with texture.*

2.3.6 Real images

In all the following experiments we chose $h = 0$ and k (the number of nearest neighbours) between 1 and 10, and proceeded with 10 to 50 iterations of the proposed algorithm. Note that the nearest neighbour of a given patch is always defined. In the worst case, if a patch is very singular, it will not be updated at once. Instead, it will stay black for several iterations and will start evolving when the surrounding patches will also get closer to the remaining of the image. Fig. 2.7 depicts the evolution of the solution for different number of iterations. One can see there how the appearance information is propagated from the boundaries to the inside of the hole, without any imposed heuristic as in [6]. We did not use any additional information, such as a confidence map [6], to improve the results, since our primary goal is to explore the effects of the non-local diffusion.

The patch size needs to be chosen large enough so that a patch can contain an entire example

Figure 2.7: *Iterative nature of our diffusion process (original image on the left). This sequence shows the current estimate at different stages. Note how the texture information is progressively diffused, while spatial TV-flow would have produced a flat area. Original image from http://www.flickr.com courtesy of* Trekking Lemon.

of the texture to reproduce. In our experiments, we fixed the diameter of the patches to values between 5 and 25 pixels. There is however a trade-off to be found here: bigger patches will allow the repeating of larger texture patterns, but will be averaged (in the non-local space) to produce a flatter estimate. Hence, images containing high frequency textures should be inpainted using small patches. This does from the fact that we back-project complete patches and not only the central pixel. For example, in the case of the bungee jumper image (Fig. 2.8) we took patches of 5-by-5 pixels, while in Fig. 2.10 we used small patches in order to reproduce the grain texture of the image. Conversely, the statue image in Fig. 2.9 required wider (25-by-25 pixels) patches in order to correctly reproduce the vegetation: there is little blur caused by the averaging here, and these patches are wide enough to capture meaningful foliage elements.

The example in Fig. 2.9 shows that our algorithm can propagate some structured texture into large holes when the mean value of the hole's surrounding are very different from the semi-local image content, which is where the nearest-neighbour search during the graph building phase takes place. If the initial solution or the vicinity of the hole contain unrelated texture, but whose first and second order moments are close to several textures in the image, then the algorithm can be misguided and output a solution that is a kind of average of these different source areas. This explains why our algorithms sometimes produces visually smooth but physically impossible or unexpected results. In Fig. 2.8, the bungee knot was replaced by a texture taken from the shoes and legs because of its initial colour and shape. This behaviour is more disturbing in Fig. 2.11, where the dark dominant colour mislead the non-local nearest-neighbour search. Consequently, the fence pattern, which is darker than the other textures, was used to fill almost the entire designated area and smoothly mixed to its surroundings. In a real graphics software, such an image should be inpainted using one step for each different texture (ground, fence, sake bottles) to circumvent this problem.

Note that in Fig. 2.11, the use of the color information did not help: the results were better on the grayscale version of the image, with less confusion between the different areas (compare with Fig. 2.12 for the grey version). Finally, Fig. 2.13 shows additional examples where the distances between the different areas (measured in the L$*$a$*$b$*$ space) were greater, thus yielding better results.

2.3.7 Is full non-locality always desirable?

Since building the non-local graph is computationally expensive, we restricted the search of a non-local neighbour in a smaller vicinity around the hole. Using principal component analysis and *kd*-tree sorting, it is possible to speed-up the whole process. However, our first experiments did not

Figure 2.8: *Bungee jumper image. The inpainting mask is the same as [34]. Note that some texture has been inpainted on the water. The pink band is not a real mistake: the algorithm chose to repeat a pattern from the legs and the shoes that was close to its initial guess.*

Figure 2.9: *Statue image (user designated area in red). Note that the algorithm successfully reproduced complex foliage texture.*

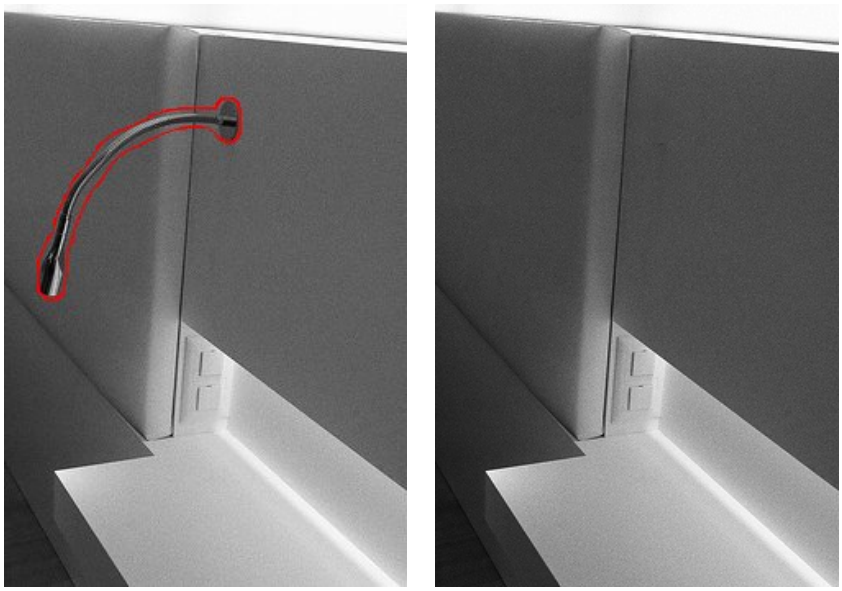


Figure 2.10: *Examples of our algorithm. Left: original image. Right: inpainted image. Red lines depict the user-selected target region. Note how the grain texture is preserved, while TV-flow would have produced a flat area. Original image from http://www.flickr.com courtesy of* Sansuiso.

show any improvement. Preliminary investigation showed that many hole pixels were mislead by the dominant colour and chose the same neighbour, producing a flat image.

2.4 Conclusion

Introducing non-locality. In this chapter, we have introduced the so-called non-local approach through the description of the inspirational NL-means algorithm. This algorithm was used as the basis for a substantial amount of derivative work, that aims to be more efficient or more general (with respect to the addressed Image Processing problem) at the cost of more complexity. In particular, we have seen that the non-local approach can be translated into various standard mathematical frameworks, thus allowing to augment regular solvers with patch-based tools.

Among these various frameworks, graph-based diffusion seemed especially interesting because it can handle both local and non-local approaches indifferently: diffusion processes exploit it the graph structure but ignore totally how the graph is obtained. Since patches can be considered as discrete samples from a manifold living in a high dimensional space, graphs are particularly convenient: patches interactions are by essence discrete.

Non-local inpainting. We have studied the application of this graph-based variational framework to tackle the problem of texture inpainting. In contrast with previous methods, this algorithm can handle naturally both geometric-based and exemplar-based approaches for inpainting, which derives from the use of graphs to implement the underlying diffusion processes. Hence, graphs seem a natural tool to extend the non-local methods to more complex problems than denoising, simply

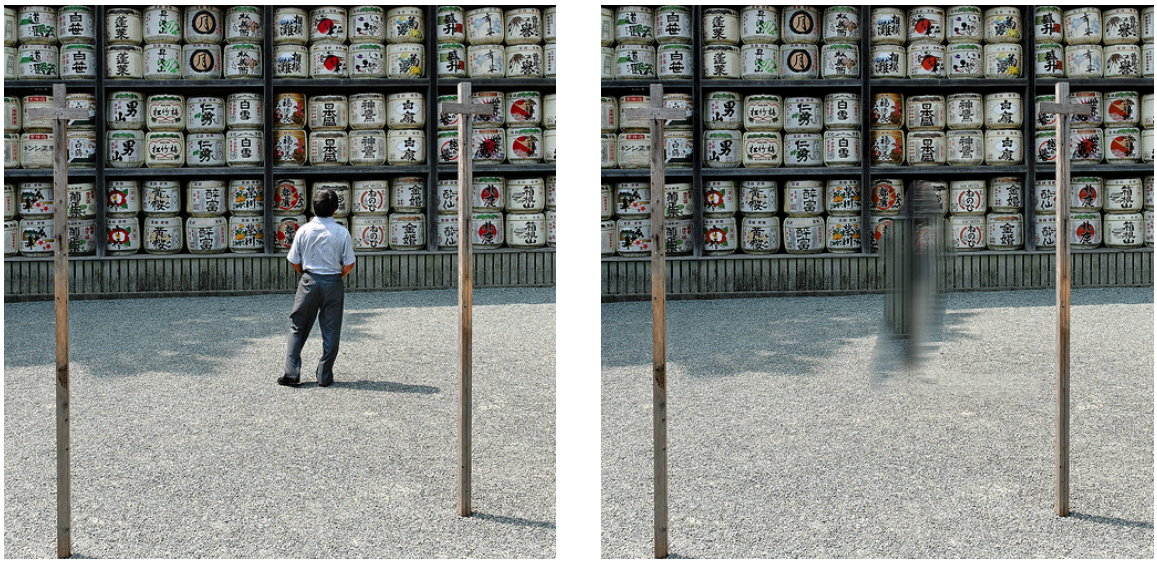


Figure 2.11: *Example of failure. Since initial colours were dark, the darker texture was extended and smoothly mixed to the surrounding areas.*

by re-using previously existing diffusion-based methods.

However, computational constraints limit in practice this approach: building the graph is otherwise too much consuming and can yield an adjacency matrix which is too big to be handled in memory (and not sparse enough to be efficiently traversed). Furthermore, as pointed out in the experiments, several extensions seem to be desirable to visually improve the results, but they cannot fit in the proposed framework. This includes patches of variable size in order to locally adapt the size of the patches with the size of the synthesized texture, and building a patch scale-space to derive a coarse-to-fine filling-in algorithm. Finally, an important limitation of the proposed algorithm lies in the back-projection of the patches: when patches overlap, their back projection induces some undesirable blur. Hence, a functional approach with the ability to join local and non-local constraints seems highly desirable.

Alternative: a low rank approach. From our experiments, one can see that non-local diffusion on graphs is a viable approach to inpaint random missing points, both theoretically (via the links to a manifold smoothing interpretation) and in practice. Furthermore, we have shown that under good circumstances (few neighbours in the graph, no textures with a very fine scale) our algorithm was able to recover large holes inside an image, but at the cost of a slightly blurry result. Hence, it is still not as efficient as the "block copy-and-paste" family of algorithms [5, 6] that remain the state of the art [44] with respect to the visual performance.

A promising alternative that is currently emerging seems to be to consider the matrix formed by gathering all the patches from an image as a low rank matrix. This is quite natural, since our assumption of smooth non-local manifold can also be rephrased as claiming that there are few independent patches, while most of them are linear combinations of the basis patches. This is backed by the study of the Principal Component Analysis (PCA) of the patches (see chapter 3) and by the success of the non-local dictionary algorithms (which are studied in chapter 4). Recently, a successful method to inpaint a single texture was proposed in [45], while we have independently

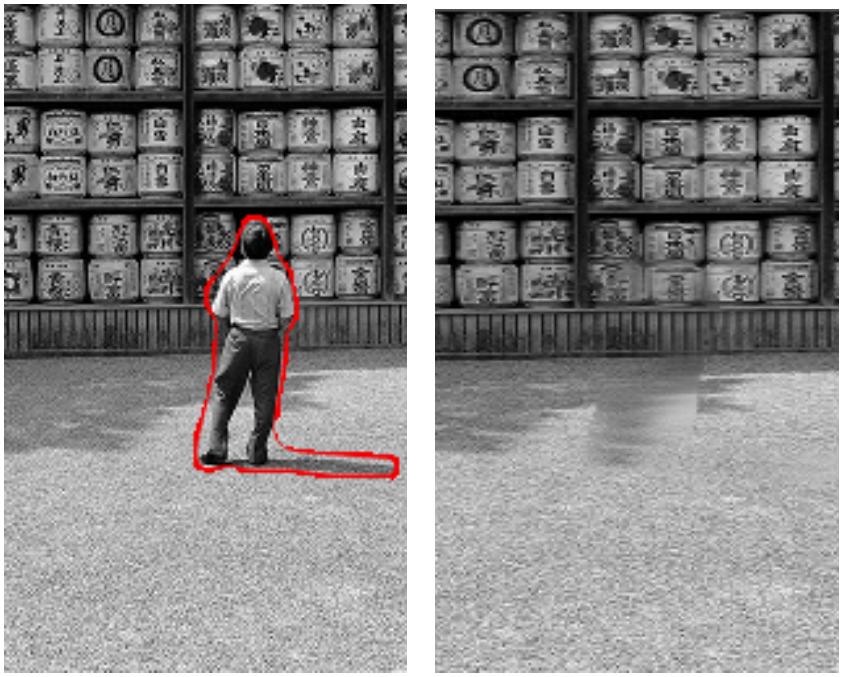


Figure 2.12: *Examples of our algorithm. Left: original image. Right: inpainted image. Red lines depict the user-selected target region. Our algorithm successfully recovered texture information of three different types simultaneously. Note that, without any additional information such as a confidence map, the proposed method did accurately restore the wooden barrier while also recovering painted barrels.*

developed a more general non-local low-rank inverse problem also suited to inpainting. However, our results were mitigated and we did not manage either to fully reproduce the results from [45]. Since it is still a work in progress, an analysis of this approach was postponed to the concluding chapter of this thesis.

Upcoming chapters. The approaches described in this chapter are somewhat limited. While they exploit local patch redundancy, they are not completely non-local in the sense that patch comparisons are always limited in the vicinity of the pixel of interest, thus yielding semi-non-local implementations. Furthermore, the powerful intuition of sparsity of the signals of interest in some well chosen representation frames was not exploited at all, thus allowing the building of a completely different family of non-local algorithms. We will see in the next chapter how the patch comparisons can be made fast enough to become fully non-local via the use of dedicated acceleration structures. This will be useful in chapter 4, where sparse representations of patches will be obtained from non-local samples.

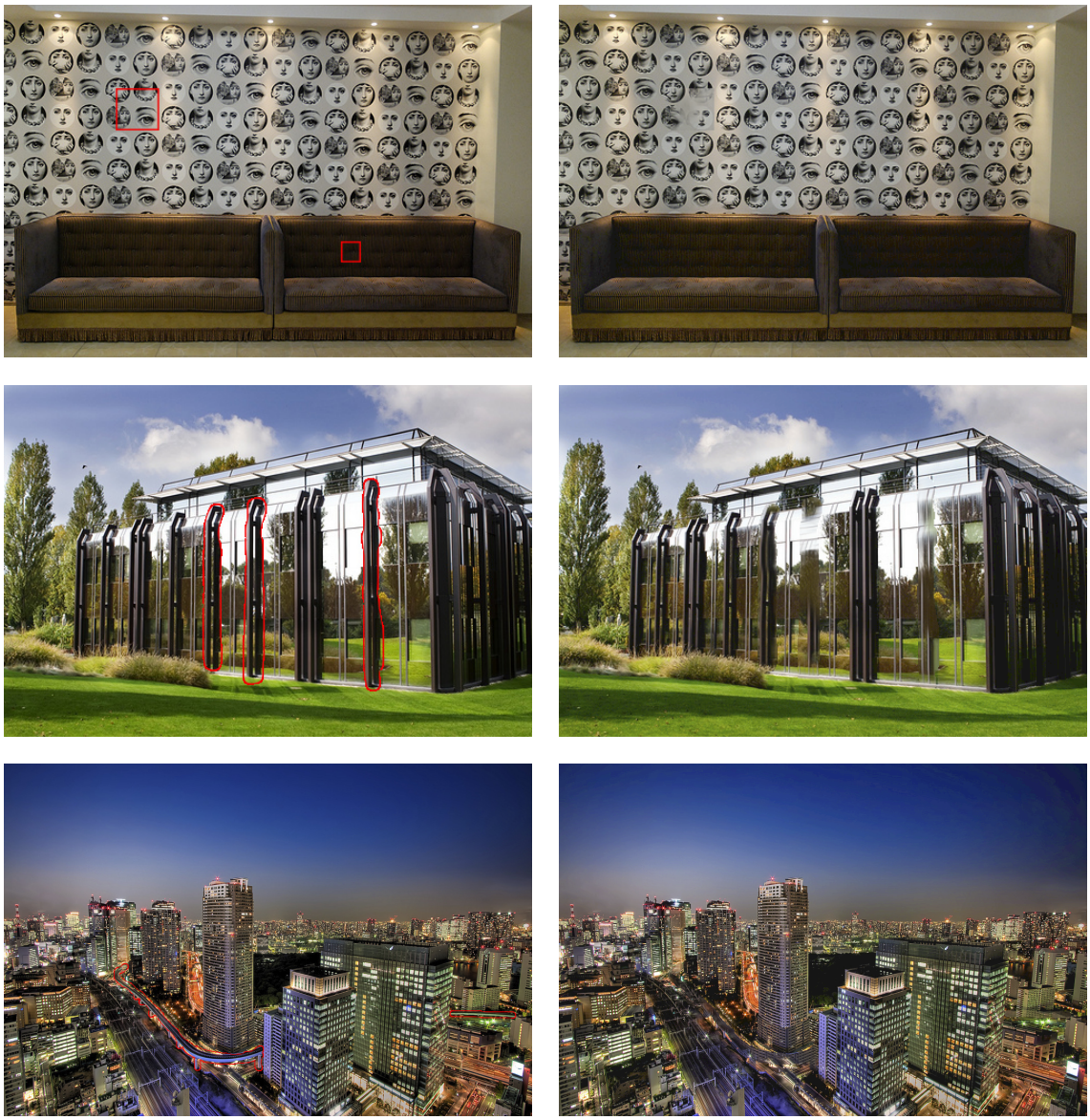


Figure 2.13: *Examples of color results. Original images are shown in the left column. Red lines depict the user-selected region. On the top row the inpainted result should be as close as possible to the original image, since there is no object to remove in the designated areas. Note however that the reconstructed mean value on the wall creates a small subjective contour due to the adjacent spotlight. This default should be removed in a future version of this work. Original pictures from http://www.flickr.com courtesy of* Sansuiso *(top),* Trekking Lemon *(middle) and* Too Much UV *(bottom).*

3 Intermezzo: fast non-locality

Much of the elegance of the genuine non-local means algorithm [46] lies in its simplicity, which makes it straightforward to implement. Naive implementations however are computationally demanding. Using patches of size P pixels and learning windows of L pixels, the overall complexity of the naive algorithm is $N \times L \times P$ to process an image of N pixels. This reproach was particularly accurate at the time of the original disclosure of the algorithm: the intrinsically parallel nature of NL-means [1] could not be exploited since multi-core and multi-CPU workstations were still expensive and not yet generalized as they are nowadays. Consequently, the publication of the algorithm was immediately followed by the development of several fast variants.

In this chapter, we review several of these fast methods and classify them into two main groups: *approximate* algorithms (where the acceleration is achieved by early termination or implementation tricks) and *pre-selection* based algorithms that aim at reducing the number of the non-local comparisons when denoising a given pixel. Then, we present two novel approaches that allow for fast and convenient exploration of non-local neighborhoods based on dimensionality reduction and feature space partitioning. These methods have the practical advantage over the previously proposed methods of relying on data structures that are easy to maintain. Hence, the proposed approaches will have a practical advantage: we will be able to easily insert or remove patches while preserving the quality of the pre-selection process. Hence, our methods will be more suitable to the processing of movies and for sliding-window based denoising procedures.

1. Significantly, a GPGPU implementation of NL-means has been available inside the NVIDIA CUDA SDK since its inclusion of image processing examples.

3.1 Fast non-local algorithms

3.1.1 Fast implementations

Parallel variants. Fully aware of the computational complexity of his algorithm, Antoni Buades, the original author of NL-means, proposed two fast variants [3] [47]: one using a *multiscale* and another using a *blockwise* approach, although they have not become very popular. In the multiscale setup, the standard NL-means algorithm is applied on a downscaled version of the input image. The result is then used to denoise the original fine scale image without resorting to non-local comparisons.

The blockwise algorithm is identical to the genuine NL-means up to two points: weighted averages of patches are used instead of pixel averages in Eq. (2.6), and a final overlapping patch merging step is added at the end. Since a given pixel can belong to several patches, this approach allows to have a loose sampling of the blocks, hence limiting the number of costly non-local explorations. Note that the chosen patches should overlap to avoid the introduction of spurious discontinuities in the result. The fusion step is implemented as a simple averaging, which caused the blur artifacts in the inpainting task from the section 2.3.2 of chapter 2. Consequently, this simple fusion strategy was criticized in subsequent work, e.g., [48] and notably in BM3D [7] (see chapter 4) because of this blur after the back-projection of the patches.

Propagating information. [1]

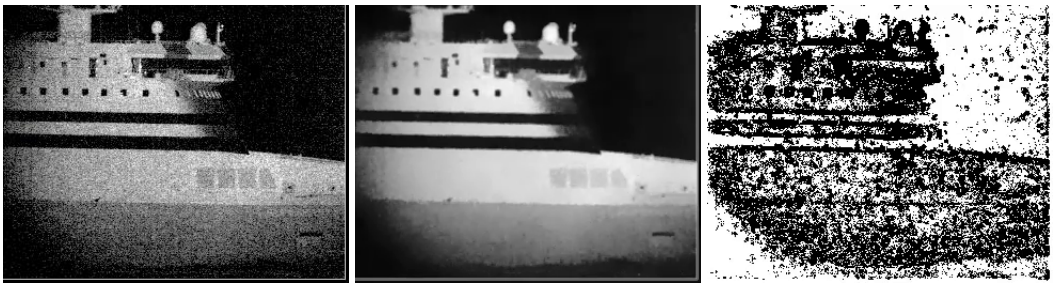
Surprisingly, there exists an even simpler acceleration strategy that was never published, to the best of our knowledge. We designate it under the name of *copy-paste NL-means*. It proceeds as follows: during the exploration of the non-local neighborhood of a given pixel, the algorithms keeps track of the locations where the patch-based Euclidean distance was smaller than a chosen (small) threshold. Since these pixels share an identical neighborhood (up to a small amount of noise) with the pixel of interest, one can infer that they will share the same denoised value. Hence, the output of the non-local averaging process is assigned not only to the pixel at hand but also to the pixels in this list, that are then tagged as visited. When a pixel belongs to several such near-identical neighbor lists, the different values are averaged together to avoid shock effects.

The speed-up comes from the fact that pixels tagged as "visited" are ignored by the non-local learning procedure. Hence, the greedy exploration loop is executed on a small subset of pixels: in our experiments on movie denoising, up to 70% of the pixels in a single frame were actually denoised by the copy-paste procedure instead of the non-local means (Fig. 3.1).

Fast numerical algorithms. Another tempting acceleration strategy is to keep the overall structure of the non-local means algorithm but to make the computations faster. This approach was explored in [49], where Wang and his co-authors leverage several techniques to speed up the patch Euclidean distance estimation. Sticking with the notations of the previous chapter, we write R a patch extraction operator and $R(\boldsymbol{x}) \in \mathbb{R}^d$ the (vectorized) patch extracted around the pixel $\boldsymbol{x} = (x_1, x_2)$ of the image. A pixel $(x_1 + i, x_2 + j)$ belongs to the patch $R(\boldsymbol{x})$ if and only if the 2D translation $\boldsymbol{t} = (i, j)$

1. Although it remains unpublished at the time of writing, this speed-up strategy was presented to the author of this thesis for the first time by A. Buades circa 2006.

(a) $T = 1$



(b) $T = 2$

Figure 3.1: *Applying copy-paste NL-means to a movie (neighbourhoods are defined over space-time windows). From left to right, this figure shows the original noisy frame, the denoising result and the copy-paste mask. In the mask image, black pixels correspond to pixels that are processed by NL-means while white pixels are those obtained by pasting operations (without exploration of their neighbourhood). The top row presents the first frame of the movie, which explains why all the pixels are processed by NL-means. The bottom row shows the second frame, and one can see that more than half of the pixels were already computed by propagating the results from the previous frame.*

belongs to the ball of $\mathbb{R}^2$ of radius a (the diameter of a patch) and that we write B_a^2. Given two patches, Wang and co-authors point out that:

$$\|R(\boldsymbol{x}) - R(\boldsymbol{y})\|_2^2 = \sum_{(i,j)\in B_a^2} (I(x_1 + i, x_2 + j) - I(y_1 + i, y_2 + j))^2 \tag{3.1}$$

$$= \sum_{\boldsymbol{t}\in B_a^2} \left(I^2(\boldsymbol{x} + \boldsymbol{t})) + I^2(\boldsymbol{y} + \boldsymbol{t})\right) - 2 \sum_{\boldsymbol{t}\in B_a^2} I(\boldsymbol{x} + \boldsymbol{t}) \cdot I(\boldsymbol{y} + \boldsymbol{t}), \tag{3.2}$$

$$= S_a(\boldsymbol{x}) + S_a(\boldsymbol{y}) - 2 \sum_{\boldsymbol{t}\in B_a^2} I(\boldsymbol{x} + \boldsymbol{t}) \cdot I(\boldsymbol{y} + \boldsymbol{t}), \tag{3.3}$$

where $S_a(\boldsymbol{x}) = \sum_{(i,j)\in B_a^2} I^2(x_1 + i, x_2 + j)$. The terms $S_a(\boldsymbol{x})$ and $S_a(\boldsymbol{y})$ can be computed efficiently in constant time using a *summed square image* defined as:

$$SSI(x_1, x_2) = \sum_{y_1<x_1, y_2<x_2} I^2(y_1, y_2).$$

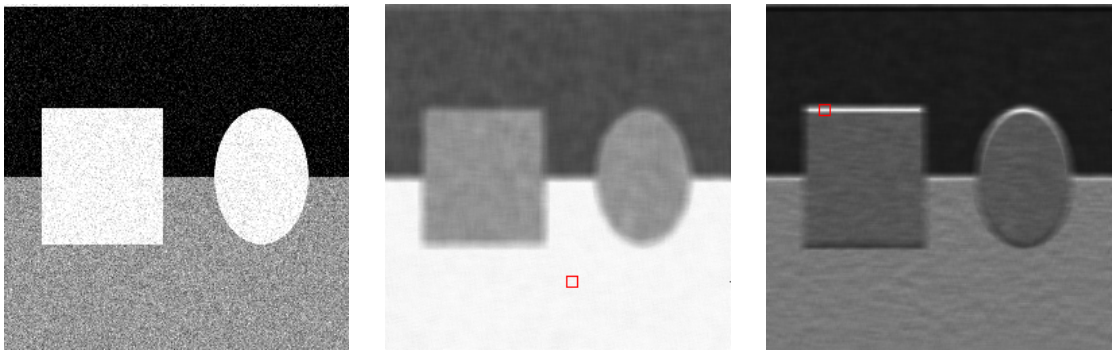
This is a squared analog to an *integral image* [50] commonly used in Computer Vision, see for example [51, 52] and that can be computed in a single pass. The last term in Eq. (3.3) can be interpreted as a convolution between two patches. Hence, it can be evaluated as a multiplication in the Fourier domain, which is also efficient thanks to the use of the Fast Fourier Transform (FFT).

All this contributes to an highly efficient implementation without any noticeable differences from the original algorithm.

3.1.2 Preselection: fast outlier rejection

The computational bottleneck of the original NL-means lies in the non-local neighborhood exploration: for each pixel, one has to loop over a (possibly large) learning window, extract patches and compute Euclidean distances, which is expensive. However, many pixels visited during this learning stage will have a negligible contribution to the denoised output because they are not visually related to the patch of interest. Consider for example the case of Fig. 3.2. When the query patch is chosen in the gray area, the weights on the black and white image parts are small but non-zero. Hence, their accumulation will degrade the clean pixel value estimation, and a semi-nonlocal algorithm would yield better results since it would take into account less outliers! Another solution is to find a preselection criterion that will detect and reject these outliers, which will both improve the denoising quality and save computational time by avoiding the estimation of the Euclidian norm with the outliers.

Hence, subsequent effort has been put in trying to detect and reject the patches that are "too different" from a given patch of interest. This can be achieved by two different strategies: either by using features that are efficiently computed on-the-fly and that approximate roughly the Euclidean distance, thus leading to a fast decision between retaining or rejecting a candidate patch; or by adding a pre-processing stage that will group together visually similar patches.



(a) Noisy image (b) Non-local weights (flat) (c) Non-local weights (line)

Figure 3.2: *The image on the left is the noisy input image. The next two images show the non-local weights for two different patches (indicated by a red square), where white is the maximum weight. Note that although lower weights are assigned to pixels that do not belong to the same image structure as the query, these values are not null. Many outliers are going to interfere with the denoising of the chosen patch since their small contributions will be summed. Hence, better results can be obtained either by limiting the learning neighbourhood (which will decrease the number of these outliers but yields a semi-nonlocal algorithm) or by setting a strict preselection threshold that will reject these outliers.*

Preselection via efficient comparisons. A first and immediate preselection criterion that comes to mind is to compare two patches if, and only if, their mean values are not too different. Indeed, computing the mean of all the patches of an image can be implemented efficiently either using

integral images (see above) or by a separable convolution with a 2D box filter. Hence, this was among the first pre-selection techniques applied, for example by Mahmoudi and Sapiro [53], Coupé *et al.* [54] or Kervrann *et al.* [15].

To add more selectivity to the preselection, these authors completed the mean with additional criteria: Mahmoudi and Sapiro [53] measured the angle between the average gradient orientation of the patches, while Coupé [54] and Kervrann [15] added the variance of the patches. The variance can be estimated efficiently from integral images and sum of square images. On the other hand, the gradient orientation is not very robust to noise and is obviously poorly defined in flat areas. Consequently, some minimum and maximum gradient magnitude thresholds are set that activate or deactivate this criterion, which raises questions about its practical interest.

Since it is non-trivial to find correct pre-filtering criteria in the original patch space, Orchard *et al.* introduced in [55] the idea of transforming the patches to a more suitable space. In a first step, they formed a matrix from the patches of the original image, by extracting each patch then reshaping it as a column vector Then, they obtained a new basis by taking the Singular Value Decomposition (SVD) of this patch matrix, and computed both the preselection threshold and the patch-wise distances in this new space. This is indeed equivalent to computing the original patch distances since the SVD is an orthonormal transform and thus preserves distances. Since SVD is closely related to Principal Component Analysis (PCA), they could obtain good approximations of the patch similarities without using all the resulting components, thus yielding an additional speedup.

Preselection via clustering. Another family of algorithms makes the choice of explicitly clustering the input noisy patches according to their appearance and replaces the learning window exploration by nearest neighbours queries (in the sense of the visually most similar). The learning step is rephrased as finding in a patch database the most similar points to the patch of interest. When implemented carefully, this approach can lead to fast fully non-local algorithms.

This idea was introduced for non-local denoising by Brox *et al.* [56] who proceeded to recursive k-means clustering of the input patches with $k = 2$. At each step of the learning process, the input patches are divided in two groups, and this operation is repeated on each of the resulting groups until either their radius is small or they contain few patches (less than 30 in the original article). The center and radius of each group are stored in the nodes of a *cluster tree*, along with the indices of the enclosed patches for the leaves of the tree. Hence, getting the patches at a small distance of an input query (or the 30 closest neighbors if the recursive clustering stopped early) becomes a binary tree lookup, which is a constant time operation. In order to avoid the introduction of shock artifacts for patches that are close to the border of a cluster, the authors have used groups with a slight overlap. This removes the need to explore the tree further than a single leave and speeds up the patch retrieval process.

Note that an independent study conducted at the same time [57] also concluded that k-means cluster trees were suitable for similar patch retrieval, but found vantage point trees [58] even more advantageous. The purpose of this study however was the search in large image databases, hence some performance constraints differed slightly from the denoising problem at hand.

Another very popular structure for fast Approximate Nearest Neighbors (ANN) queries is the kd-trees introduced by Bentley in [59]. With respect to binary space partitioning structures such as quadtrees and octrees, kd-trees typically choose some data-dependant separating hyperplanes at

each node in order to split the data along its most elongated direction. This direction is either chosen among the original space axis (as in the original kd-tree proposal [59]), but it can also be chosen after transforming the data by a PCA, yielding more efficient partitions [60].

The performance of kd-trees however degrades rapidly when the dimensionality of the data is greater than a relatively small number [57] [1] which is too small for a typical image patch size: an 8×8 patch already has 64 components. Hence, the authors of [61] introduced the so-called Gaussian kd-trees to deal with this curse of dimensionality in two ways: during the construction of the tree, Gaussian subsampling and importance sampling are used in order to create a simpler structure with less nodes by exploiting the fact that only the input data points will be used during the subsequent denoising stage. The fast tree structure is completed by a dimensionality reduction operation using the PCA of the data and by a fast GPU implementation, yielding a fast running method (typically less than 1 minute per megapixel after PCA reduction according to the authors).

In a related work [62], Manzenera used regular kd-trees combined with local jets instead of PCA to perform the dimensionality reduction step. For a given pixel, a local jet is a feature vector formed by the concatenation of the Gaussian derivatives (up to the order 2) of the image, *i.e.*, the result of the convolution between the image and the corresponding derivative of a 2D Gaussian function. This yields a more efficient dimensionality reduction than PCA that stays compatible with kd-trees. As a side note, the local jets used in place of the patches (thus replacing image content by a differential descriptor) can be interpreted as a far non-quantized ancestor of the Local Binary Descriptors presented in chapter 5.

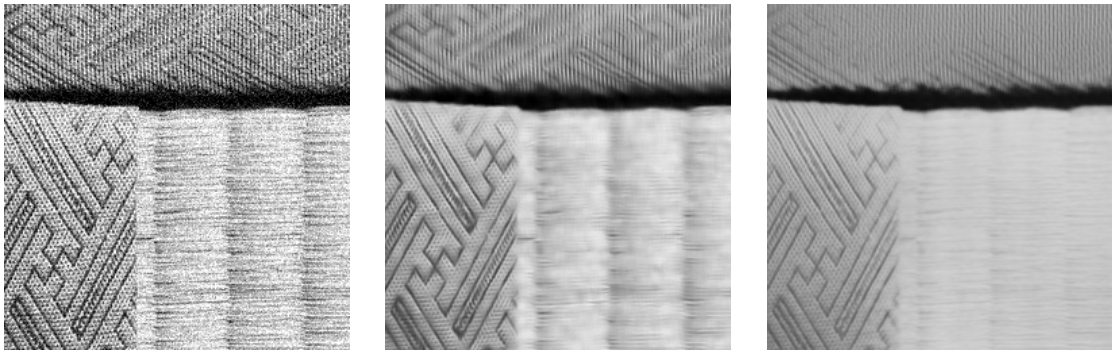
3.1.3 Full non-locality, preselection and image quality

Experimentally, it is often remarked that better results are obtained through semi-non-locality, *i.e.*, limiting the size of the learning window. Hence, this limited spatial range seems to be more than an implementation artifact. For the original NL-means, it seems clear that the limited range avoids the accumulation of many small weights coming from unrelated patches: except for pure texture images, the occurrence of similar patches diminishes with the distance, since objects usually have a finite spatial extent. When the search window is very large, it leads to a non negligible sum of very small weights that lower the importance of the relevant patches through the normalization factor $Z(\boldsymbol{x})$ in the denoising formula of NL-means (see Eq. (2.6) in chapter 2). An example of this phenomenon is shown in Fig. 3.3.

Surprisingly, this is also true for preselection based algorithms. For example in [56], the authors first divide the image into overlapping blocks (with a very large overlap of 50%). Each block is then processed independently with its own preselection tree, and the results are fused by spatial averaging.

For a given learning window size, the influence of the preselection is important. This was pointed out first in [53] with experiments on images mixing texts and natural scenes: the preselection process avoids mixing patches from these two unrelated image areas, yielding a sharper, more readable text in the output. On the other hand, this can lead to undesirable block artifacts, that can be removed using either a smoother clean patch estimator [63] or through a proper strategy of overlapping patch fusion [16, 48].

