From Theory to Practice, a Tour of Image Denoising

Marc Lebrun

N° ENSC-2011

THÈSE DE DOCTORAT
DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

Présentée par
Monsieur Marc LEBRUN

pour obtenir le grade de

DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

Domaine: **MATHEMATIQUES APPLIQUÉES**

Sujet de la thèse:

From Theory to Practice, a Tour of Image Denoising

Rapporteurs :

Luis Alvarez	Professor	Universidad de Las Palmas de Gran Canaria
Javier Portilla	Director of Research	Instituto de Óptica "Daza de Valdés"
Patrick Pérez	Distinguished Scientist	Technicolor

Thèse présentée et soutenue à Cachan le 12 juin 2014 devant le jury composé de :

Andrés Almansa	Research Scientist	CNRS, Telecom Paris Tech
Luis Alvarez	Professor	Universidad de Las Palmas de Gran Canaria
Frédéric Cao	Deputy Chief Scientist	DxO Labs
Frédéric Guichard	Chief Scientist	DxO Labs
Jean-Michel Morel	Professor	ENS Cachan
Patrick Pérez	Distinguished Scientist	Technicolor
Javier Portilla	Director of Research	Instituto de Óptica "Daza de Valdés"

Centre de Mathématiques et de Leurs Applications
(ENS CACHAN/CNRS/UMR 8536)
61, avenue du Président Wilson, 94235 CACHAN CEDEX (France)

Résumé:

Cette thèse CIFRE se situe dans le cadre de la collaboration étroite entre DxO Labs et le CMLA depuis de nombreuses années, comportant une veille bibliographique active sur différents sujets du traitement d'image, ainsi que la recherche de solutions académiques aux problèmes concrets soulevés par DxO Labs.
Le but de la thèse était la recherche et l'implémentation d'une nouvelle génération d'algorithmes de débruitage d'images raw dans le produit phare de DxO Labs : Optics Pro.
Pendant toute la durée de la thèse, une analyse détaillée de l'état-de-l'art du débruitage et des nouveautés a été menée, menant à l'implémentation open-source de deux méthodes emblématiques de débruitage. Parallèlement, les besoins de DxO Labs en terme de qualité image ont été précisément définis, ce qui a mené à lélaboration d'une nouvelle méthode académique de débruitage, surclassant celles déjà existantes, la méthode NL-Bayes. Son implémentation au cœur du code d'Optics Pro a permis de développer une approche multi-échelle et dépendante du signal. Cette méthode est implémentée dans la dernière version d'Optics Pro 9 de 2013 sous le nom de PRIME, qui a reçu un accueil médiatique très positif. Finalement, en étroite collaboration avec Miguel Colom et m'appuyant sur ses travaux d'estimation de bruit, j'ai développé un algorithme de débruitage aveugle permettant de débruiter efficacement toute sorte d'images naturelles. Cette dernière méthode est quasiment pionnière en la matière, puisque l'état-de-l'art antérieur dans ce domaine se réduisait à une unique méthode.

Abstract:

This CIFRE thesis is part of a close and long-standing collaboration between DxO Labs and CMLA. The main goal of this collaboration is to maintain a technological awareness in digital image processing, and to bring academic solutions to concrete problems raised by DxO Labs.
The aim of my thesis was to establish the state of the art in image denoising, with in view the development of a new generation of raw images denoising algorithms into the flagship product of DxO Labs: Optics Pro.
I performed a detailed analysis of state of the art denoising methods leading to the open-source implementation of two emblematic denoising algorithms. Simultaneously, the needs of DxO Labs in term of image quality were precisely defined. This led in a second phase to the conception of a new academic denoising method, NL-Bayes, outperforming existing ones. In a third phase, its implementation into the code of Optics Pro allowed us to develop a multiscale and signal-dependent approach. The result -renamed PRIME- is currently available in the last version of Optics Pro 9, 2013, with excellent press reviews. Finally, in a close collaboration with Miguel Colom, who developed a single image noise estimation, I developed a multiscale blind denoising algorithm, which gives convincing results on most natural noisy images. This last method is rather pioneering, as the state of the art in blind denoising was reduced to only one method.

Press Review of PRIME (Optics pro v9):

- *Cardinal Photo (David Cardinal, 2013):* "it was with great interest that I read about DxO's new PRIME noise reduction [...] it is far superior."
- *Life After Photoshop (Rod Lawton, 2013):* "DxO Optics Pro has excellent high ISO noise reduction", "as good as DxO says it is."
- *Experts Graphistes (Daniel Barrios, 2013):* DxO Optics Pro "traite le bruit très efficacement et donne un résultat tout à fait époustouflant."
- *La Nouvelle République (Jean-Christophe Solon, 2013):* "DxO Optics Pro 9 s'impose comme un incontournable." "L'essayer c'est l'adopter."
- *Mac 4 Ever (Arnaud, 2013):* "Le module de débruitage Prime est ce que nous avons vu de meilleur à ce jour en la matière."
- *Alpha-Numérique (Patrick Moll, 2013):* "La technologie PRIME [...] offre une réduction du bruit d'une qualité exceptionnelle qui laisse les logiciels concurrents bien loin derrière."

Remerciements

Je remercie tout particulièrement Jean-Michel Morel pour m'avoir dirigé ces trois dernières années. J'ai beaucoup aimé travailler avec lui, et j'ai énormément appris à ses côtés. Ses réunions de travail qui dérivaient souvent sur des sujets de discussions philosophiques ou de culture générale me manqueront beaucoup. Merci d'avoir été présent et disponible quand il le fallait tout en me laissant beaucoup de latitude dans l'organisation de mon travail.
Je tiens à remercier Frédéric Cao pour son encadrement à DxO, et notamment pour toutes les bonnes idées qu'il a su m'apporter lors du développement de PRIME. Sans son aide, PRIME ne serait sans doute jamais sorti dans la v9 d'Optics Pro. Je remercie aussi Luis Alvarez, Patrick Pérez et Javier Portilla d'avoir bien voulu être mes rapporteurs. Merci de plus à Andrés Almansa, Luis Alvarez, Frédéric Cao, Frédéric Guichard, Jean-Michel Morel, Patrick Pérez et Javier Portilla pour leur participation au jury. Je tiens à remercier tous les employés de DxO que j'ai rencontrés, avec une pensée particulière pour Karima qui a toujours su m'écouter et m'aider pour toutes les démarches administratives et la gestion chaotique de mes jours de congés, Benoit, Raffi, Clément et tous ceux du 3N pour avoir su m'aider à porter mon code dans celui d'Optics Pro, et soutenu malgré toutes les erreurs de compilations. Merci à Noémie et Guillaume pour avoir créé la pause "Mots Croisés" et partagé avec moi toutes leurs frustrations (et joies) de stagiaires (vous serez toujours, pour moi, les meilleurs stagiaires). Un grand merci à Wolf et Thomas pour avoir bien voulu m'accueillir au sein de l'équipe embarquée (promis Wolf, un jour je saurai effectuer un *commit* sans tout casser), et pour m'avoir toujours considéré comme l'un des vôtres malgré mes déménagements et éloignements successifs au sein du 2N.
Je remercie toutes les personnes que j'ai eu la chance de croiser au CMLA, en particulier Virginie, Véronique et Micheline pour leur efficacité et leur soutien moral, ainsi qu'à tous les participants du GTTI pour m'avoir entendu présenter la Noise Clinic pour la vingtième fois, merci à Toni pour m'avoir aidé à développer NL-Bayes, Yves et Gabriele pour m'avoir accompagné à Hong-Kong. Merci à Enric, Nicola et Gabriele pour m'avoir dépanné avec enthousiasme lorsque mon linux crashait pour la 4ème fois de l'année, Nicolas (Limare) pour m'avoir soutenu et conseillé lors de ma recherche constante d'accélération de mes codes et Nicolas (Pajor) pour m'avoir fourni d'aussi parfaits ordinateurs. Je tiens également à remercier plus particulièrement Miguel pour toute son aide, sans qui aucune de mes démo IPOL n'aurait pu voir le jour. Merci de m'avoir soutenu et conseillé lors de nos innombrables sessions de débuggage de mes codes, et surtout pour m'avoir aidé à développer la Noise Clinic tout au long de ces 8 longues versions. Et merci pour toutes nos discussions passionnante en hispano-franco-anglais.
Merci au thésard anonyme qui m'a conseillé d'utiliser dropbox: sans toi, j'aurai dû recommencer à zéro trois fois ma thèse. Qui a dit que Linux ne crashait jamais ?
Enfin, comme la vie de thésard ne se résume pas uniquement au travail, j'aimerais remercier la LIKA pour m'avoir offert un défouloir parfait lors de matchs et spectacles d'improvisations déchaînées. Merci à mes amis Flavien, Allan, Warren et Timothée de m'avoir écouté parler d'imagerie numérique à longueur de temps sans jamais vous plaindre, ainsi que Benjamin pour avoir bien voulu être mon stagiaire et pour tout ce dont je lui suis redevable. Merci à mes parents pour m'avoir fourni un soutien constant malgré l'éloignement, et un merci spécial à Mélissa pour m'avoir supporté durant cette dernière année, malgré mes horaires anarchiques et mes accès de rage de travail impromptus au beau milieu de la nuit.

Contents

Preface

Numerical images are nowadays everywhere in our daily life: on computers, smartphones, tablets. Almost all images that we are confronted to, from the vacation pictures of our friends to advertising in the subway, including instagram pictures or facebook profiles, are numerical images. Moreover, as numerical images are now mainstream, a lot of applications are developed, such as Google street view, pictures of Mars by the rover Curiosity, 3D cinema, face recognition, car detection, stereo vision, enhanced reality, and so on. And all of these applications have at least one thing in common: they need good quality images with as little noise as possible.
Denoising is one of the most crucial issue for the quality of numerical images, and it has to be done before anything else. But if we want to denoise correctly an image, we need first to understand where the noise comes from.

Part I: Denoising Methods

Introduction

Chapter 1 introduces some generic knowledge about the noise, how it appears and how one can try to deal with it. It will also briefly introduce the notion of "patch based" denoising algorithm, before being more developed in chapter 2. At the end of the chapter we will see that it is possible to focus the research on the general case of white Gaussian noise.

Denoising Principles

After having introduced the source and the nature of the noise, we will review in chapter 2 the main algorithmic principles which have been proposed for noise removal. All of them use of course a model for the noise, which in part I will always be a white Gaussian noise. More interestingly, each principle implies a model for the ideal noiseless image. The Bayesian principle is coupled with a Gaussian (or a mixture of Gaussians) model for noiseless patches. Transform thresholding assumes that most image coefficients are high and sparse in a given well-chosen orthogonal basis, while noise remains white (and therefore with homoscedastic coefficients in any orthogonal basis). Sparse coding assumes the existence of a dictionary of patches on which most image patches can be decomposed with a sparse set of coefficients. Finally the averaging principle relies on an image self-similarity assumption. Thus four considered denoising principles are:

- Bayesian patch-based methods (Gaussian patch model), which we illustrate by NL-Bayes (see chapter 12);
- transform thresholding (sparsity of patches in a fixed basis), which we illustrate by BM3D (see chapter 11);
- sparse coding (sparsity on a learned dictionary), which we illustrate by K-SVD (see chapter 10);
- pixel averaging and block averaging (image self-similarity, which is used in neighborhood filters and NL-means, see chapter 2).

As we will see in chapter 2, the current state of the art denoising recipes are actually a smart combination of *all* of these ingredients.

Generic Tools

Chapter 3 describes three generic tools that permit to increase the performance of *any* denoising principle. We shall illustrate them on DCT denoising. Starting from the application of a simple DCT transform threshold, the three generic tools (uniform and weighted aggregation, iteration and "oracle" filters, color space transform) will be applied successively. We shall observe a dramatic improvement of the denoising performance, as shown in image 1.

Figure 1: From top to bottom and left to right: crop of the noisy input image ($\sigma = 25$), and denoised images by sliding DCT thresholding filter and incrementally adding use of a $Y_oU_oV_o$ colour system, uniform aggregation, variance based aggregation and iteration with the "oracle" given by the first step. The corresponding PSNR are 26.85, 27.33, 30.65, 30.73, 31.25.

This observation is valid for all denoising principles, as we will see in chapter 4. Those three principles are shared by almost all state of the art algorithms (see in particular chapters 10, 11 and 12) with the notable exception of DDID which does not use the aggregation tool but only the other two (see section 4.10).

Detailed Analysis

In the long following chapter 4, a detailed description and analysis of ten denoising methods (K-SVD, BM3D, NL-Bayes, NL-means, BLS-GSM, NLDD, PLOW, EPLL, PLE, and Shotgun NL-means) is provided. The first six methods in addition to the sliding DCT filter specified in chapter 3, for which reliable faithful implementations are available, will also be compared in chapter 5. Moreover, the first three will be more extensively studied respectively in chapters 10, 11, and 12. This analysis shows that we are probably close to understanding digital images at a "patch" scale.

As shown in section 4.4, the mathematical and experimental evidence of two recent articles suggests that we might even be close to the best attainable performance in image denoising ever. This suspicion is supported by a remarkable convergence of all analyzed methods, as one can see on PSNR tables shown in chapter 5. They certainly converge in performance. We intend to demonstrate that, under different formalisms, their methods are almost equivalent. Working in the 64-dimensional "patch space", all recent methods estimate local "sparse models" and restore a noisy patch by finding its likeliest interpretation knowing the noiseless patches.

σ	NLDD	NL-Bayes	BM3D	BLS-GSM	K-SVD	NL-means	DCT denoising
2	46.06	**46.80**	46.34	46.02	45.09	45.46	45.92
5	41.64	**42.16**	41.64	41.18	41.36	40.49	41.08
10	38.19	**38.29**	38.10	37.48	37.60	37.05	37.45
20	**35.73**	34.84	34.60	33.64	34.15	33.27	33.52
30	**33.28**	32.76	32.56	32.04	31.79	31.21	31.09
40	**32.31**	31.61	31.07	30.62	30.14	29.63	29.13

Table 1: PSNR table showing averaging results on six noise-free images for different values of standard deviation. Only the three first digits are actually significant; the last one may vary with different white noise realizations.

Comparison

In chapter 5 we compare the following "state of the art" denoising algorithms introduced in chapters 2 and 4: the sliding DCT filter, the wavelet neighborhood Gaussian scale mixture (BLS-GSM) algorithm, the classical vector valued NL-means, the BM3D algorithm, the K-SVD denoising method, the Non-local Bayes algorithm and the NLDD algorithm. These algorithms have been chosen for two reasons. First they have a public and completely transparent code available, which is in agreement with their present description. Second, they all represent distinct denoising principles and therefore illustrate the methodological progress and the diversity of denoising principles.

The comparison, using when possible the public IPOL algorithms `http://www.ipol.im/`, will be based on four quantitative and qualitative criteria: the visualization of the *method noise*, namely the part of the image that the algorithm has taken out as noise, the visual verification of the *noise to noise* principle, and the *mean square error* or *PSNR* tables. Typical results may be found in table 1.

Last but not least the *visual quality* of the restored images must of course be the ultimate criterion. One typical comparison of visual results is shown on Image 2.