1. Between 5 and 9 depending on the size of the data and the criticality of the retrieval speed.



(a) Noisy input 256×256 pixels (b) 7×7 windows (c) 65×65 windows

Figure 3.3: *Applying NL-means on an image with fine texture. The patch size was set to* 5×5 *pixels. The only difference between the two applications of NL-means is the size of the learning windows: very local in the first case (*7×7 *pixels in the first case versus* 65×65 *in the second). Note that mechanically, just by extending the learning window (i.e., by making NL-means more non-local!) we have introduced a lot of blur on the finest textures.*

3.2 Patch Spectral Hashing

3.2.1 What makes for a good patch clustering function ?

From the prior work described in section 3.1, we can deduce two important properties that a non-local algorithm should respect to be both fast and effective:

- it should be block oriented: using blocks allows for a sparser sampling of the input image, yielding less computational needs. Furthermore, more aggressive denoising methods can be used because the necessary block fusion step that follows will smooth out most of the ringing or shock artifacts;
- it should use some kind of clustering as a preselection process. The clustering algorithm should be robust to noise, because it can be trained either from the noisy image or from a database of clean examples. It should thus correctly capture the slower variations of the patches, since higher frequencies are more disturbed by the noise. It should however remain efficient for ANN queries in high dimensions: even after dimensionality reduction patches are typically described by 10 to 20 components.

The question of which block-oriented denoising process to use will be treated in chapter 4. For now, we focus on the choice of the preselection algorithm. Since we also want to target video applications of non-local denoising, we add the following constraints:

- it should be fast to train or compute in order to be updated with minor penalties in case of scene changes. For example, the environment observed by a moving camera is subject to frequent changes in textures, luminosity…;
- for a given clustering function, the underlying accelerating structure should be easy to update: even when the scene properties are stable, we want to easily remove some outdated patches (for example that belong to an outdated frame of the movie) and also easily insert new patches from an upcoming frame;
- it should scale nicely with the number of patches: although digital movies are usually of a

much lower resolution than still images [1] the use of a temporal window during the denoising process multiplies the number of patches to consider.

While their purpose is fast ANN retrieval, kd-trees and their derivatives were designed under the hypothesis that the time used for their training was not a constraint. Hence, the building time can get long, depending on the size and the dimensionality of the data. Using several smaller trees in parallel could be a solution to this problem, at the risk of introducing visual incoherences in the output. This is anyway made impractical by the easy update requirement: since the shape of a kd-tree depends on the specific training dataset, deleting the nodes corresponding to outdated patches should be implemented through complex tree rotation operations and would probably lead to unbalanced (and hence less efficient) trees. Consequently, it would be easier to to actually train a new tree than to update a previous one. For the same reasons, cluster-trees and k-means do not appear as good candidates.

3.2.2 Multidimensional hashing

Fortunately, the problem of Approximate Nearest Neighbour (ANN) in high dimension has been widely studied in the Computer Science literature and alternative solutions have come up. Among them, *hash tables* were specifically developed to ensure an efficient access to some possibly sparse data and are widely implemented for 1-dimensional data [2].

A hash tables consists of a set of buckets (hence indexed in 1D) that contain the sorted data. In order to access a data point, a *hash function* is applied to the key describing the query, outputting a number which is the index of the corresponding bucket. Given this index, accessing the data point is simply a lookup in the bucket structure. Hence, if we forget about the hash function for now, hash tables grant a constant time access to their inner content.

While the exact implementation of the bucket structure may vary, the problem of designing a good hash table is primarily to choose a good hash function. An ideal hash function is

- deterministic: the same input should obviously yield the same bucket index for the retrieval to be correct;
- specific: most of the time, a given code should correspond to only one input point, otherwise there is a collision between two data points which will slow down the queries;
- fast to evaluate: in the converse case, the hash function will annihilate the gains from using a hash table since it is called each time there is an insertion/deletion or an access to the table.

Regarding Vision and Graphics applications, hashing-based approaches were mostly studied with 3D data because they are of important practical use for tasks like collision detection or 3D functions lookups, see for example [64] and the references therein.

However, since designing a good hash function for high dimensional data is an arduous task, hashing-based approaches were seldom used beyond 3D applications until the introduction of Locally Sensitive Hashing (LSH) for visual search tasks in [65, 66]. Given a data point $\boldsymbol{x} \in \mathbb{R}^n$, the main idea of LSH is to produce a binary code $\boldsymbol{y} \in \{-1, 1\}^m$ by choosing randomly m couples

1. A Full HD frame contains slightly more than 2 millions of pixels, while a typical DSLR outputs images between 10 and 30 MP at the time of writing.

2. Most utility libraries and popular programming languages such as Python implement dictionary structures with string keys via hash tables.

$(\boldsymbol{a}_i, \boldsymbol{b}_i)_{i=1}^m$ in $\mathbb{R}^n \times \mathbb{S}^{n-1}$ and computing the i-th component of $\boldsymbol{y}$ as

$$y_i = LSH(\boldsymbol{x})_i = \text{sign}\,\langle \boldsymbol{x} - \boldsymbol{a}_i, \boldsymbol{b}_i \rangle, \tag{3.4}$$

where $\langle ., . \rangle$ is the usual dot product in $\mathbb{R}^n$ and $\mathbb{S}^{n-1}$ is the unit sphere of $\mathbb{R}^n$. Qualitatively, the i-th component of the code is the position (above or below) of the query point $\boldsymbol{x}$ with respect to the hyperplane of normal $\boldsymbol{b}_i$ going through $\boldsymbol{a}_i$. The final code $\boldsymbol{y}$ is formed by the concatenation of its components y_i [1]. This random projection scheme has the main quality to produce hash results that are close to each other if the inout data points are already close neighbours in the original space: in this sense, LSH preserves the locality information.

While LSH has some nice properties and theoretically ensures a correct hash function in the limit, in practice it leads to suboptimal partitions of the space with respect to the distribution of the data [67]: intuitively, the randomness in the code creation process can lead to trivial projections of the dataset, hence introducing hyperplanes that do not separate the points in two groups that are roughly of the same size. Hence, more complex variants have been proposed, where some biases and pre-defined spatial cells are applied in order to have better practical behavior, but with the risk of over-fitting to the input assumptions about the distribution of the data, see for example [68].

3.2.3 Spectral Hashing

On the other hand, Weiss *et al.*proposed a deterministic algorithm to ally the original simplicity of the LSH and data-adaptive partitioning functions. In [69], they leverage the following variational approach for the optimal code computation problem.

Initial problem. Given N points $\{\boldsymbol{x}_i\}_{i=1}^N$ of $\mathbb{R}^n$, one can measure their visual similarity based on their Euclidean distance as

$$W_{ij} = W(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/\epsilon^2\right), \tag{3.5}$$

where ϵ is a sensitivity parameter. Finding a good set of codewords $\{\boldsymbol{y}_i : y_{ij} = \pm 1\}_{i=1}^N$ to describe the points $\{\boldsymbol{x}_i\}_{i=1}^N$ can be cast as a constrained minimization problem. The optimal code indeed preserves the locality between images (or patches). This property can be cast as a minimization problem, where the distance between two codewords is penalized by the visual similarity between the corresponding images:

$$\{\boldsymbol{y}_i^*\} = \arg\min \sum_{i,j} W_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2, \tag{3.6}$$

under the additional constraints that half the bits are activated over the training dataset and that the bits are uncorrelated:

$$\sum_i \boldsymbol{y}_i = 0 \text{ (activation)}, \tag{3.7}$$

$$\frac{1}{n} \sum_i \boldsymbol{y}_i \boldsymbol{y}_i^T = Id \text{ (uncorrelation)}. \tag{3.8}$$

1. Practical implementations of LSH replace of course the negative value -1 by 0 in order to have compact binary codes that can be handled by computers as bit sets

Spectral relaxation. Unfortunately, this problem is also shown to be NP-hard in [69] because of the discrete nature of the unknowns $\boldsymbol{y}$. Hence, the authors propose to consider a *relaxed* version instead, where the codewords belong to $\mathbb{R}^n$ instead of being binary vectors. Similarly to what we have done in chapter 2 when building a non-local patch graph, the matrix of weights W can be considered as the adjacency matrix of a graph defined over the set of vertices $\{\boldsymbol{x}_i\}_{i=1}^N$. The diagonal matrix D then arises naturally as the corresponding degree matrix:

$$D_{ij} = \begin{cases} \sum_k W_{ik} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

Also introducing the code matrix Y whose i-th row is made of the codeword $\boldsymbol{y}_i$, one can drop the constraint $Y_{ij} \in \{-1, 1\}$ to yield the simpler problem:

$$\min \mathit{trace}\left(Y^T(D - W)Y\right), \tag{3.10}$$

$$s.t. \begin{cases} Y^T 1 = 0 & \text{(activation)}, \\ Y^T Y = Id & \text{(uncorrelation)}. \end{cases} \tag{3.11}$$

This last problem is a classical Spectral Graph Clustering problem, and as such it has been extensively studied. Its solutions are known to be the eigenvectors of the graph Laplacian $D - W$. Hence, a binary code of length m could be generated by thresholding the m first eigenvectors with minimal eigenvalue (after excluding the trivial eigenvector 1 with eigenvalue 0 that corresponds to the mean of the training set).

Generalization. There is still an important question left: how well does this result generalize to out-of-sample data points? To compute this generalization, the authors of [69] make the general assumption that the data points are drawn from some separable probability function $P(\boldsymbol{x}) = \prod_i p_i(x_i)$. Using expectation operations, the final relaxed version of the code generation problem then writes:

$$\min \int \|\boldsymbol{y}(\boldsymbol{x}_1) - \boldsymbol{y}(\boldsymbol{x}_2)\|_2^2 W(\boldsymbol{x}_1, \boldsymbol{x}_2) P(\boldsymbol{x}_1) P(\boldsymbol{x}_2) d\boldsymbol{x}_1 d\boldsymbol{x}_2, \tag{3.12}$$

$$s.t. \begin{cases} \int \boldsymbol{y}(\boldsymbol{x}) P(\boldsymbol{x}) d\boldsymbol{x} & = 0, \\ \int \boldsymbol{y}(\boldsymbol{x}) \boldsymbol{y}(\boldsymbol{x})^T P(\boldsymbol{x}) d\boldsymbol{x} & = Id. \end{cases} \tag{3.13}$$

This is another classical problem linked to learning on manifolds and Laplace-Beltrami operators [70, 71]. The solutions to this last problem are now the eigenfunctions of the weighted Laplacian L_p, that can be computed for simple probability distributions on $\boldsymbol{x}$.

Specifically, in the one dimensional case and for a uniform distribution of x in $[a, b]$, the eigenfunctions $\Phi_k(x)$ and eigenvalues λ_k are given by:

$$\Phi_k(x) = \sin\left(\frac{\pi}{2} + \frac{k\pi}{b-a}x\right), \tag{3.14}$$

$$\lambda_k = 1 - e^{-\frac{\epsilon^2}{2}|\frac{k\pi}{b-a}|^2}, \tag{3.15}$$

and n-dimensional eigenfunctions are obtained by an outer product formulation: $\boldsymbol{\Phi}_k(\boldsymbol{x}) = \prod_{i=1}^n \Phi_i^n(x_i)$, with eigenvalue $\Lambda_k = \prod_{i=1}^n \lambda_i$. The final codes of length m are obtained by computing

first all the single-dimensional functions $\Phi_i(x)$, then building the set of the n-dimensional independent eigenfunctions $\Phi(\boldsymbol{x})$, and keeping the functions with the m smallest eigenvalues. These eigenfunctions are then applied to a query point $\boldsymbol{x}$ and the sign of each of these projections are concatenated to form the codeword.

Note that enforcing the constraint that the code basis eigenfunctions are independent means in practice that each final eigenfunction $\boldsymbol{\Phi}_k$ separates the data along one and only one axis. Functions that split the data along two or more axes can be computed as the product of several of these single-dimensional functions. Hence, it would be redundant to include them, as the corresponding signs are already computed.

3.2.4 Spectral Hashing with patches

The Spectral Hashing algorithm was designed for image similarity-based tasks, hence extending it to patches is straightforward. The visual similarity formula in Eq. (3.5) is actually identical (up to the substitution of a constant kernel instead of the Gaussian one) to the weights used by the NL-means algorithm defined in Eq. 2.5. In order to meet the conditions used to obtain the eigenfunctions in Eq. (3.14), namely uniform distribution of the components and uncorrelation, we first compute a Principal Component Analysis (PCA) of the training patches. While PCA components are not strictly speaking independent and the more complex Independent Component Analysis (ICA) should be used instead, the Spectral Hashing algorithm is claimed by its authors to be robust against this deviation, which is confirmed by our own experiments. This remark also holds for deviations from the uniform distribution: the spectral hashing proved to be robust to the actual distribution of the data.

The PCA decomposition from the learning stage is saved in memory for the later projection of new points onto the set of retained eigenfunctions. These points can come either from the input image (in order to denoise an image from its own self-similarities) or from independent and unrelated pictures (in the case of denoising a movie after learning the PCA from the first frames, or if one wants to use a specific patch space as in [4]).

Finally, to stick with the idea of *patch preselection*, we consider that all the patches that correspond to a given codeword are identical up to the chosen precision given by the number of bits used for a codewords. This is the only parameter of the method. Hence, this simplifies the hash table structure in two ways:

1. we do not need to handle collisions inside a bucket. By definition, all the patches that project with the same code are considered as noisy samples of an identical underlying clean patch, hence they are added to a list for later processing;
2. when querying the neighbors of a given patch, we directly return the list of patches inside the corresponding bucket, *i.e.*, points that are at null Hamming distance from the query.

Note that this last assumption allows us to avoid slower and more complex queries that consider points at Hamming distances of 1 or 2. These complex queries are useless in our case due to the inherent semantic information stored in the bits of a Spectral Hashing code: the first bits of a code come from the most meaningful principal components. Hence, querying identical patches at some non zero Hamming distance (for example equal to 1) does not make sense if we change the sign of a more meaningful bit: it leads to mixing principal components instead. On the other hand, changing the sign of less meaningful bits is far less efficient than just truncating the codes at the desired

precision. Hence, the preselection threshold of previous works is replaced here by the number of bits used to build the codes.

The Spectral Hashing was recently extended by its authors in [72]. However, the goal of this extension is to have the non-zero Hamming distances that are more correlated to the actual Euclidean distance when it grows, *i.e.*, improving the performance of the Spectral Hashing for regimes where the similarity parameter ϵ in Eq. (3.5) is not small anymore. The code construction mechanics remain the same, and it consists mostly in adding bits to the original codes that correspond to the outer-product-shaped modes ignored previously. Since we are only concerned with queries for the nearest neighbors, *i.e.*, Hamming distance equal to 0, this extension does not bring any improvement for the application at hand.

Algorithm 1: Patch Spectral Hashing

1: Take $\{\boldsymbol{x}_i\}_{i=1}^N$ training patches in $\mathbb{R}^n$, fix a length of m bits for the codewords.
2: Compute the PCA of $\{\boldsymbol{x}_i\}_{i=1}^N$.
3: Find the min/max bounds of the data after PCA $[a_i, b_i]$ $(i = 1, \ldots, n)$
4: Form the modes and eigenvalues using Eq. (3.14) and Eq. (3.15).
5: Retain the m eigenfunctions with the greatest eigenvalues and that are not obtained by an outer-product.

3.3 Experiments

3.3.1 Patches and PCA

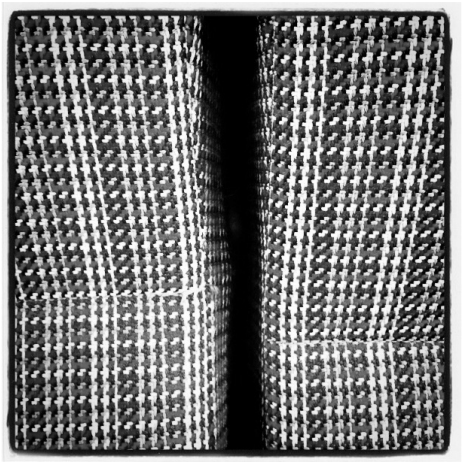
Patch PCA is at the heart of the spectral hashing algorithm. Hence, we need to check that there is no bias introduced by the set of patches used during the training phase.

Intuitively, the possible content of patches has a small dimensionality, especially when patches get smaller. This assumption is backed by several facts. First, there is the success of patch dictionary-based methods (see chapter 4) and image synthesis by a dead leaves model, that would be otherwise intractable. Second, if we look at the content of patches that are not too big, all we can see is a flat area, an edge or a corner. Hence, at this scale, image content gets very simple (see also remark 5 in chapter 2).

To confirm this intuition, we proceeded with a very simple experiment. We chose three very different images shown in Fig. 3.4: the classical Lena image, a photograph of some Japanese tatami with very fine textures of different orientations and a picture of a seat blanket with a strong geometric pattern. Then, we computed the principal components for each image and discarded the first one, that corresponds to the eigenvalue 0 (which is the mean patch of each image). The first 8 components are shown Fig. 3.5. The results are strikingly identical, except for the 8th component. For the first 7 ones, the components are either the same or with the inverted polarization (black becomes white and vice versa). The principal components of the tatami image also tend to be more aligned with the image axis, but this phenomenon is not very pronounced and can be explained by the original orientation of the textures covering almost the entire image. Furthermore, the chosen patches used to compute the illustration were taken quite large (15×15 pixels) in order to emphasize the point, and this allowed more capture of texture. You can also remark that in spite of the

(a) Lena

(b) Tatami



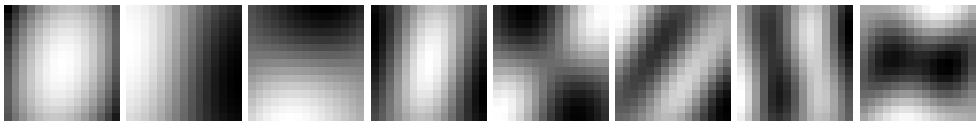
(c) Seat

Figure 3.4: *Test images for the PCA experiment.*

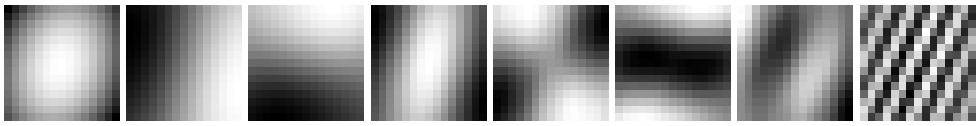
strongly geometric nature of its texture, the seat blanket picture produces PCA components that are also very close to the Lena image, while their visual content is clearly different.

If we analyze the shape of the components, the first is similar to computing the Gaussian mean of the patches, then we can find horizontal and vertical edge detectors and corner detectors, which constitute the basic semantic elements of local image content. As a side note, one can also remark the visual proximity of these principal components with the construction of the 2D Haar wavelets.

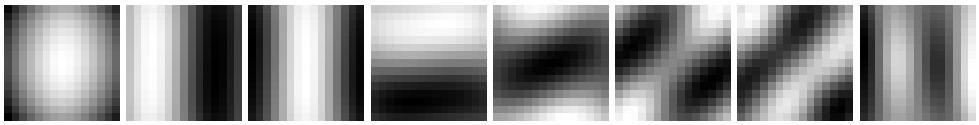
Note that an alternative experiment with the same conclusion and applied to the problem of parameterless patch matching can be found in [73].



(a) Lena principal components



(b) Tatami principal components



(c) Seat principal components

Figure 3.5: *Eight first principal components (after removing the component with eigenvalue 0) of three images (Lena and two strongly textured pictures) for patches of size* 15×15 *pixels. Although both images are very different, and in spite of the large size of the patches (that allows to capture more texture), one has to wait until the 8th component to see a difference. The seven first ones are identical up to a polarization change and are very close to the first Haar wavelets, capturing essentially edges and corners.*

3.3.2 How many bits for correct patch retrieval?

The only parameter of the spectral hashing method (apart from the patch size which is determined by the application and the input images) is the length of the codewords. In order to find an optimum value, we have measured the precision of the hash function in the noise-free then in the noisy case for different code sizes. For each image, the accuracy of the spectral hashing was measured by computing the average Mean Squared Error (MSE) of the patches in each bucket after training the hash function on this image.

Some results in the noise-free case can be seen in Fig. 3.6, They confirm the good prevision results obtained by the authors of the original paper. For small codewords (8 bits), the Tatami image seems to yield worse results than the other images. This comes from the fact that it contains several different textures (two fine textures with orthogonal directions, and one coarser texture) than the other two images which are more homogeneous. Hence, it requires more bits in order to be correctly segmented.

We reproduced the same experience by adding additive Gaussian white noise to the images before training the hash function. The results in this case can be seen in Fig. 3.7. While the magnitude of the error is larger for small codewords, it decreases quickly, which shows the robustness of the method to the corruption of the training set by additive random noise. Note that the Tatami image does not stand as an outlier anymore: the addition of noise has introduced a created variety of textures in the Lena picture. Hence, it does require more bits to be correctly segmented.

Finally, an important parameter to maintain the computational efficiency of the nonlocal filtering step that will be applied after the spectral hashing preselection is the number of patches per non-empty bucket of the table. As illustrated in Fig. 3.8, the population size of each bucket remains

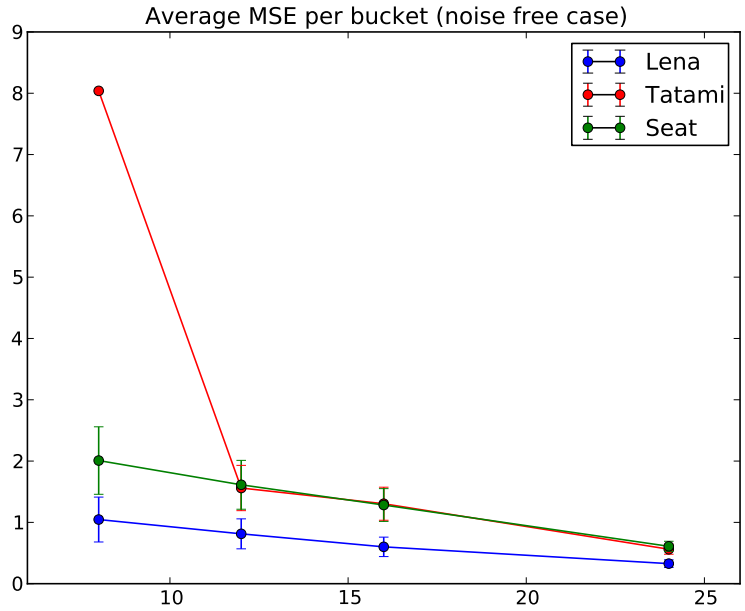


Figure 3.6: *Average Mean Squared Error (MSE) per bucket of the hash table in the noise-free case. Codewords of 12 bits or more yield very tight patch clusters.*

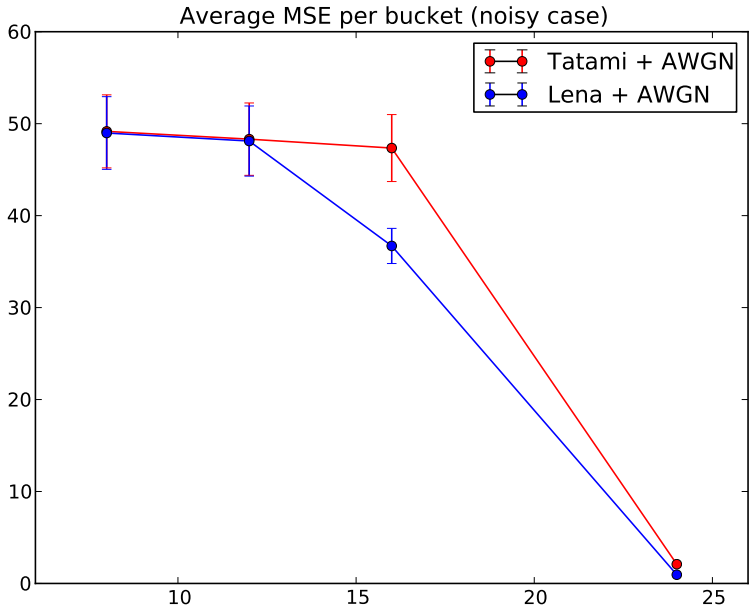


Figure 3.7: *Average Mean Squared Error (MSE) per bucket of the hash table in presence of additive Gaussian white noise. The lower bound for the code size is now larger, around 16 bits per codeword. However, one needs to be careful and check if there was an overfitting of the hash function with the noisy patches.*

low, and the comparison between the noisy and noise-free cases advocates for the use of codewords between 8 and 16 bits long.

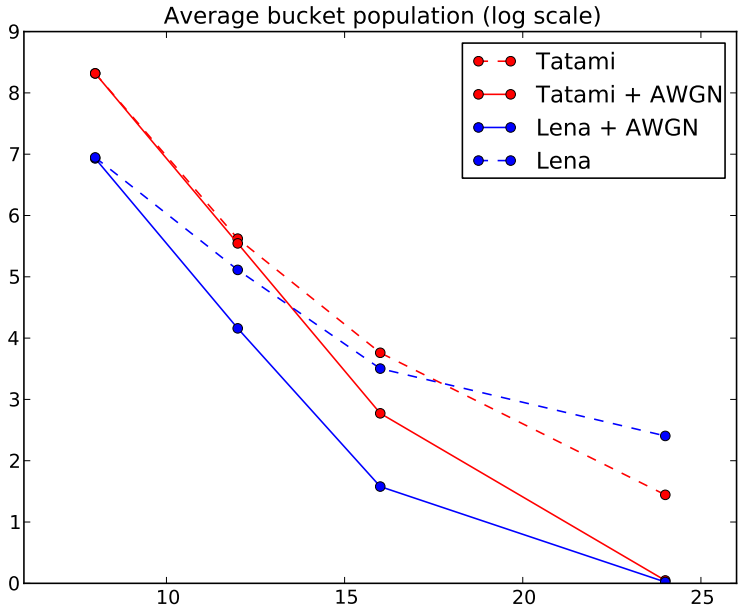


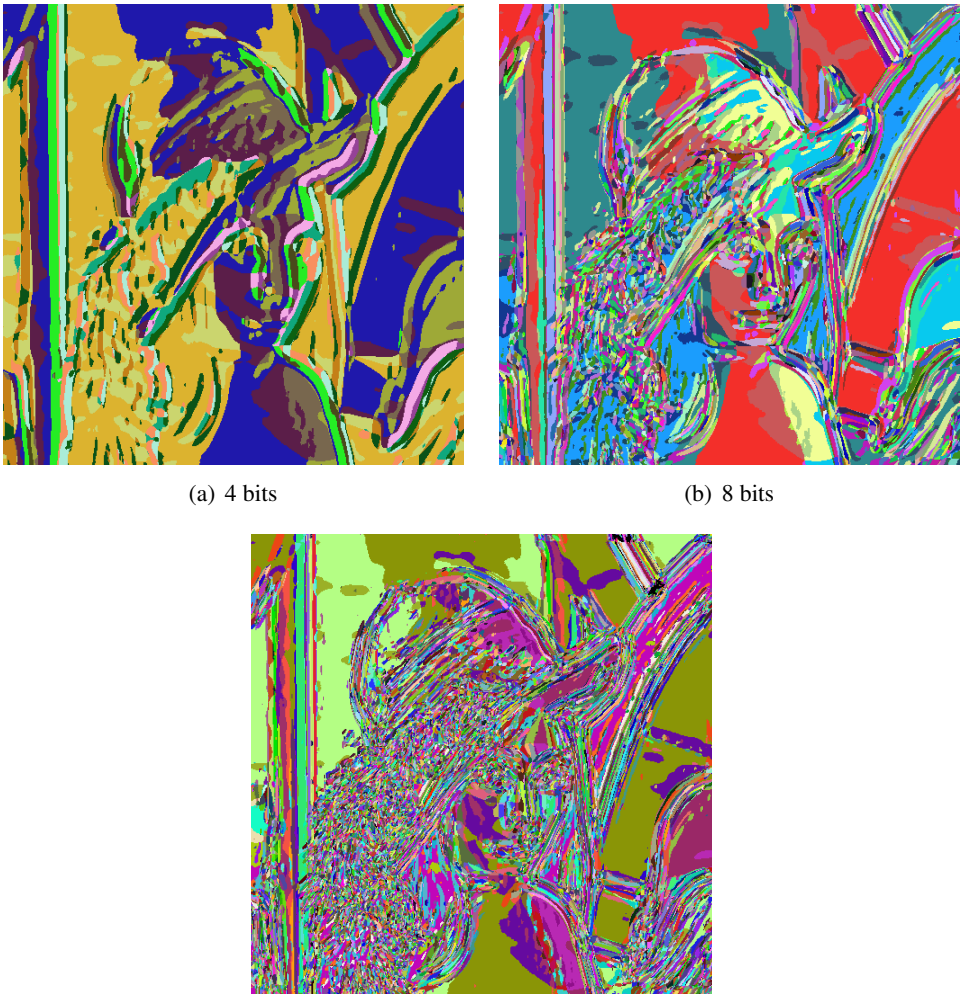
Figure 3.8: *Average population of a bucket of the spectral hash table (in log scale to account for the different size of the two test images). Without noise, long codewords yield buckets that have very few elements: this means that we are in presence of overfitting to the training set. When the patches are corrupted by some white noise, the actual bucket population are only slightly changed for small codewords (less than 16 bits). Hence, spectral hashing is a good candidate for noisy patches clustering.*

3.3.3 An accidental image segmentation tool

In order to easily identify the cases of overfitting of the spectral hash function on the patches used for training, we visually inspected the segmentation output induced by the hash function. Each bucket was considered as a region and assigned a random color. Since the buckets contain patches without any spatial information, this corresponds to an appearance-based nonlocal image segmentation process. As can be observed in Fig. 3.9 and Fig. 3.10(c), 8 bits already yield satisfying segmentations of different images. Furthermore, in the presence of noise, the hash function becomes quickly too selective and also learns the appearance of the noise (see Fig. 3.10). Hence, codewords of 8 bits seem to be a reasonable choice for denoising applications.

3.4 Example application: non-local super-resolution

Super-Resolution (SR) is the task of creating an high resolution image from a low resolution input sequence. It has many purposes, such as producing high quality stills from a video sequence or



(a) 4 bits (b) 8 bits

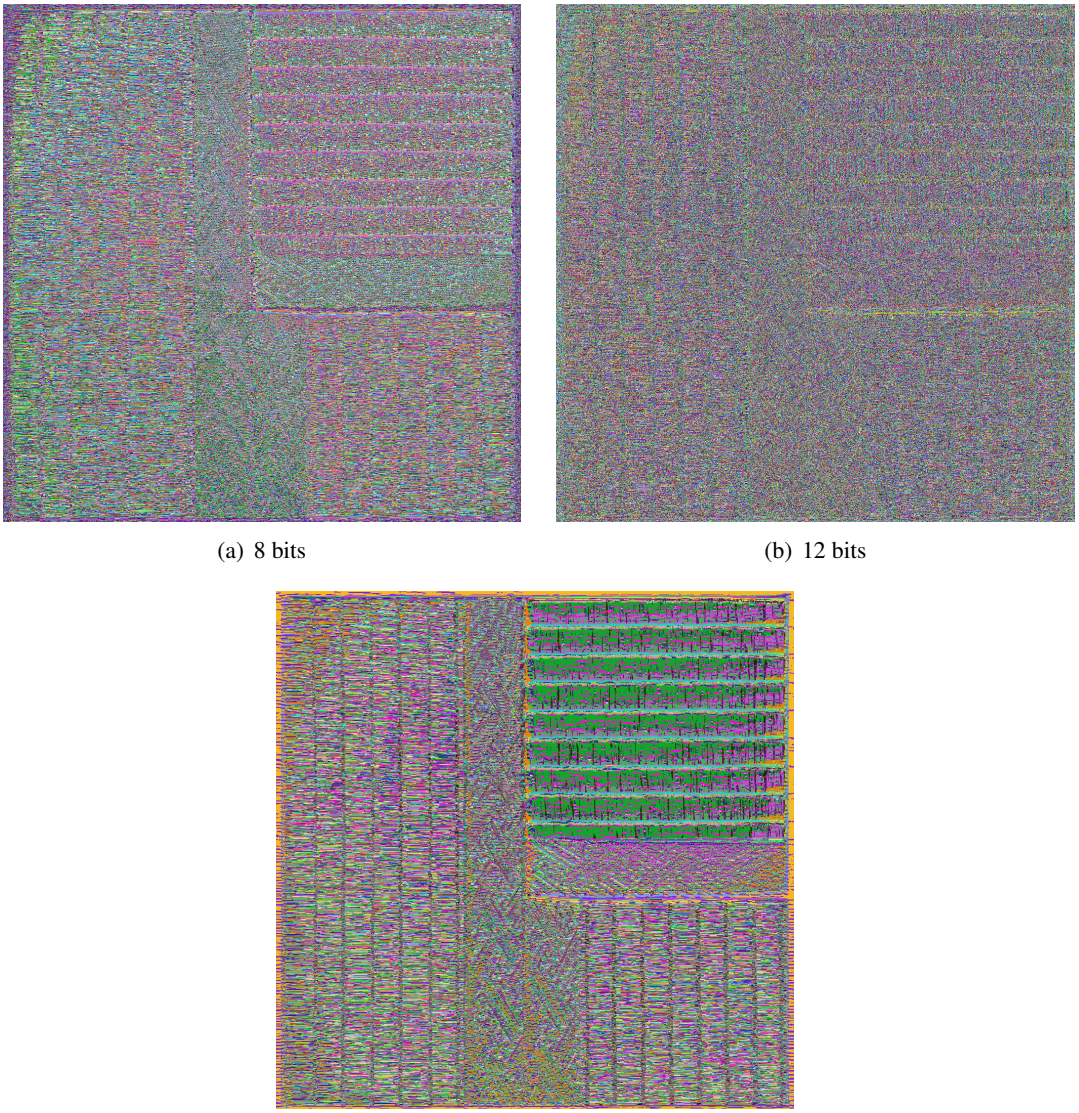
(c) 16 bits

Figure 3.9: *Visual illustration of patch clusters in the noise free case for patches of* 9×9 *pixels. Random colors were assigned to each non-empty bucket. When the codewords are too small (4 bits), the mean value of a patch becomes the most important clustering criterion and the hair gets mixed with the background. On the other hand, when the codewords are too long (16 bits), the segmentation result becomes sensitive to the digitization noise of the picture and is unstable in shaded areas and on the hat.*

upscaling a movie to an higher resolution. The main intuition to SR methods is to exploit subpixel motions between the frames of the input sequence to infer the missing data on the target high resolution grid.

To overcome the difficulties of fine image registration, several methods have been proposed exploiting the non-local intuition, *i.e.*, any datapoint can contribute to the final result if it is relevant. These algorithms, however, limit in practice the search region for relevant points in order to lower the corresponding computational cost. Furthermore, they define the non-local relations in the high resolution space, where the true images are unknown.

In the rest of this chapter, we introduce the use of spectral hashing to efficiently compute fully



(a) 8 bits (b) 12 bits

(c) Reference: 8 bits, noise free

Figure 3.10: *Visual illustration of the buckets trained with the Tatami image in the presence of noise for patches of 9×9 pixels. When the codewords are too long (12 bits), there is no more structure to be found in the segmentation result that looks like noise. There was some overfitting of the hash function to the noise in this case. The 8 bits segmentation output shows the most important structures of the image. This picture is best viewed large on screen.*

non-local neighbours. We also restate the super-resolution functional using fixed weights in the low resolution space, allowing us to use resolution schemes that avoid many artifacts.

While early work involved the fusion of several image spectra, this approach was later replaced by an inverse problem formulation, which is less sensitive to noise and does naturally handle arbitrary motions. In this context, SR can be stated as the following minimization problem, where the Low Resolution (LR) image sequence $(\mathbf{I}_{LR}^k)_{k=0}^{n-1}$ is obtained by applying a blur B, a downsampling D

and geometric warping W_k to the unknown High Resolution (HR) image $\mathbf{I}_{HR}$:

$$\mathbf{I}_{HR} = \underset{\mathbf{I}_{HR}}{\operatorname{argmin}} \sum_{k=0}^{n-1} \rho\left(\mathbf{I}_{LR}^{k}, W_k D B \mathbf{I}_{HR}\right) + 2\lambda\kappa\left(\mathbf{I}_{HR}\right), \tag{3.16}$$

The function $\rho(\cdot,\cdot)$ measures the error between the k-th frame in the sequence and the current solution after applying the registration, blurring and downsampling operators, while $\kappa(\cdot)$ penalizes solutions that are too far away from the chosen prior.