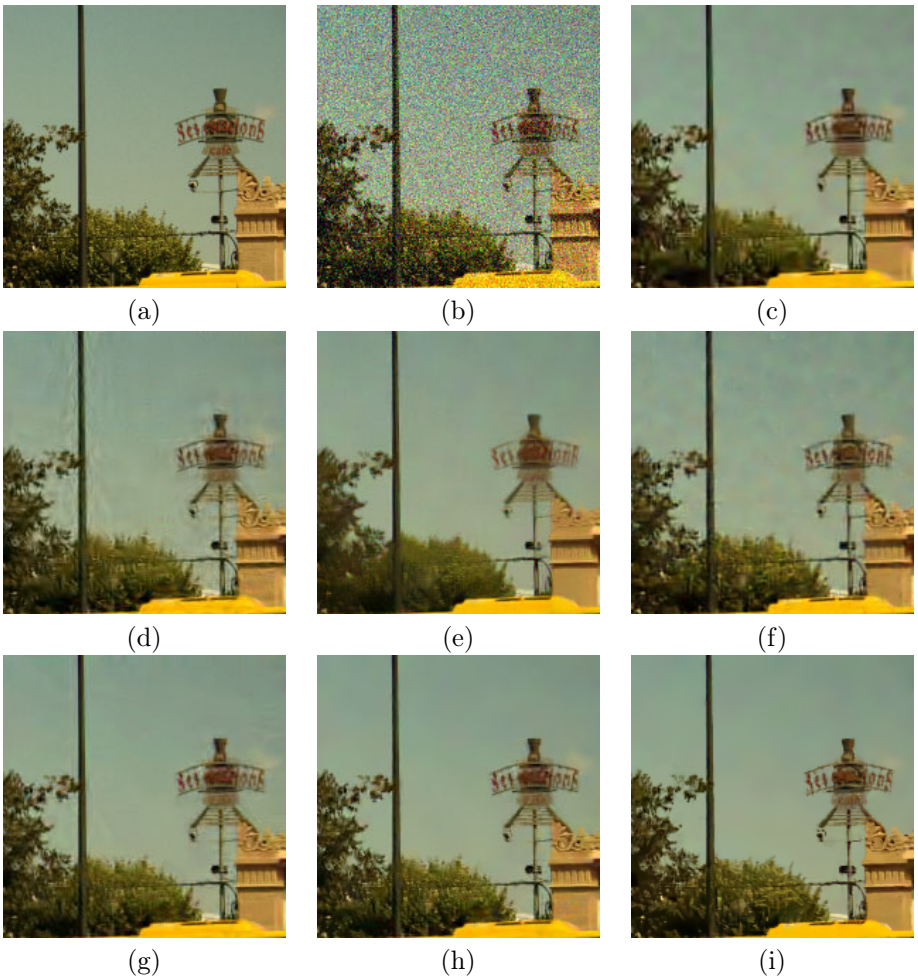


Figure 2: Comparison of visual quality. The noisy image was obtained adding a Gaussian white noise of standard deviation 30. From top to bottom and left to right: original, noisy, DCT sliding window, BLS-GSM, NL-means, K-SVD, BM3D, Non-local Bayes and NLDD.

It is easily seen that a single criterion is not enough to judge a restoration method. A good denoising solution must have a high performance under all mentioned criteria.

Synthesis

After having presented the vast and prolific field of denoising (chapter 1), some common generic tools (chapter 3) shared by state of the art denoising algorithms (chapter 4) and compared them both visually and in term of PSNR (chapter 5), it is time now to summarize our knowledge and classify those methods according to their principles, their use of patches, the size of those patches, the tools they used, and their complexity. Therefore **chapter 6** concludes this review of denoising methods in the case of white Gaussian noise, before starting to work in a more generic case in part II.

Part II: Noise Clinic

White Noise

As shown in part I, denoising methods require a noise model and an image model. As explained in chapter 1, natural images have a signal-dependent noise. Therefore, the Gaussian noise model used in part I is no longer valid when it comes to work on real images. However, it is relatively easy to obtain a signal-dependent noise model. As will be explained in **chapter 7**, it is even possible to estimate it from a single noisy image. Obtaining a convincing statistical image model is quite another story.

Generic Noise

Whereas it is possible to estimate a signal-dependent "white" noise from a single image as presented in **chapter 7**, this estimation is not accurate enough for blind denoising applications on any kind of images, especially JPEG images.

In chapter 8 we propose a non-parametric method to estimate both intensity and frequency dependent noise which obtains the noise model from the noisy image itself. Since demosaicing is the first step of the digital camera processing chain, most of the images contain correlated noise and hence the interest of estimating not only signal-dependent, but also frequency-dependent noise. The method gives the noise model for patches and therefore can be used as the input to a patch-based denoiser. In particular, this proposed method applies to cases where no access is granted to the image noise model, in particular to scanned photographs and JPEG images.

In order to evaluate the accuracy of the method, we validate it by comparing its estimations to the ground-truth noise curves for both raw and JPEG-encoded images and also by visual inspection of the denoising results of real images that present correlated noise, especially at the low frequencies. The proposed method overcomes the state-of-the-art.

Noise Clinic

Arguably several thousands papers are dedicated to image denoising. As explained in part I, most papers assume a fixed noise model, mainly white Gaussian or Poissonian. This assumption is only valid for raw images.

Yet in most images handled by the public and even by scientists, the noise model is imperfectly known or unknown. End users only dispose of the result of a complex image processing chain effectuated by uncontrolled hardware and software (and sometimes by chemical means). For such images, we have shown in chapter 8 that recent progress in noise estimation permits to estimate from a single image a noise model which is simultaneously signal and frequency dependent.

As most of denoising algorithms mainly focus on Gaussian Noise (chapter 4) and then only work for signal-independent noise, they need to be adapted to be able to deal with signal-dependent noise. The NL-Bayes algorithm described in details in chapter 12 has both advantages to give really good results without providing any artefacts and to be simple enough to provide signal-dependent denoising with only small modifications.

Therefore we propose in chapter 9 a multiscale denoising algorithm -based on NL-Bayes- adapted to this broad noise model. This leads to a blind denoising algorithm which we demonstrate on real JPEG images and on scans of old photographs for which the formation model is unknown. The consistency of this algorithm is also verified on simulated distorted images. This algorithm is finally compared to the unique state of the art blind denoising method: blind BLS-GSM. One typical result of the Noise Clinic is shown in Figure 3.

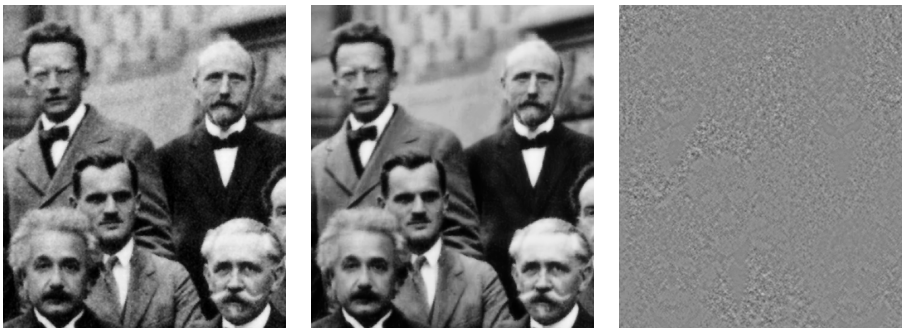


Figure 3: Blind denoising on "Solvay conference, 1927", using three scales. From left to right, top to bottom : crop of the input noisy image, crop of the output denoised image, crop of the difference image.

Part III: Reproducible Research Contributions

In order to make a fair comparison of denoising algorithms, as shown in chapter 5, we need reliable open source codes of the corresponding methods compared. For some emblematic state of the art methods, such implementations were not available. Part III regroups the detailed analysis of three methods, for which both theoretical analysis and open-source code are now available on IPOL.

K-SVD

As described in chapter 4, K-SVD is a signal representation method which, from a set of signals, can derive a dictionary able to approximate each signal with a sparse combination of the atoms. As this method is emblematic of dictionary-based denoising methods, we present in chapter 10 a detailed description of its theory and a precise analysis of its parameters which ends with a reliable implementation.

BM3D

A brief description of the well known denoising method BM3D was presented in chapter 4. This state of the art method is emblematic of patch-based methods. It was (and still is) the absolute reference for comparisons to any new denoising algorithm. For those reasons, chapter 11 will focus on a detailed description of its theory and practicable implementation. Its parameters will also be precisely analysed. Finally, we produced a reliable and open-source implementation of this method.

NL-Bayes

Chapter 4 has provided a brief description of the NL-Bayes algorithm by focusing on its genesis. This method is one of the final product of this thesis, and outperforms actual state of the art algorithms. This is why chapter 12 will present in details its implementation, as well as an analysis of the parameters of the method. As for the other methods presented in chapters 10 and 11, an open source reliable implementation is available on line on IPOL (Image Processing On Line).

Publications

During this PhD, I published and submitted as main author or co-author some papers, referenced here. Starting with the detailed study of two emblematic state of the art denoising algorithms

(K-SVD in [80] with the collaboration of Arthur Leclaire and BM3D in [74]) and a review of denoising in [77], in collaboration with Miguel Colom, Antoni Buades and Jean-Michel Morel, I then published two articles about NL-Bayes, developed as joint work with Antoni Buades and Jean-Michel Morel in [75] and [76]. From my collaboration with Miguel Colom about noise estimation, that we submitted in [33] and published in [32] with Antoni Buades and Jean-Michel Morel, it has led us to the development of the Noise Clinic, submitted in [79] and [78]. More recently, I contributed to the development of NLDD in collaboration with Nicola Pierazzo, Martin Rais, Jean-Michel Morel and Gabriele Facciolo, which is submitted in [108]. Finally, a very recent paper about image denoising [31] has been submitted in the ICM conference in collaboration with Miguel Colom, Gabriele Facciolo, Nicola Pierazzo, Martin Rais, Yi-Qing Wang and Jean-Michel Morel.

Part I

Denoising methods

Digital images are matrices of regularly spaced pixels, each containing a photon count. This photon count is a stochastic process due to the quantic nature of light. As explained in chapter 1, it follows that all images are noisy. Ever since digital images exist, numerical methods have been proposed to improve the signal to noise ratio. Such "denoising" methods require a noise model and an image model. It is relatively easy to obtain a noise model. As will be explained in part II, it is even possible to estimate it from a single noisy image.

Obtaining a convincing statistical image model is quite another story. Images reflect the world and are as complex as the world. Thus, any progress in image denoising signals a progress in our understanding of image statistics. The present part contains an analysis of ten recent state of the art methods in chapter 4. This analysis shows that we are probably close to understanding digital images at a "patch" scale.

The mathematical and experimental evidence of two recent articles suggests that we might even be close to the best attainable performance in image denoising ever. This suspicion is supported by a remarkable convergence of all analyzed methods. They certainly converge in performance as shown in chapter 5. We intend to demonstrate that, under different formalisms, their methods are almost equivalent. Working in the 64-dimensional "patch space", all recent methods estimate local "sparse models" and restore a noisy patch by finding its likeliest interpretation knowing the noiseless patches.

The story will be told in an ordinate manner. Denoising methods are complex and have several indispensable ingredients. Chapter 2 describes the four main image models used for denoising: the Markovian-Bayesian paradigm, the linear transform thresholding, the so-called image sparsity, and image self-similarity hypothesis. The performance of all methods depends on three generic tools: colour transform, aggregation, and an "oracle" step. Their recipes will also be given in chapter 3. These preparations will permit to present, in a unified terminology, the complete recipes of ten different state of the art patch-based denoising methods. Three quality assessment recipes for denoising methods will also be proposed in chapter 5 and applied to compare all methods. This part presents an ephemeral state of the art in a burgeoning subject, but many of the presented recipes will remain useful. Most denoising recipes can be tested directly on any digital image at *Image Processing On Line*, `http://www.ipol.im/`.

Chapter 1

Introduction: denoising methods and noise

Numerical images are nowadays everywhere in our daily life: on computers, smartphones, tablets. Almost all images that we are confronted to, from the vacation pictures of our friends to advertising in the subway, including instagram pictures or facebook profiles, are numerical images. Moreover, as numerical images are now mainstream, a lot of applications are developed, such as Google street view, pictures of Mars by the rover Curiosity, 3D cinema, face recognition, car detection, stereo vision, enhanced reality, and so on. And all of these applications have at least one thing in common: they need good quality images with as little noise as possible.

Denoising is one of the most crucial issue for the quality of numerical images, and it has to be done before anything else. But if we want to denoise correctly an image, we need first to understand where the noise comes from.

This first chapter introduces some generic knowledge about the noise, how it appears and how one can try to deal with it. It will also briefly introduce the notion of "patch based" denoising algorithm, before being more developed in chapter 2. At the end of this chapter we will see that it is possible to focus the research on the general case of white Gaussian noise.

1.1 Introduction

Digital images are matrices of regularly spaced pixels, each containing a photon count. This photon count is a stochastic process due to the quantic nature of light. It follows that all images are noisy. Ever since digital images exist, numerical methods have been proposed to improve the signal to noise ratio.

Most digital images and movies are currently obtained by a CCD device. The value $\tilde{u}(\mathbf{i})$ observed by a sensor at each pixel $\mathbf{i}$ is a Poisson random variable whose mean $u(\mathbf{i})$ would be the ideal image. The difference between the observed image and the ideal image $\tilde{u}(\mathbf{i}) - u(\mathbf{i}) = n(\mathbf{i})$ is called "shot noise". The standard deviation of the Poisson variable $\tilde{u}(\mathbf{i})$ is equal to the square root of the number of incoming photons $\tilde{u}(\mathbf{i})$ in the pixel captor $\mathbf{i}$ during the exposure time. The Poisson noise n adds up to a thermal noise and to an electronic noise which are approximately additive and white. On a motionless scene with constant lighting, $u(\mathbf{i})$ can be approached by simply accumulating photons for a long exposure time, and by taking the temporal average of this photon count, as illustrated in figure 1.1.

Accumulating photon impacts on a surface is therefore the essence of photography. The first Nicéphore Niépce photograph [25] was obtained after an eight hours exposure. The problem of a long exposure is the variation of the scene due to changes in light, camera motion, and incidental motions of parts of the scene. The more these variations can be compensated, the longer the exposure can be, and the more the noise can be reduced. If a camera is set to a long exposure

time, the photograph risks motion blur. If it is taken with short exposure, the image is dark, and enhancing it reveals the noise.

A recently available solution is to take a burst of images, each with short-exposure time, and to average them after registration. This technique, illustrated in Fig. 1.1, was evaluated recently in a paper that proposes fusing bursts of images taken by cameras [19]. This paper shows that the noise reduction by this method is almost perfect: fusing m images reduces the noise by a $\sqrt{m}$ factor.

It is not always possible to accumulate photons. There are obstacles to this accumulation in astronomy, biological imaging and medical imaging. In day to day images, the scene is moving, which limits the exposure time. The main limitations to any imaging system are therefore the noise and the blur. In this review, experiments will be conducted on photographs of scenes taken by normal cameras. Nevertheless, the image denoising problem is a common denominator of all imaging systems.

A naive view of the denoising problem would be: how to estimate the ideal image, namely the mean $u(\mathbf{i})$, given only one sample $\tilde{u}(\mathbf{i})$ of the Poisson variable? The best estimate of this mean is of course this unique sample $\tilde{u}(\mathbf{i})$. Getting back a better estimate of $u(\mathbf{i})$ by observing only $\tilde{u}(\mathbf{i})$ is impossible. Getting a better estimate by using also the rest of the image is obviously an ill-posed problem. Indeed, each pixel receives photons coming from different sources.

Nevertheless, a glimpse of a solution comes from image formation theory. A well-sampled image u is band-limited [124]. Thus, it seems possible to restore the band-limited image u from its degraded samples $\tilde{u}$, as was proposed in 1966 in [61]. This classic Wiener-Fourier method consists in multiplying the Fourier transform by optimal coefficients to attenuate the noise. It results in a convolution of the image with a low-pass kernel.

From a stochastic viewpoint, the band-limitedness of u also implies that values $\tilde{u}(\mathbf{j})$ at neighboring pixels $\mathbf{j}$ of a pixel $\mathbf{i}$ are positively correlated with $\tilde{u}(\mathbf{i})$. Thus, these values can be taken into account to obtain a better estimate of $u(\mathbf{i})$. These values being nondeterministic, Bayesian approaches are relevant and have been proposed as early as 1972 in [121].

In short, there are two complementary early approaches to denoising, the Fourier method, and the Bayesian estimation.

The Fourier method has been extended in the past thirty years to other linear space-frequency transforms such as the windowed DCT [135] or the many wavelet transforms [65].

Being first parametric and limited to rather restrictive Markov random field models [58], the Bayesian method are becoming non-parametric. The idea for the recent non parametric Markovian estimation methods is a now famous algorithm to synthesize textures from examples [52]. The underlying Markovian assumption is that, in a textured image, the stochastic model for a given pixel $\mathbf{i}$ can be predicted from a local image neighborhood P of $\mathbf{i}$, which we shall call "patch".

The assumption for recreating new textures from samples is that there are enough pixels $\mathbf{j}$ similar to $\mathbf{i}$ in a texture image $\tilde{u}$ to recreate a new but similar texture u. The construction of u is done by nonparametric sampling, amounting to an iterative copy-paste process. Let us assume that we already know the values of u on a patch P surrounding partially an unknown pixel $\mathbf{i}$. The Efros-Leung [52] algorithm looks for the patches $\tilde{P}$ in $\tilde{u}$ with the same shape as P and resembling P. Then a value $u(\mathbf{i})$ is sorted among the values predicted by $\tilde{u}$ at the pixels resembling $\mathbf{j}$. Indeed, these values form a histogram approximating the law of $u(\mathbf{i})$. This algorithm goes back to Shannon's theory of communication [124], where it was used for the first time to synthesize a probabilistically correct text from a sample.

As was proposed in [16], an adaptation of the above synthesis principle yields an image denoising algorithm. The observed image is the noisy image $\tilde{u}$. The reconstructed image is the denoised image $\hat{u}$. The patch is a square centered at $\mathbf{i}$, and the sorting yielding $u(\mathbf{i})$ is replaced by a weighted average of values at all pixels $\tilde{u}(\mathbf{j})$ similar to $\mathbf{i}$. This simple change leads to the "non-local means" algorithm, which can therefore be sketched in a few rows.

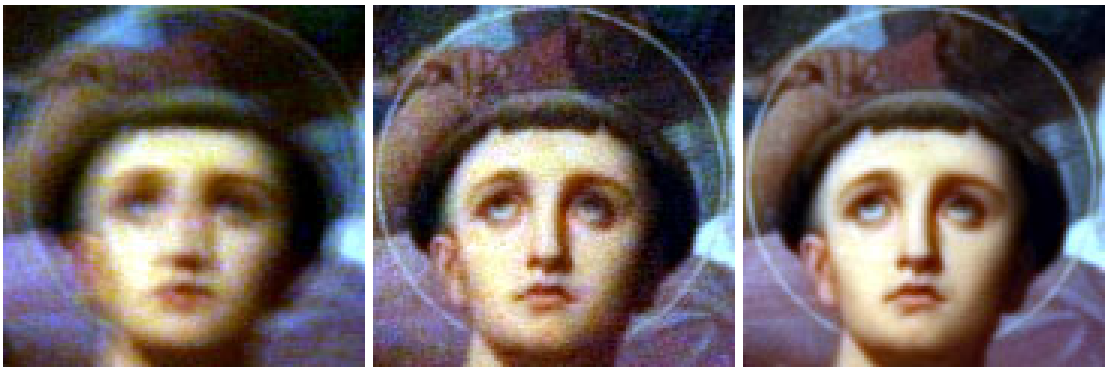


Figure 1.1: From left to right: (a) one long-exposure image (time=0.4 s, ISO=100), one of 16 short-exposure images (time=1/40 s, ISO=1600) and their average after registration. The long exposure image is blurry due to camera motion. (b) The middle short-exposure image is noisy. (c) The third image is about **four times** less noisy, being the result of averaging 16 short-exposure images. From [19].

Algorithm 1 Non-local means algorithm

Input: noisy image $\tilde{u}$, σ noise standard deviation.
Output: denoised image $\hat{u}$.
Set parameter $\kappa \times \kappa$: dimension of patches.
Set parameter $\lambda \times \lambda$: dimension of research zone in which similar patches are searched.
Set parameter C.
for each pixel **i do**
  Select a square reference sub-image (or "patch") $\tilde{P}$ around **i**, of size $\kappa \times \kappa$.
  Call $\hat{P}$ the denoised version of $\tilde{P}$ obtained as a weighted average of the patches $\tilde{Q}$ in a square neighborhood of **i** of size $\lambda \times \lambda$. The weights in the average are proportional to $w(\tilde{P}, \tilde{Q}) = e^{-\frac{d^2(\tilde{P},\tilde{Q})}{C\sigma^2}}$ where $d(\tilde{P}, \tilde{Q})$ is the Euclidean distance between patches $\tilde{P}$ and $\tilde{Q}$.
end for
Aggregation: recover a final denoised value $\hat{u}(\mathbf{i})$ at each pixel **i** by averaging all values at **i** of all denoised patches $\hat{Q}$ containing **i**

It was also proved in [16] that the algorithm gave the best possible mean square estimation if the image was modeled as an infinite stationary ergodic spatial process (see sec. 4.1 in chapter 4 for an exact statement). The algorithm was called "non-local" because it uses patches $\tilde{Q}$ that are far away from $\tilde{P}$, and *even patches taken from other images*. NL-means was not the state of the art denoising method when it was proposed. As we shall see in the comparison section 8.4.1, the 2003 Portilla et al. [114] algorithm described in sec. 4.6 has a better PSNR performance. But quality criteria show that NL-means creates less artifacts than wavelet based methods. This may explain why patch-based denoising methods have flourished ever since. By now, 1500 papers have been published on nonlocal image processing. Patch-based methods seem to achieve the best results in denoising. Furthermore, the quality of denoised images has become excellent for moderate noise levels. Patch-based image restoration methods are used in many commercial software.
An exciting recent paper in this exploration of nonlocal methods raises the following claim [86]: *For natural images, the recent patch-based denoising methods might well be close to optimality.* The authors use a set of 20000 images containing about 10^{10} patches. This paper provides a second answer to the question of absolute limits raised in [23], "Is denoising dead?". The Cramer-Rao type lower bounds on the attainable RMSE performance given in [23] are actually more optimistic: they allow for the possibility of a significant increase in denoising performance. The two types of performance bounds considered in [86] and [23] address roughly the same class of

patch-based algorithms. It is interesting to see that these same authors propose denoising methods that actually approach these bounds, as we shall see in chapter 4.
The denoising method proposed in [86] is actually based on NL-means (algorithm 1), with the adequate parameter C to account for a Bayesian linear minimum mean square estimation (LMMSE) estimation of the noisy patch given a database of known patches. The only and important difference is that the similar patches Q are found on a database of 10^{10} patches, instead of on the image itself. Furthermore, by a simple mathematical argument and intensive simulations on the patch space, the authors are able to approach *the best average estimation error* which will *ever be attained by any patch-based denoising algorithm* (see sec. 4.4.)
These optimal bounds are nonetheless obtained on a somewhat restrictive definition of patch-based methods. A patch-based algorithm is understood as an algorithm that denoises each pixel by using the knowledge of: a) the patch surrounding it, and b) the probability density of all existing patches in the world. It turns out that state of the art patch-based denoising algorithms use more information taken in the image than just the patch. For example, most algorithms use the obvious but powerful trick to denoise all patches, and then to *aggregate* the estimation of all patches containing a given pixel to denoise it better. Conversely, these algorithms generally use much less information than a universal empirical law for patches. Nevertheless, the observation that at least one algorithm, BM3D [35] might be arguably very close to the best predicted estimation error is enlightening. Furthermore, doubling the size of the patch used in the [86] paper would be enough to cover the aggregation step. The difficulty is to get a faithful empirical law for 16×16 patches.
The "convergence" of all algorithms to optimality will be corroborated here by the thorough comparison of ten recent algorithms (chapter 5). These state of the art algorithms seem to attain a very similar qualitative and quantitative performance. Although they initially seem to rely on different principles, our final discussion will argue that these methods are equivalent.
Image restoration theory cannot be reduced to an axiomatic system, as the statistics of images are still a widely unexplored continent. Therefore, a complete theory, or a single final algorithm closing the problem are not possible. The problem is not fully formalized because there is no rigorous image model. Notwithstanding this limitation, rational recipes shared by all methods can be given, and the methods can be shown to rely on only very few principles. More precisely, this part will present the following recipes, and compare them whenever possible:

- the four denoising principles in competition (chapter 2);
- three techniques that improve every denoising method (chapter 3);
- ten complete and recent denoising algorithms. For these algorithms complete recipes will be given (chapter 4);
- three complementary and simple recipes to evaluate and compare denoising algorithms (chapter 5).
- several families of noise estimation techniques (chapter 7);

Using the three comparison recipes, seven emblematic or state of the art algorithms, based on reliable and public implementations, will be compared in chapter 5. This comparison is followed by a synthesis (chapter 6) hopefully demonstrating that, under very different names, the state of the art algorithms share the same principles.
Nevertheless, this convergence of results and techniques leaves several crucial issues unsolved. (This is fortunate, as no researcher likes finished problems.) With one exception, (the BLS-GSM algorithm, sec. 4.6), state of the art denoising algorithms are not multiscale. High noises and small noises also remain unexplored.
In a broader perspective, the success of image denoising marks the discovery and exploration of one of the first densely sampled high dimensional probability laws ever (numerically) accessible to mankind: the "patch space". For 8×8 patches, by applying a local PCA to the patches surrounding a given patch, one can deduce that this space has a dozen significant dimensions (the others being very thin). Exploring its structure, as was initiated in [81], seems to be the first step toward the statistical exploration of images. But, as we shall see, this local analysis of the *patch space* already enables state of the art image denoising.

1.2 Noise

Most digital images and movies are obtained by a CCD device and the main source of noise is the so-called *shot noise*. Shot noise is inherent to photon counting. The value $\tilde{u}(\mathbf{i})$ observed by a sensor at each pixel $\mathbf{i}$ is a Poisson random variable whose mean would be the ideal image. The standard deviation of this Poisson distribution is equal to the square root of the number of incoming photons $\tilde{u}(\mathbf{i})$ in the pixel captor $\mathbf{i}$ during the exposure time. This noise adds up to a thermal noise and to an electronic noise which are approximately additive and white.
For sufficiently large values of $\tilde{u}(\mathbf{i})$, $(\tilde{u}(\mathbf{i}) > 1000)$, the normal distribution $\mathcal{N}(\tilde{u}(\mathbf{i}), \sqrt{\tilde{u}(\mathbf{i})})$ with mean $\tilde{u}(\mathbf{i})$ and standard deviation $\sqrt{\tilde{u}(\mathbf{i})}$ is an excellent approximation to the Poisson distribution. If $\tilde{u}(\mathbf{i})$ is larger than 10, then the normal distribution still is a good approximation if an appropriate continuity correction is performed, namely $\mathbb{P}(\tilde{u}(\mathbf{i}) \leq a) \simeq \mathbb{P}(\tilde{u}(\mathbf{i}) \leq a + 0.5)$, where a is any non-negative integer.
Nevertheless, the pixel value is *signal dependent*, since its mean and variance depend on $\tilde{u}(\mathbf{i})$. To get back to the classic "white additive Gaussian noise" used in most researches on image denoising, a *variance-stabilizing transformation* can be applied: When a variable is Poisson distributed with parameter $\tilde{u}(\mathbf{i})$, its square root is approximately normally distributed with expected value of about $\sqrt{\tilde{u}(\mathbf{i})}$ and variance of about $1/4$. Under this transformation, the convergence to normality is faster than for the untransformed variable[1]. The most classic VST is the Anscombe transform [3] which has the form $f(u_0) = b\sqrt{u_0 + c}$.
The denoising procedure with the standard variance stabilizing transformation (VST) procedure follows three steps,

1. apply VST to approximate homoscedasticity;
2. denoise the transformed data;
3. apply an inverse VST.

Note that the inverse VST is not just an algebraic inverse of the VST, and must be optimized to avoid bias [95].
Consider any additive signal dependent noisy image, obtained for example by the Gaussian approximation of a Poisson variable explained above. Under this approximation, the noisy image satisfies $\tilde{u} \simeq \tilde{u} + g(\tilde{u})n$ where $n \simeq \mathcal{N}(0, 1)$. We can search for a function f such that $f(\tilde{u})$ has uniform standard deviation,

$$f(\tilde{u}) \simeq f(\tilde{u}) + f'(\tilde{u})g(\tilde{u})n.$$

Forcing the noise term to be constant, $f'(\tilde{u})g(\tilde{u}) = c$, we get

$$f'(\tilde{u}) = \frac{c}{g(\tilde{u})},$$

and integrating

$$f(\tilde{u}) = \int_0^{\tilde{u}} \frac{c\,dt}{g(t)}.$$

When a linear variance noise model is taken, this transformation gives back an Anscombe transform. Most classical denoising algorithms can also be adapted to signal dependent noise. This requires varying the denoising parameters at each pixel, depending on the observed value $\tilde{u}(\mathbf{i})$. Several denoising methods indeed deal directly with the Poisson noise. Wavelet-based denoising methods [103] and [73] propose to adapt the transform threshold to the local noise level of the Poisson process. Lefkimmiatis et al. [85] have explored a Bayesian approach without applying a VST. Deledalle et al., [41] argue that for high noise level it is better to adapt NL-means than to apply a VST. These authors proposed to replace the Euclidean distance between patches by a likelihood estimation taking into account the noise model. This distance can be adapted to

[1] See `http://en.wikipedia.org/wiki/Poisson_distribution`.

each noise model such as the Poisson, the Laplace or the Gamma noise [44], and to more complex (speckle) noise occurring in radar (SAR) imagery [45].
Nonetheless, dealing with a white uniform Gaussian noise makes the discussion on denoising algorithms far easier. The recent papers on the Anscombe transform [95] (for low count Poisson noise) and [54] (for Rician noise) argue that, when combined with suitable forward and inverse VST transformations, algorithms designed for homoscedastic Gaussian noise work just as well as ad-hoc algorithms signal-dependent noise models. This explains why in the rest of this paper the noise is assumed uniform, white and Gaussian, having previously applied, if necessary, a VST to the noisy image. This also implies that we deal with *raw* images, namely images as close as possible to the direct camera output before processing. Most reflex cameras, and many compact cameras nowadays give access to this raw image.
But there is definitely a need to denoise current image formats, which have undergone unknown alterations. For example, the JPEG-encoded images given by a camera contain a noise that has been altered by a complex chain of algorithms, ending with lossy compression. Noise in such images cannot be removed by the current state of the art denoising algorithms without a specific adaptation. The key is to have a decent noise model. For this reason, the fundamentals to estimate noise from a single image will be given in chapter 7.