Standard choices are the ℓ_2-norm for the error term $\rho(\cdot)$, and the Tikhonov regularization: $\kappa(\mathbf{I}_{HR}) = \|\nabla\mathbf{I}_{HR}\|^2$, for the prior. Eq. (3.16) can then be solved using various frameworks: gradient descent, Iterated Back-Projection (IBP) [74], maximum likelihood estimation [75], etc. (see for example [76] for a review of SR algorithms).

Although they lead to well known resolution schemes, these choices have inherent drawbacks. The ℓ_2 norm is not robust to misalignements between the images. Also, the Tikhonov regularization is known to produce over-smoothed results by penalizing abrupt changes in the gradient of the solution, actually blurring the boundaries of the objects. Hence, subsequent works [77, 78]considered using the more robust ℓ_1-norm and the Total Variation (TV) regularization to produce sharper outputs. However, these methods still rely on the accuracy of the motion estimation step.

Related work. To bypass the limitations of motion estimation methods, recent works have taken advantage of example-based regularization, successfully introduced by [4] for the SR of a single image. This algorithm however required the training of a Markov random field on a huge database, limiting its practical interest.

Since a movie contains many redundancies, a subsequent work [25] followed the same intuition as the NL-means image denoising algorithm [79] and exploited self-similarities anywhere in the image sequence. The pixels are described by patches, which are simply all the values comprised in a small square neighborhood. If two patches are very similar, then the corresponding central pixels are very likely to represent the same phenomenon and should be exploited together even when they are spatially far from each other. Hence, the algorithm in [25] uses non-local averaging when fusing together the LR frames interpolated to the final resolution. However, the authors limit the search of self similarities to a small learning window around each pixel to limit the computational overhead. In [80], the authors adapt the non-local approach to the IBP algorithm, again considering non-locality as a post-processing constraint on the upscaled images, and within the limits of a search window.

We propose instead to enforce the non-local constraints on the low resolution images. This has two main advantages. First, the weights are fixed by the input LR sequence, which allows the computation of the exact gradient of the non-local error term. Second, since there are less LR pixels, this makes less weights to compute. Furthermore, we propose to use the spectral hashing introduced above to sort all the input LR patches in a single hash table, hence leveraging fully non-local SR at a moderate computational cost.

3.4.1 Variational TV non-local super-resolution

We consider that the immutable, and hence reliable, non-local information is contained in the LR frames. Consequently, we do not proceed with a standard SR process followed by an NL-means-like enhancement step, but compute instead a non-local error on the LR frames. Hence, our SR functional to be minimized is:

$$\mathbf{I}_{HR} = \arg\min_{\mathbf{I}_{HR}} \sum_{k=0}^{N-1} \left(\sum_{\boldsymbol{x}} \sum_{\boldsymbol{y}} w_k^*(\boldsymbol{x},\boldsymbol{y})(\mathbf{I}_{LR}^k(\boldsymbol{x}) - (DB\mathbf{I}_{HR})(\boldsymbol{y}))^2 \right) + 2\lambda TV(\mathbf{I}_{HR}), \tag{3.17}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ are pixels i the low-resolution image domain. Unlike previous work, the data term includes only pixels that belong to the original, low-resolution image domain. Consequently, no prior interpolation or upsampling of the input sequence is required. We also assume that there is no additional temporal interpolation, *i.e.*, we are given the index of the frame of the input sequence that should be upsampled. This reference frame is depicted by the superscript $(*)$ in Eq. 3.17, while the subscript k designates the current LR frame. Consequently, the patch similarity weight $w_k^*(\boldsymbol{x},\boldsymbol{y})$ is obtained from the patch centered on $I_k(\boldsymbol{x})$ and the one centered on $I_*(\boldsymbol{y})$:

$$w_k^*(\boldsymbol{x},\boldsymbol{y}) = \tfrac{1}{Z_k(\boldsymbol{x})} \exp\left(-\frac{\|R(I_k(\boldsymbol{x})) - R(I_*(*(\boldsymbol{y}))\|^2}{h^2} \right), , \tag{3.18}$$

where R stands as before for the patch extraction operator. It corresponds to the classical non-local weight, except that the two patches may come from different images.

Qualitatively, Eq. (3.17) gathers in its data term the non-local error with respect to the low resolution frames: the errors between the hallucinated HR image and the LR frames are computed for all the possible positions and are weighted by the reference non-local weights w_k^* computed from the LR sequence. Finally, for each (vectorized) LR frame, the weights can be gathered in a single matrix W_k^{NL} for a more elegant formulation using matrix vector multiplication.

Note that unlike traditional SR algorithms, the weights $w_k^*(\boldsymbol{x},\boldsymbol{y})$ can be interpreted as a probabilistic motion, or as a multi-valued optical flow. This does qualitatively explain the efficiency of non-local super-resolution: it integrates over the possible destinations of a pixel and does not get trapped by motion estimation errors. Again, the parameter h is a *selectivity* parameter: for small values of h, a candidate patch $\boldsymbol{x}$ needs to be very similar to the reference patch $\boldsymbol{y}$ to have a significant contribution.

Inspired by [25, 77], we solve Eq. (3.17) in two steps iteratively:

1. we first look for an HR blurred estimate $\mathbf{Z}_{HR} = B\mathbf{I}_{HR}$ using non-local back-projection;
2. we compute $\mathbf{I}_{HR}$ by TV deblurring of $\mathbf{Z}_{HR}$.

The TV deblurring subproblem is solved using the Monotonous FISTA (MFISTA) algorithm described in [81].

Our specific form of the SR objective function in Eq. (3.17) calls for several comments:

– since the weight matrices W_k^{NL} are fixed throughout the minimization process, it is possible to define the gradient of the non-local error. Hence, we can use a procedure similar to the FISTA algorithm [81] to iteratively solve the two steps non-local error minimization - TV minimization. This minimization process has the advantages of faster convergence and a tighter control of the solution;

– the normalization of each line of the matrix W_k^{NL} leads to a straightforward interpretation of non-local SR as a standard SR process with a probabilistic motion estimation instead of the usual univocal motion model: each line $W_k^{NL}(i,\cdot)$ is indeed the motion probability density of the pixel i with respect to the k-th image;
– finally, note that it is possible to separate the constant from the similarity score in Eq. (3.18). The matrix W_k^{NL} can then be written as the product of a diagonal matrix (storing the constants $\frac{1}{Z_k(\boldsymbol{x})}$) by a similarity matrix).

3.4.2 Non-local super-resolution algorithm

Putting the spectral hashing and the proposed non-local variational problem together, we obtain the following SR algorithm:

Initialization:

1. Form all the patches from the LR sequence, and compute their PCA.
2. Compute the desired number of eigenfunctions of the similarity matrix.
3. Sort all the input patches in a table, using the retained eigenfunctions to obtain binary codes.

Super-Resolution:

1. Select the frame to upsample; form an initial estimate by interpolation.
2. Apply the degradation model (blur and downsampling) to the current solution.
3. For each point in the downsampled solution, compute the non-local error using the corresponding patch in the selected frame and its non-local neighbors retrieved by SH.
4. Back-project the non-local error and deblur the updated solution with MFISTA.

Note that, since the hash table contains patches from the whole input sequence, our algorithm is fully non-local: we do not limit the search of relevant patches to a space-time search window. Furthermore, unlike bilateral regularization, there is no additional attenuation factor due to the space or time distance between the patch to update and its neighbors.

3.4.3 Experiments

Implementation details. The proposed algorithm was implemented in C++ and tested with a standard laptop. Although a Matlab implementation of spectral hashing is available online [1], we used our own C++ re-implemention in order to work with the OpenCV library.

In the nearest neighbor search, we did not use all the patches returned by a query, but only those with a similarity score above 0.9. The parameter h is fixed depending on the noise level of the input sequence. For clean movies without noise, we used a small value of 0.08, and values between 0.5 and 1 for noisy inputs. In all our experiments, we have chosen patches of size 5-by-5 pixels.

The initialization of the algorithm is very fast: computing the PCA of a whole LR sequence is the longest part and takes only a few seconds on a laptop for the patch size considered. Then, we

1. http://www.cs.huji.ac.il/ yweiss/SpectralHashing/

used a straightforward implementation of the algorithm described in Sec. 3.4.2 that did not take advantage of any acceleration and processed the pixels sequentially. Note however that the operations for each pixel are independent: this parallelism can be exploited to design faster multithreaded implementations.

3.4.4 Super-resolution results

We have tested the proposed algorithm on several sequences and reference data available on the internet [1]. Fig. 3.11 shows the interest of our functional: having fixed weights on the LR input avoids the apparition of the high frequency bright artifacts around the mouth. On the other hand, since we have numerous non-local neighbors from the whole sequence, this is counter-balanced by a more pronounced visual blur with respect to [25].

Figure 3.11: *Comparison with related work. Left: original image. Middle: result of Generalized Non-Local Means [25] for a zoom factor of 3 along each dimension. Right: our result. Our minimization scheme avoided the artifacts around the mouth.*

Fig. 3.12 illustrates the application of our algorithm to other noise-free sequences.

Finally, since accurate and near real-time optical flow algorithms are becoming available [78, 82], one may wonder why we should still persevere with non-local SR. If there is only one reason, it should be to deal with noisy sequences. In the presence of noise, the precision of optical flow diminishes quickly. Non-local algorithms, on the other hand, are able to integrate information all along the sequence without explicitly needing any motion estimation, and are designed to naturally deal with independent random noise. This is demonstrated in Fig. 3.13, using either wider patches or relaxed similarity conditions.

3.5 An alternative to spectral hashing: using space-partioning trees

In the last section of this chapter, we present an alternative to the spectral hashing. While spectral hashing has many qualities and was our final choice, hardware constraints can interfere

1. From `http://users.soe.ucsc.edu/~milanfar/software/sr-datasets.html` and `http://www.cs.technion.ac.il/~matanpr/Research.html`.

Figure 3.12: *Results in the noise free case. In both experiments, we used a zoom factor of 3 along each dimension and patches of size 5-by-5 pixels.*

with the usage of a hash table, and most notably the lack of *memory pointers* on the target hardware. In this case, a less efficient algorithm can be used as a replacement for the spectral hashing.

In this early version of our work, we considered using fixed space-partitioning functions instead of the data dependent hash function. While we abandoned it because it did not scale well with the dimensionality of the patches, we present it here because it can still have some interest for small patches (sizes up to 9×9 pixels) and for hardwares such as GPUs where random access structures penalize the overall performance.

The main ideas were:

1. to reduce the dimensionality of the patches by projecting the patches onto a PCA basis. Since the PCA is an orthonormal transform, the distances were also computed from PCA coefficients, hence diminishing the overall computational burden and allowing to discard the original patch values;
2. to use fixed space partitioning hierarchical trees to sort the patches. This is similar to the cluster trees mentioned previously from [56] [1], but without the k-means clustering stage. Ignoring the k-means clustering permits a simpler tree building procedure, at the cost of having an unbalanced tree and many empty internal nodes. This is however tempered by the possibility of easier implementation on GPUs, as demonstrated for octrees (in dimension 3) in [83] and [84].

Furthermore, this simplified tree implementation also yields a linear data layout in the memory, which is a key factor in exploiting the hardware fast cache memory pipe-lines and contributes to the overall speed of the implementation.

The algorithm proceeds with the following steps, that should be obvious from the above discussion introducing the Patch Spectral Hashing:

1. Note that Brox's approach was developed independently from our work simultaneously to the method described here. However, our approach was submitted to publication until this thesis.

Figure 3.13: *Noisy example. The input sequence was corrupted with a Gaussian noise of variance 0.05 (the images were rescaled between 0 and 1). In the first case (middle image), we took wider patches (of size 7-by-7 pixels) to make the patch comparison process more robust, which also removed some details with the noise. In the second experiment (right image), we considered a lower threshold to declare a reference patch meaningful, preserving more details.*

Learning: the first step is to build a space where non-local queries become local and fast. Classically, it is made of:

1. dimensionality reduction by PCA;
2. embedding of the input patches in an acceleration structure (described below);

Denoising: the sorted patches are denoised using radius-search queries, where the radius corresponds to the preselection threshold. For each patch:

1. its non-local neighbors list is initialized with the patches in the same cell of the tree;
2. the list is expanded with the patches in the neighboring cells to account for query patches that are located near the border of adjacent cells;
3. perform the averaging from this list.

A schematic overview of this algorithm is shown in Fig. 3.14.

3.5.1 Building and querying the patch tree

Building a fixed space partition. In order to design an efficient (and computationally friendly) data structure to embed the patches after PCA, we chose to extend the idea of quadtrees [85] [86]

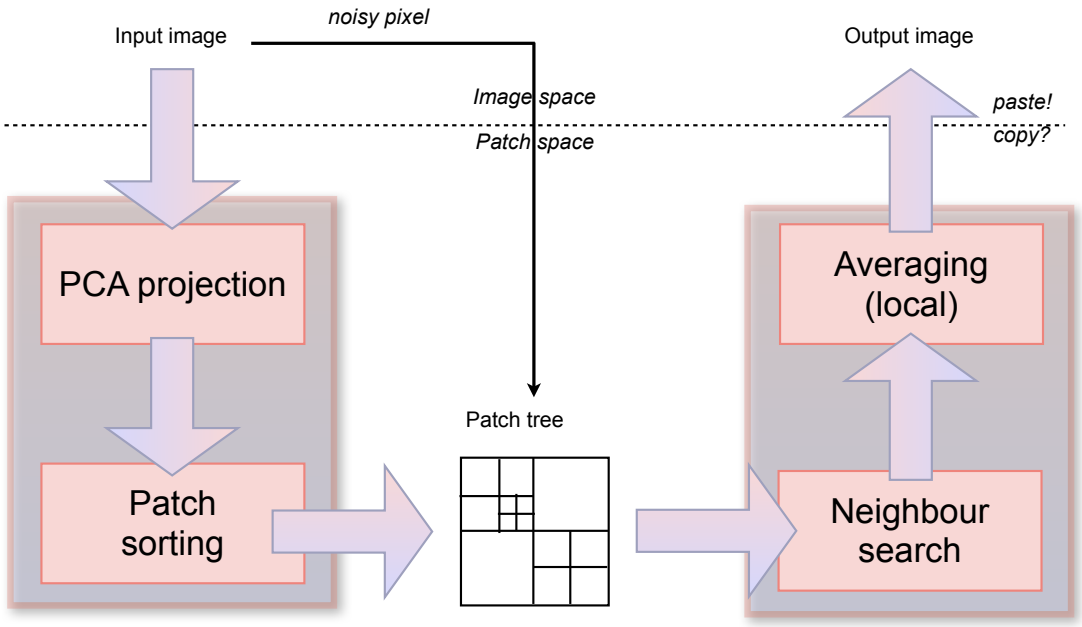


Figure 3.14: *Accerating NL-means with binary space partitioning trees*

and octrees to spaces of arbitrary dimension d. An initial node contains the complete space and patches are inserted sequentially. During the insertion, when a node contains enough patches and is considered full, it splits into 2^d child cells that are axis aligned and subdivides the original volume into cells of equal volume. The subdivision process is repeated if needed until the leaves do nnot have more elements than a predetermined maximal size.

This process is illustrated in Fig. 3.15 in the case $d = 2$. Since a cell is divided into $2^2 = 4$ child nodes, the tree corresponding to this special case is called a *quadtree*, and the case $d = 3$ yields to $2^3 = 8$ children per internal node and the corresponding trees are called *octrees*.

Since the number of children of a node grows exponentially with the dimension d of the space, these structures clearly lead to an extra memory consumption (including many empty inner nodes) in high dimensions, and in practice the case $d > 3$ is almost never met (kd-trees are used instead). However, there are several advantages in using them especially when the implementation platform does not have a *pointer* notion allowing random access to memory places:

- the child nodes can be allocated in contiguous memory because their number is constant, thus allowing the use of linear arrays in the implementations;
- the total number of possibly allocated nodes can be computed from the maximum depth of the tree;
- since the space partition is fixed [1] it is possible to compute *a priori* the path from the root of the tree to the leaf containing a given point. This path is constant regardless of the inner nodes that were already allocated: the traversal procedure only needs to stop following it

1. While their coordinates are known *a priori*, the inner nodes are of course allocated only if needed in order to limit the number of nodes to explore in a query.

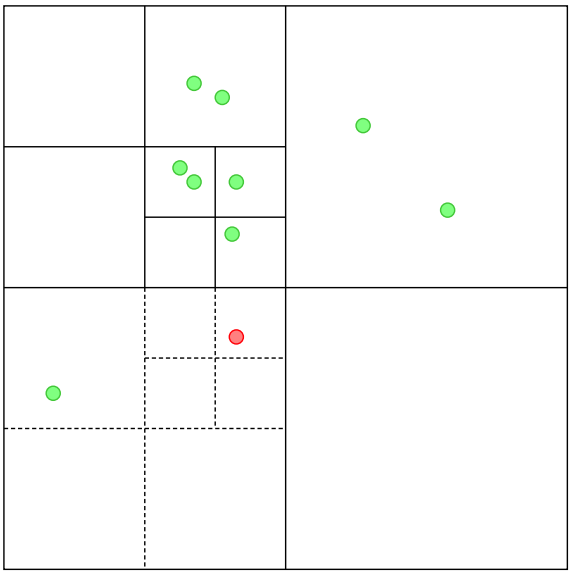


Figure 3.15: *Quadtree example. In this case, a node was split when it contained more than 2 data points. Dashed lines depict cell separations (or equivalently internal nodes) that have not been allocated because there are not enough points in the lower-left quadrant. The path to the red point however can be computed independently of what nodes were created or not: the full path is computed, then one simply stops the tree traversal when a leaf is reached.*

when a leaf is reached (see also Fig. 3.15).

Fast queries. The two first properties (locality in memory and pre-determined maximal partition) were exploited for example in [87] for GPU implementations of various physical simulation algorithms. Furthermore, the third property was pointed out by Frisken and Perry [88] who derived simple formula for tree traversal situations, including finding the leaf enclosing a given point.

These formula leverage an IEEE-compliant floating-point format implementation which is widespread in the industry [1] and exploit the fact that a child cell os obtained by halving its parent along each dimension. Let us consider the simpler one dimensional case and a tree of maximum depth equal to 3 used to sort numbers between 0 and 1 in Fig. 3.16. In this case, the root has the level **ROOT_LEVEL** = 3 and a leaf has a level 1 or 0 (corresponding to the smallest possible cells). Then, a locational code $LC(x)$ is computed as:

$$LC(x) = binary(\lfloor x \cdot 2^{\mathbf{ROOT\_LEVEL}} \rfloor), \tag{3.19}$$

where $binary(x)$ stands for taking the binary representation of the integer x. Note that the conversion from a real number to an integer in Eq. (3.19) is fast if the floating-point representation follows the IEEE standard [89]. The binary representation of $LC(x)$ is interpreted as a branching pattern to follow from the root of the tree to reach the corresponding leaf, and the leftmost bit corresponding to the root is always 0.

Let us take two examples from Fig. 3.16. First, with $x_1 = 0.15$, we have:

$$LC(x_1) = binary(\lfloor 0.15 \cdot 2^3 \rfloor = binary(1) = 001,$$

1. When in doubt, this representation can be enforced on demand by current C/C++ compilers such as GCC.

i.e., we need to branch first left, then right. Then, with $x_2 = 0.55$, we compute:

$$LC(x_2) = binary(\lfloor 0.55 \cdot 2^3 \rfloor = binary(4) = 010.$$

Hence, we need to branch right, then left, but since we are already on a leaf the traversal stops.

These formulas are computed and implemented independently for each dimension, in a direct extension of [88] that was limited to the 2D case. Hence, the produced codes are not interleaved and we have one locational code per dimension (up to 7 in our experiments).

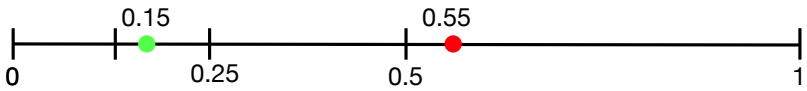


Figure 3.16: *Illustration of binary code computations on the real line. The green dot corresponds to the point x_1 in the text, and the red dot to x_2. The branching pattern from the root to the points can be computed using simply the knowledge of the maximum tree depth. Please refer to the text for more details.*

Fighting the curse of dimensionality. By design, each time a node is split in the tree, the enclosed volume in the feature space is divided in 2^d subvolumes, a number that can quickly become huge. To fix the ideas with actual numbers, we have tested the proposed algorithm for a feature space dimension d equal to 2, 3 and 7. In the latter case, the number of siblings of a node is equal to $2^7 = 128$. As a consequence, the feature space becomes highly fragmented when d grows, and a query point is more and more likely to stand near the border of its cell.

This phenomenon creates a kind of *curse of dimensionality* [90] [91] for the queries in the patch tree: for a given search radius, the number of nodes that could be expired in the worst case grows exponentially. In order to avoid visiting all the potential neighbors, we adopt the following strategy:

- we fix a maximum search radius ρ. This is the same as the minimal radius of a bounding volume in the feature space, since this is our obvious preselection parameter;
- for each dimension of the feature space, we compute the sign s_i of the difference $x_i - c_i$, where $\boldsymbol{x} \in \mathbb{R}^d$ is the query point and $\boldsymbol{c} \in \mathbb{R}^d$ is the center of the leaf containing $\boldsymbol{x}$;
- for each dimension, we form a new query point defined as:

$$\boldsymbol{y}^i = (x_1, \cdots, x_i + s_i\rho, x_{i+1}, \cdots, x_d)^T;$$

- we query the corresponding cells via their locational codes, as described above.

This simple trick allows us to divide by 2 the number of neighbors to be explored.

3.6 Conclusion

In this chapter, we have explored several ways to make the non-local exploration fast, thus enabling real non-local algorithms instead of semi-non-local ones. This comes via the use of several patch preselection strategies that avoid comparing non-related patches. Furthermore, we have proposed two novel preselection algorithms via clustering in the principal components space of the patches. The first and most efficient one is an adaptation of the spectral hashing that was proposed in the context of content-based image retrieval. The second one does not scale very well with the

patch dimensionality. It is based on fixed binary space partitions, and can be advantageous if the target implementation platform is limited (especially in terms of random memory access, such as GPUs).

In the next chapter, we will leverage the fast non-local query algorithms presented here to build a novel non-local denoising algorithm by a sparsifying transformation of the noisy data.

4 Leveraging non-local sparsity

We have seen in chapter 2 through the example of NL-means and its derivatives that the redundancy inside an image could be exploited in order to yield efficient yet simple algorithms. However, this simplicity is obtained by considering only simple combinations of the input data, such as weighted averaging and iterative diffusion processes. Hence, these algorithms do not exploit another powerful property that empowers many Image and Signal Processing algorithms: *sparsity*.

The past 20 years have seen a powerful trend in leveraging sparsity constraints for various Signal, Image, and Information Processing problems, including very different tasks such as denoising or classifier training. How dos it come that this family of constraints has become so popular? The sparsity of a phenomenon in a suitably chosen basis is indeed a very appealing and generic property to enforce, because it is verified by most of the signals of interest including of course images. Qualitatively, its interpretation is that for a given class of signals there usually exists at least one overcomplete dictionary in which any instance of this class will have a spare representation, *i.e.*, it is represented by few meaningful coefficients while the remaining ones are zero (sparse signals) or quickly negligible (compressible signals). One can think for example to a song signal: while any frequency could theoretically be present in the measured signal, in practice only frequencies corresponding to notes and to the tones of the instruments will be present.

Sparsity is obviously leveraged by image compression: e.g., JPEG compressed files are obtained from retaining the most important coefficients of the Discrete Cosine Transform (DCT) of each block of 8×8 or 16×16 pixels inside the original image [1], but it was explicitly introduced as a Signal Processing tool by Donoho *et al.* in several steps (most notably [92, 93]). Significant subsequent works have shown that it can be used for image restoration too, and more generally for inverse problems in imaging.

In this chapter, we will build upon the patch spectral hashing introduced in the previous chapter

1. The final JPEG file is obtained after some subsequent and non-trivial quantization and coding operations. The heart of the process is that fewer coefficients than the original complete set are needed for a faithful compressed image.

to form groups of visually related patches, that we will denoise jointly by a sparsity-promoting operation. The procedure that we propose exploits the very high similarity between the patches grouped together in a bucket of the spectral hash table, and thus avoids the need for an explicit dictionary learning or sparse coding stage. Furthermore, we will see that group norms can be used to enforce *social and structured sparsity*, *i.e.*, each output patch is sparse and these sparse representations do not vary much inside a given group. The minimization of these mixed norms is obtained by a thresholding operation, yielding a simpler resolution scheme than the previously proposed iterative ones.

4.1 Non-local sparsity

We have previously introduced in chapter 3 a few algorithms that accelerate NL-means via a prior clustering of the patches. Given a cluster of similar patches, a natural way of leveraging a non-local sparsity property is to find patch estimates that:

– correctly represent a smoothed (denoised) version of the input data;
– are sparse in the chosen representation frame.

These constraints were popularized by the BM3D algorithm, that we will describe next and that is still considered one of the top-ranking denoising method several years after its introduction. Unlike NL-means, BM3D is algorithmically complex, and very few derivative works outside the original team were subsequently proposed. The idea of designing algorithms enforcing non-local sparsity constraints became on the other hand popular, and we will review different mathematical approaches that were applied for this purpose.

4.1.1 Introducing non-local sparsity: BM3D

Block Matching 3D (BM3D) is built around the two ideas of *i)* grouping together similar patches and *ii)* finding a suitable set of jointly sparse representations for a group of patches. While BM3D was presented initially as a two-stage method, we present first the inner denoising process that we will call *BM3D-core* in order to avoid introducing some unnecessary complexity. We then show with *BM3D-full* how this building block is employed inside the global two-tier scheme.

BM3D-core. In its core part, BM3D builds groups of patches by visual similarity, denoises these patches by enforcing their sparsity in some transform domain, along with the overall sparsity of the group, and back-projects the resulting patches into an output image. We will study these three steps successively.

The input image is first divided into overlapping patches, as with NL-means and the other non-local algorithms. The sampling of the patches is explicitly taken loose in order to have fewer patches to process.

The search for visually similar patches is directly inspired by algorithms from the video compression field. Successful video compression algorithms (such as the ones defined in the MPEG standards) do not encode explicitly all the frames, but instead compress a subset of reference frames and use *motion estimation* to propagate this result. Given a patch (or *block*) in a reference frame, its temporal trajectory is computed by looking for the most similar block in the subsequent frames. In

order to be fast, this motion estimation usually relies on so-called *Block Matching* algorithms that exploit different tricks such as using a Sum of Absolute Differences (SAD) instead of the classical Euclidean (squared) norm, or diamond-shaped search patterns to maximize the chance of finding a suitable temporal neighbor at the beginning of the search (see [94] for a review of block matching methods and [95] for a popular diamond search algorithm). BM3D-core uses a block matching procedure to gather all the patches that are similar enough to a given input patch inside a search window, the only difference being that the search is not interrupted as soon as one neighbour is found (as is the case for motion estimation).

An example of a patch stack can be observed in Fig 4.1. It clearly shows the 3 dimensions of BM3D. The first two dimensions serve as the frame of reference inside a given patch, with the slight abuse that a patch can be transparently replaced by its transform: domain transforms such as the DCT, the Hadamard transform or a wavelet transform map 2D signals to 2D sets of coefficients. Constraints that are applied in this 2D frame aim at creating patches with sparse representations. The third axis goes along the stack of patches. The constraint that will be enforced along this axis aims at finding and preserving the common structure of the patches in the stack.

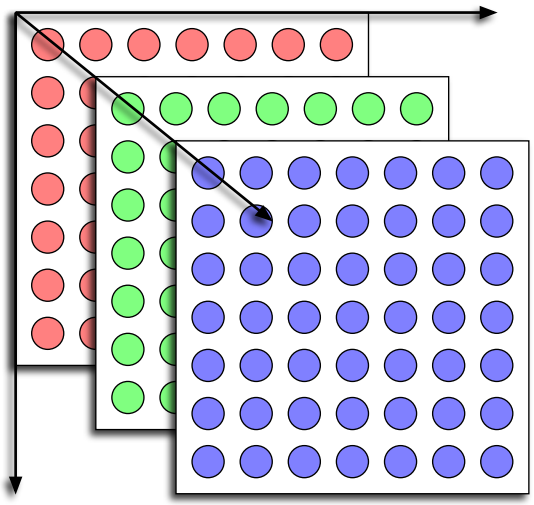


Figure 4.1: *Example of a patch stack. Each patch has a 2D frame of reference, while a third dimension is used to navigate in the stack.*

The second stage of BM3D-core is the joint denoising of similar patches by a sparsifying process, called *collaborative filtering* in BM3D publications. Each group of patches is interpreted as a *stack* where a given patch is a 2D slice. Each 2D slice then undergoes a transformation to a domain where image blocks are known to be sparse, usually a DCT [1]. A subsequent transform is applied vertically along each column of coefficients considered as an 1D vector, and a clean sparse 1D vector is computed using either Hard Thresholding (HT, see section 4.2.2 below) or Wiener filtering [2]. These operations aim at enforcing the joint sparsity of the group with respect to the current transform coefficients and perform the actual denoising. This chain of transforms is depicted by the 3D

1. The authors also mention the Walsh-Hadamard, Haar and bi-orthogonal wavelet transforms.
2. While the sparsifying nature of Wiener filtering seems unclear, it is nonetheless used here in this role by the authors of BM3D. It is true, however, that it will remove some high frequency components due to the noise.

(actually a 2D + 1) in the name BM3D. Finally, the 3D transformation is inverted to produce a stack of clean patches (see Fig. 4.2 for an more visual overview).

Note that this 3D transform process does not explicitly enforce some stack or group sparsity measure. Qualitatively, it proceeds by computing an underlying clean mean patch. This mean patch is sparse because of the application of the hard thresholding operation. The variations from this mean to the original blocks of the stack are sparse (again, thanks to the HT step) or correspond to some least squares criterion (when using Wiener filtering as the third transform).

In the last stage of BM3D-core, clean patches are back-projected to their original location in the image. Since the blocks overlap, one needs to find a criterion to merge the different estimates for a given pixel. In BM3D, this is done by giving a weight to each estimate which is inversely proportional to the variance of its original stack: when the denoising is successful, the variance of the 3D stack is likely to be low, yielding more confidence in the estimated value. The final denoised value at each pixel is obtained through a weighed average of all the reprojected values.

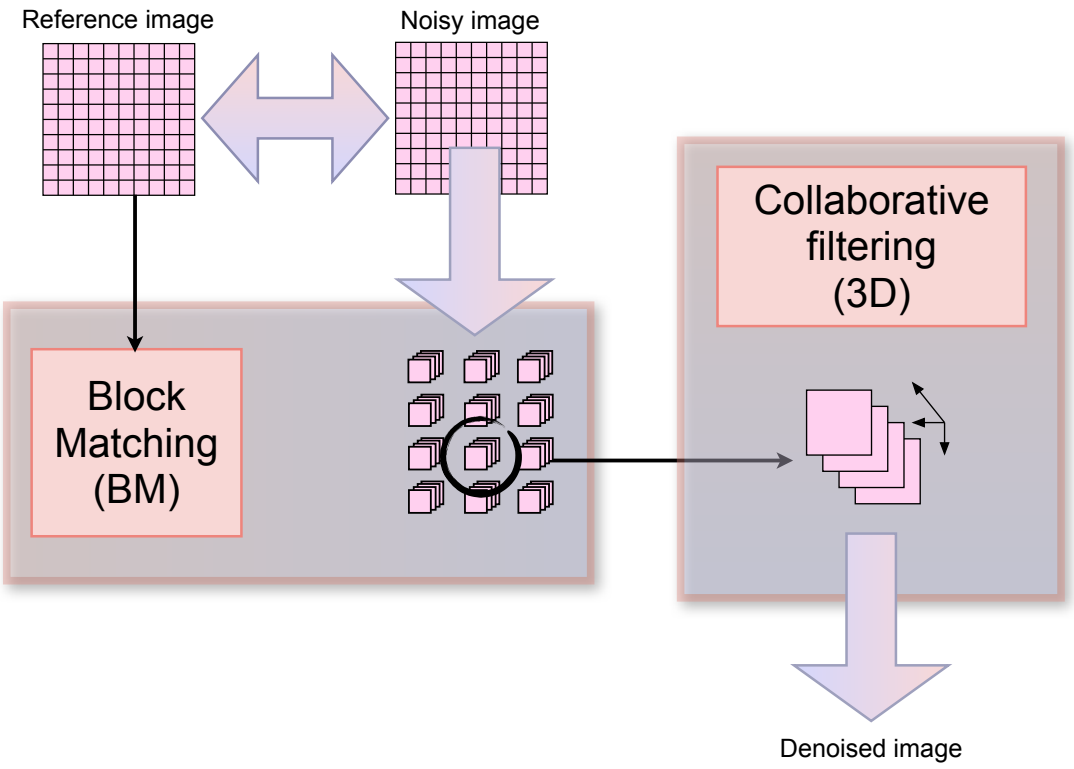


Figure 4.2: *Overview of the BM3D-core process. The reference image is used for the block matching only. It can be equal to the noisy image, but this is not mandatory. The 3D collaborative filtering is usually a sparsity-promoting operation (typically the Hard Thresholding) after applying a 2D domain transform to each slice of a patch stack followed by a domain transform along each column of the stack. For example, the 2D transform could be a DCT to identify the important spatial frequencies of each patch, and the last 1D transform a Haar wavelet transform to analyze the variations of these frequencies inside the stack.*

BM3D-full. Since patch similarities computed from the noisy image are not always very reliable, it makes sense to have a two-tier process:

1. an intermediate clean image is produced from the input noisy data only. A more aggressive denoising process (such as hard thresholding) can be used during this stage;
2. the final image is estimated by denoising the input noisy image with similarities computed on the intermediate denoising product.

Hence, a standard implementation of BM3D comprises two successive applications of BM3D-core:

1. computation of an intermediate image using a 2D DCT followed by a 1D Haar wavelet transform as the 3D transform, and hard thresholding as the sparsifying operation;
2. production of the final estimate by using the intermediate product to compute the patch stacks. The 3D transform is still a 2D DCT + 1D Haar wavelet transform, but the final denoising process is an implementation of Wiener filtering.

With respect to the scheme presented in Fig. 4.2, the findl BM3D denoising procedure is to apply the block BM3D-core twice: the first time with the reference image equal to the noisy image, and the second time with the output of the first denoising process as reference image.

Remark 6. *This two-tier denoising procedure has also been found to be effective for NL-means in [15]. Hence, a fair comparison between NL-means and BM3D should include also a two-tier NL-means or be limited to one application of BM3D-core.*

Extensions of BM3D. The genuine BM3D described above was extended in several ways. Some of them are straightforward, such as applying BM3D to movies by extending the similarity search domain to include neighboring frames in the input stream. Among the most interesting ones, we can pick:

– the Shape Adaptive (SA-PCA, SA-DCT) variants [96] [97]. In these algorithms, the size of the patches is not fixed. A polynomial approximation is used instead and a patch can grow as long as the approximation error is low. Hence, this patch extraction procedure can integrate information on a large spatial neighborhood in flat areas. Note that a similar idea for NL-means was proposed by Kervrann and Boulanger in [98], and a Shape Adaptive NL-means was also proposed recently in [99];
– BM4D [100] and V-BM4D [101] extend the BM3D principles to stacks of volumetric data (BM4D) and stacks of spatiotemporal cubes (V-BM4D) instead of 2D image slices.

The case of V-BM4D is interesting. Unlike other non-local movie denoising such as [102] it uses spatiotemporal blocks instead of 2D patches, although spatiotemporal blocks are said to be less efficient than purely spatial patches in the related non-local literature. V-BM4D does not use raw spatiotemporal data however, which may explain this difference of appreciation. Namely, the blocks of V-BM4D are obtained by following the temporal trajectory of each patch in the first frame of the given time window to process. Hence, it relies partially on motion estimation, but this dependancy is likely to produce more relevant spatiotemporal blocks than the original noisy data. This assumption is backed up by the results of V-BM4D [101], but also by a prior paper on high quality video denoising by Liu and Freeman [103] who integrated information along optical flow trajectories, and moderates the claims from [102] [1].

1. In particular, the title of this paper, which reads *Denoising image sequences does not require motion estimation* seems a bit exaggerated in light of subsequent work.