Chapter 2

Four Denoising Principles

In this chapter, we will review the main algorithmic principles which have been proposed for noise removal. All of them use of course a model for the noise, which in part I will always be a white Gaussian noise. More interestingly, each principle implies a model for the ideal noiseless image. The Bayesian principle is coupled with a Gaussian (or a mixture of Gaussians) model for noiseless patches. Transform thresholding assumes that most image coefficients are high and sparse in a given well-chosen orthogonal basis, while noise remains white (and therefore with homoscedastic coefficients in any orthogonal basis). Sparse coding assumes the existence of a dictionary of patches on which most image patches can be decomposed with a sparse set of coefficients. Finally the averaging principle relies on an image self-similarity assumption. Thus four considered denoising principles are:

- Bayesian patch-based methods (Gaussian patch model), which we illustrate by NL-Bayes (see chapter 12 for a detailed technical description);
- transform thresholding (sparsity of patches in a fixed basis), which we illustrate by BM3D (see chapter 11 for a detailed technical description);
- sparse coding (sparsity on a learned dictionary), which we illustrate by K-SVD (see chapter 10 for a detailed technical description);
- pixel averaging and block averaging (image self-similarity).

As we will see in chapter 4, the current state of the art denoising recipes are actually a smart combination of *all* of these ingredients.
This chapter uses the results of joint work with Antoni Buades and Jean-Michel Morel.

2.1 Bayesian patch-based methods

Given u the noiseless ideal image and $\tilde{u}$ the noisy image corrupted with Gaussian noise of standard deviation σ so that

$$\tilde{u} = u + n, \tag{2.1}$$

the conditional distribution $\mathbb{P}(\tilde{u} \mid u)$ is

$$\mathbb{P}(\tilde{u} \mid u) = \frac{1}{(2\pi\sigma^2)^{\frac{M}{2}}} e^{-\frac{||u-\tilde{u}||^2}{2\sigma^2}}, \tag{2.2}$$

where M is the total number of pixels in the image.
In order to compute the probability of the original image given the degraded one, $\mathbb{P}(u \mid \tilde{u})$, we need to introduce a prior on u. In the first models [58], this prior was a parametric image model describing the stochastic behavior of a patch around each pixel by a Markov random field, specified by its Gibbs distribution. A Gibbs distribution for an image u takes the form

$$\mathbb{P}(u) = \frac{1}{Z} e^{-E(u)/T},$$

where Z and T are constants and E is called the energy function and writes

$$E(u) = \sum_{C \in \mathcal{C}} V_C(u),$$

where $\mathcal{C}$ denotes the set of cliques associated to the image and V_C is a potential function. The maximization of the *a posteriori* distribution writes by Bayes formula

$$\operatorname*{Arg\,max}_u \mathbb{P}(u \mid \tilde{u}) = \operatorname*{Arg\,max}_u \mathbb{P}(\tilde{u} \mid u)\mathbb{P}(u),$$

which is equivalent to the minimization of $-\log \mathbb{P}(u \mid \tilde{u})$,

$$\operatorname*{Arg\,min}_u \|u - \tilde{u}\|^2 + \frac{2\sigma^2}{T} E(u).$$

This energy writes as a sum of local derivatives of pixels in the image, thus being equivalent to a classical Tikhonoff regularization, [58] and [10].
Recent Bayesian methods have abandoned as too simplistic the global patch models formulated by an *a priori* Gibbs energy. Instead, the methods build local non parametric patch models learnt from the image itself, usually as a local Gaussian model around each given patch, or as a Gaussian mixture. The term "patch model" is now preferred to the terms "neighborhood" or "clique" previously used for the Markov field methods. In the nonparametric models, the patches are larger, usually 8×8, while the cliques were often confined to 3×3 neighborhoods. Given a noiseless patch P of u with dimension $\kappa \times \kappa$, and $\tilde{P}$ an observed noisy version of P, the same model gives by the independence of noise pixel values

$$\mathbb{P}(\tilde{P}|P) = c \cdot e^{-\frac{\|\tilde{P}-P\|^2}{2\sigma^2}} \tag{2.3}$$

where P and $\tilde{P}$ are considered as vectors with κ^2 components and $||P||$ denotes the Euclidean norm of P. Knowing $\tilde{P}$, our goal is to deduce P by maximizing $\mathbb{P}(P|\tilde{P})$. Using Bayes' rule, we can compute this last conditional probability as

$$\mathbb{P}(P|\tilde{P}) = \frac{\mathbb{P}(\tilde{P}|P)\mathbb{P}(P)}{\mathbb{P}(\tilde{P})}. \tag{2.4}$$

$\tilde{P}$ being observed, this formula can in principle be used to deduce the patch P maximizing the right term, viewed as a function of P. This is only possible if we have a probability model for P, and these models will be generally learnt from the image itself, or from a set of images. For example [24] applies a clustering method to the set of patches of a given image, and [145] applies it to a huge set of patches extracted from many images. Each cluster of patches is thereafter treated as a set of Gaussian samples. This permits to associate to each observed patch its likeliest cluster, and then to denoise it by a Bayesian estimation in this cluster. Another still more direct way to build a model for a given patch $\tilde{P}$ is to group the patches similar to $\tilde{P}$ in the image. Assuming that these similar patches are samples of a Gaussian vector yields a standard Bayesian restoration. We shall now discuss this particular case, where all observed patches are noisy.
Why Gaussian? As usual when we dispose of several observations but of no particular guess on the form of the probability density, a Gaussian model is adopted. In the case of the patches Q similar to a given patch P, the Gaussian model has some pertinence, as it is assumed that many contingent random factors explain the difference between Q and P: other details, texture, slight lighting changes, shadows, etc. The Gaussian model in presence of a combination of many such random and independent factors is heuristically justified by the central limit theorem. Thus, for good or bad, assume that the patches Q similar to P follow a Gaussian model with (observable, empirical) covariance matrix $\mathbf{C}_P$ and (observable, empirical) mean $\overline{P}$. This means that

$$\mathbb{P}(Q) = c.e^{-\frac{(Q-\overline{P})^t \mathbf{C}_P^{-1}(Q-\overline{P})}{2}} \tag{2.5}$$

From (12.2) and (12.4) we obtain for each observed $\tilde{P}$ the following equivalence of problems:

$$\begin{aligned}\max_P \ \mathbb{P}(P|\tilde{P}) &\Leftrightarrow \max_P \ \mathbb{P}(\tilde{P}|P)\mathbb{P}(P) \\ &\Leftrightarrow \max_P e^{-\frac{\|P-\tilde{P}\|^2}{2\sigma^2}} e^{-\frac{(P-\overline{P})^t \mathbf{C}_P^{-1}(P-\overline{P})}{2}} \\ &\Leftrightarrow \min_P \ \frac{\|P-\tilde{P}\|^2}{\sigma^2} + (P-\overline{P})^t \mathbf{C}_P^{-1}(P-\overline{P}).\end{aligned}$$

This expression does not yield an algorithm. Indeed, the noiseless patch P and the patches similar to P are not observable. Nevertheless, we can observe the noisy version $\tilde{P}$ and compute the patches $\tilde{Q}$ similar to $\tilde{P}$. An empirical covariance matrix can therefore be obtained for the patches $\tilde{Q}$ similar to $\tilde{P}$. Furthermore, using (12.1) and the fact that P and the noise n are independent,

$$\mathbf{C}_{\tilde{P}} = \mathbf{C}_P + \sigma^2\mathbf{I}; \qquad E\tilde{Q} = \overline{P}. \tag{2.6}$$

Notice that these relations assume that we searched for patches similar to $\tilde{P}$ at a large enough distance, to include all patches similar to P, but not too large either, because otherwise it can contain outliers. Thus the safe strategy is to search similar patches in a distance slightly larger than the expected distance caused by noise. If the above estimates are correct, our MAP (maximum *a posteriori* estimation) problem finally boils down by (12.6) to the following feasible minimization problem:

$$\max_P \ \mathbb{P}(P|\tilde{P}) \Leftrightarrow \min_P \ \frac{\|P-\tilde{P}\|^2}{\sigma^2} + (P-\overline{\tilde{P}})^t(\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I})^{-1}(P-\overline{\tilde{P}}).$$

Differentiating this quadratic function with respect to P and equating to zero yields

$$P - \tilde{P} + \sigma^2(\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I})^{-1}(P-\overline{\tilde{P}}) = 0.$$

Taking into account that $\mathbf{I} + \sigma^2(\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I})^{-1} = (\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I})^{-1}\mathbf{C}_{\tilde{P}}$, this yields

$$(\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I})^{-1}\mathbf{C}_{\tilde{P}}P = \tilde{P} + \sigma^2(C_{\tilde{P}} - \sigma^2\mathbf{I})^{-1}\overline{\tilde{P}}.$$

and therefore

$$\begin{aligned}P &= \mathbf{C}_{\tilde{P}}^{-1}(\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I})\tilde{P} + \sigma^2\mathbf{C}_{\tilde{P}}^{-1}\overline{\tilde{P}} \\ &= \tilde{P} + \sigma^2\mathbf{C}_{\tilde{P}}^{-1}(\overline{\tilde{P}} - \tilde{P}) \\ &= \overline{\tilde{P}} + \left[\mathbf{I} - \sigma^2\mathbf{C}_{\tilde{P}}^{-1}\right](\tilde{P} - \overline{\tilde{P}}) \\ &= \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I}\right]\mathbf{C}_{\tilde{P}}^{-1}(\tilde{P} - \overline{\tilde{P}})\end{aligned}$$

Thus we have proved that a restored patch $\hat{P}_1$ can be obtained from the observed patch $\tilde{P}$ by the one step estimation

$$\hat{P}_1 = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I}\right]\mathbf{C}_{\tilde{P}}^{-1}(\tilde{P} - \overline{\tilde{P}}), \tag{2.7}$$

which resembles a local Wiener filter.

Remark 1. *It is easily deduced that the expected estimation error is*

$$E||P - \hat{P}_1||^2 = Tr\left[\left(\boldsymbol{C}_P^{-1} + \frac{\boldsymbol{I}}{\sigma^2}\right)^{-1}\right].$$

In chapter 4, sections 4.2, 4.3, 4.4, 4.5, 4.6, 4.9 will examine not less than **six Bayesian algorithms** deriving patch-based denoising algorithms from variants of (2.7). The first question when looking at this formula is obviously how the matrix $\mathbf{C}_{\tilde{P}}$ can be learnt from the image itself. Each method proposes a different notion to learn the patch model.
Of course, other, non Gaussian, Bayesian models are possible, depending on the patch density assumption. For example [120] assumes a local exponential density model for the noisy data, and gives a convergence proof to the optimal (Bayes) least squares estimator as the amount of data increases.

2.2 Transform thresholding

Classical transform coefficient thresholding algorithms like the DCT or the wavelet denoising use the observation that images are faithfully described by keeping only their large coefficients in a well-chosen basis. By keeping these large coefficients and setting to zero the small ones, noise should be removed and image geometry kept. By any orthogonal transform, the coefficients of an homoscedastic de-correlated noise remain de-correlated and homoscedastic. For example the wavelet or the DCT coefficients of a Gaussian white noise with variance σ^2 remain a Gaussian diagonal vector with variance σ^2. Thus, a threshold on the coefficients at, say, 3σ removes most of the coefficients that are only due to noise. (The expectation of these coefficients is assumed to be zero.) The *sparsity* of image coefficients in certain bases is only an empirical observation. It is nevertheless invoked in most denoising and compression algorithms, which rely essentially on coefficient thresholds. The established image compression algorithms are based on the DCT (in the JPEG 1992 format) or, like the JPEG 2000 format [4], on biorthogonal wavelet transforms [26].
Let $\mathcal{B} = \{G_i\}_{i=1}^M$ be an orthonormal basis of $\mathbb{R}^M$, where M is the number of pixels of the noisy image $\tilde{U}$ (in staircase to recall that it is handled here as a vector). Then we have

$$\langle \tilde{U}, G_i \rangle = \langle U, G_i \rangle + \langle N, G_i \rangle \,, \tag{2.8}$$

where $\tilde{U}$, U and N denote respectively the noisy, original and noise images. We always assume that the noise values $N(\mathbf{i})$ are uncorrelated and homoscedastic with zero mean and variance σ^2. The following calculation shows that the noise coefficients in the new basis remain uncorrelated, with zero mean and variance σ^2:

$$\begin{aligned} E[\langle N, G_i \rangle \langle N, G_j \rangle] &= \sum_{\mathbf{r},\mathbf{s}=1}^{M} G_i(\mathbf{r}) G_j(\mathbf{s}) E[\mathbf{w}(\mathbf{r})\mathbf{w}(\mathbf{s})] \\ &= \langle G_i, G_j \rangle \sigma^2 = \sigma^2 \delta[j-i]. \end{aligned}$$

Each noisy coefficient $\langle \tilde{U}, G_i \rangle$ is modified independently and then the solution is estimated by the inverse transform of the new coefficients. Noisy coefficients are modified by multiplying by an attenuation factor $a(i)$ and the inverse transform yields the estimate

$$\mathbf{D}\tilde{U} = \sum_{i=1}^{M} a(i) \, \langle \tilde{U}, G_i \rangle \, G_i. \tag{2.9}$$

$\mathbf{D}$ is also called a *diagonal operator*. Noise reduction is achieved by attenuating or setting to zero small coefficients of order σ, assumedly due to noise, while the original signal is preserved by keeping the large coefficients. This intuition is corroborated by the following result.

Theorem 1. *The operator $\boldsymbol{D}_{inf}$ minimizing the mean square error (MSE),*

$$\boldsymbol{D}_{inf} = arg\min_{\boldsymbol{D}} E\{\|U - \boldsymbol{D}\tilde{U}\|^2\}$$

is given by the family $\{a(i)\}_i$, where

$$a(i) = \frac{|\langle U, G_i \rangle|^2}{|\langle U, G_i \rangle|^2 + \sigma^2}, \tag{2.10}$$

and the corresponding expected mean square error (MSE) is

$$E\{\|U - \boldsymbol{D}_{inf}\tilde{U}\|^2\} = \sum_{i=1}^{M} \frac{|\langle U, G_i \rangle|^2 \sigma^2}{|\langle U, G_i \rangle|^2 + \sigma^2}. \tag{2.11}$$

The previous optimal operator attenuates all noisy coefficients. If one restricts $a(i)$ to be 0 or 1, one gets a projection operator. In that case, a subset of coefficients is kept, and the rest are set to zero. The projection operator that minimizes the MSE under that constraint is obtained with

$$a(i) = \begin{cases} 1 & |\langle U, G_i \rangle|^2 \geq \sigma^2, \\ 0 & \text{otherwise} \end{cases}$$

and the corresponding MSE is

$$E\{\|U - \mathbf{D}_{inf}\tilde{U}\|^2\} = \sum_i \min(|\langle U, G_i \rangle|^2, \sigma^2). \tag{2.12}$$

A *transform thresholding* algorithm therefore keeps the coefficients with a magnitude larger than the noise, while setting the zero the rest. Note that both above mentioned filters are "ideal", or "oracular" operators. Indeed, they use the coefficients $\langle U, G_i \rangle$ of the original image, which are not known. These algorithms are therefore usually called *oracle filters*. We shall discuss their implementation in the next sections. For the moment, we shall introduce the classical thresholding filters, which approximate the oracle coefficients by using the noisy ones.
We call, as is classical, *Fourier–Wiener filter* the optimal operator (2.10) when $\mathcal{B}$ is a Fourier basis. By the use of the Fourier basis, global image characteristics may prevail over local ones and create spurious periodic patterns. To avoid this effect, the bases are usually more local, of the wavelet or block DCT type.

Sliding window DCT. The local adaptive filters were introduced by Yaroslavsky and Eden [135] and Yaroslavsky [137]. The noisy image is analyzed in a moving window, and at each position of the window its DCT spectrum is computed and modified by using the optimal operator (2.10). Finally, an inverse transform is used to estimate only the signal value in the central pixel of the window.
This method is called the *empirical Wiener filter*, because it approximates the unknown original coefficients $\langle u, G_i \rangle$ by using the identity

$$E|\langle \tilde{U}, G_i \rangle|^2 = |\langle U, G_i \rangle|^2 + \sigma^2$$

and thus replacing the optimal attenuation coefficients $a(i)$ by the family $\{\alpha(i)\}_i$,

$$\alpha(i) = \max\left\{0, \frac{|\langle \tilde{U}, G_i \rangle|^2 - c\sigma^2}{|\langle \tilde{U}, G_i \rangle|^2}\right\}.$$

where c is a parameter, usually larger than one.

Wavelet thresholding. Let $\mathcal{B} = \{G_i\}_i$ be a wavelet orthonormal basis [96]. The so-called *hard wavelet thresholding method* [47] is a (nonlinear) projection operator setting to zero all wavelet coefficients smaller than a certain threshold. According to the expression of the MSE of a projection operator (2.12), the performance of the method depends on the ability of the basis to approximate the image U by a small set of large coefficients. There has been a strenuous search for wavelet bases adapted to images [107].
Unfortunately, not only noise, but also image features can cause many small wavelet coefficients, which are nevertheless lower than the threshold. The brutal cancelation of wavelet (or DCT) coefficients near the image edges creates small oscillations, a Gibbs phenomenon often called *ringing*. Spurious wavelets can also be seen in flat parts of the restored image, caused by the undue cancelation of some of the small coefficients. These artifacts are sometimes called *wavelet outliers* [50]. These undesirable effects can be partially avoided with the use of a soft thresholding [48],

$$\alpha(i) = \begin{cases} \frac{\langle \tilde{U}, G_i \rangle - sgn(\langle \tilde{U}, G_i \rangle)\mu}{\langle \tilde{U}, G_i \rangle}, & |\langle \tilde{U}, G_i \rangle| \geq \mu, \\ 0 \quad \text{otherwise}, \end{cases}$$

The continuity of this soft thresholding operator reduces the Gibbs oscillation near image discontinuities.

Several orthogonal bases adapt better to image local geometry and discontinuities than wavelets, particularly the "bandlets" [107] and "curvelets" [126]. This tendency to adapt the transform locally to the image is accentuated with the methods adapting a different basis to each pixel, or selecting a few elements or "atoms" from a huge patch dictionary to linearly decompose the local patch on these atoms. This point of view is sketched in the next section on sparse coding.

2.3 Sparse coding

Sparse coding algorithms learn a redundant set $\mathbf{D}$ of vectors called *dictionary* and choose the right atoms to describe the current patch.
For a fixed patch size, the dictionary is encoded as a matrix of size $\kappa^2 \times n_{dic}$, where κ^2 is the number of pixels in the patch and $n_{dic} \geq \kappa^2$. The dictionary patches, which are columns of the matrix, are normalized (in Euclidean norm). This dictionary may collect usual orthogonal bases (discrete cosine transform, wavelets, curvelets ...), but also patches extracted (or learnt) from clean images or even from the noisy image itself.
The dictionary permits to compute a sparse representation α of each patch P, where α is a coefficient vector of size n_{dic}^2 satisfying $P \approx \mathbf{D}\alpha$. This sparse representation α can be obtained with an ORMP (orthogonal recursive matching pursuit) [34]. ORMP gives an approximate solution to the (NP-complete) problem

$$\underset{\alpha}{\text{Arg min}} \, ||\alpha||_0 \quad \text{such that} \quad ||P - \mathbf{D}\alpha||_2^2 \leq \kappa^2 (C\sigma)^2 \tag{2.13}$$

where $\|\alpha\|_0$ refers to the l^0 norm of α, i.e. the number of non-zero coefficients of α. This last constraint brings in a new parameter C. This coefficient multiplying the standard deviation σ guarantees that, with high probability, a white Gaussian noise of standard deviation σ on κ^2 pixels has an l^2 norm lower than $\kappa C\sigma$. The ORMP algorithm is introduced in [34]. Details on how this minimization can be achieved are given in the section describing the K-SVD algorithm 4.7 and in the dedicated chapter 10. (It has been argued that the l^0 norm of the set of coefficients can be replaced by the much easier l^1 convex norm. This remark is the starting point of the compressive sampling method [20].)
In K-SVD and other current sparse coding algorithms, the previous denoising strategy is used as the first step of a two-steps algorithm. The selection step is iteratively combined with an update of the dictionary taking into account the image and the sparse codifications already computed. More details will be found in chapter 10 on the K-SVD algorithm.
Several of our referees have objected to considering *sparse coding* and *transform thresholding* as two different denoising principles. As models, both indeed assume the *sparsity* of patches in some well chosen basis. Nevertheless, some credit must be given to historical development. The notion of sparsity is associated with a recent and sophisticated variational principle, where the dictionary and the sparse decompositions are computed simultaneously. Transform thresholding methods existed before the term sparsity was even used. They simply pick a local wavelet or DCT basis and threshold the coefficients. In both algorithms, the sparsity is implicitly or explicitly assumed. But transform threshold methods use orthogonal bases, while the dictionaries are redundant. Furthermore, the algorithms are very different.

2.4 Image self-similarity leading to pixel averaging

The principle of many denoising methods is quite simple: they replace the colour of a pixel with an average of the colours of nearby pixels. It is a powerful and basic principle, when applied directly on noisy pixels with independent noise. If m pixels with the same colour (up to the fluctuations due to noise) are averaged the noise is reduced by a $\sqrt{m}$ factor.

The MSE between the true (unknown) value $u(\mathbf{i})$ of a pixel $\mathbf{i}$ and the value estimated by a weighted average of pixels $\mathbf{j}$ is

$$\begin{aligned} E\|u(\mathbf{i}) - \sum_{\mathbf{j}} w(\mathbf{j})\tilde{u}(\mathbf{j})\|^2 &= E\|\sum_{\mathbf{j}} w(\mathbf{j})(u(\mathbf{i}) - u(\mathbf{j})) - \sum_{\mathbf{j}} w(\mathbf{j})n(\mathbf{j})\|^2 \\ &= \sum_{\mathbf{j}} w(\mathbf{j})^2(u(\mathbf{i}) - u(\mathbf{j}))^2 + \sigma^2 \sum_{\mathbf{j}} w(\mathbf{j})^2, \end{aligned} \tag{2.14}$$

where we assume that the noise, the image and the weights are independent and that the weights $\{w(\mathbf{j})\}_{\mathbf{j}}$ satisfy $\sum_{\mathbf{j}} w(\mathbf{j}) = 1$.
The above expression implies that the performance of the averaging depends on the ability to find many pixels $\mathbf{j}$ with an original value $u(\mathbf{j})$ close to $u(\mathbf{i})$. Indeed, the variance term $\sum_{\mathbf{j}} w(\mathbf{j})^2$ is minimized by a flat distribution probability $w(\mathbf{j}) = 1/m$, where m is the number of averaged pixels. The first term measures the bias caused by the fact that pixels do not have exactly the same deterministic value. Each method must find a tradeoff between the bias and variance terms of equation (2.14).

Averaging of spatially close pixels A first rather trivial idea is to average the closest pixels to a given pixel. This amounts to convolve the image with a fixed radial positive kernel. The paradigm of such kernels is the Gaussian kernel.
The convolution of the image with a Gaussian kernel ensures a fixed noise standard deviation reduction factor that equals the kernel standard deviation. Yet, nearby pixels do not necessarily share their colours. Thus, the first error term in (2.14) can quickly increase. This approach is valid only for pixels for which the nearby pixels have the same colour, that is, it only works inside the homogeneous image regions, but not for their boundaries.

Averaging pixels with similar colours A simple solution to the above mentioned dilemma is given by the sigma-filter [83] or neighborhood filter [136]. These filters average only nearby pixels of $\mathbf{i}$ having also a similar colour value. We shall denote these filters by *YNF*, (for Yaroslavsky neighborhood filter). Their formula is simply

$$YNF_{h,\rho}\tilde{u}(\mathbf{i}) = \frac{1}{C(\mathbf{i})} \sum_{\mathbf{j} \in B_\rho(\mathbf{i})} \tilde{u}(\mathbf{j})\, e^{-\frac{|\tilde{u}(\mathbf{i}) - \tilde{u}(\mathbf{j})|^2}{h^2}}, \tag{2.15}$$

where $B_\rho(\mathbf{i})$ is a ball of center $\mathbf{i}$ and radius $\rho > 0$, $h > 0$ is the filtering parameter and $C(\mathbf{i}) = \sum_{\mathbf{j} \in B_\rho(\mathbf{i})} e^{-\frac{|\tilde{u}(\mathbf{j}) - \tilde{u}(\mathbf{i})|^2}{h^2}}$ is the normalization factor. The parameter h controls the degree of colour similarity needed to be taken into account in the average. According to the Bayesian interpretation of the filter we should have $h = \sigma$. The filter (2.15), due to Yaroslavsky and Lee, has been reinvented several times, and has received the alternative names of *SUSAN filter* [125] and of *Bilateral filter* [127]. The relatively minor difference in these algorithms is that instead of considering a fixed spatial neighborhood $B_\rho(\mathbf{i})$, they weigh the spatial distance to the reference pixel $\mathbf{i}$ by a
Neighborhood filters choose the "neighboring" pixels by comparing their noisy colour. The weight distribution is therefore computed by using noisy values and is not independent of the noise. Therefore the error formula (2.14) is not applicable. For a flat zone and for a given pixel with colour value a, the nearby pixels with an intensity difference lower than h will be independent and identically distributed with a probability distribution which is the restriction of the Gaussian to the interval $(a - h, a + h)$. If the search zone (or spatial neighborhood) is broad enough, then the average value will tend to the expectation of this random variable. Thus, the increase of the search zone and therefore of the number of pixels being averaged beyond a reasonable value will not increase the noise reduction capability of the filter. More precisely, the asymptotic noise reduction factor is given in the next theorem, taken from [11].

Theorem 2. *Assume that $n(\mathbf{i})$ are i.i.d. with zero mean and variance σ^2, then a noise n filtered by the neighborhood filter YNF_h satisfies*

$$Var\ YNF_{h,\rho}\, n = f\left(\frac{h}{\sigma}\right)\sigma^2,$$

where

$$f(x) = \frac{1}{(2\pi)^{3/2}} \int_{\mathbb{R}} \frac{1}{\beta^2(a,x)} (e^{2xa} - 1)^2 e^{(a+x)^2} e^{\frac{-a^2}{2}}\, da$$

and

$$\beta(a,x) = \frac{1}{\sqrt{2\pi}} \int_{a-x}^{a+x} e^{-t^2/2}\, dt\,.$$

The function $f(x)$ is a decreasing function with $f(0) = 1$ and $\lim_{x\to\infty} f(x) = 0$ (see plot in Fig. 2.1). The noise reduction increases with the ratio h/σ. We see that $f(x)$ is close to zero for values of x over 2.5 or 3, that is, values of h over $2.5\,\sigma$ or $3\,\sigma$. This corresponds to the values proposed in the original papers by Lee and Yaroslavsky. However, for a Gaussian variable, the probability of observing values at a distance to the average above 2.5 or 3 times the standard deviation is very small. Thus, taking these large values excessively increases the probability of mismatching pixels belonging in fact to other objects. This explains the observed decaying performance of the neighborhood filter when the noise standard deviation or the search zone $B(\mathbf{i}, \rho)$ increase too much.

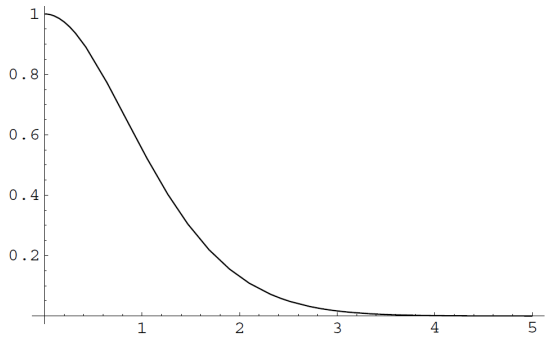


Figure 2.1: Noise reduction function $f(x)$ given by Theorem 2.

The image model underlying neighborhood filters is the *image self-similarity*, namely the presence in the image of pixels **j** which have the same law as **i**. We will introduce in section 4.1 the NL-means algorithm [16] which can be seen as an extension of the neighborhood filters attenuating their main drawbacks. In NL-means, the "neighborhood of a pixel **i**" is defined as any set of pixels **j** in the image such that a patch around **j** looks like a patch around **i**. In other terms NL-means estimates the value of **i** as an average of the values of all the pixels **j** whose neighborhood looks like the neighborhood of **i**.

Chapter 3

Noise Reduction, Generic Tools

This chapter describes three generic tools that permit to increase the performance of *any* denoising principle. We shall illustrate them on DCT denoising. Starting from the application of a simple DCT transform threshold, the three generic tools (uniform and weighted aggregation, iteration and "oracle" filters, color space transform) will be applied successively. We shall observe a dramatic improvement of the denoising performance. This observation is valid for all denoising principles, as we will see in chapter 4. Those three principles are shared by almost all state of the art algorithms (see in particular chapters 10, 11 and 12) with the notable exception of DDID which does not use the aggregation tool but only the other two (see section 4.10).
This chapter uses the results of joint work with Antoni Buades and Jean-Michel Morel.