4.1.2 Is BM3D the ideal non-local denoising method?

From its description in the previous section, it is clear that BM3D and its derivatives are technically sound, building on now well established ideas (non-locality and sparsity). However, and in spite of being a top-performing algorithm, BM3D did not generate the same trend of derivative works as NL-means.

Like most non-local algorithms, BM3D is semi-nonlocal: in order to stay fast, block matching is performed in smaller search windows. Beyond block matching, implementing BM3D can be tricky and there are several caveats:

- the choice of the transforms (DCT, wavelets, Walsh-Hadamard transform in 1 or 2D) requires that, in practice, the blocks and the stacks have a size that is a power of 2 for efficient implementations. While this can be easily enforced for 2D patches, this is an important drawback for the stacks: it is easy to have 2 patches in a stack, it becomes difficult to have 8, and if ones really wants to have 16 or 32 patches in a stack for better denoising then a stack will very likely include some outliers because of the noise;
- the method involves many parameters to tune, and their value inside an execution of BM3D-core will obviously vary depending on the fact that it is applied on a noisy image or on a noisy image accompanied by the intermediate result;
- more subtle, parameters of the algorithm also include the choice of the transforms to use. In order to obtain the best results, the transforms used inside BM3D-core also vary depending on the fact that it is applied on the noisy image or on pairs of noisy and intermediate) images. Hence, a good instance of BM3D (in the sense of good denoising performance) must include implementations of various transforms, yielding an increased complexity.

The interested reader will find in [104] a complete study of BM3D with respect to its parameters.

Finally, the complex step of collaborative filtering hides the fact that some kind of group sparsity is enforced by the algorithm, although the authors insist on this point in their communications (see for example in [105]). Therefore, it seems legitimate to further study how this group sparsity property could be enforced in a more straightforward and intuitive way.

Remark 7. *Independently of BM3D, the same ideas of* i) *finding a good representation for a set of similar patches (though not sparse) and* ii) *iterating twice the grouping process, were developed by Kervrann et al. in [15] under the name of* Bayesian non-local means filter. *In this work, the stacks of patches are considered as spatially-varying dictionaries from which optimal estimators (under a white noise assumption) are derived for each patch involved in the group.*

4.1.3 Non-local sparsity through dictionary training

Non-local sparse coding. To tackle the drawbacks of BM3D (including a more straightforward algorithmic formulation), subsequent researchers did propose non-local algorithms relying explicitly on sparsity. An important step was the introduction of algorithms based on patch dictionaries learned from a database of images. The main motivation behind this family of methods is that the dictionaries will capture only the meaningful variations of the appearance of the patches. Hence, even if trained on noisy data, they will more likely retain the lower frequencies and non-random parts of the patches and reject the noise. Requiring that the patches have a sparse decomposition on the dictionary then prevents from introducing further blur in the output, because it forces the

reconstruction algorithm to select a linear combination of only a few significant atoms from the dictionary, instead of equally spreading the error over many of them.

Adopting the same approach as BM3D, *i.e.*, stacking similar patchs then computing sparse estimates, the problem of non-local denoising is then to find a process that, for a given group of patches, will have the following properties:

(Property 1) it should enforce the sparsity of each group member by discarding small coefficients that are due to the noise;

(Property 2) it should exploit at the same time the common structure of the patches inside a given group to obtain a better estimate, effectively taking advantage of the preselection process.

In this regard, the K-SVD based algorithm in [106] can be considered as an important step in effectively inspiring several other non-local algorithms. Note however that it only relies on the exploitation of Property 1. In a first step of *sparse coding*, a sparse approximation of the current image patches on an overcomplete dictionary is obtained using standard techniques such as Orthogonal Matching Pursuit [107]. During this stage, a data term enforces the proximity between the image obtained from the estimated patches and the input image. In a second step, the patches are kept fixed and the dictionary is updated using the K-SVD algorithm [108] in order to find a new dictionary that produces sparser representations of the patches. This process is iterated several times, and the output image is obtained by merging the final patch estimates. The initial dictionary is obtained by training on a dataset of high quality images or from the noisy input image.

This algorithm is not compared to BM3D in the original paper. On the negative side, it is slower (because of the K-SVD part and the iterative process) and it can get trapped in local minima. The constraints applied to the patches are nonetheless made more obvious. This work was further extended in [109] to color images by imposing an additional constraint that aims at preserving the relative importance of the different color channels (red, green and blue): a naive application of the grayscale algorithm outputs grayish images by averaging the residual errors among the three colors.

Non-local simultaneous sparse coding. This K-SVD based algorithm served as a basis for the non-local sparse models of Mairal *et al.* [110], who attempt to also leverage the Property 2 above, *i.e.*, exploiting some common structures shared by the patches within a stack. In order to improve the quality of the results, they proposed to replace the sparse coding part by a Learned Simultaneous Sparse Coding (LSSC) stage [111] . The goal is still to find patches that have a sparse representation on the chosen dictionary (and that fit with the data term), with the additional constraint that visually similar patches should also have a close decomposition in the dictionary. Hence, a first step of block matching is introduced in the algorithm in order to group together related patches, as in BM3D. However, by limiting this search to a smaller spatial neighbourhood and retaining up to 64 patches per stack, this yields again a kind of semi-nonlocal constraint. The simultaneous sparse coding brings in this case a performance increase (measured with the Peak Signal-to-Noise Ratio) of almost 1 dB compared to the sole sparse coding constraint, which makes the proposed algorithm perform slightly better than BM3D and the K-SVD based method from [106].

Note that the simultaneous sparse coding constraint resembles the smooth underlying appearance manifold assumption [112, 113] that we have already met in chapter 2 for non-local inpainting. It means that when traveling from a patch to one of its neighbours in the stack, there should not be any abrupt change in their manifold-based coordinates.

Another improvement of the LSSC-based algorithm [110] over [106] and [110] is the use of an

online dictionary learning algorithm developed by the authors in a previous paper [114] that is more efficient than the K-SVD: it can be trained on larger training sets. This yields a more representative generic patch dictionary which helps in improving the results and in generalizing the method to simultaneous denoising and demosaicking.

Finally, Dong *et al.*[115] sought to improve the efficiency of the simultaneous sparse coding step by adding an explicit patch clustering operation in the minimization process. The objective function is then composed of a classical squared ℓ_2 data term, plus two additional ℓ_1 regularization terms: one to enforce the sparsity of the patch decompositions and another to measure the distances between the center of a cluster (expressed in the dictionary) and the dictionary-based coordinates of the patches assigned to the cluster at hand.

It is not clear whether this last ℓ_1 penalization term is relevant or not: it will promote sparse disparities between the center of a cluster and the patches assigned to it. Random noise however does not create this kind of error pattern, and since the chosen clustering algorithm is the k Nearest Neighbours (kNN) a correct clustering stage will produce cluster centers that minimize the Euclidean distance to the corresponding group members, and not the ℓ_1 distance. The resulting complex functional is minimized by a reweighted ℓ_1 scheme, and the complete algorithm also involves a step of clustering by kNN to adapt the clusters to the data being processed. This algorithm brings some minor improvements over BM3D in terms of PSNR, and is sometimes better, sometimes worse than BM3D in terms of the structured similarity metric SSIM [116] which captures better the errors introduced in the geometric content of the images.

Note that a prior paper [117] introduced the same idea of explicitly clustering the patches and computing a best dictionary per cluster. However, the authors of this paper did run the clustering subroutine only at the first iteration of their algorithm, which may not yield the best possible clusters for the current image estimate. Furthermore, since they used Gaussian kernel regression to compute the reconstruction frame their results suffered from artifacts in flat uniform areas, where they are the most disturbing.

4.1.4 Motivations for a novel non-local sparse algorithm

As we have seen in this section, several successful sparse non-local image restoration algorithms were proposed. Although they do not always reference it, they are closer to BM3D than to NL-means and build on similar principles:

- patches can be grouped together if they are very similar, and considered in this case as independent realizations of an underlying process. The exploration loop of NL-means is unnecessary, and actually may even be undesirable under certain conditions for the reasons encoutenered before (see chapter 3);
- denoised patches can be obtained by a sparse approximation in some well chosen basis, such as DCT or Haar-wavelet decomposition;
- the patches of a given group should be considered altogether in order to obtain more relevant sparse representations and to avoid artifacts.

However, these algorithms include practical inconveniences, such as being only partially non-local, unintentionally hiding the sparsity constraint behind the complexity, mixing prior learning and data adaptive dictionaries, or including iterative optimization subroutines. In the following sections, we propose a new non-local denoising functional designed to be fully non-local and to remain efficient even when processing a sequence of frames coming from a movie. The resolution

scheme will rely on the spectral hashing introduced in the previous chapter to adaptively cluster the input data without requiring an explicit dictionary learning step and without adapting this dictionary during the denoising process.

Furthermore, we propose to change the norm used to enforce the simultaneous sparsity constraint for a more suitable one. Indeed, one of the main difficulties that previous works have silently or explicitly tried to address is the fact that there are 3 dimensions involved in the non-local sparsity:

- sparse patches: as small images, it is quite intuitive that each patch should have a sparse representation in some dictionary, since this property is already exploited successfully on full size images. This corresponds to *individual coefficients* sparsity;
- as low-dimensional signals, the coefficients of each patch in some well chosen dictionary should be *jointly sparse*. Note that we also exploit this low-dimensionality property in the dimensionality reduction step of the patch spectral hashing. This corresponds to promoting some *group sparsity* property, since a patch yields a group of coefficients;
- since we have non-local groups of patches assumed to be different realizations of the same phenomenon, the relations between the patches of a given cluster should be taken into account to improve the result. This corresponds to the *social sparsity* introduced by Kowalski in [118] [1] and [119], *i.e.*, group sparsity computed by considering stacks of patches.

Because of these 3 dimensions, the choice of a triple mixed-norm (see Definition 3) seemed a natural choice to handle the patches and their relationship, instead of the hidden dependancies of the 2+1D transforms of BM3D or of the simultaneous joint sparse coding that adds an optimization subproblem in the solver.

4.2 Choosing a sparsity inducing penalization

In this section, we formally introduce the notion of sparsity in a mathematical framework. As seen before, there are different types of sparsity that can be enforced, from individual coefficients to neighborhood structures. Hence, we will start with the simplest one (individual coefficients sparsity and single norms) and gradually increase the complexity.

4.2.1 Mathematical background

We suppose that we observe a signal $\boldsymbol{y} \in \mathbb{R}^L$ of length L. This signal was formed from an original signal $\boldsymbol{x}$ corrupted by some additive noise $\boldsymbol{n}$. Furthermore, $\boldsymbol{x}$ can be decomposed in some dictionary $\boldsymbol{\Phi} \in \mathbb{R}^{L \times N}$ and we note $\boldsymbol{\alpha} \in \mathbb{R}^N$ its decomposition:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n} = \boldsymbol{\Phi}\boldsymbol{\alpha} + \boldsymbol{n}.$$

Then, we consider the task of recovering $\boldsymbol{x}$ from $\boldsymbol{y}$ under the constraint that $\boldsymbol{\alpha}$ is sparse. In the case of white noise, this can be cast as an inverse problem:

$$\hat{\boldsymbol{x}} = \boldsymbol{\Phi} \arg\min_{\alpha \in \mathbb{R}^N} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\alpha\|_2^2 + \lambda F(\alpha), \tag{4.1}$$

where $F(\alpha)$ is some sparsity-promoting *convex* but possibly non-smooth constraint on $\boldsymbol{\alpha}$ and λ is a constant balancing the importance of each term.

1. This special type of sparsity is introduced by Kowalski in this paper, but without explicitly naming it. The expression of *social sparsity* comes from later papers from the same author.

Let us also recall the definition of a proximate operator introduced by Moreau in [18]:

Definition 1 (Proximity operator)**.** *Let $F : \mathbb{R}^N \to \mathbb{R}^N$ be a lower semi-continuous convex function. The proximity operator (or equivalently the proximal mapping) of F, written $\text{prox}_{\lambda F} : \mathbb{R}^N \to \mathbb{R}^N$, is defined as:*

$$\text{prox}_{\lambda F}(\boldsymbol{z}) = \underset{\boldsymbol{\alpha} \in \mathbb{R}^N}{\arg\min}\ \lambda F(\boldsymbol{\alpha}) + \tfrac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{z}\|^2. \tag{4.2}$$

Then, if one knows a solution or an explicit formulation to the proximity operator, the Iterative Shrinkage/Thresholding Algorithm (ISTA) described in Alg. 2 solves the problem (4.1) [17, 120]. Note that ISTA is related to the broad family of the forward-backward minimization algorithms [17]: it combines at each iteration a forward term (the inner gradient descent step) followed by a backward term (the proximity operator).

Algorithm 2: Iterative Shrinkage/Thresholding Algorithm (ISTA).

1: Take $\boldsymbol{\alpha}^0 \in \mathbb{R}^N, \gamma = \|\boldsymbol{\Phi}\boldsymbol{\Phi}^*\|$.
2: **for** $k = 1$ to n **do**
3: $\quad \boldsymbol{\alpha}^k = \text{prox}_{\frac{\lambda}{\gamma}F}\left(\boldsymbol{\alpha}^{k-1} + \frac{1}{\gamma}\boldsymbol{\Phi}^*(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}^{k-1})\right)$;
4: **end for**
5: **return** $\hat{\boldsymbol{x}} = \boldsymbol{\Phi}\boldsymbol{\alpha}^n$.

Remarks. A well known example of a proximal operator is the soft-thresholding obtained in the case $F(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|_1$. The interested reader is referred to [17] for more examples of proximity operators and proximal algorithms. While ISTA can converge slowly (but is still widely used thanks to its simplicity), faster variants such as as FISTA [23] have recently been proposed. Finally, note that in the case where $\boldsymbol{\Phi}$ is orthonormal [1] (for example in the case of the DCT or wavelet transform), then $\boldsymbol{\Phi}\boldsymbol{\Phi}^* = Id$, yielding $\gamma = 1$ and ISTA reduces to the single iteration

$$\hat{\boldsymbol{x}} = \boldsymbol{\Phi}\,\text{prox}_{\lambda F}\left(\boldsymbol{\Phi}^*(\boldsymbol{y})\right).$$

Soft and generalized thresholding. Given $\boldsymbol{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$, the soft thresholding operator is defined by:

$$\mathcal{S}_\lambda(x_i) = \text{sign}(x_i) \cdot \max\left(|x_i| - \lambda, 0\right) := \text{sign}(x_i)\left(|x_i| - \lambda\right)^+, \tag{4.3}$$

where λ is the designated threshold.

Assuming that $|x_i| \neq 0$ (otherwise the result is obviously 0), the soft thresholding in Eq. (4.3) of threshold τ_i can also be cast in the form of a generalized thresholding defined as follows:

$$\mathcal{S}_\lambda(x_i) = \text{sign}(x_i)\left(|x_i| - \lambda\right)^+ \tag{4.4}$$

$$= \frac{x_i}{|x_i|}\left(|x_i| - \lambda\right)^+ \tag{4.5}$$

$$= x_i\left(1 - \frac{\lambda}{|x_i|}\right)^+ \tag{4.6}$$

$$\mathcal{S}^g_{v_i}(x_i) = x_i\left(1 - v_i(x_i)\right)^+. \tag{4.7}$$

1. This result does actually hold more generally for tight frames, but we will consider only orthonormal transforms in this chapter.

The coefficient $v_i(x_i)$ is a shrinkage coefficient that can vary from coefficient to coefficient. Hence, the standard soft thresholding operator corresponds to a shrinkage coefficient $\lambda/|x_i|$. While the soft thresholding is usually expressed in the form of Eq (4.3), the generalized thresholding variant can be preferable in the sequel in order to provide a unified and simple way to express the various proximal mappings.

4.2.2 Sparsity of coefficients: the lasso

We consider first the case where we only want to promote a sparse solution, without any structure or dependancies between the coefficients. Strictly speaking, the real sparsity measure is the ℓ_0 pseudo-norm that counts the number of non-zero coefficients of a given vector $\boldsymbol{x} = (x_1 \dots x_n)$:

$$\|\boldsymbol{x}\|_0 := \sharp\{i : |x_i| > 0\}, \tag{4.8}$$

The sparse optimization problem in Eq (4.1) then writes:

$$\hat{\boldsymbol{x}} = \boldsymbol{\Phi} \underset{\alpha \in \mathbb{R}^N}{\arg\min} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_0.$$

However, the associated minimization problem is known to be NP-hard and hence is intractable in general. In practice, this pseudo-norm can be used by fixing the maximum number K of non-zero components, and the associated proximity operator enforcing this constraint is the Hard Thresholding operator $\mathcal{H}_K$ which is computed by keeping the K coefficients of greatest amplitude and setting the remaining ones to 0. Note that in this case the ISTA algorithm is known as IHT for Iterative Hard Thresholding. Finally, recall from Sec. 4.1.1 that hard thresholding is the sparsifying transform used in BM3D and its derivatives.

In order to bypass the problems of the ℓ_0 pseudo-norm, it is now classical to use the ℓ_1 norm instead, because it is a convex relaxation of the ℓ_0 one. This can be easily seen in 2D (see Fig. 4.3). The ℓ_1 norm was implicitly introduced as a sparsity-promoting constraint for image restoration tasks by Donoho *et al.* in a 1992 paper [92] entitled *Maximum Entropy and the Nearly Black Object* and dedicated to spectroscopic images restoration. It was then re-introduced in the context of sparse representations in overcomplete dictionaries in another later paper by Chen, Donoho and Saunders [93] and by Tibshirani [121] for sparse regression. The ℓ_1 norm has the nice property to promote sparse solutions because of the pointed shape of its unit ball: it is very likely to be tangential to the hyperplane corresponding to a set of constraints in only one point near an axis (Fig. 4.3). Hence, the solution of finding a point that respects these constraints while having the smallest ℓ_1 norm has many chances to be sparse.

The corresponding sparse optimization problem reads:

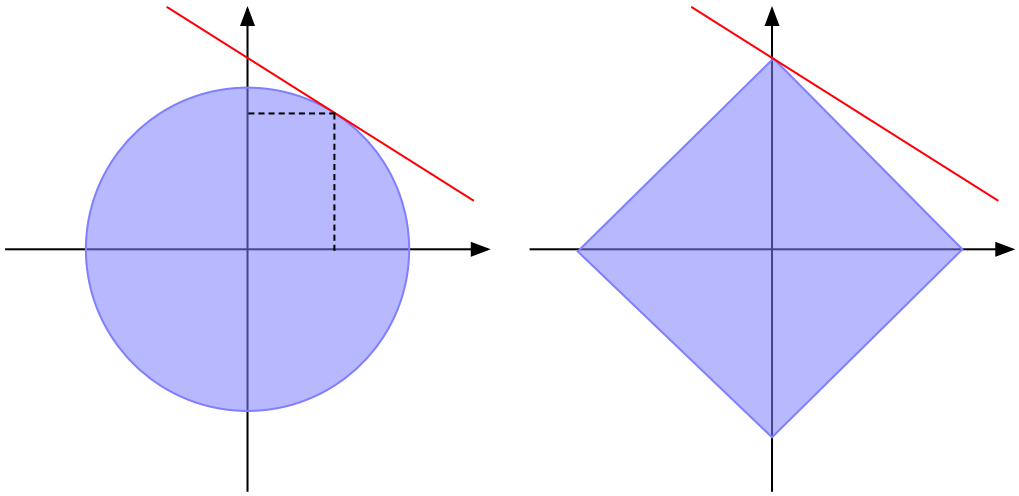
$$\hat{\boldsymbol{x}} = \boldsymbol{\Phi} \underset{\alpha \in \mathbb{R}^N}{\arg\min} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1.$$

This problem is now a classical one known under various names, such as Basis Pursuit DeNoising (BPDN) [93] in the context of representations in dictionaries or Least Absolute Shrinkage and Selection Operator (LASSO) [121] in sparse regression and feature selection.

It can be solved using ISTA, in which case one needs the proximity operator of the ℓ_1 norm which is known to be given coefficient-wise by:

$$\mathrm{prox}_{\lambda\|.\|_1}(\alpha_i) = \mathrm{sign}(\alpha_i) \cdot (|\alpha_i| - \lambda)^+, \tag{4.9}$$

i.e., it is the soft thresholding operator introduced in Eq (4.3) and that was already known in image restoration [1].



(a) Constrained ℓ_2 minimization (b) Constrained ℓ_1 minimization

Figure 4.3: *Constrained optimization with ℓ_2 and ℓ_1 penalties. The red line depicts the problem constraints. Then, one looks for an optimum on this line with minimum ℓ_2 or ℓ_1 norm by scaling the corresponding unit ball (depicted in blue in the pictures). From this simple 2D example, one can intuitively understand why the ℓ_1-norm, unlike the ℓ_2-norm, promotes the sparsity of the solution: unless they are parallel to its sides, the constraints (red line) will intersect the diamond shape at one of its tips, thus putting some components of the solution to 0. The ℓ_2 constrained problem does not suffer from the parallelism problem, but it can produce non-sparse solutions.*

4.2.3 Sparsity of groups of coefficients: mixed norms and group lassos

The lasso problem introduced previously promotes coefficient-wise sparse solutions, but does not consider the possible inter-coefficients dependancies that can arise in many cases. For example, if $\boldsymbol{\alpha}$ is computed via a multi-scale analysis of $\boldsymbol{x}$, one can wish to group together the coefficients corresponding to the same sub-band and process them independently of the other bands. In order to introduce these dependancies, we suppose now that they can be divided into non-overlapping groups. Hence, a coefficient becomes indexed by a pair (g, m) (instead f a single index i) where g indexes the groups and m indexes each member inside a group. Note that the groups need not have the same size and null elements can be added if this property is required in practice.

To measure the norm of a vector of coefficients with respect to the given structure, we need to introduce a two-level mixed norm:

Definition 2 (Two-level mixed norm)**.** *Let $\boldsymbol{x} \in \mathbb{R}^N = \mathbb{R}^{G \times M}$ be a point indexed by the pair $(g, m) \in \mathbb{N}^2$. The $\ell_{p,q}$ mixed-norm of $\boldsymbol{x} \in \mathbb{R}^N$ is defined by:*

$$\|\boldsymbol{x}\|_{p,q} = \left(\sum_{g=1}^{G} \left(\sum_{m=1}^{M} |x_{g,m}|^p \right)^{q/p} \right)^{1/q}. \tag{4.10}$$

The sparse estimation problem in Eq (4.1) becomes:

$$\hat{\boldsymbol{x}} = \boldsymbol{\Phi} \arg\min_{\alpha \in \mathbb{R}^N} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_{p,q}^q.$$

Different types of solutions will arise from different choices for p and q.

Remark 8. *In order to ease the understanding of the notations that can get confusing for two or three level mixed norms, one can remark that the order used in the $\ell_{p,q,r}$ norm corresponds to the norms applied respectively (and from left o right) to the individual coefficients, then the groups of coefficients, the hierarchy of the groups, etc. Note that coefficients however are indexed with the higher hierarchical level first: in $x_{p,q,r}$, the subscript r corresponds to the position of x in its groups, followed by its group index q and finally its cluster p.*

The group lasso. A classical choice is to take $(p, q) = (2, 1)$ corresponding to the $\ell_{2,1}$ norm. This choice is known under the names of Group-lasso (G-LASSO) [122, 123] or joint sparsity [124]. It has been applied in various fields such as Machine Learning (for joint feature selection, see [122]), Image Processing [125, 126] and more generally in Signal Processing [123, 124].

The corresponding proximity operator is a group version of the generalized thresholding [123] where the shrinkage factor $\nu_{g,m}(\boldsymbol{\alpha})$ depends on each group:

$$\nu_{g,m}(\boldsymbol{\alpha}) = \frac{\lambda}{\|\boldsymbol{\alpha}_{g,.}\|_2},$$

where $\|\alpha_{g,.}\|_2$ is the ℓ_2-norm of the group of index g. This yields the following coefficient-wise expression for the proximity operator:

$$\text{prox}_{\lambda\|.\|_{2,1}}(\alpha_{g,m}) = \alpha_{g,m}\left(1 - \frac{\lambda}{\|\boldsymbol{\alpha}_{g,.}\|_2}\right)^+. \tag{4.11}$$

The qualitative interpretation of Eq (4.11) is straightforward: for each group, the shrinkage coefficient is constant and depends on the group only. If the norm of the group is smaller than λ, then the whole group is discarded, otherwise all the non-zero coefficients are kept and shrunk. Hence, the group lasso constraint corresponds to « turning either on or off » each group of coefficients.

The elitist lasso. The opposite choice $(p, q) = (1, 2)$ yields a very different type of result. It corresponds to the $\ell_{1,2}$ norm, *i.e.*, the ℓ_2 norm of the ℓ_1 norm of each group. To understand why it is called E-LASSO for Elitist lasso (in audio signal processing [123]) or Exclusive lasso (in Machine Learning [127]), one needs to have a look at the corresponding problem and proximity operator. The sparse estimation problem reads in this case:

$$\begin{aligned}\hat{\boldsymbol{x}} &= \boldsymbol{\Phi} \arg\min_{\alpha \in \mathbb{R}^N} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2 + \lambda \sum_{g=1}^{G}\left(\sum_{m=1}^{M} |\alpha_{g,m}|^2\right)^{1/2} \\ &= \boldsymbol{\Phi} \arg\min_{\alpha \in \mathbb{R}^N} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2 + \lambda \sum_{g=1}^{G} \|\boldsymbol{\alpha}_{g,.}\|_1.\end{aligned}$$

Hence, it penalizes the ℓ_1-norm of each group while giving an equal importance to each group [1] groups. G-LASSO instead penalizes the ℓ_1 norm of a vector computed from the ℓ_2 norm of each group. Hence, the ℓ_1 norm of G-LASSO (applied on the groups as a whole) yields a sparse set of non-zero groups, and for each of these non-zero groups all the coefficients are equally shrunk.

To compute the proximity operator of E-LASSO, we need to introduce an intermediate sequence $z \in \mathbb{R}^N$ where each group $z_{g,.}$ is formed by the amplitudes of the corresponding group $\alpha_{g,.}$ [128]:

$$z_{g,m} = |\alpha_{g,m}| \ \forall (g,m).$$

Then, for each group $z_{g,.}$ we sort its members in descending order, obtaining a new group $\check{z}_{g,.}$:

$$\check{z}_{g,1} \geqslant \check{z}_{g,2} \geqslant \ldots \geqslant \check{z}_{g,M} \ \forall g.$$

Finally, we write M_g the index of the member of $\check{z}_{g,.}$ defined by

$$\begin{cases} \check{z}_{g,M_g} & > \lambda \sum_{m=1}^{M_g} (\check{z}_{g,m} - \check{z}_{g,M_g}) \\ \check{z}_{g,M_g+1} & \leqslant \lambda \sum_{m=1}^{M_g+1} (\check{z}_{g,m} - \check{z}_{g,M_g}) \end{cases}. \tag{4.12}$$

Qualitatively, M_g is linked to the non-uniformity of the amplitudes of the coefficients: it corresponds to the index of the largest meaningful member of $\check{z}_{g,.}$ and points to the last element in the limit case where they are all equal. Finally, define the quantity $\|\check{z}_{g,1:M_g}\|_1$ as the sum of the M_g coefficients of a group with the greatest modulus:

$$\sum_{m=1}^{M_g} |\check{z}_{g,m}|. \tag{4.13}$$

The proximity operator of E-LASSO is again a generalized thresholding operation. The corresponding shrinkage factor $\nu_{g,m}$ is given by:

$$\nu_{g,m} = \frac{\lambda}{1 + M_g \lambda} \frac{\sum_{m'=1}^{M_g} |\check{z}_{g,m'}|}{|\alpha_{g,m}|} = \frac{\lambda}{1 + M_g \lambda} \frac{\|\check{z}_{g,1:M_g}\|_1}{|\alpha_{g,m}|}. \tag{4.14}$$

This calls two remarks. First, the shrinkage is not inversely proportional to the the energy of the group anymore, but *inversely proportional* to the cumulative norm of it greatest coefficients instead. Second, the shrinkage is not fixed for a given group anymore, by it varies between coefficients of a same group.

The proximal operator of the $\ell_{1,2}$ mixed norm is finally given by:

$$\text{prox}_{\lambda \|.\|_{1,2}}(\alpha_{g,m}) = \alpha_{g,m} \left(1 - \frac{\lambda}{1 + M_g \lambda} \frac{\|\check{z}_{g,1:M_g}\|_1}{|\alpha_{g,m}|} \right)^+. \tag{4.15}$$

Eq (4.15) is easier to interpret if it is written in the classical soft-thresholding form:

$$\text{prox}_{\lambda \|.\|_{1,2}}(\alpha_{g,m}) = \text{sign}(\alpha_{g,m}) \left(|\alpha_{g,m}| - \frac{\lambda}{1 + M_g \lambda} \|\check{z}_{g,1:M_g}\|_1 \right)^+.$$

If a coefficient $\alpha_{g,m}$ has an amplitude which is only a fraction of the cumulative norm of the most meaningful coefficients in its group, it is discarded.

Hence, E-LASSO selects the members that are the most influential inside a group and sets the others to 0, which explains why it is called the Elitist lasso. Again, we emphasize that this behaviour is very different from G-LASSO that selects the best groups and discards the others: E-LASSO selects the meaningful coefficients of each group instead.

1. Recall that the ℓ_2 norm (applied by E-LASSO on the groups as a whole) is related to a least squares problem.

4.2.4 Social sparsity: three-level mixed norms and one last elitist lasso

In the previous section, we considered groups of coefficients. We now add the last tier by considering that we can have clusters of groups. Hence, a coefficient is now indexed by a triplet (c, g, m) where c stands for cluster, and again g for group and m for member.

The norm of a candidate solution is computed with respect to this structure by using a three-level mixed norm, defined as:

Definition 3 (Three-level mixed norm)**.** *Let $\boldsymbol{x} \in \mathbb{R}^N = \mathbb{R}^{C\times G\times M}$ be a point indexed by the triplet $(c, g, m) \in \mathbb{N}^3$. The $\ell_{p,q,r}$ mixed-norm of $\boldsymbol{x} \in \mathbb{R}^N$ is defined by:*

$$\|\boldsymbol{x}\|_{p,q,r} = \left(\sum_{c=1}^{C}\left(\sum_{g=1}^{G}\left(\sum_{m=1}^{M}|x_{c,g,m}|^p\right)^{q/p}\right)^{r/q}\right)^{1/r}. \tag{4.16}$$

PE-LASSO. Among the possible choices for the $\ell_{p,q,r}$ mixed norm we will consider only the triplet $(p, q, r) = (2, 1, 2)$. Intuitively, at the patch level this norm behaves as an elitist lasso ($\ell_{1,2}$ norm) and corresponds exactly to E-LASSO when there is no cluster structure. Hence, it favors sparse patches, which is a property that we are seeking to enforce. Then, it adds a layer of structured sparsity by computing the ℓ_2 norm of the cluster. This leads to an additional phenomenon of selecting the columns in the patch stack that have an important contribution to the norm of the cluster, while discarding the others. This last selection process extracts the common structure of the groups in a stack despite the presence of noise.

The sparse estimation problem reads:

$$\begin{aligned}\hat{\boldsymbol{x}} &= \boldsymbol{\Phi}\arg\min_{\alpha\in\mathbb{R}^N}\tfrac{1}{2}\|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2+\lambda\|\alpha_{c,g,m}\|_{2,1,2}^2\\ &= \boldsymbol{\Phi}\arg\min_{\alpha\in\mathbb{R}^N}\tfrac{1}{2}\|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2+\lambda\sum_{c=1}^{C}\left(\sum_{g=1}^{G}\left(\sum_{m=1}^{M}|\alpha_{c,g,m}|^2\right)^{1/2}\right)^{2/1}.\end{aligned}$$

Hence, in the absence of clusters ($C = 1$) it corresponds exactly to the elitist lasso problem.

The construction of the proximity operator for the $\ell_{2,1,2}$ mixed norm is parallel to the case of the $\ell_{1,2}$-norm, with only one noticeable difference: the intermediate groups $z_{c,g,m}$ are now obtained on a per-cluster basis as:

$$z_{c,g,m} = \left(\sum_{g=1}^{G}|\alpha_{g,m}|^2\right)^{1/2} = \|\boldsymbol{\alpha}_{c,.,m}\|_1.$$

Then, $\check{\boldsymbol{z}}_{c,.,.}$ is again obtained by sorting the members $\{\check{z}_{c,.,m}\}_{m=1}^{M}$ in descending order and an index M_c is computed according to Eq (4.12). The proximity operator is then similarly given by [119]:

$$\operatorname{prox}_{\lambda\|.\|_{2,1,2}}(\alpha_{c,g,m}) = \alpha_{c,g,m}\left(1-\frac{\lambda}{1+M_c\lambda}\frac{\|\check{\boldsymbol{z}}_{c,.,1:M_g}\|_1}{|\alpha_{c,g,m}|}\right)^+. \tag{4.17}$$

Hence, it is the same as E-LASSO with one exception: instead of considering the distribution of the amplitudes of the coefficients of a group, it is the contribution of one coefficient to the cluster energy that matters.

The horizon of the $\ell_{2,1,2}$ norm when assessing the importance of a coefficient extends to the whole cluster instead of one group. This property enforces some similarity between the groups of a given cluster: if one coefficient $\alpha_{c,g,m}$ has an important amplitude but that is not confirmed by the other groups of the cluster, then it will be shrunk strongly. This norm does not select the most meaningful coefficients, but the coefficients that are persistently the most meaningful inside the cluster. Hence, it is called Persistent Elitist lasso (PE-LASSO) in [119].

Remark 9. *If some overlap between the clusters is allowed, then they correspond to the neighborhood systems of [119]. In this case the notations are slightly more complex since the summation over the index c has to be replaced by a summation over some kind of neighborhood relationship such as* $\sum_{g' \in \mathcal{N}(g)}$ *where* $\mathcal{N}(g)$ *defines the set of neighbors of the group g. However, in the sequel we will only handle non-overlapping clusters: a group will belong to one and only one cluster, which allows us this simplification in the equations.*

4.2.5 Conclusion: choosing PE-LASSO

In what follows, we will consider stacks of patches as in BM3D. After a suitable transform (DCT...) each patch corresponds to a set of coefficients, hence a group will naturally correspond to each patch. Furthermore, the existence of a stack of similar, nearly-identical patches pleads for the exploitation of the additional cluster structure. Hence, the use of the three level mixed norm $\ell_{2,1,2}$ appears as the natural constraint to promote the output of sparse patches and exploiting at the same time their redundancy.

Recall that qualitatively the behavior of the corresponding PE-LASSO estimator is to pick the coefficients that have a meaningful contribution throughout the whole cluster and strongly shrink the others. Hence, this estimator will exploit the redundancies inside a stack of patches in order to repair individual coefficients that would appear as outliers because they are too different from one patch to the remaining others, be it too strong or too weak. This seems to be the correct way to promote sparse representations while at the same time exploiting the stack structure. A G-LASSO variant instead would shut down entire patches with respect to the norm of the stack. Such a situation is however very unlikely to happen because of the prior preselection of the patches.

Consequently, we propose to replace the 3D transform of BM3D by a PE-LASSO thresholding.

4.3 A novel sparse non-local denoising algorithm

4.3.1 The DANSE algorithm

From the studies of fast non-local patch retrieval in chapter 3 and of sparse non-locality at the beginning of the current chapter, we have obtained the recipe for a fast and efficient non-local denoising algorithm:

1. a fast preselection process is needed to sort the patches with respect to their appearance and to build group of identical patches (see Fig 4.4);

2. a sparsity-promoting optimization process should be applied to each group of patches in order to obtain sparse patches that also exhibit a shared sparse structure;
3. an image estimate is then formed by back-projecting the resulting patches.

These steps are essentially the same as in BM3D. Furthermore, this process can be applied twice: the first time to obtain a rough, strongly denoised intermediate product, and the second time by using the intermediate result as descriptor for the patches, thus producing slightly different groups and hopefully a final result with less artifacts.

From chapter 3, it seems logical to retain the spectral hashing as clustering method, and from what precedes the $\ell_{2,1,2}$ mixed norm is a good candidate as a sparsity promoting constraint. Consequently, we propose the following Denoising Algorithm via Non-local Spectral hashing and Elitist lasso (DANSE):

1. learn a patch hash function for a given patch size and a number of bits;
2. extract the noisy patches from the input image and sort them in a table using the previously computed hash function;
3. apply a domain transform $\mathcal{T}$ (typically a DCT or a wavelet decomposition) on the patches;
4. compute the result of the PE-LASSO estimator by considering clusters of patches with a Hamming diameter of 0;
5. invert the domain transform by applying its inverse $\mathcal{T}^{-1}$;
6. back-project the resulting patches.

The steps 2–6 can be repeated by using the intermediate result as patch coordinates for the projection in the hash table.

Note that many variants can be derived from this core procedure, for example:
- the patch hash function can be computed beforehand using a database of clean images representing the typical context of the application, or from a generic image database;
- the learning patches (for the hash function) and the noisy ones need not be the same. This can eventually lead to iterative schemes where the noisy inputs are used to create an intermediate estimate. Then, either a refined hash function is estimated from this intermediate result, and the final image is obtained by applying this novel function to the inout data, or the same initial hash function is used on the intermediate patch values (as in BM3D);
- video denoising schemes can be produced by accumulating the patches from the frame of a sliding time widow in the hash table. When a frame exits the current window, its patches are simply removed from the hash table.

This last example shows the utility of using a hash table: it is quite easy to insert or remove patches inside the hash table, while other structures would lead to speed or memory hiccups.

Interpretation. Unlike BM3D, the proposed DANSE algorithm can be cast as a convex optimization problem. Writing $\mathbb{R}$ the patch extraction operator, the DANSE algorithm can be stated as a minimization problem on a set of unknown patches $\boldsymbol{\theta}$, whose solution is then back-projected to the spatial domain in order to form the final denoised image. In this case, the operator R can be expressed as a matrix that transforms an input image. It proceeds at the same time with the patch extraction and windowing operations, and its inverse is equal to R^T if the windows obey the Princen-Bradley condition (see below). We call this minimization problem *danse-core* by analogy

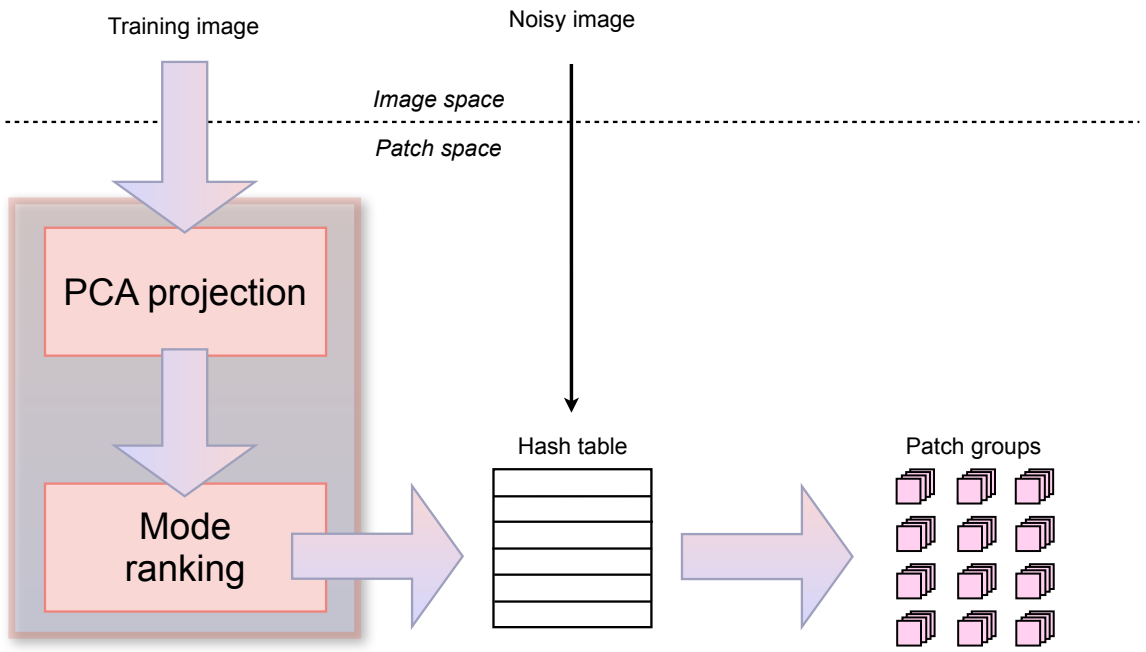


Figure 4.4: *Overview of the group creation process based on spectral hashing (chapter 3). As in BM3D, the training image can be equal to the noisy one, be an intermediate denoising product or even a completely unrelated image (like in sparse coding based algorithms).*

the the previous *bm3d-core*:

$$\hat{\boldsymbol{x}} = R^T \underset{\boldsymbol{\theta}\in\mathbb{R}^{N\times n}, \boldsymbol{\theta}=R\boldsymbol{x}}{\arg\min} \|\boldsymbol{\theta} - R\boldsymbol{y}\|_2^2 + \lambda \left(\sum_{c=1}^{C} \sum_{g=1}^{G} \left(\sum_{m=1}^{M} |\mathcal{T}(\boldsymbol{\theta})_{c,g,m}|^2 \right)^{1/2} \right)^{2/2}, \tag{4.18}$$

or in a more compact form

$$\hat{\boldsymbol{x}} = R^T \underset{\boldsymbol{\theta}\in\mathbb{R}^{N\times n}, \boldsymbol{\theta}=R\boldsymbol{x}}{\arg\min} \left(\|\boldsymbol{\theta} - R\boldsymbol{y}\|_2^2 + \lambda \|\mathcal{T}(\boldsymbol{\theta})\|_{2,1,2}^2 \right). \tag{4.19}$$

If we suppose that the images have an infinite spatial support, we can choose an *apodization* window based patch extractor. Instead of extracting one patch per pixel, the patches are separated by the half size of the windows along each dimension. Each patch is then multiplied by a window respecting the Princen-Bradley condition, which writes (for rectangular windows w of size $H \times W$):

$$w_{i,j}^2 + w_{i+H,j}^2 + w_{i,j+W}^2 + w_{i+H,j+W}^2 = 1.$$

Hence, the patches can be multiplied by the same weights during the extraction and the back-projection steps.

4.3.2 Experiments

Implementation. The proposed algorithm was implemented in C++ with the OpenCV library. The hash functions involved in the spectral hashing step were computed beforehand in a separate

step in order to save some computation time. Note that they could have been computed online (from the noisy data), as we have seen in chapter 3 that it did not vary much under noise. Finally, note that although the Peak SNR measures did show some minor improvements, we preferred to report here mean structured similarity scores (SSIM, see [116]): the SNR is slightly biased towards blurred estimates and does not report actual image quality improvements when applied to textured images. On the other hand, SSIM measures the consistency of the local image content (textures, edges...) between the reference image and the candidate one. In order to report a single number, we averaged the SSIM measurements over the whole images, and it ranges from 0 (poor quality) to 1 (exactly the same content).

All the experiments were conducted by adding a Gaussian white noise of standard deviation 0.2 [1] to the input images. In our tests, the DCT led to the best results. Hence, all the results presented below were obtained by choosing the DCT as the 2D domain transform. Depending on the image size (that is directly related to the total number of patches), the computational time varied from immediate to a few seconds on a 2010 laptop. Most of the computation time is actually spent in the forward and reverse DCT routines.

Non-iterative image denoising experiments. Our first experiment consisted of denoising images with a spectral hasher trained on the same input images, which we call *self-denoising*. Fig. 4.5 shows the SSIM curve for various patch sizes and codeword lengths in the case of the classical Lena image expressed as a function of the hyperparameter λ. The curves are very flat, which demonstrates that the proposed algorithm is robust to the exact setting of λ, provided it stays in the numerical range $[0.1, 0.8]$. Furthermore, for a fixed patch size, the performance is not improved by using codewords of 12 bits instead of 8. Hence, this conforms the experiments from the previous chapter: 8 bits are sufficient to correctly cluster patches even under strong noise. Note that we also confirm that a good patch size is around 8×8 pixels, which is close to the optimal size found for NL-means (patches of 7×7 pixels according to [3]). Fig. 4.6 illustrates the denoising of an image given a hasher learned from its noisy patches. You can note that the visual impression correlates well with the MSSIM metric: the texture of the image is well preserved by our algorithm.

Another important lesson from the previous chapter was also that the hash function computed by the patch spectral hashing procedure was very generic. This is confirmed by the results from Fig. 4.7. In this experiment, we have fixed the patch size (8×8 pixels) and the number of bits (8), and we have denoised the classical Cameraman image with hash functions trained from different images: Lena, Barbara, and the Seat picture from Fig. 3.4 (chapter 3). The three SSIM curves are almost perfectly superimposed: hence, the clustering functions obtained from various images are almost identical. Furthermore, the Seat image seems to provide the best spectral hashing function (observe the red curve in Fig. 4.7), although the Cameraman and the Seat images are clearly not related to each other (Fig. 4.8).

Finally, to confirm the influence (or actually the lack of influence) of the length of the codewords after 8 bits, we have fixed the hasher training image and varied the codeword size. Some typical results are displayed in Fig. 4.9 where the hash function was trained with the Lena picture and applied to denoise Barbara. The curves corresponding to the SSIM metric for 8 and 12 bits are almost perfectly superimposed, and there is no great interest in moving to 12 bits. As a side note, one can note that the peak MSSIM score in this case is fairly low (around 0.6) compared to the

1. The dynamic range of the input images was set to $[0, 1]$.

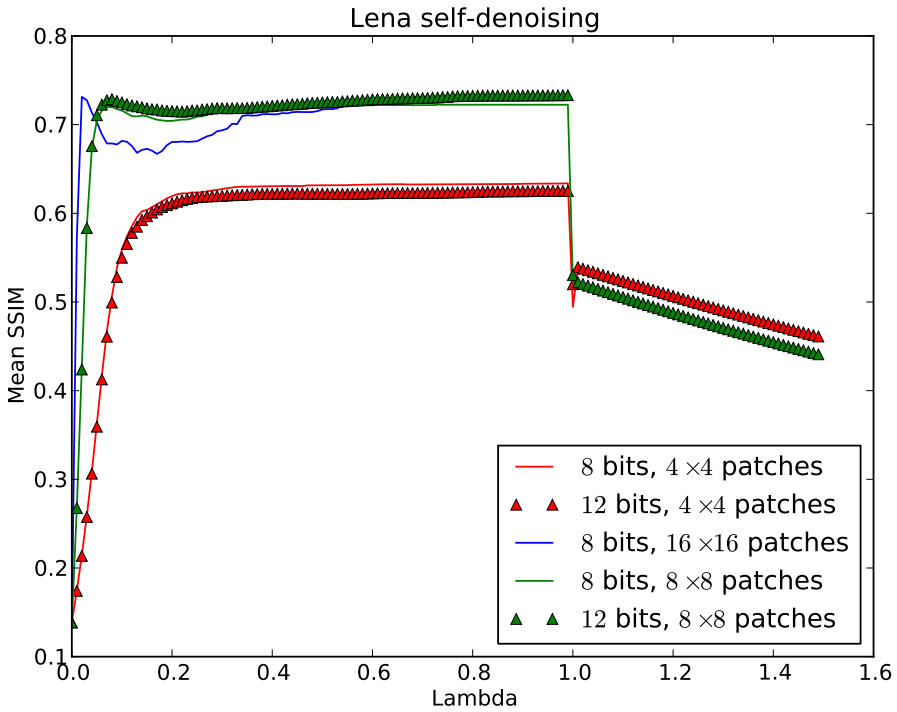
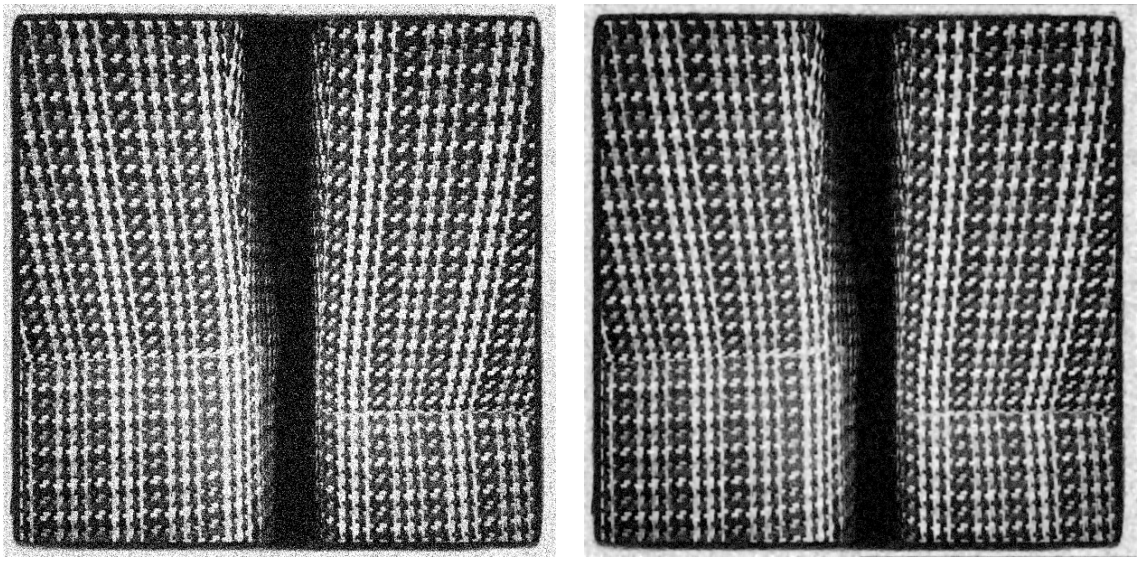


Figure 4.5: *Mean SSIM scores obtained after the denoising of Lena in the case of Gaussian white noise of standard deviation* 0.2*.*



(a) Noisy input (b) DANSE output

Figure 4.6: *Visual illustration of the proposed DANSE algorithm qualities. Note that the structure of the original image was well preserved by our method.*

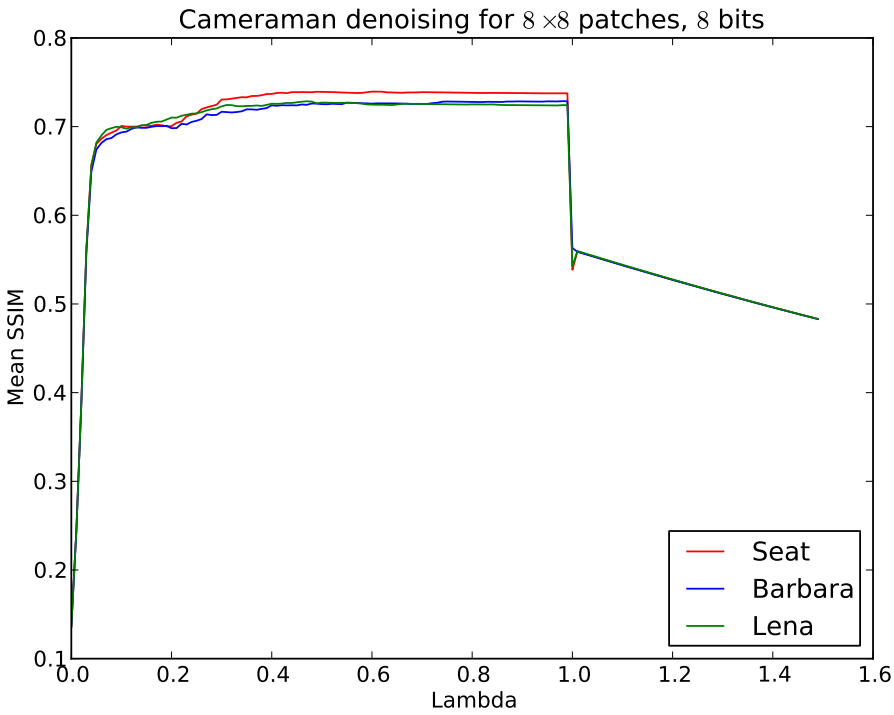


Figure 4.7: *SSIM curves for a fixed patch size and codeword length, but using spectral hash functions trained on different images. Note that the three curves are almost merged into a single one, although the three training images are very different. This confirms that local image content tends to be highly generic.*

other experiments. Our interpretation of this fact is that the Barbara image contains a lot of specific texture (crossing stripes). On the other hand, Lena contains few textures (except for the feather of the hat). Hence, Lena is probably not a good training image to obtain a generic patch spectral hash function: it does not contain enough diversity of local image content.

Image denoising: 2 iterations. Like BM3D, our DANSE algorithm can be applied in a two-tier fashion, by computing an intermediate denoising result that is used to refine the patch grouping phase in a second iteration. In all our experiments, this brought only a minor enhancement in the demising quality metric: the MSSIM gain was always around 0.05. Hence, we did not find it useful to report more results using this iterative scheme.

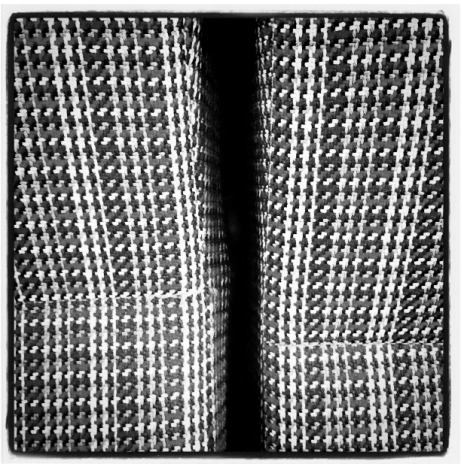
4.4 Conclusion

In this chapter, we have developed an interpretation of non-local processing as a block-oriented sparsity-promoting process. Following the successful approach of BM3D and subsequent development such as simultaneous sparse coding (LSSC), we have considered stacks of nearly-identical patches. Interpreting these stacks as clusters and the patches as groups of coefficients, we could derive a mathematically sound and simple formulation of this problem using a three level mixed

(a) Cameraman

(b) Lena

(c) Barbara



(d) Seat

Figure 4.8: *The images used for the experiments in Fig. 4.7.*

norm that takes into account the structural similarities inside the noisy data. Furthermore, this novel formulation comes with a proximal solver that consists in practice in point-wise thresholding operations. Furthermore, a solution is computed easily and without additional iterative procedures, and the proposed DANSE algorithm remains quite intuitive.

In order to create these stacks of visually similar patches, we have taken advantage of the spectral hashing algorithm presented in the the previous chapter. The clusters are formed by the result of nearest neighbor queries in this sectorial hash table with a Hamming radius of 0, *i.e.*, one bucket in the table is equivalent to one cluster, and there is no need to further optimize a dictionary during the denoising process. Consequently, the number of bits used to generate the locational codes is determined by the amount of noise and the size of the patches, but our experiments showed that the fine tuning of this parameter was not necessary. Finally, thanks to the simplicity of handling the patches in the table, we are able to process an important amount of data in a fully and truly non-local way, that also avoids the iterative procedures that were proposed before in the non-local

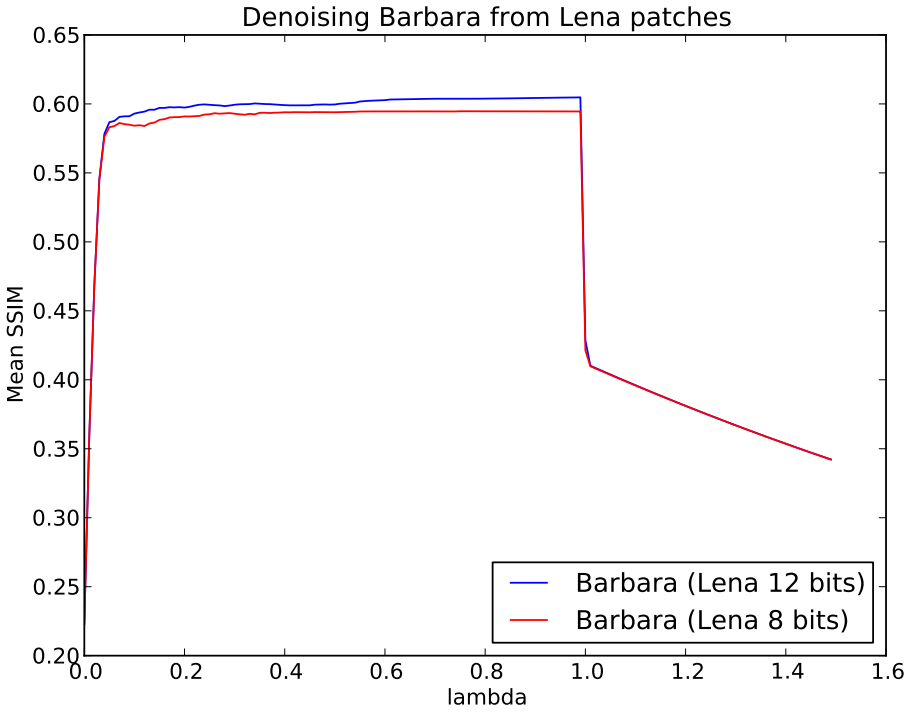


Figure 4.9: *SSIM metrics measured after denoising Barbara with a hash function trained from Lena for different lengths of codewords.*

sparse coding literature.

From Bits to Images: Inversion of Local Binary Descriptors

5

In this chapter, we will consider a radically different problem than in the previous chapters, but we will still continue to work with image patches. The atypical question that we will consider is the following: given some feature vectors used in Pattern Recognition to describe patches, can we invert them and recreate the original image content, or at least a plausible version?

We will show that, in the case of the very recent Local Binary Descriptors such as BRIEF [129] and FREAK [130] that are getting increasingly popular because of their efficiency in mobile computing, the answer is yes. Inverting feature descriptors is a very uncommon task: their merits are usually assessed in practical setups. However, this field is progressively emerging both under the pressure of user privacy protection [131] and because of the need for a deeper understanding of the internal behavior of these descriptors in order to properly analyze failure cases (and especially false positives) [132].

We will present several variational approaches that allow to invert this family of descriptors, whether they are binarized or not. These algorithms were disclosed for the first time in a conference presentation [133] (for the preliminary results on non-quantized descriptors) and in a preprint available online ([134], submitted). The description of the inversion algorithms is augmented with some arguments on how to deal with the ambiguities that arise during the reconstruction process. Apart from a unified view of this work so far, we also derive here a theorem inspired by [135] that represents a first step towards a full demonstration of why these descriptors are actually successful.

5.1 Introduction

Local Binary Descriptors are becoming more and more popular for image matching tasks, especially when going mobile. While they are extensively studied in this context, their ability to carry enough information in order to infer the original image is seldom addressed. In this work,

we leverage an inverse problem approach to show that it is possible to directly reconstruct the image content from Local Binary Descriptors. This process relies on very broad assumptions besides the knowledge of the pattern of the descriptor at hand. This generalizes previous results that required either a prior learning database or non-binarized features. Furthermore, our reconstruction scheme reveals differences in the way different Local Binary Descriptors capture and encode image information. Hence, the potential applications of our work are multiple, ranging from privacy issues caused by eavesdropping image keypoints streamed by mobile devices to the design of better descriptors through the visualization and the analysis of their geometric content.

How much, and what type of information is encoded in a keypoint descriptor ? Surprisingly, the answer to this question has seldom been addressed directly. Instead, the performance of keypoint descriptors is studied extensively through several image-based benchmarks following the seminal work of Mikolajczyk and Schmid [136] using Computer Vision and Pattern Recognition task-oriented metrics. These stress tests aim at measuring the stability of a given descriptor under geometric and radiometric changes, which is a key to success in matching templates and real world observations. While precision/recall scores are of primary interest when building object recognition systems, they do not tell much about the intrinsic quality and quantity of information that are embedded in the descriptor. Indeed, these benchmarks are informative about the context in which a descriptor performs well or poorly, but not why. As a consequence, descriptors were mostly developed empirically by benchmarking new ideas against some image matching datasets.

Furthermore, there is a growing trend towards the use of image recognition technologies from mobile handheld devices such as the smartphones combining high quality imaging parts and a powerful computing platform. Application examples include image search and landmark recognition [137] or augmented media and advertisement [138]. To reduce the amount of data exchanged between the mobile and the online knowledge database, it is tempting to use the terminal to extract image features and send only these features over the network. This data is obviously sensitive since it encodes what the user is viewing. Hence it is legitimate to wonder if its interception could lead to a privacy breach.

5.1.1 Related work

Recently, an inspirational paper [131] showed that ubiquitous interest points such as SIFT [139] and SURF [140] suffice to reconstruct plausible source images. This method is based on an image patch database indexed by their SIFT descriptors and then proceeds by successive queries, replacing each input descriptor by the corresponding patch retrieved in the learning set. Although it produces good image reconstruction results, it actually tells us little about the information embedded in the descriptor: retrieving an image patch from a query descriptor leverages the matching capabilities of SIFT which are now well established by numerous benchmarks and were actually key for its wide adoption.

During the writing of this thesis, another paper on this topic was made publicly available [132]. The goal of these authors is to invert the Histogram of Oriented Gradients (HOG) descriptors [141] that are now ubiquitous for the Object Detection task (see for example the submissions to the PASCAL Visual Object Classes challenge [1]). Briefly, the HOG descriptors can be seen as an extension of the SIFT descriptor, obtained by concatenating the histograms of gradient directions from sev-

1. http://pascallin.ecs.soton.ac.uk/challenges/VOC/

eral subwindows inside the tentative object bounding box, thus removing the need for a keypoint detector (but still at the cost of training the subsequent classifier to accurately reject the negative examples). While they propose and compare 4 different inversion algorithms (including Linear Discriminant Analysis [142] and sparse coding on patch dictionaries), each of them depends at some point on a learning database via the prior training of a function matching HOG descriptors and image patches.

In this work, we instead propose two algorithms that aim at reconstructing image patches from local descriptors *without* any external information and with very little additional constraints. We consider descriptors made of local image intensity differences, which are increasingly popular in the Computer Vision community, for they are not very demanding in computational power and hence well suited for embedded applications. The first algorithm that we describe works on *real-valued* difference descriptors, and addresses the reconstruction process as a regularized deconvolution problem. The second algorithm leverages some recent results from 1-bit Compressive Sensing (CS) [143] to reconstruct image parts from *binarized* difference descriptors, and hence is of great practical interest because these descriptors are usually available as bitstrings rather than as real-valued vectors. Hence, we show that an inverse problem approach suffices to invert a local image patch descriptor provided that the descriptor is a local difference operator, thus avoiding the need to build an external database beforehand.

Notations

In this chapter, we make extensive use of the following notations. Matrices and vectors are denoted by bold letters or symbols (e.g., $\boldsymbol{\Phi}$, $\boldsymbol{x}$) while light letters are associated to scalar values (e.g., scalar functions, vector components or dimensions). The scalar product between two N-length vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is written $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{N} x_i y_i$, while their Hadamard product $\boldsymbol{x} \odot \boldsymbol{y}$ is such that $(\boldsymbol{x} \odot \boldsymbol{y})_i = x_i y_i$ for $1 \leq i \leq N$. Since we work only with real matrices, the adjoint of a matrix $\boldsymbol{A}$ is $\boldsymbol{A}^* = \boldsymbol{A}^T$. The vector of ones is written $\mathbf{1} = (1, \cdots, 1)^T$ and the identity matrix is denoted Id.

Most of the time, we will "vectorize" 2-D images, *i.e.*, an image or a patch image $\boldsymbol{x}$ of dimension $N_1 \times N_2$ is represented as a N-dimensional vector $\boldsymbol{x} \in \mathbb{R}^N$ with $N = N_1 N_2$. This allows us to represent any linear operation on $\boldsymbol{x}$ as a simple matrix-vector multiplication. One important linear operator is the 2-D wavelet analysis operator $\boldsymbol{W}$ with $\boldsymbol{W}^T$ the corresponding synthesis operator. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$, $\boldsymbol{W}\boldsymbol{x}$ is then a vector of wavelet coefficients and $\boldsymbol{W}^T\boldsymbol{y}$ a patch with the same size as $\boldsymbol{x}$.

We denote by $\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$ with $p \geq 1$ the ℓ_p-norm of $\boldsymbol{x} \in \mathbb{R}^N$, reserving the notation $\|\cdot\|$ for $p = 2$ and with $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$. The ℓ_0 "norm" of $\boldsymbol{x}$ is $\|\boldsymbol{x}\|_0 = \#\{i : x_i \neq 0\}$. Correspondingly, for $1 \leq p \leq +\infty$, a ℓ_p-ball of radius λ is the set $B_p(\lambda) = \{\boldsymbol{x} \in \mathbb{R}^N : \|\boldsymbol{x}\|_p \leq \lambda\}$.

We use also the following functions. We denote by $(\boldsymbol{x})_+$ the non-negativity thresholding function, which is defined componentwise as $(\lambda)_+ = (\lambda + |\lambda|)/2$, and $(\boldsymbol{x})_- = -(-\boldsymbol{x})_+$. The sign function $\operatorname{sign} \lambda$ is equal to 1 if $\lambda > 0$ and -1 otherwise.

In the context of convex optimization, we denote by $\Gamma^0(\mathbb{R}^N)$ the class of proper, convex and lower-semicontinuous functions of the finite dimensional vector space $\mathbb{R}^N$ to $(-\infty, +\infty]$ [144]. The indicator function $\iota_{\mathcal{S}} \in \Gamma^0(\mathbb{R}^N)$ of a set $\mathcal{S}$ maps $\iota_{\mathcal{S}}(\boldsymbol{x})$ to 0 if $\boldsymbol{x} \in \mathcal{S}$ and to $+\infty$ otherwise. For any $F \in \Gamma^0(\mathbb{R}^N)$ and $\boldsymbol{z} \in \mathbb{R}^N$, the *Fenchel-Legendre* conjugate function F^* is

$$F^*(\boldsymbol{z}) = \max_{\boldsymbol{x} \in \mathbb{R}^N} \langle \boldsymbol{z}, \boldsymbol{x} \rangle - F(\boldsymbol{x}), \tag{5.1}$$

while, for any $\lambda > 0$, its *proximal operator* reads

$$\operatorname{prox}_{\lambda F} z = \underset{x \in \mathbb{R}^N}{\arg\min}\ \lambda F(x) + \tfrac{1}{2}\|x - z\|^2. \tag{5.2}$$

For $F = \iota_{\mathcal{S}}$ for some convex set $\mathcal{S} \subset \mathbb{R}^N$, the proximal operator of $\operatorname{prox}_{\lambda F}$ simply reduces to the orthogonal projection operator on $\mathcal{S}$ denoted by $\operatorname{proj}_{\mathcal{S}}$.

5.2 Local Binary Descriptors

We are interested in reconstructing image patches from binary descriptors obtained by quantization of local image differences, such as BRIEF [145] or FREAK [130]. Hence, we will refer to these descriptors as Local Binary Descriptors (LBDs) in the text. In a standard Computer Vision and Pattern Recognition application, such as object recognition or image retrieval, an interest point detector such as Harris corners [146], SIFT [139] or FAST [147] is first applied on the images to locate interest points. The regions surrounding these keypoints are then described by a feature vector, thus replacing the raw light intensity values by more meaningful information such as histograms of gradient orientation or Haar-like analysis coefficients. In the case of LBDs, the feature vectors are made of local binarized differences computed according to the generic process described below.

5.2.1 Generic Local Binary Descriptor model

A LBD of length M describing a given image patch of $\sqrt{N} \times \sqrt{N} = N$ pixels can be computed by iterating M times the following three-step process:

1. compute the Gaussian average of the patch at two locations x_i and x'_i with variance σ_i and σ'_i respectively;
2. form the difference between these two measurements;
3. binarize the result by retaining only its sign.

Reshaping the input patch as a column vector $\mathbf{p} \in \mathbb{R}^N$, the first two steps in the above procedure can be merged into the application of a single linear operator $\mathcal{L}$:

$$\begin{aligned} \mathcal{L} : \mathbb{R}^N &\to \mathbb{R}^M \\ p &\mapsto \left(\langle \mathcal{G}_{q_i,\sigma_i}, p\rangle - \langle \mathcal{G}_{q'_i,\sigma'_i}, p\rangle\right)_{1 \leqslant i \leqslant M}, \end{aligned} \tag{5.3}$$

where $\mathcal{G}_{q,\sigma} \in \mathbb{R}^N$ denotes a (vectorized) two-dimensional Gaussian of width σ centered in $q \in \mathbb{R}^2$ (Fig. 5.1, top) with $\|\mathcal{G}_{q,\sigma}\|_1 = 1$. As illustrated in Fig. 5.1-bottom, since $\mathcal{L}$ is a linear operator, it can be represented by a matrix $\mathcal{L} \in \mathbb{R}^{M \times N}$ multiplying p and whose each row $\mathcal{L}_i$ is given by

$$\mathcal{L}_i = \mathcal{G}_{q_i,\sigma_i} - \mathcal{G}_{q'_i,\sigma'_i}, \quad 1 \leq i \leq M. \tag{5.4}$$

We will take advantage of this decomposition interpretation to avoid explicitly writing $\mathcal{L}$ later on.

The final binary descriptor is obtained by the composition of this sensing matrix with a component-wise quantization operator $\mathcal{B}$ defined by $\mathcal{B}(x)_i = \operatorname{sign} x_i$, so that, given a patch p, the corresponding LBD reads:

$$\bar{p} := \mathcal{B}(\mathcal{L}p) \in \{-1, +1\}^M. \tag{5.5}$$

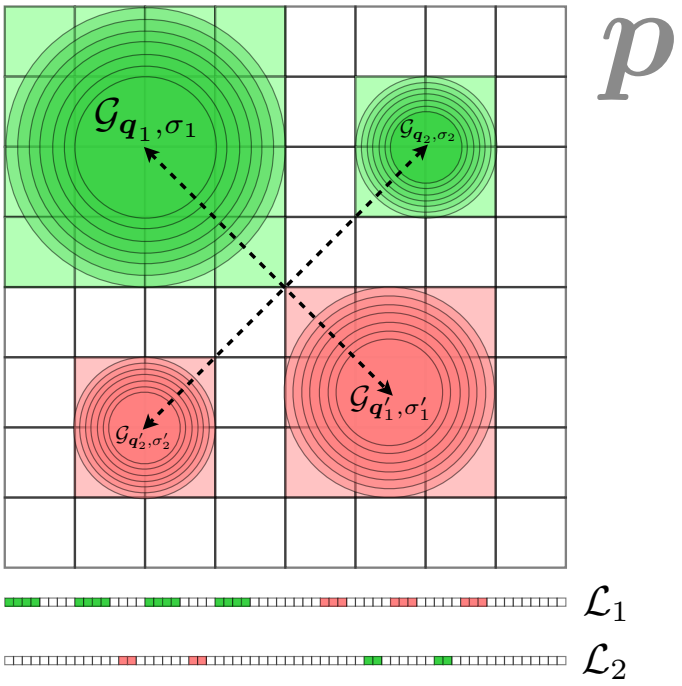


Figure 5.1: *Example of a local descriptor for an* 8×8 *pixels patch and the corresponding sensing matrix. Only two measurements of the descriptor are depicted; each one is produced by subtracting the Gaussian mean in the lower (red) area from the corresponding upper (green) one. All the integrals are normalized by their area to have values in* $[0, 1]$*. Below this, the corresponding vectors.*

Note that we have chosen this definition of $\mathcal{B}$ to be consistent with the notations of [135]. Implementations of LBDs will of course use the binary space $\{0, 1\}^M$ instead, since it fits naturally with the digital representation found in computers.

From the description of LBDs, it is clear that they involve only simple arithmetic operations. Furthermore, the distance between two LBDs is measured using the Hamming distance, which is a simple bitwise exclusive-or (XOR) instruction [130, 145]. Hence, computation and matching of LBDs can be implemented efficiently, sometimes even using hardware instructions (XOR), allowing their use on mobile platforms where computational power and electric consumption are strong limiting constraints. Since they also provide good matching performances, LBDs are getting more and more popular over SIFT and SURF: combined with FAST for the keypoint detection, they provide a fast and efficient feature extraction and matching pipe-line, producing compact descriptors that can be streamed over networks.

Typically, a 32-by-32 pixels image patch (1024 bytes in 8 bit grayscale format) can be reduced to a vector of only 256 measurements [145] coded with 256 *bits*. A typical floating-point descriptor such as SIFT or SURF would require instead 64 float values, *i.e.*, 256 *bytes* for the same patch, eight times the LBD size, and the distances would be measured with the ℓ_2-norm using slower floating-point instructions.