3.1 Aggregation of estimates

Aggregation techniques combine for any pixel a set of m possible estimates. If these estimates were independent and had equal variance, then a uniform average would reduce this estimator variance by a factor m. Such an aggregation strategy was the main proposition of the *translation invariant wavelet thresholding algorithm* [27]. This method denoises several translations of the image by a wavelet thresholding algorithm and averages these different estimates once the inverse translation has been applied to the denoised images.
An interesting case is when one is able to estimate the variance of the m estimators. Statistical arguments lead to attribute to each estimator a weight inversely proportional to its variance [102]. For most denoising methods the variance of the estimators is high near image edges. When applied without aggregation, the denoising methods leave visible "halos" of residual noise near edges. For example in the sliding window DCT method, patches containing edges have many large DCT coefficients which are kept by thresholding. In flat zones instead, most DCT coefficients are canceled and the noise is completely removed. The proposition of [60] is to use the aggregation for DCT denoising, approximating the variance of each estimated patch by the number of non zero coefficients after thresholding. In the online paper [139] one can test an implementation of DCT denoising. It actually uses an aggregation with uniform weights: "translation invariant DCT denoising is implemented by decomposing the image to sliding overlapping patches, calculating the DCT denoising in each patch, and then aggregating the denoised patches to the image averaging the overlapped pixels. The translation invariant DCT denoising significantly improves the denoising performance, typically from about 2 to 5 dB, and removes the block artifact".
The same risk of "halo" occurs with non-aggregated NL-means (section 4.1), since patches containing edges have many less similar instances in the image than flat patches. Thus the non-local averaging is made over less samples, and the final result keeps more noise near image edges. The same phenomenon occurs with BM3D (see section 4.8 and dedicated chapter 11) if the aggregation step is not applied [35]. As a consequence, an aggregation step is applied in all patch-based denoising algorithms. This weighted aggregation favors, at each pixel near an edge, the estimates

given by patches which contain the pixel but do not meet the edge.
Aggregation techniques aim at a superior noise reduction by increasing the number of values being averaged for obtaining the final estimate or selecting those estimates with lower variance. Kervrann et al [71] considered the whole Bias+Variance decomposition in order to also adapt the search zone of neighborhood filters or of NL-means. Since the bias term depends on the original image, it cannot be computed in practice, and Kervrann et al. proposed to minimize both bias and variance by choosing the smallest spatial neighborhood attaining a stable noise reduction.
Another type of aggregation technique considers the risk estimate rather than the variance to locally attribute more weight to the estimators with small risks. In [130], Van De Ville and Kocher give a closed form expression of Stein's Unbiased Estimator of the Risk (SURE) for NL-Means. (See also generalizations of the SURE estimator to the non-Gaussian case in [119].) The aim is to select globally the best bandwidth for a given image. In [51], Duval et al. also use the SURE technique for minimizing the risk by selecting locally the bandwidth. Deledalle et al. [42] apply the same technique for combining the results of NL-means with different window sizes and shapes. A similar treatment can be found in [120], but with the assumption of a local exponential density for the noisy patches.

3.2 Iteration and "oracle" filters

Iterative strategies to remove residual noise would drift from the initial image. Instead, a first step denoised image can be used to improve the reapplication of the denoising method to the initial noisy image. In a second step application of a denoising principle, the denoised DCT coefficients, or the patch distances, can be computed in the first step denoised image. They are an approximation to the true measurements that would be obtained from the noise-free image. Thus, the first step denoised image is used as an "oracle" for the second step.
For averaging filters such as neighborhood filters or NL-means, the image u can be denoised in a first step by the method under consideration. This first step denoised image denoted by $\hat{u}_1$ is used for computing more accurate colour distances between pixels. Thus, the second step neighborhood filter is

$$YNF_{h,\rho}\tilde{u}(\mathbf{i}) = \frac{1}{C(\mathbf{i})} \sum_{\mathbf{j}\in B_\rho(\mathbf{j})} \tilde{u}(\mathbf{j}) e^{-\frac{|\hat{u}_1(\mathbf{j})-\hat{u}_1(\mathbf{i})|^2}{h^2}},$$

where $\tilde{u}$ is the observed noisy image and $\hat{u}_1$ the image previously denoised by (2.15).
Similarly, for linear transform Wiener-type methods, the image is first denoised by its classical definition, which amounts to approximate the oracle coefficients of Theorem 1 using the noisy ones. In a second iteration, the coefficients of the denoised image approximate the true coefficients of the noise-free image. Thus, the second step filter following the first step (2.9) is

$$\mathbf{D}\tilde{U} = \sum_i a(i)\ \langle \tilde{U}, G_i\rangle\ G_i\,, \quad \text{with} \quad a(i) = \frac{|\langle \hat{U}_1, G_i\rangle|^2}{|\langle \hat{U}_1, G_i\rangle|^2 + \sigma^2},$$

where $\hat{U}_1$ is the denoised image by applying a first time the thresholding algorithm to the observed image $\tilde{U}$.

Alternatives and extensions: "twicing" and Bregman iterations In the recent review paper [100], many denoising operators are formalized in a general linear framework, noting that they can be associated with a doubly stochastic diffusion matrix W with nonnegative coefficients. For example in NL-means, this matrix is obtained by the symmetrization of the matrix of the NL-means weights $w_{\tilde{P},\tilde{Q}}$ defined in Algorithm 1. Unless it is optimal, as is the case with an ideal Wiener filter, the matrix W associated with the denoising filter can be iterated. A study of MSE evolution with these iterations is proposed in [100] for several denoising operators, considering several different patch types (texture, edge, flat). Iteration is, however, different from the oracle iteration described above. In the oracle iteration, the matrix W is changed at each step, using

its better estimate given by the previously denoised image. One does not generally observe much improvement by iterating the oracle method more than once. [100] points out another generic tool, used at least for total variation denoising, the so-called "twicing", term due to Tukey [128]. Instead of repeated applications of a filter, the idea is to process the residual obtained as the difference between the estimated image and the initial image. If the residuals contain some of the underlying signal, filtering them should recover part of it. The author shows that the Bregman iterations [106] used for improving total variation based denoising are a twicing and so is the matching pursuit method used in the K-SVD filter described in section 4.7 and in the dedicated chapter 10.

3.3 Dealing with colour images

The straightforward strategy to extend denoising algorithms to colour or multivalued images is to apply the algorithm independently to each channel. The use of this simple strategy often introduces colour artifacts, easily detected by the eye. Two different strategies are observable in state of the art denoising algorithms.
Depending on the algorithm formulation, a vector-valued version dealing at the same time with all colour channels can be proposed. This solution is adopted by averaging filters like neighborhood filters or NL-means. These algorithms compute colour differences directly in the vector valued image, thus yielding a unified weight configuration which is applied to each channel.
The alternative option is to convert the usual RGB image to a different colour space where the independent denoising of each channel does not create noticeable colour artifacts. Most algorithms use the YUV system which separates the geometric and chromatic parts of the image. This change writes as a linear transform by multiplication of the RGB vector by the matrix

$$YUV = \begin{pmatrix} 0.30 & 0.59 & 0.11 \\ -0.15 & -0.29 & 0.44 \\ 0.61 & -0.51 & -0.10 \end{pmatrix}, \qquad Y_oU_oV_o = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

The second colour transform to the space $Y_oU_oV_o$ is an orthogonal transform. It has the advantage of maximizing the noise reduction of the geometric component, since this component is an average of the three colours. The geometric component is perceptually more important than the chromatic ones, and the presence of less noise permits a better performance of the algorithm in this part. It also permits a higher noise reduction on the chromatic components U_o and V_o, due to their observable regularity.
This latter strategy is adopted by transform thresholding filters for which the design of an orthonormal basis coupling the different colour channels is not trivial.

3.4 Trying all generic tools on an example

This section applies incrementally the previous generic denoising tools to the DCT sliding window to illustrate how these additional tools permit to drastically improve the algorithm performance. We start with the basic DCT "neighborhood filter" as proposed by Yaroslavsky [135]. Its principle is to denoise a patch around each pixel, and to keep only the central denoised pixel.
Fig. 3.1 displays the denoised images obtained by incrementally applying each of the following ingredients:

- Basic DCT thresholding algorithm by the neighborhood filter technique (keeping only the central pixel of the window). Each colour channel is treated independently.

- Use of an orthogonal geometric and chromatic decomposition colour system $Y_oU_oV_o$; grey parts are better reconstructed and colour artifacts are reduced.

- Uniform aggregation; the noise reduction is superior and isolated noise points are removed.

- Adaptive aggregation using the estimator variance; the noise reduction near edges is increased, "halo" effects are removed.

- Additional iteration using “oracle” estimation; residual noise is totally removed and the sharpness of details is increased.

The $PSNR$'s obtained by incrementally applying the previous strategies respectively are 26.85, 27.33, 30.65, 30.73, 31.25. This experiment illustrates how the use of these additional tools is crucial to achieve competitive results. This last version of the DCT denoising algorithm, incorporating all the proposed generic tools, will be the one used in the comparison section. A complete description of the algorithm can be found in Algorithm 2. The colour version of the algorithm applies the denoising independently to each $Y_oU_oV_o$ component. This version is therefore slightly better than the version online in [139], which does not use the oracle step.

Algorithm 2 DCT denoising algorithm. DCT coefficients lower than 3σ are canceled in the first step and a Wiener filter is applied in the “oracle” second step. The colour DCT denoising algorithm applies the current strategy independently to each $Y_oU_oV_o$ component.

Input: noisy image $\tilde{u}$, σ noise standard deviation.
Optional: prefiltered image $\hat{u}_1$ for “oracle" estimation.
Output: output denoised image.

Set parameter $\kappa = 8$: size of patches.
Set parameter $h = 3\,\sigma$: threshold parameter.

for each pixel **i** **do**

Select a square reference patch $\tilde{P}$ around **i** of size $\kappa \times \kappa$.
if $\hat{u}_1$ **then**
Select a square reference patch P_1 around **i** in $\hat{u}_1$.
end if
Compute the DCT transform of $\tilde{P}$.
if $\hat{u}_1$ **then**
Compute the DCT transform of P_1.
end if
if $\hat{u}_1$ **then**
Modify DCT coefficients of $\tilde{P}$ as

$$\tilde{P}(i) = \tilde{P}(i)\frac{P_1(i)^2}{P_1(i)^2 + \sigma^2}$$

else
Cancel coefficients of $\tilde{P}$ with magnitude lower than h.
end if
Compute the inverse DCT transform obtaining $\hat{P}$.
Compute the aggregation weight $w_{\tilde{P}} = 1/\#\{\text{number of non-zero } DCT \text{ coefficients}\}$.
end for

for each pixel **i** **do**
Aggregation: recover the denoised value at each pixel **i** by averaging all values at **i** of all denoised patches $\hat{Q}$ containing **i**, weighted by $w_{\tilde{Q}}$.
end for

Figure 3.1: Top: original and noisy images with an additive Gaussian white noise of standard deviation 25. Below, from top to bottom and left to right: crop of denoised images by sliding DCT thresholding filter and incrementally adding use of a $Y_oU_oV_o$ colour system, uniform aggregation, variance based aggregation and iteration with the "oracle" given by the first step. The corresponding PSNR are 26.85, 27.33, 30.65, 30.73, 31.25.

Chapter 4

Detailed Analysis of Ten Methods

In this long chapter, a detailed description and analysis of ten denoising methods (K-SVD, BM3D, NL-Bayes, NL-means, BLS-GSM, NLDD, PLOW, EPLL, PLE, and Shotgun NL-means) is provided. The first six methods in addition to the sliding DCT filter specified in chapter 3, for which reliable faithful implementations are available, will also be compared in chapter 5. Moreover, the first three will be more extensively studied respectively in chapters 10, 11, and 12. This analysis shows that we are probably close to understanding digital images at a "patch" scale.
As shown in section 4.4, the mathematical and experimental evidence of two recent articles suggests that we might even be close to the best attainable performance in image denoising ever. This suspicion is supported by a remarkable convergence of all analyzed methods, as one can see on PSNR tables shown in chapter 5. They certainly converge in performance. We intend to demonstrate that, under different formalisms, their methods are almost equivalent. Working in the 64-dimensional "patch space", all recent methods estimate local "sparse models" and restore a noisy patch by finding its likeliest interpretation knowing the noiseless patches.
This chapter uses the results of joint work with Antoni Buades and Jean-Michel Morel.

4.1 Non-local means

The Non-local means (NL-means) algorithm tries to take advantage of the redundancy of most natural images. The redundancy, or self-similarity hypothesis is that for every small patch in a natural image one can find several similar patches in the same image, as illustrated in figures 4.1 and 4.2. This similarity is true for patches whose centers stand at a one pixel distance of the center of the reference patch. In that case the self-similarity boils down to a local image regularity assumption. Such a regularity is guaranteed by Shannon-Nyquist's sampling conditions, which require the image to be blurry. In a much more general sense inspired by neighborhood filters, one can define as "neighborhood of a pixel **i**" any set of pixels **j** in the image such that a patch around **j** looks like a patch around **i**. All pixels in that neighborhood can be used for predicting the value at **i**, as was first shown in [52] for the synthesis of texture images. This self-similarity hypothesis is a generalized periodicity assumption. The use of self-similarities is actually well-known in information theory from its foundation. In his 1948 Mathematical Theory of Communication, Shannon [124] analyzed the local self-similarity (or redundancy) of natural written language, and gave probably the first stochastic text synthesis algorithm. The Efros-Leung texture synthesis method adapted this algorithm to images, and NL-Means [12] seems to be first adaptation of the same idea to denoising[1]
NL-means denoises a square reference patch $\tilde{P}$ around **i** of dimension $\kappa \times \kappa$ by replacing it by an

[1]Nevertheless, some researchers have pointed out to us the report [8] as giving an early intuition that intuition could use signal redundancy. This very short paper describes an experiment in a few sentences. It suggests that region redundancy on both sides of an edge can be detected, and used for image denoising. Nevertheless, no algorithm is specified in this paper.

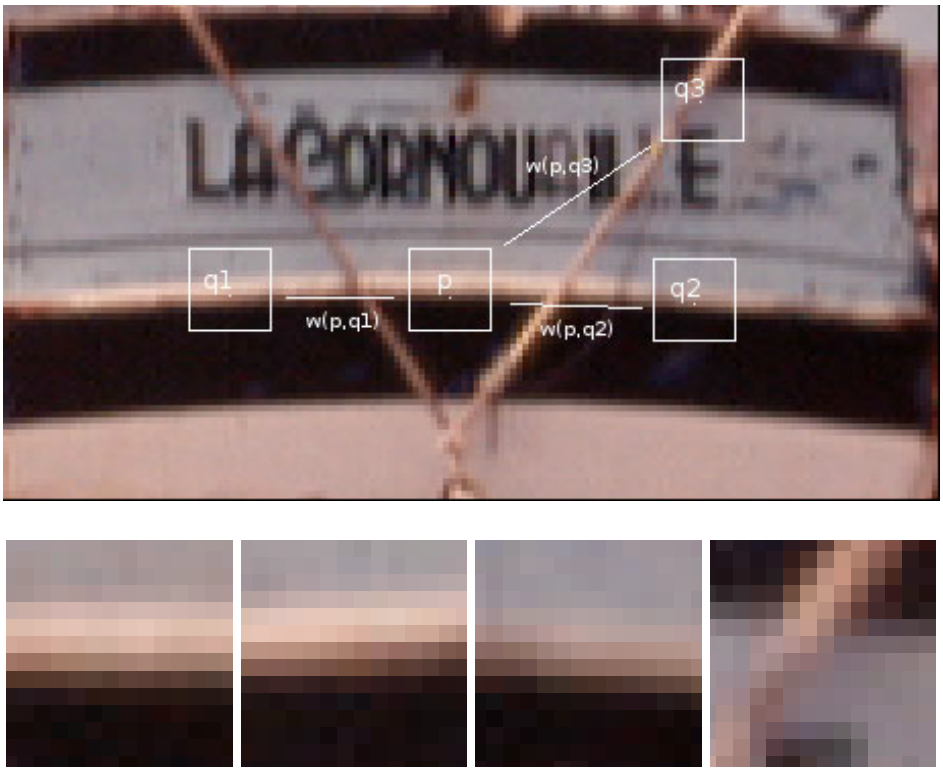


Figure 4.1: $q1$ and $q2$ have a large weight because their similarity windows are similar to that of p. On the other side the weight $w(p, q3)$ is much smaller because the intensity grey values in the similarity windows are very different.

average of all similar patches $\tilde{Q}$ in a square neighborhood of $\mathbf{i}$ of size $\lambda \times \lambda$. To do this, a normalized Euclidean distance between $\tilde{P}$ and $\tilde{Q}$, $d(\tilde{P}, \tilde{Q}) = \frac{1}{\kappa^2}\|\tilde{P} - \tilde{Q}\|^2$ is computed for all patches $\tilde{Q}$ is the search neighborhood. Then the weighted average is

$$\hat{P} = \frac{\sum_{\tilde{Q}} \tilde{Q} e^{-\frac{d(\tilde{P},\tilde{Q})^2}{h^2}}}{\sum_{\tilde{Q}} e^{-\frac{d(\tilde{P},\tilde{Q})^2}{h^2}}}.$$

The thing that helps NL-means over the neighborhood filters is the concentration of the noise law, as the number of pixels increases. Because the distances are computed on many patch samples instead of only one pixel, the fluctuations of the quadratic distance due to the noise are reduced.

Related attempts: [134] proposed a “universal denoiser" for digital images. The authors prove that this denoiser is universal in the sense “of asymptotically achieving, without access to any information on the statistics of the clean signal, the same performance as the best denoiser that does have access to this information". In [105] the authors presented an implementation valid for binary images with an impulse noise, with excellent results. Awate and Whitaker [6] also proposed a method whose principles stand close to NL-means, since the method involves comparison between patches to estimate a restored value. The objective of the algorithm is to denoise the image by decreasing the randomness of the image.

A consistency theorem for NL-means. NL-means is intuitively consistent under stationarity conditions, namely if one can find many samples of every image detail. It can be proved [16] that if the image is a fairly general stationary and mixing random process, for every pixel $\mathbf{i}$, NL-means

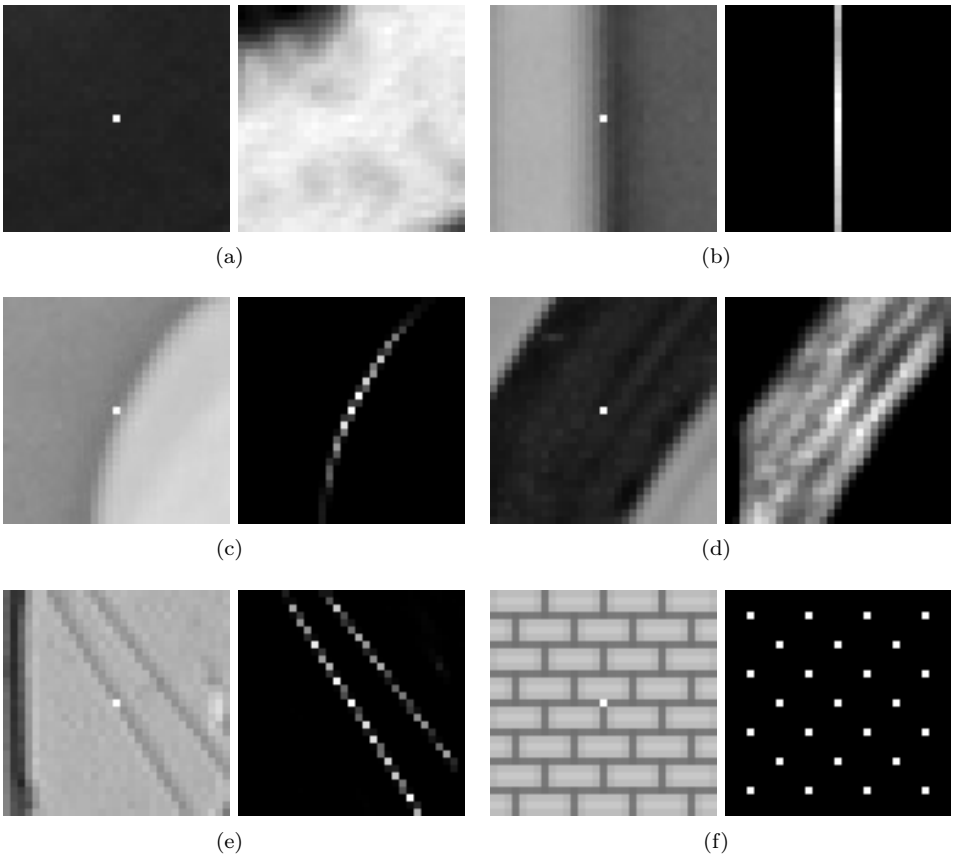


Figure 4.2: On the right-hand side of each pair, one can see the weight distribution used to estimate a centered patch of the left image by NL-means. (a) In flat zones, the weights are uniformly distributed, NL-means acts like a low pass isotropic filter. (b) On straight edges, the weights are distributed in the direction of the edge (like for anisotropic filters). (c) On curved edges, the weights favor pixels belonging to the same contour. (d) In a flat neighborhood, the weights are distributed in a grey-level neighborhood (exactly like for neighborhood filters). In the cases of (e) and (f), the weights are distributed across the more similar configurations, even though they are far away from the observed pixel. This behavior justifies the "non local" appellation.

converges to the conditional expectation of $\mathbf{i}$ knowing its neighborhood, which is the best Bayesian estimate.

NL-means as an extension of previous methods. A Gaussian convolution preserves only flat zones, while contours and fine structure are removed or blurred. Anisotropic filters instead preserve straight edges, but flat zones present many artifacts. One could think of combining these two methods to improve both results. A Gaussian convolution could be applied in flat zones, while an anisotropic filter could be applied on straight edges. Still, other types of filters should be designed to specifically restore corners, or curved edges, or periodic texture. Figure 4.2 illustrates how NL-means chooses the right weight configuration for each sort of image self-similarity.
NL-means is easily extended to the denoising of image sequences and video, involving indiscriminately pixels belonging to a space-time neighborhood. The algorithm favors pixels with a similar local configuration. When the similar configuration moves, so do the weights. Thus, as shown in [14] the algorithm is able to follow moving similar configurations without any explicit motion computation (see Fig. 4.3).
Indeed, this fact contrasts with previous classical movie denoising algorithms, which were motion

compensated. The underlying idea of motion compensation is the existence of a "ground truth" for the physical motion. Legitimate information about the colour of a given pixel should exist only along its physical trajectory. Yet, one of the major difficulties in motion estimation is the ambiguity of trajectories, the so-called *aperture problem*. The aperture problem, viewed as a general phenomenon of movies, can be positively interpreted in the following way: There are many pixels in the next or previous frames which resemble the current pixel. Thus, it seems sound to use not just one trajectory, but rather *all similar pixels* to the current pixel across time and space as NL-means does (see [14] for more details on this discussion).

Algorithm 3 NL-means algorithm (parameter values for κ, λ are indicative).

Input: noisy image $\tilde{u}$, σ noise standard deviation.
Output: output denoised image.

Set parameter $\kappa = 3$: size of patches.
Set parameter $\lambda = 31$: size of research zone for which similar patches are searched.
Set parameter $h = 0.6\,\sigma$: bandwidth filtering parameter.

for each pixel **i** **do**

Select a square reference patch $\tilde{P}$ around **i** of dimension $\kappa \times \kappa$.

Set $\hat{P} = 0$ and $\hat{C} = 0$.

for each patch $\tilde{Q}$ in a square neighborhood of **i** of size $\lambda \times \lambda$ **do**

Compute the normalized Euclidean distance between $\tilde{P}$ and $\tilde{Q}$, $d(\tilde{P}, \tilde{Q}) = \frac{1}{\kappa^2}\|\tilde{P} - \tilde{Q}\|^2$.

Accumulate $\tilde{Q}e^{-\frac{d(\tilde{P},\tilde{Q})^2}{h^2}}$ to $\hat{P}$ and $e^{-\frac{d(\tilde{P},\tilde{Q})^2}{h^2}}$ to $\hat{C}$.

end for

Normalize the average patch $\hat{P}$ by dividing it by the sum of weights $\hat{C}$

end for

for each pixel x **do**
Aggregation: recover the denoised value at each pixel **i** by averaging all values at **i** of all denoised patches $\hat{Q}$ containing **i**
end for

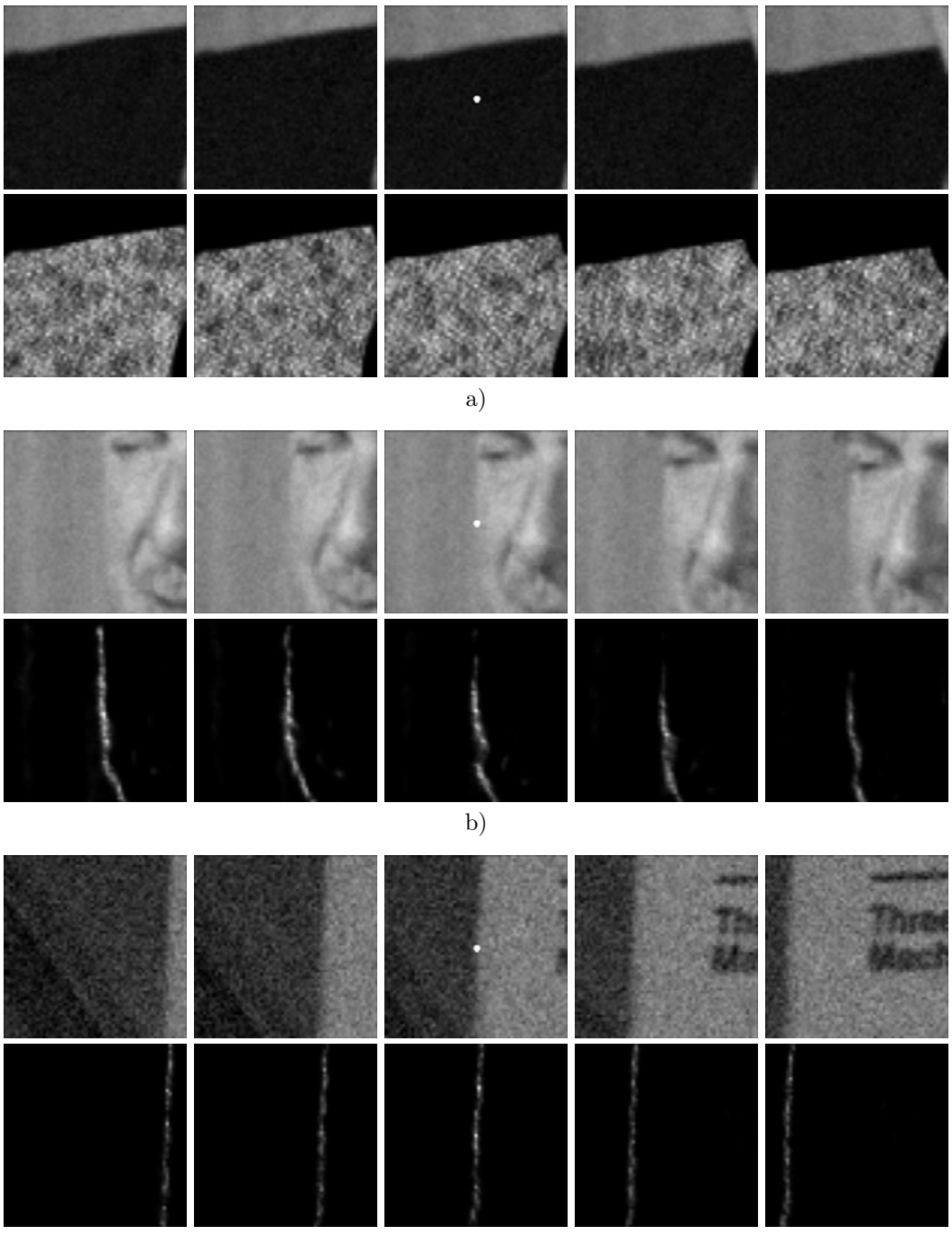


Figure 4.3: Weight distribution of NL-means applied to a movie. In a), b) and c) the first row shows a five frames image sequence. In the second row, the weight distribution used to estimate the central pixel (in white) of the middle frame is shown. The weights are equally distributed over the successive frames, including the current one. They actually involve all the candidates for the motion estimation instead of picking just one per frame. The aperture problem can be taken advantage of for a better denoising performance by involving more pixels in the average.

4.2 Non-local Bayesian denoising

It is apparent that (2.7) given in section 2.1,

$$\hat{P}_1 = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I}\right]\mathbf{C}_{\tilde{P}}^{-1}(\tilde{P} - \overline{\tilde{P}}),$$

gives by itself a denoising algorithm, provided we can compute the patch expectations and patch covariance matrices. We shall now explain how the Non-local Bayes algorithm proposed in chapter 12 and also in [75] does it.