5.2.2 LBDs, LBPs, and other integral descriptors

Unlike [131], we use LBDs in this work instead of SIFT descriptors. As we will see in section 5.3, it is actually the knowledge of the spatial measurement pattern used by an LBD that allows us to properly define the matrix of the operator $\mathcal{L}$ in (5.3) as a convolution matrix. SIFT, HOG and SURF use histograms of gradient orientation instead, thus losing the precise localization information through an integration step. Hence, it seems very unlikely that our approach could be extended to these descriptors. On the other hand, it is possible to reproduce most of the algorithm described in [131] by replacing SIFT with a correctly chosen LBD to index the reference patch database, but this would bring only minor novelty.

Note also that we have coined the descriptors used here as LBDs, which are not the same as the Local Binary Patterns (LBPs) popularized by [148] for face detection. Although both LBDs and LBPs produce bit string descriptors, LBPs are obtained after binarization of image direction histograms. As such, LBPs are integral descriptors and suffer from the same lack of spatial awareness as SIFT and SURF.

5.2.3 The BRIEF and FREAK descriptors

Given two LBDs, the differences reside in the pattern used to select the size and the location of the measurement pairs $(\mathcal{G}_{\boldsymbol{q}_i,\sigma_i}, \mathcal{G}_{\boldsymbol{q}'_i,\sigma'_i})_{i=1}^{M}$. The authors of the pioneering BRIEF [145] chose small Gaussians of fixed width to bring some robustness against image noise, and tested different spatial layouts. Among these, two random patterns outperformed the others: the first one corresponds to a normal distribution of the measurement points centered in the image patch, and the second one to a uniform distribution.

Working on improving BRIEF, the authors of ORB [149] introduced a measurement selection process based on their matching performance and retained pairs with the highest selectivity. On the other hand, BRISK [150] introduced a concentric pattern to distribute the measurements inside the patch but retained only the innermost points for the descriptor, keeping the peripheral ones to estimate the orientation of the keypoint.

Eventually, the FREAK descriptor was proposed in [130] to leverage the advantages of both approaches: the learning procedure introduced with ORB and the concentric measurement layout of BRISK. The pattern was modified to resemble the retinal sampling pattern and can be seen in Figure 5.2. Note that it allows for a wider overlap between the measurements than the BRISK pattern: Fig. 5.3 shows some rows of the sensing matrix $\mathcal{L}$ in the original 2D geometry. All the rings were allowed to contribute in the training phase. Consequently, the FREAK descriptor implicitly captures the image details at a coarser scale when going away from the center of the patch.

5.3 Reconstruction as an inverse problem

In this work, our goal is to demonstrate that the knowledge of the particular measurement layout of an LBD is sufficient to infer the original image patch *without any external information*, using only an inverse problem approach. Typically, a 32×32 pixels patch (1024 values which are typically encoded on at least 1024 bytes of 8 bits) will be represented by a descriptor with 512 binary

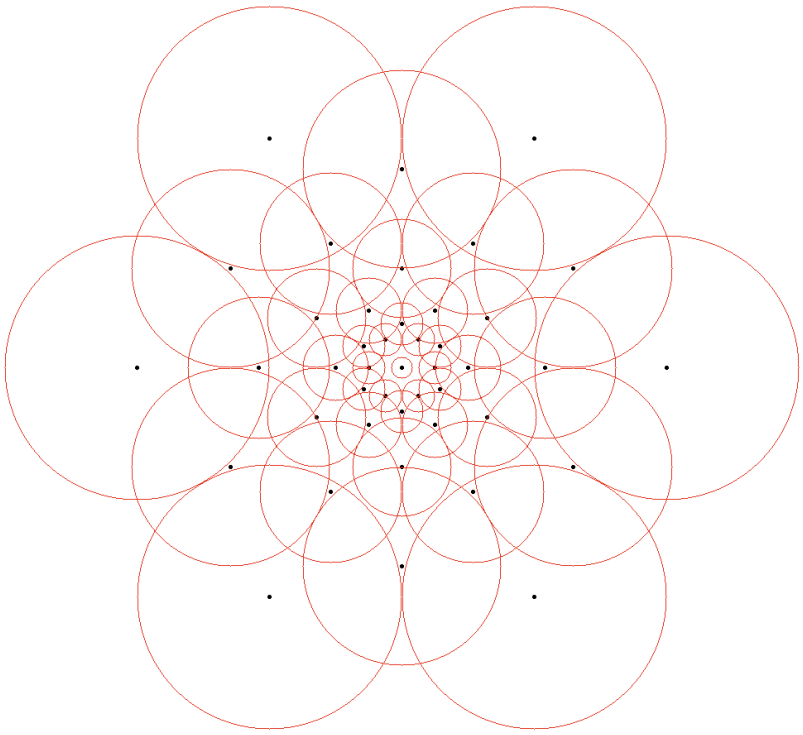
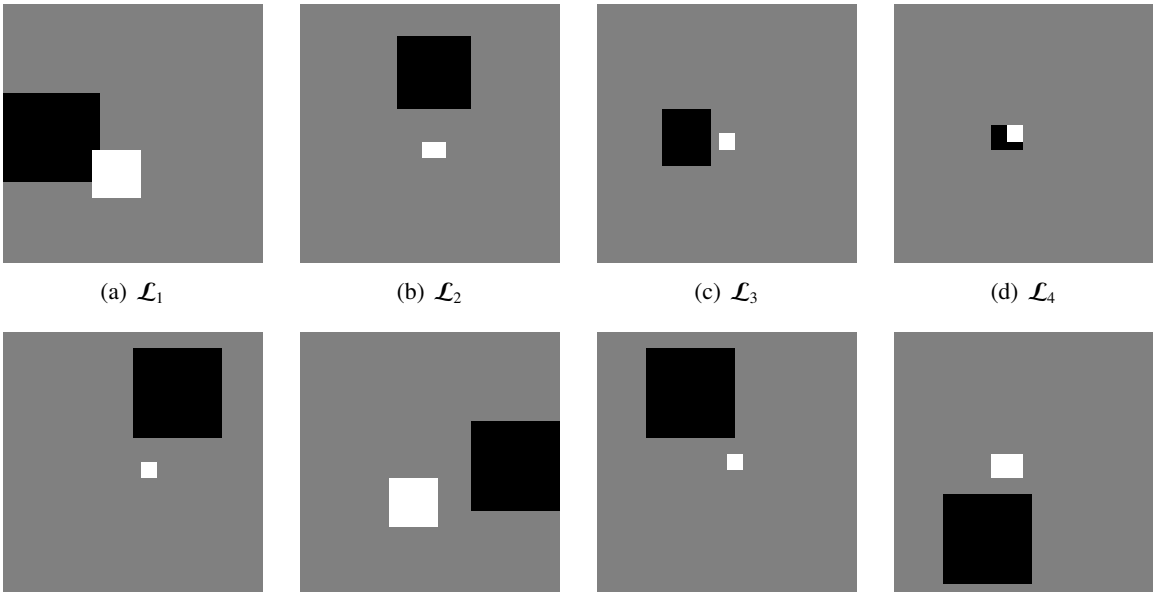


Figure 5.2: *The retinal pattern used by FREAK. The further a point is from the center, the wider the averaging area. Hence, FREAK captures the image variations at a coarser scale on the border of the patch than in the center.*



(a) $\mathcal{L}_1$ (b) $\mathcal{L}_2$ (c) $\mathcal{L}_3$ (d) $\mathcal{L}_4$

(e) $\mathcal{L}_{509}$ (f) $\mathcal{L}_{510}$ (g) $\mathcal{L}_{511}$ (h) $\mathcal{L}_{512}$

Figure 5.3: *The first and last frame vectors for a FREAK descriptor of 512 measurements (2-D representations). The white square depicts the positive lobe and the dark square the negative one. Each approximated Gaussian is normalized to be of unit ℓ_1-norm.*

components, *i.e.*, 512 bits. Hence, the reconstruction task is ill-posed: even without binarization of the features, there are half less measurements than unknowns.

Assuming (to simplify the implementation of the equations and avoid the use of fixed-point arithmetic) that a non-binarized feature vector is represented with floating-point values (4 bytes per component), the binarization will then divide by an additional factor of 32 the amount of available information! Classically, to make this problem tractable we introduce a regularization constraint that should be highly generic since we do not know *a priori* the type of image that we need to reconstruct. Thus, the sparsity of the reconstructed patch in some wavelet frame appeared as a natural choice: it only requires that a patch should have few nonzero coefficients when analyzed in this wavelet frame, which is quite general.

5.3.1 Real-valued descriptor reconstruction with convex optimization

Ignoring first the quantization operator by replacing $\mathcal{B}$ by the identity function, we choose the ℓ_1-norm to penalize the error in the data term and the ℓ_1-norm of the wavelet coefficients as a sparsity promoting regularizer. The ℓ_1-norm is more robust than the usual ℓ_2-norm to the actual value of the error and it is more connected with its sign. Hence, it is hopefully a better choice when dealing with binarized descriptors. The problem of reconstructing an image patch $\hat{\boldsymbol{p}} \in \mathbb{R}^N$ given an observed binary descriptor $\bar{\boldsymbol{p}} \in \mathbb{R}^M$ then reads:

$$\hat{\boldsymbol{p}} = \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min}\, \lambda\|\boldsymbol{\mathcal{L}x} - \bar{\boldsymbol{p}}\|_1 + \|\boldsymbol{W}\boldsymbol{x}\|_1 + \iota_{\mathcal{S}}(\boldsymbol{x}), \tag{5.6}$$

which is a sparse ℓ_1 deconvolution problem [151]. In Eq. (5.6), $\|\boldsymbol{\mathcal{L}x} - \bar{\boldsymbol{p}}\|_1$ is the *data term* that ties the solution to the observation $\bar{\boldsymbol{p}}$, $\|\boldsymbol{W}x\|_1$ is the regularizer that constrains the patch candidate $\boldsymbol{x}$ to have a sparse representation, and $\iota_{\mathcal{S}}(.)$ is the indicator function of the *validity domain* of $\boldsymbol{x}$ that we will make explicit later.

While the objective function in Eq. (5.6) is convex, it is not differentiable since the ℓ_1-norm has singular points on the axes of $\mathbb{R}^N$. Hence, we chose the *primal-dual* algorithm presented in [19] to solve this minimization problem. Instead of using derivatives of the objective which may not exist, it relies on *proximal calculus* and proceeds by alternate minimizations on the primal and dual unknowns.

The generic version of this algorithm aims at solving minimization problems of the form:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min}\, F(\boldsymbol{K}\boldsymbol{x}) + G(\boldsymbol{x}), \tag{5.7}$$

where $\boldsymbol{x} \in \mathbb{R}^N$ is the *primal* variable, $\boldsymbol{K} \in \mathbb{R}^{D\times N}$ is a linear operator, and $F \in \Gamma^0(\mathbb{R}^D)$ and $G \in \Gamma^0(\mathbb{R}^N)$ are convex (possibly non-smooth) functions. The algorithm proceeds by restating (5.7) as a *primal-dual* saddle-point problem on both the *primal* $\boldsymbol{x}$ and its *dual* variable $\boldsymbol{u}$:

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{u}} \langle \boldsymbol{K}\boldsymbol{x}, \boldsymbol{u}\rangle + G(\boldsymbol{x}) - F^*(\boldsymbol{u}). \tag{5.8}$$

For the problem at hand, we start by decoupling the data term and the sparsity constraint in Eq. (5.6) by introducing the auxiliary unknowns $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^N$ such that:

$$\hat{\boldsymbol{p}} = \underset{\boldsymbol{y},\boldsymbol{z}\in\mathbb{R}^N}{\arg\min}\, \lambda\|\boldsymbol{\mathcal{L}y} - \bar{\boldsymbol{p}}\|_1 + \|\boldsymbol{W}\boldsymbol{z}\|_1 + \iota_{\mathcal{S}}(\boldsymbol{y}) + \iota_{\{\boldsymbol{0}\}}(\boldsymbol{y} - \boldsymbol{z}). \tag{5.9}$$

The term $\iota_{\{\mathbf{0}\}}(\boldsymbol{y}-\boldsymbol{z})$ guarantees that the optimization occurs on the bisector plane $\boldsymbol{y}=\boldsymbol{z}$. Then, defining $\boldsymbol{K}=\left(\begin{smallmatrix}\boldsymbol{\mathcal{L}} & 0\\ 0 & \boldsymbol{W}\end{smallmatrix}\right)\in\mathbb{R}^{(M+N)\times 2N}$, we can perform our optimization (5.6) in the product space $\boldsymbol{x}=(\boldsymbol{y}^T,\boldsymbol{z}^T)^T\in\mathbb{R}^{2N}$ [152][153]. In this case, (5.6) can be written as (5.7) by setting $F(\boldsymbol{K}\boldsymbol{x})=F_1(\boldsymbol{\mathcal{L}}\boldsymbol{y})+F_2(\boldsymbol{W}\boldsymbol{z})$, where:

$$F_1(\cdot)=\lambda\|\cdot-\bar{\boldsymbol{p}}\|_1, \tag{5.10}$$

$$F_2(\cdot)=\|\cdot\|_1, \tag{5.11}$$

$$G\left(\begin{smallmatrix}\boldsymbol{y}\\ \boldsymbol{z}\end{smallmatrix}\right)=\iota_{\mathcal{S}}(\boldsymbol{y})+\iota_{\{\mathbf{0}\}}(\boldsymbol{y}-\boldsymbol{z}). \tag{5.12}$$

Introducing $\boldsymbol{r}\in\mathbb{R}^M$ and $\boldsymbol{s}\in\mathbb{R}^N$ the dual counterparts of $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively, and taking the Fenchel-Legendre transform of F yields the desired primal-dual formulation of (5.6):

$$\min_{\boldsymbol{y},\boldsymbol{z}}\max_{\boldsymbol{r},\boldsymbol{s}}\ \langle\boldsymbol{\mathcal{L}}\boldsymbol{y},\boldsymbol{r}\rangle+\langle\boldsymbol{W}\boldsymbol{z},\boldsymbol{s}\rangle+G\left(\begin{smallmatrix}\boldsymbol{y}\\ \boldsymbol{z}\end{smallmatrix}\right)-F_1^*(\boldsymbol{r})-F_2^*(\boldsymbol{s}). \tag{5.13}$$

An explicit formulation of $F_1^*(\boldsymbol{r})$ can be obtained by noting that, for $\varphi(\boldsymbol{r})=\varphi'(\boldsymbol{r}-\boldsymbol{u})$, $\varphi^*(\boldsymbol{r})=\varphi'^*(\boldsymbol{r})+\langle\boldsymbol{r},\boldsymbol{u}\rangle$, and that the conjugate of the ℓ_1-norm is $\iota_{B_\infty(1)}$ [144]. F_2^* is derived in [19], eventually yielding:

$$F_1^*(\boldsymbol{r})=\iota_{B_\infty(\lambda)}(\boldsymbol{r})+\langle\boldsymbol{r},\bar{\boldsymbol{p}}\rangle, \tag{5.14}$$

$$F_2^*(\boldsymbol{s})=\iota_{B_\infty(1)}(\boldsymbol{s}). \tag{5.15}$$

The algorithm presented in [19] requires explicit solutions for the proximal mappings of F_1^*, F_2^* and G. The first two are easily computed pointwise [19, 144] as:

$$(\mathrm{prox}_{\sigma F_1^*}\,\boldsymbol{r})_i=\mathrm{sign}(r_i-\sigma\bar{p}_i)\cdot\min(\lambda,|r_i-\sigma\bar{p}_i|), \tag{5.16}$$

$$(\mathrm{prox}_{\sigma F_2^*}\,\boldsymbol{s})_i=\mathrm{sign}(s_i)\cdot\min(1,|s_i|). \tag{5.17}$$

The function G is formed by the indicator of the set $\mathcal{S}$ and the indicator of the bisector plane $\{\left(\begin{smallmatrix}\boldsymbol{y}\\ \boldsymbol{z}\end{smallmatrix}\right)\in\mathbb{R}^{2N}:\boldsymbol{y}=\boldsymbol{z}\}$. An easy computation provides [153]:

$$\mathrm{prox}_{\sigma G}\left(\begin{smallmatrix}\boldsymbol{y}\\ \boldsymbol{z}\end{smallmatrix}\right)=\begin{pmatrix}\mathrm{proj}_{\mathcal{S}}\,\frac{1}{2}(\boldsymbol{y}+\boldsymbol{z})\\ \mathrm{proj}_{\mathcal{S}}\,\frac{1}{2}(\boldsymbol{y}+\boldsymbol{z})\end{pmatrix}. \tag{5.18}$$

Thus, its proximal mapping does not depend on any parameter.

Let us now precisely define our *validity domain* $\mathcal{S}$. It is chosen in order to remove ambiguities in the definition of the program (5.6) that could lead to a non-uniqueness of the solution. They are due to the differential nature of $\boldsymbol{\mathcal{L}}$, *i.e.*, the descriptor of any constant patch is zero. This involves both that $\bar{\boldsymbol{p}}$ does not include any information about the average of the initial patch $\boldsymbol{p}$, and the average of $\boldsymbol{x}$ in (5.6) cannot be determined by the optimization.

This problem is removed by defining $\mathcal{S}$ as the intersection of two convex sets $\mathcal{S}_1$ and $\mathcal{S}_2$. The first set $\mathcal{S}_1$ arbitrarily constrains the minimization domain to stay in the set of patches whose pixel dynamic lies in $[0,h_{\mathrm{pix}}]$, *i.e.*, $\mathcal{S}_1=\{\boldsymbol{x}\in\mathbb{R}^N:0\le x_i\le h_{\mathrm{pix}}\}$. In our experiments, we simply consider pixels with real values in $[0,1]$ and consequently fix $h_{\mathrm{pix}}=1$. The second domain $\mathcal{S}_2$ is associated to the space of patches whose pixel mean is equal to 0.5, *i.e.*, $\mathcal{S}_2=\{\boldsymbol{x}\in\mathbb{R}^N:\frac{1}{N}\sum_i x_i=0.5\}$.

This gives us a first set $\mathcal{S}_1$ whose proximal mapping $\mathrm{proj}_{\mathcal{S}_1}$ is a simple clipping of the values in $[0,1]$, while $\mathcal{S}_2$ is an hyperplane in $\mathbb{R}^N$ whose corresponding proximal mapping $\mathrm{proj}_{\mathcal{S}_2}$ is the

projection onto the simplex of $\mathbb{R}^N$ of vectors with mean 0.5. This projection can be solved efficiently using [154]. While the desired constraint set $\mathcal{S}$ is the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2$, we approximate the projection on $\mathcal{S}$ by $\text{proj}_{\mathcal{S}} \simeq \text{proj}_{\mathcal{S}_1} \circ \text{proj}_{\mathcal{S}_2}$. The correct treatment of $\text{proj}_{\mathcal{S}}$ would normally require to iteratively combine $\text{proj}_{\mathcal{S}_1}$ and $\text{proj}_{\mathcal{S}_2}$ (e.g., running Generalized Forward-Backward splitting [152] until convergence). In our experiments, this approximation did not lead to differences in the estimated patches.

Alg. 3 summarizes the different steps involved in the resolution scheme. It requires a bound Γ on the operator norm $\boldsymbol{K} \in \mathbb{R}^{(M+N)\times 2N}$, *i.e.*, on $\|\boldsymbol{K}\| = \max_{\boldsymbol{x}:\,\|\boldsymbol{x}\|=1} \|\boldsymbol{K}\boldsymbol{x}\|$. This is obtained by observing that $\|\boldsymbol{W}\|^2 = 1$ with a proper rescaling due to energy conservation constraints, leaving:

$$\|\boldsymbol{K}\|^2 = \left\|\begin{pmatrix} \boldsymbol{\mathcal{L}} & 0 \\ 0 & \boldsymbol{W} \end{pmatrix}\right\|^2 \leqslant \|\boldsymbol{\mathcal{L}}\|^2 + 1, \tag{5.19}$$

where $\|\boldsymbol{\mathcal{L}}\|$ can be efficiently estimated without any spectral decomposition of $\boldsymbol{\mathcal{L}}$ by using the power method [151].

While (5.13) may seem unnecessarily complicated at first because it involves both a minimization and a maximization subproblem, the resolution scheme is actually very efficient: it is a first-order method that involves mostly pointwise normalization and thresholding operations.

Algorithm 3: Primal-dual ℓ_1 sparse patch reconstruction.

Take $\Gamma \geqslant \|\boldsymbol{K}\|_2$, choose τ, σ, θ such that $\Gamma^2\sigma\tau \leqslant 1$, $\theta \in [0, 1]$ and n the number of iterations
Initialize: $\boldsymbol{x}^{(0)}, \tilde{\boldsymbol{x}}^{(0)} \leftarrow \boldsymbol{0}$, and $\boldsymbol{r}^{(0)}, \boldsymbol{s}^{(0)} \leftarrow \boldsymbol{0}$
for $i = 0$ to $n - 1$ **do**
  $\boldsymbol{r}^{(i+1)} \leftarrow \text{prox}_{\sigma F_1^*}(\boldsymbol{r}^{(i)} + \sigma\boldsymbol{\mathcal{L}}\tilde{\boldsymbol{x}}^{(i)})$
  $\boldsymbol{s}^{(i+1)} \leftarrow \text{prox}_{\sigma F_2^*}(\boldsymbol{s}^{(i)} + \sigma\boldsymbol{W}\tilde{\boldsymbol{x}}^{(i)})$
  $\boldsymbol{x}^{(i+1)} \leftarrow \text{proj}_{\mathcal{S}}(\boldsymbol{x}^{(i)} - \frac{\tau}{2}\boldsymbol{\mathcal{L}}^T\boldsymbol{r}^{(i+1)} - \frac{\tau}{2}\boldsymbol{W}^T\boldsymbol{s}^{(i+1)})$
  $\tilde{\boldsymbol{x}}^{(i+1)} \leftarrow \boldsymbol{x}^{(i+1)} + \theta(\boldsymbol{x}^{(i+1)} - \boldsymbol{x}^{(i)})$
end for
return $\hat{\boldsymbol{p}} \leftarrow \boldsymbol{x}^{(n)}$.

5.3.2 Iterative binary descriptor reconstruction

To our surprise, when implementing and testing Alg. 3 it turned out that it was able to reconstruct not only real-valued descriptors but also binarized ones, *i.e.*, it still worked without modifications for some $\bar{\boldsymbol{p}} \in \{-1, 1\}^M$ instead of $\mathbb{R}^M$. This is probably due to the choice of the ℓ_1-norm in the data term, which tends to attach more importance to the sign of the error than to its actual value. However, the behavior of our solver in the binarized descriptor case was unstable and it consistently failed to reconstruct some image patches, yielding a null solution. Hence, we chose to leverage some recent results from 1-bit Compressive Sensing [135] to work out a dedicated binary reconstruction scheme.

Keeping the same inverse-problem approach, we substantially modified the functional of (5.6) in two ways:

1. the data term was changed to enforce bitwise consistency between the LBD computed from the reconstructed patch and the input binary descriptor;

2. to apply the same Binary Iterative Hard Thresholding (BIHT) algorithm as [135], we take as sparsity measure the ℓ_0-norm of the wavelet coefficients instead of the relaxed version obtained with the ℓ_1-norm.

We are interested by the solution of this new Lasso-type program [121]:

$$\hat{\boldsymbol{p}} = \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min}\, \mathcal{J}(\boldsymbol{x}) \text{ s.t. } \|\boldsymbol{W}\boldsymbol{x}\|_0 \leqslant k \text{ and } \boldsymbol{x} \in \mathcal{S}, \tag{5.20}$$

where the constraints enforces both the validity and the k-sparsity of $\boldsymbol{x}$ in the wavelet domain.

Inspired by [135], we set our data term as

$$\mathcal{J}(\boldsymbol{x}) = \|[\,\bar{\boldsymbol{p}} \odot \mathcal{B}(\mathcal{L}\boldsymbol{x})]_-\|_1. \tag{5.21}$$

Qualitatively, $\mathcal{J}$ measures the LBD consistency of $\boldsymbol{x}$ with $\boldsymbol{p}$, with $\mathcal{J}(\boldsymbol{x}) = 0$ iff $\mathcal{B}(\mathcal{L}\boldsymbol{x}) = \bar{\boldsymbol{p}}$. Each component of the Hadamard product in the definition of $\mathcal{J}$ is either positive (both signs are the same) or negative. Since the negative function sets to 0 the consistent components, the ℓ_1-norm finally adds the contribution of each inconsistent entry. Note that at the time of writing we do not know a solution for the proximal mapping associated to this data term $\mathcal{J}$, which explains our choice for BIHT over the primal-dual solver used in the previous section.

Similarly to the way Iterative Hard Thresholding aims at solving an ℓ_0-Lasso problem [155], BIHT finds one solution of (5.20) by repeating the three following steps until convergence:

1. computing a step of gradient descent of the data term;

2. enforcing sparsity by projecting the intermediate estimate to the set of patches with at most K non-zero coefficients;

3. enforcing the mean-value constraint on the result.

This last operation was already studied in the previous section for the real-valued case: it is the projection onto the set $\mathcal{S}$. The ℓ_0-norm constraint is applied by Hard Thresholding and amounts to keeping the K biggest coefficients of the wavelet transform of the estimate and discarding the others. We write this operation $\mathcal{H}_K$. Finally, unlike in the primal-dual algorithm, the gradient descent of the data term has to be computed. The result of Lemma 5 in [135] applies in our case and a subgradient of the data term in (5.20) is:

$$\partial\mathcal{J}(\boldsymbol{x}) \;\ni\; \tfrac{1}{2}\mathcal{L}^T(\mathcal{B}(\mathcal{L}\boldsymbol{x}) - \bar{\boldsymbol{p}}), \tag{5.22}$$

i.e., the back-projection of the binary error.

Putting everything together, we obtain Alg. 4 that is the adaptation of BIHT to the reconstruction of image patches from their LBD representation. Again, this algorithm is made of simple elementary steps. The parameter $\tau = 1/M$ guarantees that the current solution $\boldsymbol{x}^{(i)}$ and the gradient step $\frac{\tau}{2}\mathcal{L}^T(\bar{\boldsymbol{p}} - \mathcal{B}(\mathcal{L}\boldsymbol{x}^{(i)}))$ have comparable amplitudes [135]. Since M is determined from the LBD size, only the patch sparsity level K in the wavelet basis must be tuned (see section 5.4.1). In our

experiments, however, the algorithm was not very sensitive to the value of K.

Algorithm 4: BIHT patch reconstruction.

Take $\tau = 1/M$, choose K the number of non-zero coefficients and n the number of iterations
Initialize: $\boldsymbol{x}^{(0)} \leftarrow 0$ and $\boldsymbol{a}_0 \leftarrow 0$.
for $i = 0$ to $n - 1$ **do**
  $\boldsymbol{a}^{(i+1)} \leftarrow \boldsymbol{x}^{(i)} + \frac{\tau}{2}\mathcal{L}^T(\bar{\boldsymbol{p}} - \text{sign}(\mathcal{L}\boldsymbol{x}^{(i)}))$
  $\boldsymbol{b}^{(i+1)} \leftarrow \mathcal{H}_K(\boldsymbol{W}\boldsymbol{a}^{(i+1)})$
  $\boldsymbol{x}^{(i+1)} \leftarrow \text{proj}_{\mathcal{S}}(\boldsymbol{W}^T\boldsymbol{b}^{(i+1)})$
end for
return $\hat{\boldsymbol{p}} \leftarrow \boldsymbol{x}^{(n)}$.

5.3.3 Extension: how to deal with ambiguities in the reconstruction?

As we have seen before, the differential nature of the operators $\mathcal{L}$ yields an ambiguity in the reconstructions: the result can be a priori known only up to an additive constant. Writing $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^N$ a constant patch equal to 1 everywhere, two patches $\boldsymbol{p}$ and $\boldsymbol{p} + \mu\mathbf{1}$ are equivalent for the operator $\mathcal{L}$. In the implementation used in the experiments below, we have partially resolved it by using generic assumptions about the dynamic range of the images and by looking for solutions that have a unit norm, as in the original BIHT algorithm [135]. In this section, we propose to investigate more accurate ways to solve these ambiguities at the possible cost of retaining small bits of additional information during the LBD extraction such as the mean value of the original image patch.

Given an image patch $\boldsymbol{p}$ and its binary descriptor $\bar{\boldsymbol{p}}$, we propose to study the following set of three constraints:

Positivity: the pixel values are lower bounded. Without loss of generality, this lower bound can be taken equal to 0:

$$p_i \geqslant 0 \ \forall i \ ; \tag{5.23}$$

Mean: the mean value of $\boldsymbol{p}$ is equal to some value α. Note that the positivity constraint imposes that α is strictly positive, otherwise the patch is constant (and equal to 0):

$$\langle \boldsymbol{p}, \frac{1}{N}\mathbf{1} \rangle = \alpha > 0 \ ; \tag{5.24}$$

Range: the ℓ_1-norm of the filtered patch is equal to some value β. Again, the positivity constraint imposes that β is strictly positive:

$$\|\mathcal{L}\boldsymbol{p}\|_1 = \beta > 0. \tag{5.25}$$

Imposing a dynamic. Let us suppose first that we know the value β, for example because it was measured before binarization. Then, one can compute:

$$\begin{aligned}
\beta &= \|\mathcal{L}\boldsymbol{p}\|_1 \\
&= \langle \bar{\boldsymbol{p}}, \mathcal{L}\boldsymbol{p} \rangle \\
&= \min_{\mu \in \mathbb{R}} \langle \bar{\boldsymbol{p}}, \mathcal{L}(\boldsymbol{p} - \mu \mathbf{1}) \rangle \\
&= \min_{\mu \in \mathbb{R}} \langle \boldsymbol{p} - \mu \mathbf{1}, \mathcal{L}^T \bar{\boldsymbol{p}} \rangle \\
\beta &\leqslant \left(\min_{\mu \in \mathbb{R}} \|\boldsymbol{p} - \mu \mathbf{1}\|_\infty \right) \cdot \|\mathcal{L}^T \bar{\boldsymbol{p}}\|_1 \quad \text{(by Hölder's inequality).}
\end{aligned}$$

The minimization subproblem has for obvious solution $\mu^\star = (\max_i p_i - \min_i p_i)/2$. Writing this quantity as $d_{\boldsymbol{p}}^\infty$, we finally get to:

$$d_{\boldsymbol{p}}^\infty \geqslant \frac{\beta}{\|\mathcal{L}^T \bar{\boldsymbol{p}}\|_1}. \tag{5.26}$$

If we set (5.26) to the equality, then we have a way to fix the range of the reconstructed patch instead of the arbitrary $[0, 1]$ interval, provided the norm of the feature vector before binarization was measured.

Imposing a mean value. If we know that the mean value of the original patch is α, then the positivity constraint trivially imposes that $\|\boldsymbol{p}\|_\infty \leqslant N\alpha$. Using the same notation as before, this implies that:

$$d_{\boldsymbol{p}}^\infty \leqslant N\frac{\alpha}{2}. \tag{5.27}$$

Hence, knowing the mean of the patch imposes an upper bound to the dynamic range.

Finally, note that combining Eq. (5.26) and (5.27) yields:

$$\alpha \geqslant \frac{2\beta}{\|\mathcal{L}^T \bar{\boldsymbol{p}}\|_1}, \tag{5.28}$$

which can be a way to check the correctness of a solution.

5.4 Results and discussion

5.4.1 Implementation details

For the reconstruction tests presented in this Section, we re-implemented two of the different LBDs: BRIEF and FREAK. For BRIEF, we chose a uniform distribution for the location of the Gaussian measurements, whose support was fixed to 3×3 pixels, following the original paper [145]. For FREAK, we did not take into account the orientation of the image patches (see [130], section 4.4) but we also implemented two variants:

- EX-FREAK, for EXhaustive-Freak, computes all the possible pairs from the retinal pattern;
- RA-FREAK, for RAndomized-FREAK, randomly selects its pairs from the retinal pattern.

All the operators were implemented in C++ with the OpenCV library [1] and used the same codebase for fair comparisons, varying only in the measurement pair selection. The code used to generate the examples in this paper is available online and can be retrieved from the page `http://lts2www.epfl.ch/software/`.

The sensing operator was implemented in the following way:
- the forward operator $\mathcal{L}$ is obtained through the use of integral images for a faster computation of the Gaussian weighted integrals $\langle \mathcal{G}_{\boldsymbol{q}_i,\sigma_i}, \mathcal{G}_{\boldsymbol{q}'_i,\sigma'_i} \rangle$. This approximation has become standard in feature descriptor implementations since it allows a huge acceleration of the computations, see for example [140];
- the backward operator $\mathcal{L}^T$ is the combination of the frame vectors of the considered LBD, weighted by the input vector of coefficients. Hence, we avoid explicitly forming $\mathcal{L}^T$ by computing on the fly the image representation of each vector $\mathcal{L}_i$ of (5.4).

In the previous sections, we have proposed two algorithms that aimed at reconstructing an image patch from the corresponding local descriptor. In order to assess their relevance and the quality of the reconstructions we have applied them on whole images according to the following protocol:

1. an image is divided into patches of size $\sqrt{N} \times \sqrt{N}$ pixels, with an horizontal and vertical offset of N_{off} pixels between each patch;
2. each patch is reconstructed independently from its LBD representation using the additional constraint that its mean should be 0.5, *i.e.*, the mean of the input dynamic range;
3. reconstructed patches are back-projected to their original image position. Wherever patches overlap, the final result is simply the average of the reconstructions.

Hence, the experiments introduce an additional parameter which is the offset between the selected patches.

Note that in contrast with [131] our methods do not require the use of a seamless patch blending algorithm. Simple averaging does not introduce artifacts. Also, we do not require the knowledge of the scale and orientation of the keypoint to reconstruct: we assume it is a patch of fixed size aligned with the image axes. This is in line with the 1-bit feature detection and extraction pipeline: the genuine FAST detector does not consider scale and orientation, and the BRIEF descriptor is not rotation or scale-invariant. Later descriptors such as FREAK were trained in an affine-invariant context; hence by considering only fixed width and orientation our algorithm is suboptimal.

We used patches of 32×32 pixels, LBDs of 512 measurements and ran Alg. 3 and Alg. 4 for 1000 and 200 iterations respectively. In Alg. 3, the trade-off parameter λ was set to 0.1. We tried different values for the sparsity K in Alg. 4 (retaining between 10% and 40% of the wavelet coefficients) but the results did not vary in a meaningful way. Thus we fixed K throughout all the experiments to keep the 40% greater coefficients of $\boldsymbol{W}\boldsymbol{p}$, choosing the Haar wavelet as analysis operator.

The original `Lena`, `Barbara` and `Kata` images can be seen in Fig. 5.4.1.

5.4.2 Reconstruction results

At first glance, the reconstruction results for non-overlapping patches visible in Fig. 5.5 seem very weird and have sometimes very little in common with the original image. However, if we

1. Freely available at `http://opencv.org`

(a) Lena (b) Barbara (c) Kata

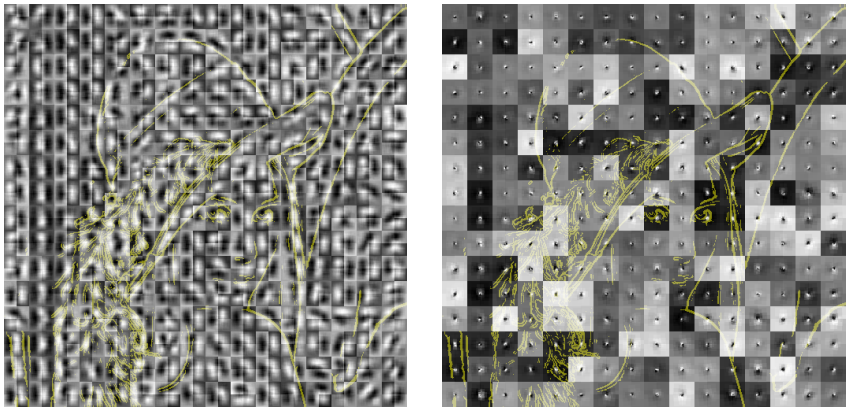
Figure 5.4: *Original images and their designated names in the text.*

overlay the original edges on top of the reconstructed images, one can see that each estimated patch contains a correct version of the original gradient direction. This shows that all the LBDs that we have experimented with capture the local gradient and that this information is enough for Alg. 4 to infer the original value. Even curved lines and cluttered area are encoded by the binary descriptors: see the shoulder of Lena and the feathers of her hat (Fig. 5.5). While there is a significant difference between the reconstruction from BRIEF and FREAK, the variants RA-FREAK and EX-FREAK lead to results which are almost identical with the original FREAK.

Keeping the patch size constant at 32×32 pixels, some results for various offsets between the patches can be seen in Fig. 5.6. An increased number of overlapping patches dramatically improves the quality of the reconstruction using FREAK. This can be understood easily by considering the peculiar shape of the reconstruction without overlap: each estimated patch contains the correct gradient information at its center only. By introducing more overlap between the patches these small parts of contour sum up to recreate the original objects.

Instead of computing patches at fixed positions and offsets, an experiment more relevant with respect to privacy concerns consists in reconstructing only the patches associated with an interest point detector. For the results shown in Fig. 5.7, we have first applied the FAST feature detector of OpenCV with its default parameters and discarded the remaining part of the image, hence the black areas, and used real-valued descriptors. Fig. 5.7 shows the results of the same experiment with binary descriptors. Since FAST keypoints tend to aggregate near angular points and corners, this leads to a relatively dense reconstruction. In each of the three images, the original content can be clearly recognized and a large part of the background clutter has been removed. Thus, one can add as a side note that FAST keypoints are a good indicator of image content saliency. The results in Fig. 5.8 and Fig. 5.12 extend to binary descriptors an important privacy issue that was raised before by [131] for SIFT: if one can intercept keypoint data sent over a network (e.g., to an image search engine), then it is possible to find out what the legitimate user was seeing.

Reconstruction results from BRIEF and FREAK are strikingly different (Fig. 5.5). While BRIEF leads to large, blurred edge estimates that almost entirely occupy the original patch, FREAK produces small accurate edges almost confined in the center of the patch. This allows us to point at a fundamental difference between BRIEF and FREAK. While the former does randomly sample a rough estimate of the dominant gradient in the neighborhood, the latter concentrates its finest measurements and allows more bits (Fig. 5.10) to the innermost part. Thus, the inversion of BRIEF



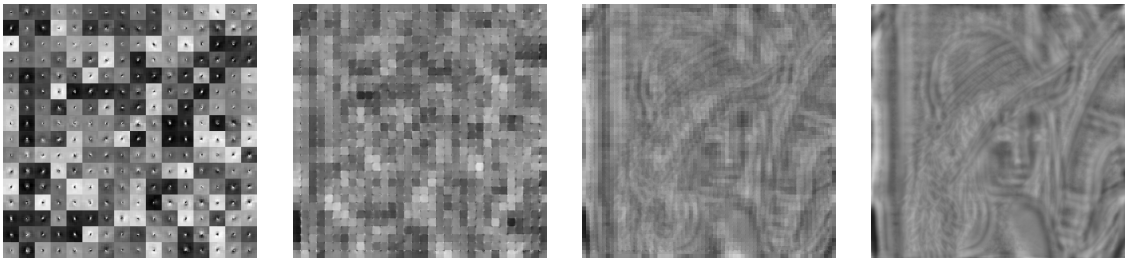
(a) BRIEF (b) FREAK

(c) RA-FREAK (d) EX-FREAK

Figure 5.5: *Reconstruction of Lena from binary LBDs using Alg. 4. There is no overlap between the patches used in the experiment, thus giving a blockwise aspect. We have overlaid some edges from the original image. In each case, the orientation selected for the output patch is consistent with the original main gradient direction. Note also the difference between BRIEF (random measurements spread over an image patch) and FREAK and its derivatives (fine measurements with higher density near the center of the patch): the former gives large blurred edges covering the whole patch, while the latter affect the dominant gradient direction to the central pixel, leaving the periphery almost untouched.*

leads to a fuzzy blurred edge dividing two areas since the information is spread spatially over the whole patch, while the reconstruction of FREAK produces a small but accurate edge surrounded by a large low-resolution area. This is confirmed by the experiments shown in Fig. 5.9. One can especially remark the eyes of `Lena` and `Barbara` and the crossed pattern of the table blanket from `Barbara` which exhibit fine details using FREAK that are missing with BRIEF. In the `Kata` image, one can almost recognize the face of the characters with FREAK, while the fingers holding the sword are clearly distinguishable.

Figure 5.10 compares the measurement strategies of BRIEF and FREAK for 512 measurements. The top row displays the sum of the absolute values of the weights applied to a pixel when computing the descriptor, *i.e.*, $\sum_{i=1}^{M} |(\mathcal{G}_{\boldsymbol{q}_i,\sigma_i})_j| + |(\mathcal{G}_{\boldsymbol{q}'_i,\sigma'_i})_j|$ in (5.3) for the N pixels $1 \leq j \leq N$. We clearly see that BRIEF measures patch intensity almost uniformly over its domain, while FREAK focuses its



(a) N_{off} = 32 pixels (b) N_{off} = 16 pixels (c) N_{off} = 8 pixels (d) N_{off} = 1 pixel

Figure 5.6: *Reconstruction of Lena from binary FREAKs. The size of the patches is kept fixed at* 32×32 *pixels, while their spacing is gradually reduced. We start with an offset of 32 pixels, i.e., no overlap, until a dense reconstruction. In the limit when each pixel is reconstructed from its neighborhood the individual edge bits chain up and one can clearly distinguish the original image contours, like after the application of a Laplacian filter.*

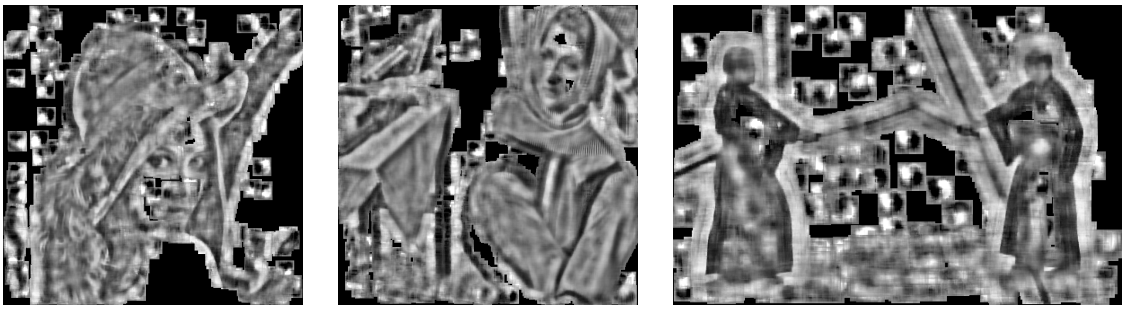


Figure 5.7: *Reconstruction of floating-point (non binarized) BRIEFs centered on FAST keypoints.*

patch observation on the patch domain center. Yet, when plotting in how many LBD measurements a pixel contributed (Fig. 5.10, bottom row) one can see that FREAK also uses peripheral pixels. Since both the weight and occurrence patterns are similar with BRIEF, it means that this LBD is democratic and gives all the pixels a similar importance.

5.4.3 Quality and stability of the reconstruction

Because of the very peculiar structure of the LBD operators, establishing strong mathematical properties on these matrices is a very arduous task, especially in the 1-bit case. As a consequence, finding indubitable theoretical grounds to the success of our BIHT reconstruction algorithm still remains to be investigated. Intuitively, one can however remark that the conditions used to ensure the existence of a reconstruction in Compressive Sensing, like the famous Restrictive Isometry Property (RIP), are only *sufficient* conditions and are by no means necessary conditions. Since LBDs were designed to accurately describe some image content, they are probably more efficient than random sensing matrices. Hence, they can capture more information from an input patch with very few measurements and with a more brutal quantization at the cost of a loss in genericity: they are specialized sensors tuned to image keypoints.

An important parameter with respect to the expected quality of the reconstructions is of course

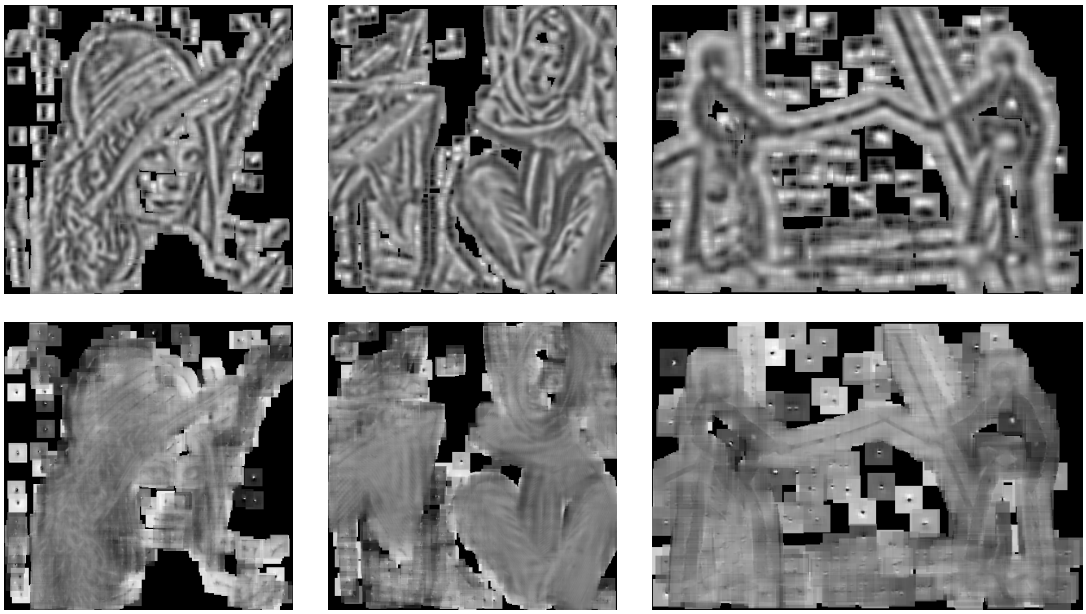


Figure 5.8: *Reconstruction of LBDs centered on FAST keypoints only.* Top row: *using BRIEF.* Bottom row: *using FREAK. Since the detected points are usually very clustered there is a dense overlap between patches, yielding a visually plausible reconstruction. The original image content has been correctly recovered by Alg. 4 from binary descriptors, and eavesdropping the communications of a mobile camera (e.g., embedded in a smartphone) could reveal private data.*

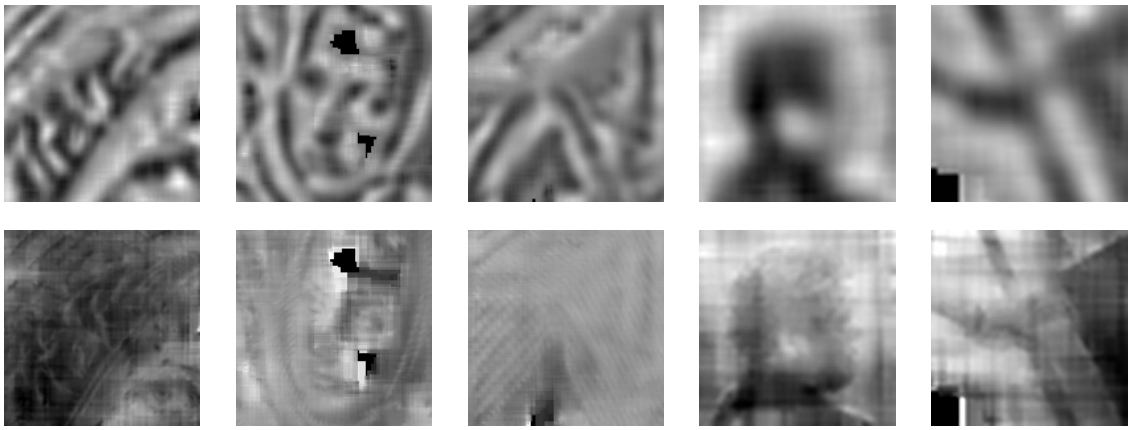
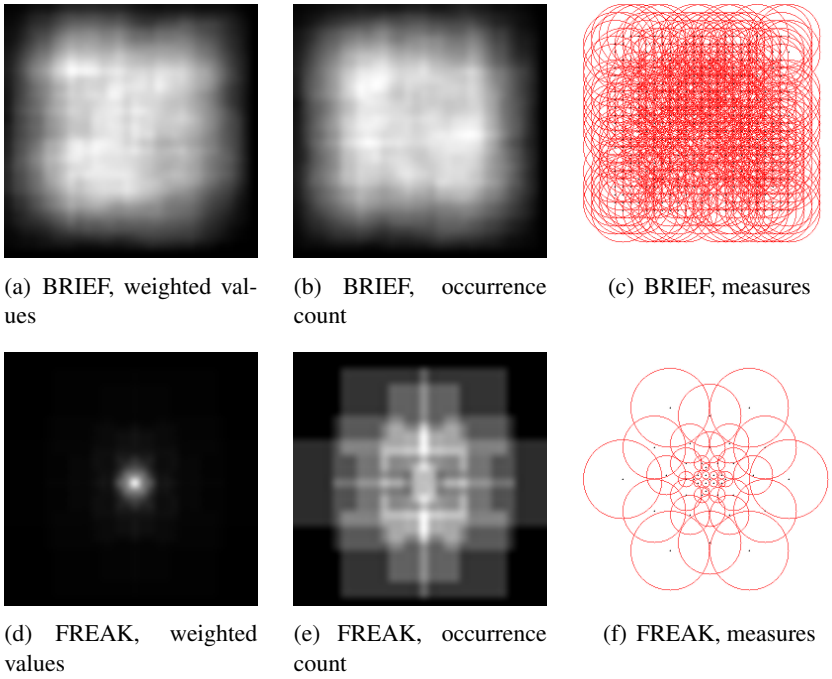


Figure 5.9: *Details of the reconstructions from Fig. 5.8.* Top row: *using BRIEF as LBD.* Bottom row: *using FREAK. The reconstructed patches were selected by the application of the FAST detector with identical parameters. While BRIEF is successful at capturing large gradient orientations, hence giving pleasant results when the image is seen from a distance, FREAK captures more accurate orientations in the center of the patches. Thus finer details are recovered: notice for example the eyes in the pictures of Lena and Barbara, the textures from Barbara or the face and the fingers in the kata image. For this Figure, some additional contrast enhancement post-processing was applied to emphasize the point.*



(a) BRIEF, weighted values (b) BRIEF, occurrence count (c) BRIEF, measures

(d) FREAK, weighted values (e) FREAK, occurrence count (f) FREAK, measures

Figure 5.10: *Comparison of the spatial weights in BRIEF and FREAK basis functions. From left to right: sum of the weights of each pixel when computing a descriptor, number of times a pixel contributes to the descriptor, spatial support of the sensing Gaussians. Brighter means higher importance (higher weight in the first column, more occurrences in the second one). One can see that BRIEF considers pixels almost equally all over the patch, while FREAK gives a very high weight to the centre. This shows that FREAK uses peripheral values, but with a much lower ponderation. BRIEF is clearly more democratic since the weight pattern is similar to the occurrence pattern. Hence, the FREAK descriptors uses more bits to encode the central geometry, while BRIEF gathers information from all over the patch.*

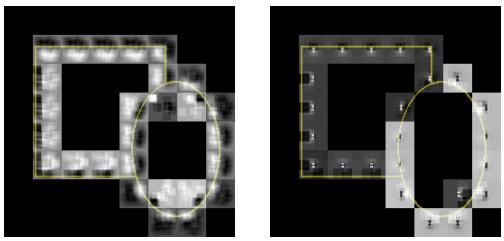
the length M of the LBDs. Since we lack a reliable quality metric to assess the reconstructions, we have proceeded to visual comparisons between the original image contours and the reconstructed gradient directions on a synthetic image. As can be seen in Fig. 5.11, dominant orientations are reconstructed correctly until $M = 128$ measures. Smaller sizes yield blocky estimated patches where it is hard to infer any edge direction.

5.5 Binary stable descriptors

5.5.1 Scrambling for secrecy

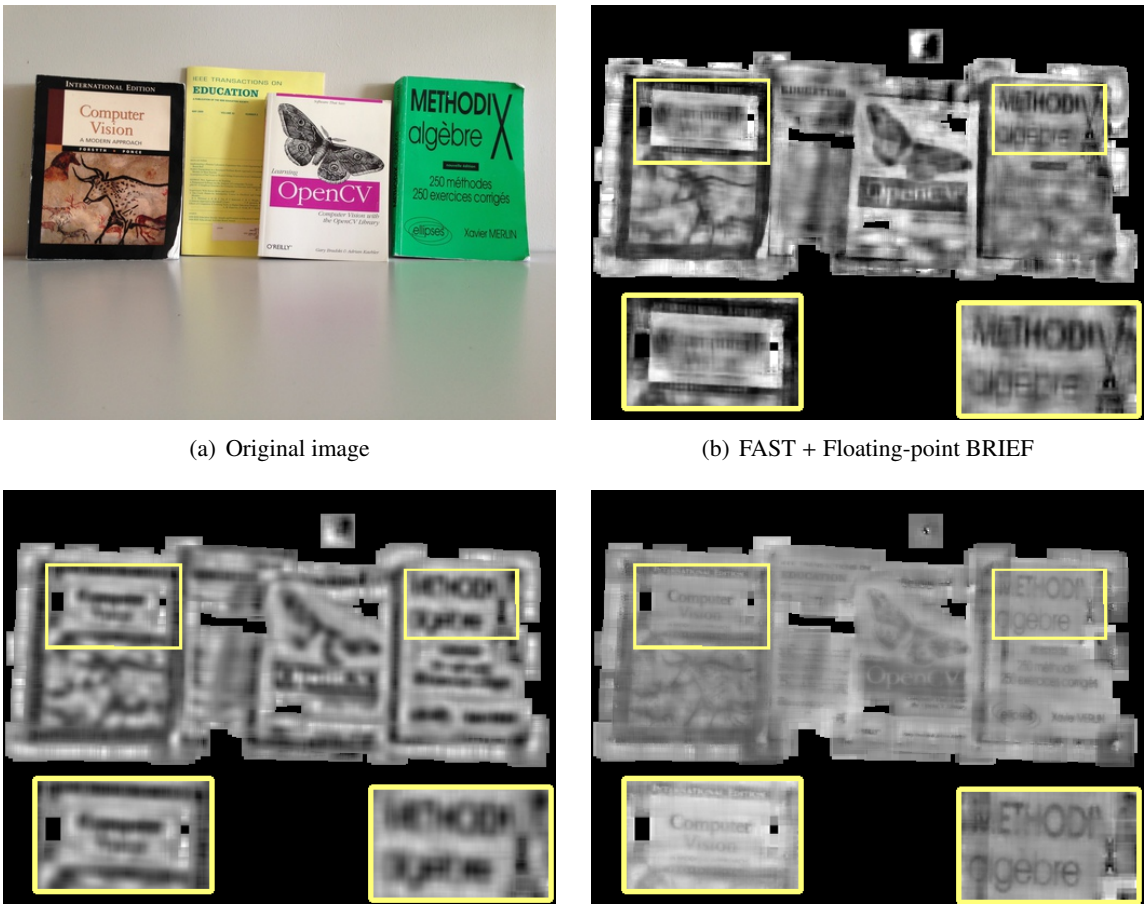
In this section, we establish a theorem that provides some bounds on the distances between different LBDs when they differ by a random projection. This result will be useful to design more secretive descriptors, and it could also yield a new way of generating future LBDs.

Let us assume explicitly that the image patch $\boldsymbol{p} \in \mathbb{R}^N$ is K sparse or compressible in a basis $\boldsymbol{\Psi}$, *i.e.*, $\boldsymbol{p} = \boldsymbol{\Psi}\alpha$ and either $K = \|\boldsymbol{\alpha}\|_0 \ll N$ or $\|\boldsymbol{\alpha}\|_1/\|\boldsymbol{\alpha}\|_2 \leq \sqrt{K}$. We write its non-binarized feature vector $\boldsymbol{d} = \mathcal{L}\boldsymbol{p} \in \mathbb{R}^M$. Given two patches $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, we further assume that $\mathcal{L}$ preserves their



(a) BRIEF, $M = 128$ (b) FREAK, $M = 128$

Figure 5.11: *Zero-overlap reconstruction of a synthetic image using LBDs of 128 measurements instead of 512 a sin the other experiments (and 256 in most image matching softwares). Note that in spite of the huge information loss (compression ratio of 256:1 for each patch) the directions of the edges are correctly estimated.*



(a) Original image (b) FAST + Floating-point BRIEF

(c) FAST + binarized BRIEF (d) FAST + binarized FREAK

Figure 5.12: *Reconstruction of book covers. The bottom part of each image shows in inset a close-up view of two book titles. This experiment confirms the difference between BRIEF and FREAK: while the former extracts salient shapes such as the auroch and the butterfly, the latter is more successful at reconstructing the text. Note that FREAK allows the reading of 3 titles out of 4, hence demonstrating the potential existing privacy breach in case of mobile communications eavesdropping.*

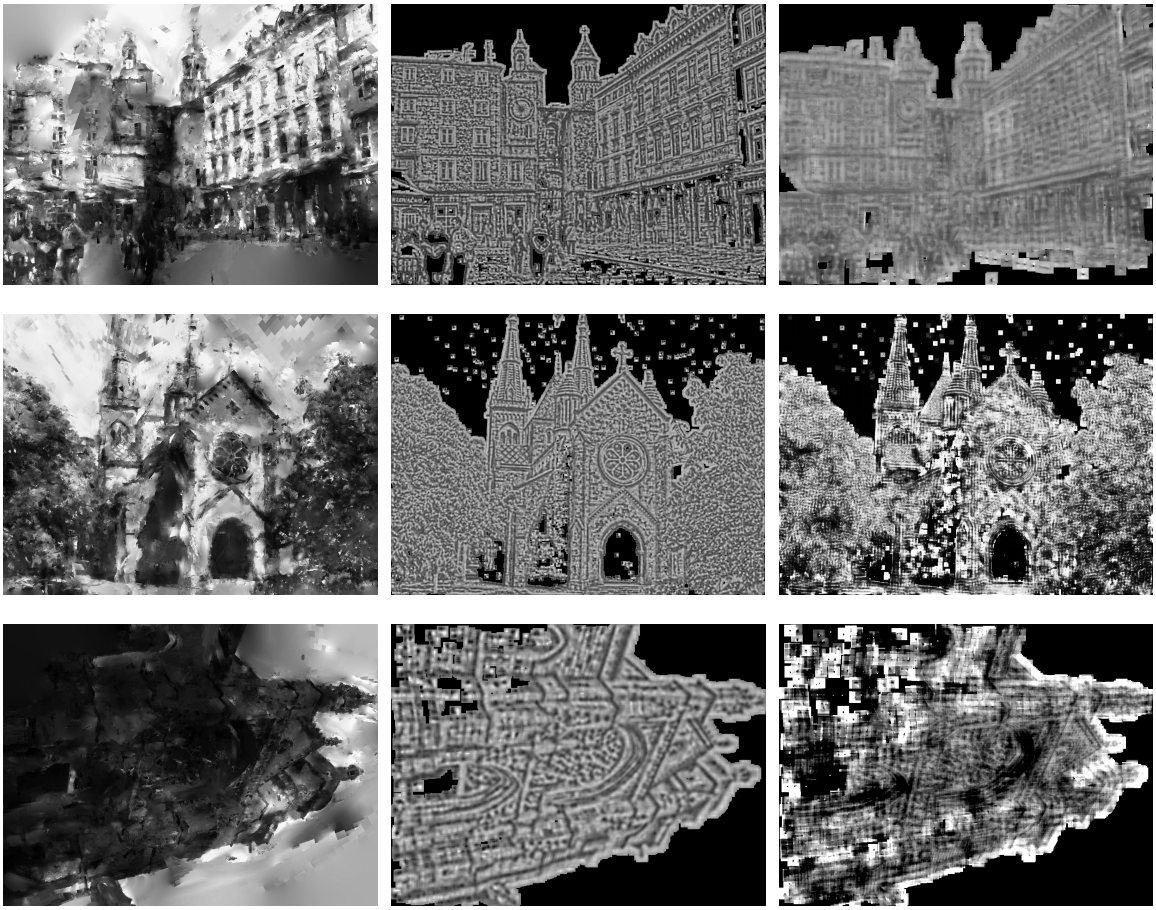


Figure 5.13: *Comparison with [131]. From left to right: image reconstructed with the method of [131], our binary reconstruction algorithm using FAST+BRIEF (middle) and FAST+FREAK (right). We used patches of* 32×32 *pixels in our algorithm. The contrast of the FREAK results was enhanced for readability, but this Figure is best viewed online in electronic version.*

locality (in the sense of the Locality-Sensitive Hashing of Andoni *et al.*[66]): if $\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|$ is small (resp. big), then $\|\boldsymbol{d}_1 - \boldsymbol{d}_2\|$ is also small (resp. big).

Let $\boldsymbol{\Phi} \in \mathbb{R}^{D\times M}$ be a randomly generated matrix. In the previous sections, we have shown that the observation of the binary descriptor $\mathcal{B}(\boldsymbol{d}) = \mathcal{B}(\mathcal{L}\boldsymbol{p})$ could lead to the reconstruction of the original patch $\boldsymbol{p}$, which is a potential privacy breach. Hence, we propose to build a *scrambled* (or secret) binary descriptor $\bar{\boldsymbol{d}}^s$ by the simple operation:

$$\bar{\boldsymbol{d}}^s = \mathcal{B}(\boldsymbol{\Phi}\boldsymbol{d}) = \mathcal{B}(\boldsymbol{\Phi}\mathcal{L}\boldsymbol{p}) \tag{5.29}$$

Note that conversely to some random projection schemes, e.g., used in dimensionality reduction, we do not impose any requirement on the dimension M relative to D, *i.e.*, it could be larger or smaller than D.

Hence, if we can prove that the random projection $\boldsymbol{\Phi}$ preserves the distance between the original descriptors, then we can possibly improve the protection of the user's privacy by generating a matrix $\boldsymbol{\Phi}$ and applying it to all the descriptors. As long as this random matrix is kept secret, it will prevent

the inversion of the descriptors: our algorithms require the knowledge of the measurements inside a patch, and the alternative works need to be trained on the exact descriptor. The remainder of this section is dedicated to proving this intuitive scheme and analyzing its implications.

5.5.2 Binary Stable Descriptors

Let us now define the normalized Hamming distance between two binary strings $\boldsymbol{a}, \boldsymbol{b}$ that both belong to $\mathcal{B}^D = \{-1, +1\}^D$ as

$$d_H(\boldsymbol{a}, \boldsymbol{b}) := \tfrac{1}{D} \sum_{i=1}^{D} a_i \oplus b_i \ \in [0, 1],$$

where $a \oplus b$ is the « exclusive or » (XOR) operation between a and b. For $\boldsymbol{r}, \boldsymbol{s} \in S^{M-1}$ (the unit sphere of $\mathbb{R}^M$), we define also the angular distance between these two vectors as

$$d_S(\boldsymbol{r}, \boldsymbol{s}) := \tfrac{1}{\pi} \arccos\langle \boldsymbol{r}, \boldsymbol{s} \rangle \ \in [0, 1].$$

Assuming $\boldsymbol{\Psi} = \mathrm{Id}$ for simplicity, we have the following theorem:

Theorem 1. *Let $\epsilon > 0$, $\eta \in (0, 1)$ and take*

$$D \ \geqslant \ \tfrac{2}{\epsilon^2}(2K \, \log(\tfrac{37N\sqrt{M}}{\epsilon K}) + \log(\tfrac{2}{\eta})). \tag{5.30}$$

Let us generate a random Gaussian matrix $\boldsymbol{\Phi} \sim \mathcal{N}^{D \times M}(0, 1)$. Then, for any two K-sparse patches $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ with descriptors $\boldsymbol{d}_1 = \mathcal{L}\boldsymbol{p}_1$ and $\boldsymbol{d}_2 = \mathcal{L}\boldsymbol{p}_2$ and scrambled descriptors $\bar{\boldsymbol{d}}_1^s, \bar{\boldsymbol{d}}_2^s$ obtained by Eq. (5.29)*, we have*

$$d_S(\boldsymbol{d}_1, \boldsymbol{d}_2) - \epsilon \leqslant d_H(\bar{\boldsymbol{d}}_1^s, \bar{\boldsymbol{d}}_2^s) \leqslant d_S(\boldsymbol{d}_1, \boldsymbol{d}_2) + \epsilon, \tag{5.31}$$

with a probability higher than $1 - \eta$.

Interpretation. We start by discussing the interpretation of the theorem 1. Its proof is postponed to the end of this section.

First, one can remark that the property in theorem 1 is not strictly equivalent to the BϵSE property [135] since the signal is not assumed sparse in the domain where the angle d_S is computed, *i.e.*, $\mathcal{L} \neq \mathrm{Id}$ and $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are not sparse. Moreover, since we do not impose strict conditions on $\mathcal{L}$, in particular it is not an orthonormal basis, we cannot simply reformulate the problem by showing that $\boldsymbol{\Phi}\mathcal{L}$ is a random Gaussian matrix (which is not *a priori* the case). However, this theorem shows our previous claim: binarizing a randomized descriptor $\mathcal{L}$ does not break its locality with the extra asset of enforcing its secrecy (if the random matrix $\boldsymbol{\Phi}$ is kept secret).

Solving Eq. (5.30) for ϵ, one obtains that $\epsilon = O(\sqrt{\tfrac{K}{D}} \log(\tfrac{N\sqrt{M}}{K}))$ [135]. Therefore, if D is sufficiently large and if $\mathcal{L}$ is sufficiently local angularly, *i.e.*, if $d_S(\boldsymbol{d}_1, \boldsymbol{d}_2)$ follows the distance between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, then $\mathcal{B}(\boldsymbol{\Phi}\mathcal{L}\cdot)$ will be local too in $\mathcal{B}^D$ since a small ϵ will preserve it according to Eq. (5.31).

Interestingly, even if the randomization by $\boldsymbol{\Phi}$ is performed after the descriptor computation by $\mathcal{L}$, the requirement imposed on the dimension D of the binary descriptor d^s depends mainly on the image sparsity K. In particular, the linear descriptor dimension M induces only a minor $\log M$ penalty in (5.30). Furthermore, this sketches a possible scheme for the design of future LBDs:

1. choose a transform $\mathbf{\Psi}$ where image patches around keypoints are sparse, which should be feasible since keypoints occur at very special locations in images (corners, blobs...);
2. design a linear transform $\mathcal{L}$ that creates the correct ordering on the patches (for example by training like FREAK). This is somewhat similar to the *kernel trick* in Machine Learning where features are transformed to a space where they obey a correct metric;
3. randomly select the projections to keep.

Proof of Theorem 1. For any creation process providing $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ in $\mathbb{R}^D$, Lemma 3 in [135] tells us that, for $\mathbf{\Phi} \in \mathcal{N}^{D\times M}(0,1)$, for $\epsilon' > 0$ and $0 \leq \delta \leq 1$, we have

(5.32)

where $B^*_\delta(\boldsymbol{u}) := B_\delta(\boldsymbol{u}) \cap S^{M-1}$.

We consider now that $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are taken in a δ-covering set $Q_\delta \subset \Sigma^*_K(\mathcal{L})$, where $\Sigma^*_K(\mathcal{L})$ is defined by

$$\Sigma^*_K(\mathcal{L}) := \{\boldsymbol{d} = \mathcal{L}\boldsymbol{x} : \|\boldsymbol{x}\|_0 \leq K, \|\boldsymbol{d}\| = 1\} \subset \mathbb{R}^M,$$

i.e., it is the subset of the unit ball that is the image of a K-sparse point $\boldsymbol{x}$ by the operator $\mathcal{L}$. By definition of a δ-covering, for any $\boldsymbol{u} \in \Sigma^*_K(\mathcal{L})$ there is a $\boldsymbol{v} \in Q_\delta$ such that $\|\boldsymbol{u} - \boldsymbol{v}\| \leq \delta$. We can always decide to normalize $\Sigma^*_K(\mathcal{L})$ to unit vectors since the probabilistic relation above does not depend on the descriptor length.

We know from [135] that $|Q_\delta| \leq \binom{N}{K}(\frac{3}{\delta})^K$. Indeed, for one specific slice of $\Sigma^*_K(\mathcal{L})$ where we fix the K-length support of the vectors $\boldsymbol{x}$, this slice is the intersection of S^{M-1} with a subspace of dimension K, *i.e.*, the slice is a $(K-1)$-sphere δ-covered with no more than $(\frac{3}{\delta})^K$ points. Since there are $\binom{N}{K}$ available such slices, the cardinality of Q_δ follows.

Therefore, by applying a union bound to all $\boldsymbol{d}_1, \boldsymbol{d}_2$ taken in Q_δ, and noting that there are no more than $|Q_\delta|^2$ such pairs, we have jointly

$$\Pr\left\{\forall \boldsymbol{u} \in B^*_\delta(\boldsymbol{d}_1), \forall \boldsymbol{v} \in B^*_\delta(\boldsymbol{d}_2), |d_H(\bar{\boldsymbol{u}}^s, \bar{\boldsymbol{v}}^s) - d_S(\boldsymbol{d}_1, \boldsymbol{d}_2)| \leqslant \epsilon' + \sqrt{\frac{\pi}{2}M\delta}\right\} \geqslant 1 - 2(\frac{3eN}{K\delta})^{2K} e^{-2\epsilon' D}, \tag{5.33}$$

using $\binom{N}{K} \leq (\frac{eN}{K})^K$.

Moreover, by definition of the covering, for any $\boldsymbol{r}, \boldsymbol{s} \in \mathbb{R}^M$, there exist two vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ in Q_δ with $\boldsymbol{r} \in B^*_\delta(\boldsymbol{d}_1)$ and $\boldsymbol{s} \in B^*_\delta(\boldsymbol{d}_2)$ such that

$$|d_H\left(\mathcal{B}(\mathbf{\Phi}\boldsymbol{r}), \mathcal{B}(\mathbf{\Phi}\boldsymbol{s})\right) - d_S(\boldsymbol{d}_1, \boldsymbol{d}_2)| \leqslant \epsilon' + \sqrt{\frac{\pi}{2}M\delta} \tag{5.34}$$

with a probability exceeding $1 - 2\,(\frac{3eN}{K\delta})^{2K} e^{-2\epsilon'^2 D}$.

Note that $\boldsymbol{r} \in B^*_\delta(\boldsymbol{d}_1)$ implies that $\pi d_S(\boldsymbol{r}, \boldsymbol{d}_1) \leqslant 2\arcsin(\delta/2) \leqslant \pi\delta/2$, and $d_S(\boldsymbol{d}_1, \boldsymbol{d}_2)$ can be similarly bounded. Thus, $d_S(\boldsymbol{r}, \boldsymbol{s}) \geqslant d_S(\boldsymbol{d}_1, \boldsymbol{d}_2) - \delta$ and $d_S(\boldsymbol{r}, \boldsymbol{s}) \leqslant d_S(\boldsymbol{d}_1, \boldsymbol{d}_2) + \delta$. Using this observation, we find that

$$|d_H\left(\mathcal{B}(\mathbf{\Phi}\boldsymbol{r}), \mathcal{B}(\mathbf{\Phi}\boldsymbol{s})\right) - d_S(\boldsymbol{r}, \boldsymbol{s})| \leqslant \epsilon' + \left(1 + \sqrt{\frac{\pi}{2}M}\right)\delta \tag{5.35}$$

with the same probability.

Let us define the probability of failure as $2\,(\frac{3eN}{K\delta})^{2K}\,e^{-2\epsilon'^2 D} = \eta$, where $0 < \eta < 1$, and set $\epsilon' = (1+\sqrt{\frac{\pi}{2}D})\,\delta$ and $2\epsilon' = \epsilon$. Solving for D, we finally get that $|d_H(\mathcal{B}(\mathbf{\Phi r}), \mathcal{B}(\mathbf{\Phi s})) - d_S(\boldsymbol{d}_1, \boldsymbol{d}_2)| \leqslant \epsilon$ with a probability bigger than $1-\eta$ if:

$$D \geqslant \frac{2}{\epsilon^2}(2K\,\log(\frac{3eN}{K}) + 2K\,\log(\frac{2+\sqrt{2\pi D}}{\epsilon}) + \log(\frac{2}{\eta})).$$

Since $M \geqslant 1$, we have that $2 + \sqrt{2\pi M} \leq (2 + \sqrt{2\pi})\sqrt{M}$.

Since $(2 + \sqrt{2\pi}) \approx 36.75 < 37$, we can make further simplifications and the previous relation is obviously satisfied if

$$D \geqslant \frac{2}{\epsilon^2}(2K\,\log(\frac{37N\sqrt{M}}{\epsilon K}) + \log(\frac{2}{\eta})),$$

which concludes. Note that the bound is not tight because of the last rounding. □

5.6 Conclusion and future work

In this chapter, we have presented two algorithms that can successfully reconstruct small image parts from a subset of local differences without requiring external data or prior training. Both algorithms leverage an inverse problem approach to tackle this task and use as regularization constraint the sparsity of the reconstructed image patches in some wavelet frame. They rely however on different frameworks to solve the corresponding problem, but are the only reconstruction algorithms proposed so far that do not rely on any kind of prior learning.

The first method relies on proximal calculus to minimize a convex non-smooth objective function, adopting a deconvolution-like approach. While this functional was not specifically designed for 1-bit LBDs, it has proved to be robust enough to provide some 1-bit reconstructions, but it does lack stability in this case. On the other hand, the second method was built from the ground up to handle 1-bit LBDs, and thus provides stable results. The reconstruction process is guided by a hard sparsity constraint in the wavelet domain.

There are several levels on which to exploit and interpret our results. First, they can have an important industrial impact. Since it is possible to easily invert LBDs without additional information, mobile application developers cannot simply move from SIFT to LBDs in order to avoid the privacy issues raised by [131]. Hence, they need to add an additional encryption tier to their feature point transmission process if the conveyed data is either sensitive or private.

Second, the differences in the reconstruction from different LBDs can help researchers to design their own LBDs. For example, our experiments have pointed out that BRIEF encodes information at a coarser scale than SIFT, and maybe both descriptors could be combined in some way to create a scale-aware descriptor taking advantage of both patterns. Hence, our work can be used as a tool to study and compare binary descriptors providing different information than standard matching benchmarks. Furthermore, the fact that real-value differences yield comparable results as binarized descriptors legitimates *a posteriori* the performance of LBDs in matching benchmarks: they encode most of the originally available information.

While it is always interesting to have alternative algorithms for a given task, we think that the most important part from [132] is not the proposed reconstruction methods. It is instead the object

classification experiment where they asked Amazon Mechanical Turk users to classify the reconstruction results according to the Pascal VOC classes. It turned out that the human classification lead to the same false positives as the automated classification. Hence, all the possible information from the HOG descriptors was already exploited by the Machine Learning algorithms. While this is not directly related to the current thesis, it does illustrate how feature inversion algorithms can be used to help understanding and developing patch descriptors.

Finally, our framework for 1-bit contour reconstruction could be combined with the previously proposed Gradient Camera concept [156], leading to the development of a 1-bit Compressive Sensing Gradient Camera. This disruptive device would ally the qualities of both worlds with an extended dynamic range and low power consumption. Exploiting the retinal pattern of FREAK and our reconstruction framework could also yield neuromorphic cameras mimicking the human visual system that could be useful for medical and physiological studies.

Interestingly, in a previous work dedicated to the understanding of human vision [157], Oliva and Torralba did point out low frequency / high frequency patterns from image blobs that are similar to the difference observed here between BRIEF and FREAK reconstructions. Hence, our reconstruction algorithms could be exploited not only for pure Computer Vision and Pattern Recognition needs but also in the analysis of early vision processes, including investigations on the origin of the retinal pattern (that served as a basis for FREAK).

Of course, the reconstruction algorithms still need to be improved before reaching an application level where smart cameras sending keypoints would replace smartphones taking pictures. Among the possible improvements, we believe that an interesting extension would be to make our framework *scale-sensitive*. While some feature point detectors provide a scale space location of the detected feature, we discarded the scale coordinate and used patches of fixed width instead. This does not depreciate our experiments with FAST points because we used an implementation that is not scale aware, but reconstructions of better quality can probably be achieved by mixing smooth coarse scale patches with finer details. Additionally, this work did not investigate the issues linked to the geometric transformation invariance enabled by most descriptors. Our model can be interpreted in terms of reconstruction of canonical image patches that correspond to a reference orientation and scale. As far as we have seen, this omission did not create artifacts in our results. This absence by itself is worth of investigating.

6 Conclusions

6.1 Summary of the thesis

In this thesis, we have seen that patches are powerful primitives in Image Processing. More precisely, our work shows that they can be embedded in different variational frameworks, thus yielding algorithms that do not suffer from the heuristic limitations of the example-based methods. The study of the space of patches is of interest. By exploiting the low dimensionality of the patches we were able to propose a fast non-local denoising algorithm that is fully non-local and clearly exploits the joint sparsity properties of the patches without needing to optimize some overcomplete dictionary. Finally, we have also shown that Local Binary Descriptors are indeed powerful description tools: they contain enough information to reconstruct the original image data, which allows the comparison of existing descriptors in new ways and could hopefully lead to the development of better descriptors.

6.2 Future work

6.2.1 Decoupling locality and non-locality

One of the major difficulties in extending non-locality to various general Image Processing problems is to find a correct yet tractable way to mix the non-local and the spatial (either local or global) constraints on the resulting image. Hence, most of the non-local algorithms that claim to have solved, for example, some non-local deconvolution or demosaicking problem are actually solving alternatively a standard deconvolution problem followed by a non-local smoothing step. This is actually tied to how the non-local constraints are expressed, which is hard to insert into more general frameworks.

An appealing and somewhat promising way to express the non-local constraints instead seems to lie in leveraging some low rank property concerning the appearance of the image. Recall that the underlying assumption of non-locality is that many image patches are redundant. Indeed, this can be restated as claiming that most of the patches are linear combinations of a few seed ones, and is backed up by the low dimensionality of the PCA of image patches. Mathematically, this last formulation can be translated as requiring that the matrix formed by all the patches of an image has a *low rank*. This low rank constraint is easy to embed into a standard variational framework (such as the ones described in [17]) because there exists a proxy for it, the nuclear norm, whose corresponding proximity operator is easy to compute, since it is simply a soft thresholding operation on the singular values of the matrix.

This low rank constraint was successfully exploited for several purposes, such as image alignment [158], camera calibration [159] and inpainting [45]. However, in this series of work, the image is assumed to correspond to only one texture, which is an important limitation in practice. We have tried instead to minimize the following functional:

$$J_{\text{Low-Rank}}(\boldsymbol{x}) = \frac{1}{2}||A\boldsymbol{x} - \boldsymbol{y}|| + \lambda||\boldsymbol{R}\boldsymbol{x}||_* + \mu||\mathcal{G}(\boldsymbol{R}'\boldsymbol{x})||_{1,2} + \nu||\boldsymbol{x}||_{\text{TV}}, \tag{6.1}$$

where A is any linear operator, $\boldsymbol{y}$ is the observation, $\boldsymbol{R}\boldsymbol{x}$ and $\boldsymbol{R}'\boldsymbol{x}$ are two patch matrices (where the patches may have different size in $\boldsymbol{R}\boldsymbol{x}$ and $\boldsymbol{R}'\boldsymbol{x}$), $\mathcal{G}(\boldsymbol{R}\boldsymbol{x})$ is a patch grouping operator (that creates groups of spatially connected patches), $||.||_*$ depicts the nuclear norm and $||.||_{\text{TV}}$ the Total Variation. Since this functional involves a smooth data term plus convex terms of known proximity operator, it can be solved using a first order algorithm such as the generalized forward.-backward splitting from [152].

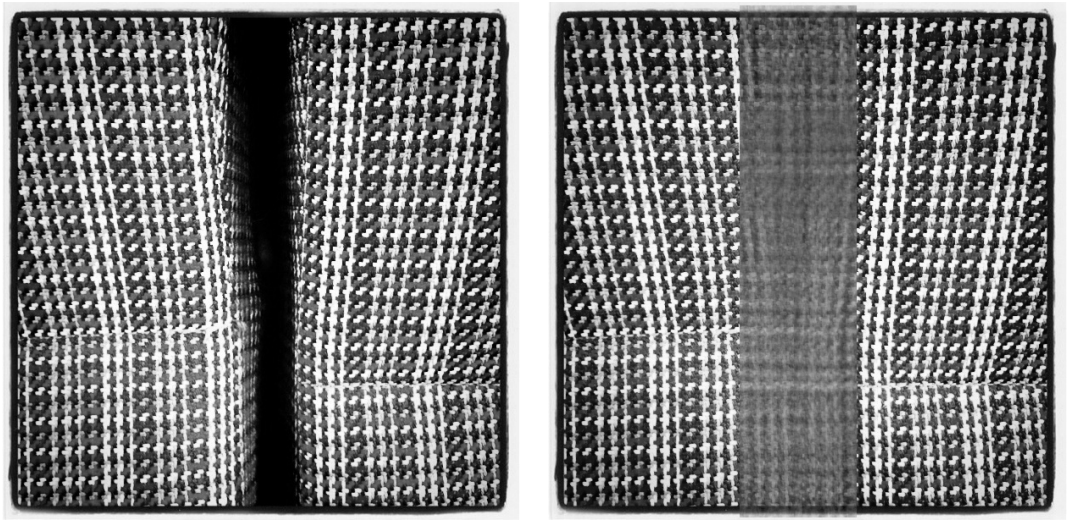
In Eq. (6.1), the constraints are decoupled:
- a classical data term involving a linear operator of any kind;
- a non-local constraint through the nuclear norm of the patch matrix;
- a local appearance regularity constraint via the norm of spatially connected patches;
- a global coherence model with the Total Variation.

The size of the patches between the local and non-local constraint can vary. While our results were clearly promising, we still have a reconstruction artifact that could not be corrected. See Fig. 6.1 for an example of inpainting: while our algorithm managed to infer a plausible missing texture, the mean value in the inpainted area is obviously wrong.

6.2.2 Towards real smart cameras

The possibility to reconstruct patches from binarized descriptor is an exciting outcome of our work. Since LBDs are explicitly designed to be low power and easy to embed into limited hardware, our reconstruction framework can be leveraged to build new cameras embedding computational hardware that would be really smart by computing and sending keypoints along with descriptors, instead of the current smartphones that are only phones equipped with cameras (and sending standard image data).

The communication of keypoints instead of images can lower the bandwidth used for the communications and provide at the same time information that is useful to the other cameras in a network. For example, different cameras could share their keypoints to compute their relative position and perform a distributed calibration of the network.



(a) Original image (b) Inpainting result

Figure 6.1: *Inpainting using a non-local low rank model. The designated inpainting area was a large rectangle in the middle of the image. While our algorithm did successfully reconstruct some texture, it clearly estimated a wrong mean value.*

Of course, this requires the improvement of the visual quality of the reconstructed patches. An important quality gain could come from the addition of a merge step via Poisson editing [160] in order to seamlessly blend the edge information from different descriptors (with possibly different scales) and to propagate the information into regions where no keypoint was detected.

Bibliography

[1] D. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 613–627, 1995.

[2] A. Alahi, Y. Boursier, L. Jacques, and P. Vandergheynst, "A sparsity constrained inverse problem to locate people in a network of cameras," *Digital Signal Processing, 2009 16th International Conference on*, pp. 1–7, 2009.

[3] A. Buades and B. Coll, "A review of image denoising algorithms, with a new one," *Multiscale Modeling and Simulation*, Jan. 2006.

[4] W. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *Computer Graphics and Applications, IEEE*, vol. 22, no. 2, pp. 56–65, 2002.

[5] A. A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer," in *the 28th annual conference*, (New York, New York, USA), pp. 341–346, ACM Press, 2001.

[6] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *Image Processing, IEEE Transactions on*, vol. 13, no. 9, pp. 1200–1212, 2004.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *Image Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 2080–2095, 2007.

[8] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, Jan. 1992.

[9] L. A. Vese and S. J. Osher, "Modeling Textures with Total Variation Minimization and Oscillating Patterns in Image Processing," *Journal of Scientific Computing*, vol. 19, no. 1/3, pp. 553–572, 2003.

[10] J. Aujol, G. Gilboa, T. Chan, and S. Osher, "Structure-texture image decomposition—modeling, algorithms, and parameter selection," *International Journal of Computer Vision*, Jan. 2006.

[11] J. Gilles, "Image decomposition: theory, numerical schemes, and performance evaluation," *Advances in Imaging and Electron Physics*, vol. 158, pp. 89–137, 2009.

[12] Yaroslavsky, L. P. and Yaroslavskij, L. P., "Digital picture processing. An introduction.," *Digital picture processing. An introduction.. L. P. Yaroslavsky (L. P. Yaroslavskij).Springer Seriesin Information Sciences, Vol. 9.Springer-Verlag, Berlin - Heidelberg - New York - Tokyo. ISBN 0-387-11934-5 (USA).*, 1985.

[13] J.-S. Lee, "Digital image smoothing and the sigma filter," *Computer Vision, Graphics, and Image Processing*, vol. 24, pp. 255–269, Nov. 1983.

[14] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *Computer Vision, 1998. Sixth International Conference on*, pp. 839–846, 1998.

[15] C. Kervrann, J. Boulanger, and P. Coupe, "Bayesian non-local means filter, image redundancy and adaptive dictionaries for noise removal," *Scale Space and Variational Methods in Computer Vision*, pp. 520–532, 2007.

[16] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3D filtering," *Proceedings of SPIE*, vol. 6064, pp. 354–365, 2006.

[17] P. Combettes and J. Pesquet, "Proximal Splitting Methods in Signal Processing," *Arxiv preprint arXiv:0912.3522*, Jan. 2009.

[18] J. Moreau, "Proximité et dualité dans un espace hilbertien," *Bull. Soc. Math. France*, vol. 93, no. 2, pp. 273–299, 1965.

[19] A. Chambolle and T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, pp. 120–145, Dec. 2010.

[20] F.-X. Dupe, J. Fadili, and J. L. Starck, "A Proximal Iteration for Deconvolving Poisson Noisy Images Using Sparse Representations," *IEEE Transactions on Image Processing*, vol. 18, no. 2, pp. 310–321, 2009.

[21] J. Boulanger, C. Kervrann, P. Bouthemy, P. Elbau, J. Sibarita, and J. Salamero, "Patch-Based Nonlocal Functional for Denoising Fluorescence Microscopy Image Sequences," *Medical Imaging, IEEE Transactions on*, vol. 29, no. 2, pp. 442–454, 2010.

[22] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1, pp. 89–97, 2004.

[23] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, Jan. 2009.

[24] J. Aujol and A. Chambolle, "Dual norms and image decomposition models," *International Journal of Computer Vision*, Jan. 2005.

[25] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the Nonlocal-Means to Super-Resolution Reconstruction," *Image Processing, IEEE Transactions on*, vol. 18, no. 1, pp. 36–51, 2009.

[26] E. d'Angelo and P. Vandergheynst, "Fully non-local super-resolution via spectral hashing," in *ICASSP 2011 - 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1137–1140, IEEE, 2011.

[27] D. Zhou and B. Schölkopf, "Regularization on Discrete Spaces," in *Pattern Recognition*, pp. 361–368, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[28] A. Elmoataz, O. Lezoray, and S. Bougleux, "Nonlocal Discrete Regularization on Weighted Graphs: A Framework for Image and Manifold Processing," *IEEE Transactions on Image Processing*, vol. 17, no. 7, 2008.

[29] G. Peyré, S. Bougleux, and L. Cohen, "Non-local regularization of inverse problems," *Computer Vision–ECCV 2008*, pp. 57–68, 2008.

[30] L. Bagnato, P. Frossard, and P. Vandergheynst, "A Variational Framework for Structure from Motion in Omnidirectional Image Sequences," *Journal of Mathematical Imaging and Vision*, vol. 41, pp. 182–193, Mar. 2011.

[31] A. Elmoataz, O. Lezoray, S. Bougleux, and V. Ta, "Unifying local and nonlocal processing with partial difference operators on weighted graphs," 2008.

[32] G. Gilboa and S. Osher, "Nonlocal operators with applications to image processing," tech. rep., 2007.

[33] E. d'Angelo and P. Vandergheynst, "Towards unifying diffusion and exemplar-based inpainting," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, 2010.

[34] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–355– I–362 vol.1, 2001.

[35] T. Chan and J. Shen, "Mathematical models for local nontexture inpaintings," *SIAM Journal on Applied Mathematics*, vol. 62, no. 3, pp. 1019–1043, 2001.

[36] T. Chan and J. Shen, "Nontexture inpainting by curvature-driven diffusions," *Journal of visual communication and image representation*, vol. 12, no. 4, pp. 436–449, 2001.

[37] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999.

[38] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," *SIGGRAPH '03: SIGGRAPH 2003 Papers*, July 2003.

[39] P. Harrison, "A non-hierarchical procedure for re-synthesis of complex textures," in *Proc Int Conf Central Europe Comp Graphics*, 2001.

[40] Y. Wexler, E. Shechtman, and M. Irani, "Space-Time Completion of Video," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 3, 2007.

[41] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," *Image Processing, IEEE Transactions on*, vol. 12, no. 8, pp. 882–889, 2003.

[42] P. Arias, V. Caselles, and G. Sapiro, "A variational framework for non-local image inpainting," *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 345–358, 2009.

[43] G. Peyré, "Manifold models for signals and images," *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 249–260, 2009.

[44] J.-F. Aujol, S. Ladjal, and S. Masnou, "Exemplar-based inpainting from a variational point of view," *SIAM Journal on Mathematical Analysis*, vol. 42, no. 3, pp. 1246–1285, 2010.

[45] X. Liang, X. Ren, Z. Zhang, and Y. Ma, "Repairing sparse low-rank texture," in *ECCV'12: Proceedings of the 12th European conference on Computer Vision*, Springer-Verlag, Oct. 2012.

[46] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 60–65 vol. 2, 2005.

[47] A. Buades, "Image and film denoising by non-local means," pp. 1–149, Feb. 2006.

[48] J. Salmon and Y. Strozecki, "From patches to pixels in Non-Local methods: Weighted-average reprojection," *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 1929–1932, 2010.

[49] J. Wang, Y. Guo, Y. Ying, Y. Liu, and Q. Peng, "Fast Non-Local Algorithm for Image Denoising," *Image Processing, 2006 IEEE International Conference on*, pp. 1429–1432, 2006.

[50] F. C. Crow, "Summed-area tables for texture mapping," in *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Request Permissions, Jan. 1984.

[51] M. Jones and P. Viola, "Robust real-time object detection," *Workshop on Statistical and Computational Theories of Vision*, 2001.

[52] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[53] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *Signal Processing Letters, IEEE*, vol. 12, no. 12, pp. 839–842, 2005.

[54] P. Coupé, P. Yger, and C. Barillot, "Fast non local means denoising for 3D MR images," *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006*, pp. 33–40, 2006.

[55] J. Orchard, M. Ebrahimi, and A. Wong, "Efficient nonlocal-means denoising using the SVD," *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 1732–1735, 2008.

[56] T. Brox, O. Kleinschmidt, and D. Cremers, "Efficient Nonlocal Means for Denoising of Textural Patterns," *Image Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1083–1092, 2008.

[57] N. Kumar, L. Zhang, and S. Nayar, "What is a good nearest neighbors algorithm for finding similar patches in images?," in *Computer Vision–ECCV 2008*, pp. 364–378, Springer, 2008.

[58] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," *SODA '93 Proceedings of the fourth annual annual ACM-SIAM Symposium on Discrete algorithms*, 1993.

[59] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[60] R. F. Sproull, "Refinements to nearest-neighbor searching ink-dimensional trees - Springer," *Algorithmica*, 1991.

[61] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high-dimensional filtering," *SIGGRAPH '09: SIGGRAPH 2009 papers*, July 2009.

[62] A. Manzanera, "Local Jet Based Similarity for NL-Means Filtering," *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 2668–2671, 2010.

[63] A. Buades, B. Coll, and J.-M. Morel, "The staircasing effect in neighborhood filters and its solution," *Image Processing, IEEE Transactions on*, vol. 15, no. 6, pp. 1499–1505, 2006.

[64] S. Lefebvre and H. Hoppe, "Perfect spatial hashing," *SIGGRAPH '06: SIGGRAPH 2006 Papers*, July 2006.

[65] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998.

[66] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Foundations of Computer Science, 2006. FOCS '06. 47th Annual IEEE Symposium on*, pp. 459–468, 2006.

[67] W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li, "Modeling LSH for performance tuning," *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, Oct. 2008.

[68] H. Jegou, L. Amsaleg, C. Schmid, and P. Gros, "Query adaptative locality sensitive hashing," *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 825–828, 2008.

[69] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," *Advances in Neural Information Processing Systems*, vol. 21, pp. 1753–1760, 2009.

[70] Y. Bengio, O. Delalleau, N. L. Roux, J.-F. Paiement, P. Vincent, and M. Ouimet, "Learning Eigenfunctions Links Spectral Embedding and Kernel PCA," *Neural Computation*, vol. 16, pp. 2197–2219, Oct. 2004.

[71] R. Coifman, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," *Proceedings of the National Academy of Sciences*, vol. 102, pp. 7426–7431, May 2005.

[72] Y. Weiss, R. Fergus, and A. Torralba, "Multidimensional Spectral Hashing," *Computer Vision - ECCV 2012*, vol. 7576, pp. 340–353, 2012.

[73] N. Sabater, A. Almansa, J. M. P. A. Morel, and M. I. I. T. on, "Meaningful Matches in Stereovision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, May 2012.

[74] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical models and image processing*, vol. 53, no. 3, pp. 231–239, 1991.

[75] D. Capel and A. Zisserman, "Computer vision applied to super resolution," *Signal Processing Magazine, IEEE*, vol. 20, no. 3, pp. 75–86, 2003.

[76] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *Signal Processing Magazine, IEEE*, vol. 20, no. 3, pp. 21–36, 2003.

[77] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super resolution," *Image Processing, IEEE Transactions on*, vol. 13, no. 10, pp. 1327–1344, 2004.

[78] D. Mitzel, T. Pock, T. Schoenemann, and D. Cremers, "Video Super Resolution Using Duality Based TV-L 1 Optical Flow," *Pattern Recognition*, pp. 432–441, 2009.

[79] A. Buades, B. Coll, and J. Morel, "Nonlocal image and movie denoising," *International Journal of Computer Vision*, Jan. 2008.

[80] W. Dong, L. Zhang, G. Shi, and X. Wu, "Nonlocal back-projection for adaptive image enlargement," *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pp. 349–352, 2009.

[81] A. Beck and M. Teboulle, "Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems," *Image Processing, IEEE Transactions on*, vol. 18, no. 11, pp. 2419–2434, 2009.

[82] E. d'Angelo, J. Paratte, G. Puy, and P. Vandergheynst, "Fast TV-L<inf>1</inf> optical flow for interactivity," in *2011 18th IEEE International Conference on Image Processing (ICIP 2011)*, pp. 1885–1888, IEEE, 2011.

[83] S. Lefebvre, S. Hornus, and F. Neyret, "Octree textures on the GPU," *GPU gems*, vol. 2, pp. 595–613, 2005.

[84] S. Laine and T. Karras, "Efficient sparse voxel octrees," in *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, ACM Request Permissions, Feb. 2010.

[85] R. A. Finkel and J. L. Bentley, "Quad Trees A Data Structure for Retrieval on Composite Keys," *Acta informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[86] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, "Computational Geometry," 2008.

[87] W. H. Wen-mei, "GPU Computing Gems Emerald Edition (Applications Of GPU Computing Series)," 2011.

[88] S. F. Frisken and R. N. Perry, "Simple and efficient traversal methods for quadtrees and octrees," *Journal of Graphics Tools*, vol. 7, no. 3, pp. 1–11, 2002.

[89] Y. King, "Floating-point tricks: Improving performance with ieee floating point," in *Game Programmin Gems 2* (M. DeLoura, ed.), (Hinghan, MA.), pp. 167–181, Charles River Media, 2001.

[90] B. Chazelle, "Computational geometry: a retrospective," in *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, ACM Request Permissions, May 1994.

[91] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM computing surveys (CSUR)*, vol. 33, no. 3, pp. 273–321, 2001.

[92] D. L. Donoho, I. M. Johnstone, J. C. Hoch, and A. S. Stern, "Maximum Entropy and the Nearly Black Object," *JSTOR: Journal of the Royal Statistical Society. Series B (Methodological), Vol. 54, No. 1 (1992), pp. 41-81*, vol. 54, pp. 41–81, 1992.

[93] S. S. Chen, D. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1998.

[94] A. Barjatya, "Block matching algorithms for motion estimation," *Final Project Paper, Utah State University*, 2004.

[95] S. Zhu and I. T. o. Kai-Kuang Ma Image Processing, "A new diamond search algorithm for fast block-matching motion estimation," *Image Processing*, vol. 9, no. 2, 2000.

[96] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D image denoising with shape-adaptive principal component analysis," *SPARS'09–Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[97] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images," *Image Processing, IEEE Transactions on*, vol. 16, no. 5, pp. 1395–1411, 2007.

[98] C. Kervrann and J. Boulanger, "Optimal Spatial Adaptation for Patch-Based Image Denoising," *Image Processing*, vol. 15, no. 10, pp. 2866–2878, 2006.

[99] C.-A. Deledalle, V. Duval, and J. Salmon, "Non-local Methods with Shape-Adaptive Patches (NLM-SAP)," *Journal of Mathematical Imaging and Vision*, May 2011.

[100] M. Maggioni and A. Foi, "Nonlocal transform-domain denoising of volumetric data with groupwise adaptive variance estimation," *Proceedings of SPIE*, vol. 8296, p. 82960O, 2012.

[101] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, "Video Denoising, Deblocking, and Enhancement Through Separable 4-D Nonlocal Spatiotemporal Transforms," *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 3952–3966, 2012.

[102] A. Buades, B. Coll, and J.-M. Morel, "Denoising image sequences does not require motion estimation," pp. 70–74, 2005.

[103] C. Liu and W. Freeman, "A high-quality video denoising algorithm based on reliable motion estimation," *Computer Vision–ECCV 2010*, pp. 706–719, 2010.

[104] M. Lebrun, "An Analysis and Implementation of the BM3D Image Denoising Method," *Image Processing On Line*, 2012.

[105] V. Katkovnik and A. Foi, "Advanced Image Processing Based on Spatially Adaptive Nonlocal Image Filtering and Regularization," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 1–296, Department of Signal Processing, Tampere University of Technology, Tampere, Finland, Sept. 2010.

[106] M. Elad and M. Aharon, "Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, pp. 3736–3745, Dec. 2006.

[107] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pp. 40–44, 1993.

[108] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[109] J. Mairal, M. Elad, and G. Sapiro, "Sparse Representation for Color Image Restoration," *Image Processing*, vol. 17, no. 1, pp. 53–69, 2008.

[110] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2272–2279, 2009.

[111] J. Tropp, A. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit," *Signal Processing*, vol. 86, no. 3, pp. 572–588, 2006.

[112] N. Sochen, R. Kimmel, and R. Malladi, "A general framework for low level vision," *Image Processing*, vol. 7, Mar. 1998.

[113] X. Bresson, P. Vandergheynst, and J. Thiran, "Multiscale active contours," *International Journal of Computer Vision*, vol. 70, no. 3, pp. 197–211, 2006.

[114] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, June 2009.

[115] W. Dong, X. Li, L. ZHANG, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011.

[116] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[117] P. Chatterjee and P. Milanfar, "Clustering-Based Denoising With Locally Learned Dictionaries," *IEEE Transactions on Image Processing*, vol. 18, pp. 1438–1451, Nov. 2009.

[118] A. Gramfort and M. Kowalski, "Improving M/EEG source localizationwith an intercondition sparse prior," in *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, pp. 141–144, IEEE, 2009.

[119] M. Kowalski, K. Siedenburg, and M. Dörfler, "Social Sparsity! Neighborhood Systems Enrich Structured Shrinkage Operators," 2012.

[120] M. A. T. Figueiredo and R. D. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Transactions on Image Processing*, vol. 12, no. 8, 2003.

[121] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[122] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[123] M. Kowalski and B. Torrésani, "Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients," *Signal, Image and Video Processing*, vol. 3, pp. 251–264, Sept. 2008.

[124] M. Fornasier and H. Rauhut, "Recovery algorithms for vector-valued data with joint sparsity constraints," *SIAM Journal on Numerical Analysis*, vol. 46, no. 2, pp. 577–613, 2008.

[125] G. Peyré, J. M. Fadili, and C. Chesneau, "Adaptive Structured Block Sparsity Via Dyadic Partitioning," *EUSIPCO 2011*, 2011.

[126] G. Peyré and J. Fadili, "Group Sparsity with Overlapping Partition Functions," 2011.

[127] Y. Zhou, R. Jin, and S. C. H. Hoi, "Exclusive lasso for multi-task feature selection," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 988–995, 2010.

[128] M. Kowalski, K. Siedenburg, and M. Dörfler, "Social Sparsity! Neighborhood Systems Enrich Structured Shrinkage Operators," 2012.

[129] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1281–1298.

[130] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 510–517, 2012.

[131] P. Weinzaepfel, H. Jegou, and P. Pérez, "Reconstructing an image from its local descriptors," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 337–344, 2011.

[132] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba, "Inverting and Visualizing Features for Object Detection," *arXiv.org*, vol. cs.CV, Dec. 2012.

[133] E. d'Angelo, A. Alahi, and P. Vandergheynst, "Beyond Bits: Reconstructing Images from Local Binary Descriptors," *International Conference On Pattern Recognition (ICPR)*, pp. 1–4, Sept. 2012.

[134] E. d'Angelo, L. Jacques, A. Alahi, and P. Vandergheynst, "From Bits to Images: Inversion of Local Binary Descriptors," *arXiv.org*, vol. cs.CV, Nov. 2012.

[135] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-Bit Compressive Sensing via Binary Stable Embeddings of Sparse Vectors," *arXiv.org*, vol. cs.IT, Apr. 2011.

[136] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[137] "Googles Goggles," `http://www.google.com/mobile/goggles/`, 2013.

[138] "Kooaba Image Recognition", `http://www.kooaba.com/`, 2013.

[139] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[140] H. Bay, T. Tuytelaars, and L. van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[141] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893 vol. 1, 2005.

[142] B. Hariharan, J. Malik, and D. Ramanan, "Discriminative decorrelation for clustering and classification," in *ECCV'12: Proceedings of the 12th European conference on Computer Vision*, Springer-Verlag, Oct. 2012.

[143] P. T. Boufounos and R. G. Baraniuk, "1-Bit compressive sensing," in *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pp. 16–21, 2008.

[144] P. L. Combettes and J. C. Pesquet, "Proximal splitting methods in signal processing," *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212, 2011.

[145] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[146] C. Harris and M. Stephens, "A combined corner and edge detector," *Alvey vision conference*, vol. 15, p. 50, 1988.

[147] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision–ECCV 2006*, pp. 430–443, 2006.

[148] T. Ahonen, A. Hadid, and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 12, pp. 2037–2041, 2006.

[149] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, 2011.

[150] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2548–2555, 2011.

[151] E. Sidky and J. Jørgensen, "Convex optimization problem prototyping with the Chambolle-Pock algorithm for image reconstruction in computed tomography," *arXiv.org*, 2011.

[152] H. Raguet, J. Fadili, and G. Peyré, "Generalized Forward-Backward Splitting," 2011.

[153] A. Gonzalez, L. Jacques, C. De Vleeschouwer, and P. Antoine, "Compressive Optical Deflectometric Tomography: A Constrained Total-Variation Minimization Approach," *arXiv.org*, vol. cs.CV, Sept. 2012.

[154] C. Michelot, "A finite algorithm for finding the projection of a point onto the canonical simplex of Rn," *Journal of Optimization Theory and Applications*, vol. 50, no. 1, pp. 195–200, 1986.

[155] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, pp. 265–274, Nov. 2009.

[156] J. Tumblin, A. Agrawal, and R. Raskar, "Why I want a gradient camera," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 103–110 vol. 1, 2005.

[157] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in brain research*, vol. 155, p. 23, 2006.

[158] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "RASL: Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2012.

[159] Z. Zhang, Y. Matsushita, and Y. Ma, "Camera calibration with lens distortion from low-rank textures," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2321–2328, 2011.

[160] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 313–318, 2003.

Emmanuel d'Angelo
emmanuel.dangelo@epfl.ch
French nationality

Profile

Former military engineer, technical expert in Image Processing and Image Intelligence for the French defense procurement agency "Délégation Générale de l'Armement" (DGA, Ministry of Defense, France). Now starting a new career of developer in a high-tech company.

Teaching

2008-2012	• Teaching assistant at EPFL (Lausanne, Switzerland) for Prof. P. Vandergheynst *Bachelor courses* (in French): analog and digital signal processing *Master courses* (in English): advanced signal processing labs with `Matlab`, image processing labs with `OpenCV`
Since 2005	• Teacher for the post-grad education company EuroSAE, Paris (F). Lessons taught: active contours, photogrammetry, motion estimation, hands-on sessions for the MTS014 course
2005 – 2008	• Assistant professor at ENSTA-Paris Tech school of engineering, Paris (F) Programming lessons in language C/C++ Encadrement for first year and foreign students
2005 – 2007	• Supervision of a Computer Science engineer trainee from the ETGL institute, Saclay, (F), 14 months *Various improvements on an automated image processing algorithm evaluation platform*

Education

2008 – 2012	• Doctoral school in Eletrical Engineering, EPFL, Lausanne, (CH)
2006	• Joint military intelligence certificate, CFIAR Strasbourg (F)
2005	• Master of Science degree "Mathematics - Vision - Machine learning", ENS Cachan (F) *Sandwich courses done while working as a full time image processing expert*
2003	• Master of Engineering in *Spatial imaging*, ISAE (Sup'Aéro), Toulouse (F)

Skills

Programming	• Languages: C/C++/Objective-C; high performance computations: OpenCL/CUDA Good knowledge of the open source image processing library `OpenCV` *Example code on github:* http://github.com/sansuiso/ComputersDontSee
Tools	• System administration on Linux and Mac OS X platforms Frequent use of `Matlab`, `Cmake`, `Git`, office tools (including LaTeX)
Languages	• French (native), English, German, basic skills in Japanese

Hobbies

Sports	• 4th dan of Kendo (Japanese fencing), former member of the French junior national team Deputy teacher at "Ecole-Atelier Rudra" (Béjart Ballet), Lausanne (CH), 2009 (6 months) Third rank in the junior European Kendo Championships, Basel (CH), 1998
Internet	• Personal homepage, centered on image processing technologies: http://www.computersdontsee.net
Arts	• Enthusiast analog and digital photographer

Personal Publications

[1] E. d'Angelo, L. Jacques, A. Alahi, P. Vandergheynst, "From Bits to Images: Inversion of Local Binary Descriptors," *submitted to IEEE Transaction on PAMI*, 2012.

[2] E. d'Angelo, A. Alahi, P. Vandergheynst, "Beyond Bits: Reconstructing Images from Local Binary Descriptors," *21st International Conference on Pattern Recognition (ICPR)*, Oral Presentation (oral acceptance rate: 15%), Tsukuba Science City, Japan, 2012.

[3] E. d'Angelo, J. Paratte, G. Puy, P. Vandergheynst, "Fast TV-L1 Optical Flow for Interactivity," *IEEE International Conference on Image Processing (ICIP 2011)*, Brussels, Belgium, 2011.

[4] E. d'Angelo, P. Vandergheynst, "Fully Non-Local Super-Resolution via Spectral Hashing," *International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011)*, Prague, Czech Republic, 2011.

[5] E. d'Angelo, P. Vandergheynst, "Towards unifying diffusion and exemplar-based inpainting," *IEEE International Conference on Image Processing (ICIP 2010)*, Hong Kong, People's Republic of China, 2010.

[6] E. d'Angelo, P. Vandergheynst, "Method to compensate the effect of the rolling shutter effect", US Patent App. 12/662,731, Publication 2011/0267514 A1.