In order to select the set of similar patches $\mathcal{P}(\tilde{P})$, the distance L^2 between the reference patch $\tilde{P}$ and all patches $\tilde{Q}$ in a neighborhood around $\tilde{P}$ is computed. Then, only a fixed number of similar patches is kept, obviously those with the lowest distance to the reference one. This fixed number must be larger than the dimension of the patch to obtain an invertible covariance matrix $\mathbf{C}_{\tilde{P}}$. Then, the mean $\overline{\tilde{P}}$ and covariance matrix $\mathbf{C}_{\tilde{P}}$ write

$$\mathbf{C}_{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P}) - 1}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\Big(\tilde{Q} - \overline{\tilde{P}}\Big)\Big(\tilde{Q} - \overline{\tilde{P}}\Big)^t, \qquad \overline{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\tilde{Q}. \tag{4.1}$$

The selection of similar patches can be improved in a second step by using the first estimate $\hat{P}_1$ as *oracle*. Indeed, the patch similarity is better estimated with the denoised patches and a better approximation of $\overline{P}$ and $\mathbf{C}_P$ can be obtained. The new set of similar patches $\mathcal{P}(\hat{P}_1)$ is set to contain neighbouring denoised patches $\hat{Q}_1$ with L^2 distance to $\hat{P}_1$ below a certain threshold τ_0. Then,

$$\mathbf{C}_{\hat{P}_1} \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1) - 1}\sum_{\hat{Q}_1\in\mathcal{P}(\hat{P}_1)}\Big(\hat{Q}_1 - \overline{\hat{P}_1}\Big)\Big(\hat{Q}_1 - \overline{\hat{P}_1}\Big)^t, \qquad \overline{\hat{P}_1} \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1)}\sum_{\hat{Q}_1\in\mathcal{P}(\hat{P}_1)}\tilde{Q}_1. \tag{4.2}$$

respectively approximate $\mathbf{C}_P$ and $\overline{P}$. This approximation is different from (12.6) where $\mathbf{C}_P$ was approximated by using the covariance matrix of noisy patches, $\mathbf{C}_{\tilde{P}} \equiv \mathbf{C}_P + \sigma^2\mathbf{I}$. Thus, we obtain a second estimation Bayesian formula

$$\hat{P}_2 = \overline{\tilde{P}_1} + \mathbf{C}_{\hat{P}_1}\left[\mathbf{C}_{\hat{P}_1} + \sigma^2\mathbf{I}\right]^{-1}(\tilde{P} - \overline{\tilde{P}_1}). \tag{4.3}$$

The estimates (2.7) and (4.3) may appear equivalent, but they are not in practice. $\mathbf{C}_{\hat{P}_1}$, obtained after a first denoising step, is a better estimation than $\mathbf{C}_{\tilde{P}}$. Furthermore, $\overline{\tilde{P}_1}$ is a more accurate mean than $\overline{\tilde{P}}$. Nevertheless, this second step cannot be considered as an iteration of the first step algorithm. Indeed, it is still the original patch $\tilde{P}$ which is being denoised at the second step. Since in Algorithm 4 the number of independent samples (the number of patches) used for computing the covariance matrix is larger than the dimension of the vectors (size of patches) the empirical covariance matrices of the form $\mathbf{C}_P$ are always symmetric nonnegative. They are actually positive with high probability on natural images. Thus, it is always possible in practice to inverse them. Numerically, a Cholesky based symmetric matrix inversion algorithm is used. For the sake of completeness (not happening in practice, but which could perhaps happen on a synthetic image) if the matrix cannot be inverted, the algorithm sets $\hat{P}_1 = \tilde{P}$ or $\hat{P}_2 = \hat{P}_1$.

As pointed out in chapter 12 and also in [75], the Nonlocal Bayes algorithm only is an interpretation (with some generic improvements like the aggregation) of the PCA based algorithm proposed in [142]. This paper has a self-explanatory title: "Two-stage image denoising by principal component analysis with local pixel grouping." It is equivalent to apply a PCA on the patches similar to $\tilde{P}$, followed by a Wiener filter on the coefficients of $\tilde{P}$ on this PCA, or to apply formula (2.7) with the covariance matrix of the similar patches. Indeed the PCA computes nothing but the eigenvalues of the empirical covariance matrix. Thus, the method in [142] gets its Bayesian interpretation. A study on the compared performance of local PCA versus global PCA for TSID is actually proposed in [43].

Algorithm 4 Non local Bayes image denoising

Input: noisy image
Output: denoised image

for all patches $\tilde{P}$ of the noisy image **do**

Find a set $\mathcal{P}(\tilde{P})$ of patches $\tilde{Q}$ similar to $\tilde{P}$.

Compute the expectation $\overline{\tilde{P}}$ and covariance matrix $\mathbf{C}_{\tilde{P}}$ of these patches by

$$\mathbf{C}_{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P}) - 1} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \Big(\tilde{Q} - \overline{\tilde{P}}\Big)\Big(\tilde{Q} - \overline{\tilde{P}}\Big)^t, \qquad \overline{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \tilde{Q}.$$

Obtain the first step estimation:

$$\hat{P}_1 = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2 \mathbf{I}\right] \mathbf{C}_{\tilde{P}}^{-1} (\tilde{P} - \overline{\tilde{P}}).$$

end for
Obtain the pixel value of the basic estimate image $\hat{u}_1$ as an average of all values of all denoised patches $\hat{Q}_1$ which contain $\mathbf{i}$.
for all patches $\tilde{P}$ of the noisy image **do**

Find a new set $\mathcal{P}_1(\tilde{P})$ of noisy patches $\tilde{Q}$ similar to $\tilde{P}$ by comparing their denoised "oracular" versions Q_1 to P_1.

Compute the new expectation $\overline{\tilde{P}}^1$ and covariance matrix $\mathbf{C}_{\hat{P}_1}$ of these patches:

$$\mathbf{C}_{\hat{P}_1} \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1) - 1} \sum_{\hat{Q}_1 \in \mathcal{P}(\hat{P}_1)} \left(\hat{Q}_1 - \overline{\tilde{P}}^1\right)\left(\hat{Q}_1 - \overline{\tilde{P}}^1\right)^t, \qquad \overline{\tilde{P}}^1 \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1)} \sum_{\hat{Q}_1 \in \mathcal{P}(\hat{P}_1)} \tilde{Q}.$$

Obtain the second step patch estimate

$$\hat{P}_2 = \overline{\tilde{P}}^1 + \mathbf{C}_{\hat{P}_1} \left[\mathbf{C}_{\hat{P}_1} + \sigma^2 \mathbf{I}\right]^{-1} (\tilde{P} - \overline{\tilde{P}}^1).$$

end for
Obtain the pixel value of the denoised image $\hat{u}(\mathbf{i})$ as an average of all values of all denoised patches $\hat{Q}_2$ which contain $\mathbf{i}$.

4.3 Patch-based near-optimal image denoising (PLOW)

While in the Non-local Bayes method of section 4.2 a local model is estimated in a neighborhood of each patch, in the PLOW [24] method the idea is to learn from the image a sufficient number of patch clusters, actually 15, and to apply the LMMSE estimate to each patch *after* having assigned it to one of the clusters obtained by clustering. Thus, this empirical-Bayesian algorithm starts by clustering the patches by the classic K-means clustering algorithm. To take into account that similar patches can actually have varying contrast, the inter-patch distance is photometrically neutral, and the authors call it a "geometric distance". The clustering phase is accelerated by a dimension reduction obtained by applying a principal component analysis to the patches. The clustering is therefore a segmentation of the set of patches, and the denoising of each patch is then performed within its cluster. Since each cluster contains geometrically similar, but not necessarily photometrically similar patches, the method identifies for each patch in the cluster the photometrically similar patches as those whose quadratic distance to the reference patch are within the bounds allowed by noise. Then a LMMSE [70] estimate is obtained for the reference patch by a variant of (2.7). The algorithm uses a first phase, which performs a first denoising before constituting the clusters. Thus the main phase is actually using the first phase as oracle to get the covariance matrices of the sets of patches.

Algorithm 5 Algorithm 1: PLOW denoising

Input: image in vector form $\tilde{U}$.
Output: denoised image in vector form $\hat{U}$.

Set parameters: patch size $\kappa \times \kappa = 11 \times 11$, number of clusters $K = 15$;
Estimate noise standard deviation $\hat{\sigma}$ by $\hat{\sigma} = 1.4826\text{median}(|\nabla\tilde{U} - \text{median}(\nabla\tilde{U})|)$;
Set parameter: $h^2 = 1.75\hat{\sigma}^2\kappa^2$;
Pre-filter image $\tilde{U}$ to obtain a pilot estimate $\hat{U}_1$;
Extract overlapping patches of size $\kappa \times \kappa$, $\tilde{Q}$ from $\tilde{U}$ and $\hat{Q}_1$ from U_1;
Geometric clustering with K-Means of the patches in $\hat{U}_1$ (using a variant of PCA for the patches). The distance is a geometric distance, photometrically neutral.
for each patch cluster Ω_k **do**
  Estimate from the patches $\hat{Q}_1 \in \Omega_k$ the mean patch $\overline{P}_k \simeq \sum_{\hat{Q}_1 \in \Omega_k} \hat{Q}_1$ and the cluster covariance $\mathbf{C}_P^k$.
  for each patch $\hat{Q}_{1,i} \in \Omega_k$ **do**
    Consider its associated noisy patch $\tilde{Q}_i$. Identify photometrically similar patches $\tilde{Q}_j$ in the cluster as those with a quadratic distance to $\tilde{Q}_i$ within the bounds allowed by noise, namely $\gamma^2 + 2\kappa^2\hat{\sigma}^2$, with $\gamma = \gamma(\kappa)$ a "small" threshold.
    Compute similarity weights $w_{ij} = e^{-\frac{||\tilde{Q}_i - \tilde{Q}_j||^2}{h^2}}$.
    Compute the slightly more complex than usual LMMSE estimator for the noisy patch $\tilde{Q}_i$, (because the cluster contains patches that are geometrically similar but not necessarily photometrically similar):

$$\hat{Q}_i = \overline{P} + \left[\mathbf{I} - \left(\sum_j w_{ij}\mathbf{C}_P^k + \mathbf{I}\right)^{-1}\right] \sum_j \frac{w_{ij}}{\sum_j w_{ij}} (\tilde{Q}_j - \overline{P}).$$

  end for
end for
At each pixel aggregate multiple estimates from all $\hat{P}$ containing it, with weights given as inverses of the variance of each estimator.

4.4 Inherent bounds in image denoising

By "Shotgun" patch denoising methods, we mean methods that intend to denoise patches by a fully non-local algorithm, in which the patch is compared to a patch model obtained from a large or *very large* patch set. The "sparse-land" methods intend to learn from a single image or from a small set of images a sparse patch dictionary on which to decompose any given patch. The shotgun methods learn instead from a very large patch set extracted from tens of thousands of images (up to 10^{10} patches). Then the patch is denoised by deducing its likeliest estimate from the set of all patches. In the case of [145], this patch space is organized as a Gaussian mixture with about 200 components. Shotgun methods have started being used in several image restoration methods. For example in [62], for image inpainting, with an explicit enough title: "Scene completion using millions of photographs".

The approach of [86] is to define the simplest universal "shotgun" method, where a huge set of patches is used to estimate the upper limits a patch-based denoising method will ever reach. The results support the "near optimality of state of the art denoising results", the results obtained by the BM3D algorithm being only 0.1 decibel away from optimality for methods using small patches (typically 8×8.)

This experiment uses to evaluate the MMSE a set of 20, 000 images from the LabelMe dataset [123]. The method, even if certainly not practical, is of exquisite simplicity. Given a clean patch P the noisy patch $\tilde{P}$ with Gaussian noise of standard deviation σ has probability distribution

$$\mathbb{P}(\tilde{P} \mid P) = \frac{1}{(2\pi\sigma^2)^{\frac{\kappa^2}{2}}} e^{-\frac{||P-\tilde{P}||^2}{2\sigma^2}}, \tag{4.4}$$

where κ^2 is the number of pixels in the patch. Then given a noisy patch $\tilde{P}$ its optimal estimator for the Bayesian minimum squared error (MMSE) is by Bayes' formula

$$\hat{P} = \mathbb{E}[P \mid \tilde{P}] = \int \mathbb{P}(P \mid \tilde{P}) P dP = \int \frac{\mathbb{P}(\tilde{P} \mid P)}{\mathbb{P}(\tilde{P})} \mathbb{P}(P) P dP. \tag{4.5}$$

Using a huge set of M natural patches (with a distribution supposedly approximating the real natural patch density), we can approximate the terms in (4.5) by $\mathbb{P}(P)dP \simeq \frac{1}{M}$ and $\mathbb{P}(\tilde{P}) \simeq \frac{1}{M}\sum_i \mathbb{P}(\tilde{P} \mid P_i)$, which in view of (4.4) yields

$$\hat{P} \simeq \frac{\frac{1}{M}\sum_i \mathbb{P}(\tilde{P} \mid P_i) P_i}{\frac{1}{M}\sum_i \mathbb{P}(\tilde{P} \mid P_i)}.$$

Thus the final MMSE estimator is nothing but the exact application of NL-means, denoising each patch by matching it to the huge patch database. Clearly this is not just a theoretical algorithm. Web based application could provide a way to denoise online any image by organizing a huge patch data base. The final algorithm is summarized in Algorithm 6.

The main focus of [86] is, however, as we mentioned, elsewhere: it uses this shotgun denoising to estimate universal upper and lower bounds of the attainable PSNR by any patch based denoising algorithm. More precisely, the algorithm gives upper and lower bounds to the following problem: *Given a noisy patch $\tilde{P}$, given the law $p(P)$ of all patches in the world, find the best possible estimate (in the sense of MMSE).* The shotgun algorithm gives a best possible estimate for *any patch based denoising algorithm* of this kind.

The upper bound obtained by the authors turns out to be very close to results obtained with BM3D (see sec. 4.8), and the authors conclude that for small window sizes, or moderate to high noise levels, the chase for the best denoising algorithm might be close to the finish. More precisely, only fractions of decibels separate the current best algorithms from these demonstrated upper bounds. The EPLL method [145] can be viewed as a first (slightly) more practical realization of this quasi-optimality by a shotgun algorithm, and there is no doubt that other more practical ones will follow. We now describe how the [86] lower and upper bounds can be estimated from a sufficient set of natural images.

Algorithm 6 Shotgun NL-means

Input: Noisy image $\tilde{u}$ in vectorial form.
Input: Very large set of M patches P_i extracted from a large set of noiseless natural images.
Output: Denoised image $\hat{u}$.

for all patches $\tilde{P}$ extracted from $\tilde{u}$ **do**
Compute the MMSE denoised estimate of $\tilde{P}$

$$\hat{P} \simeq \frac{\sum_{i=1}^{M} \mathbb{P}(\tilde{P} \mid P_i) P_i}{\sum_{i=1}^{M} \mathbb{P}(\tilde{P} \mid P_i)}$$

where $\mathbb{P}(\tilde{P} \mid P_i)$ is known from (4.4).
end for
At each pixel $\mathbf{i}$ get $\hat{u}(\mathbf{i})$ as $\hat{P}(\mathbf{i})$, where the patch P is centered at $\mathbf{i}$.
(optional Aggregation) : for each pixel $\mathbf{j}$ of u, compute the denoised version $\hat{u}_{\mathbf{j}}$ as the average of all values $\hat{P}(\mathbf{j})$ for all patches containing $\mathbf{j}$. (This step in not considered in [86].)

The MSE for a given denoising algorithm can be obtained by randomly sampling patches P, then add noise to generate noisy patches $\tilde{P}$, and measure the reconstruction error $\|P - \hat{P}\|^2$. Then the mean reconstruction error is

$$\text{MSE} = \int \mathbb{P}(P) \int \mathbb{P}(\tilde{P} \mid P) \|P - \hat{P}\|^2 d\tilde{P} dP. \tag{4.6}$$

Conversely, one can start from a noisy patch $\tilde{P}$, measure the variance of $\mathbb{P}(P \mid \tilde{P})$ around it. This amounts according to the authors of [86] to compute the sum of weighted distances between the restored $\hat{P}$ and all possible P explanations:

$$\text{MSE} = \int \mathbb{P}(\tilde{P}) \int \mathbb{P}(P \mid \tilde{P}) \|P - \hat{P}\|^2 d\tilde{P} dP. \tag{4.7}$$

This last equation follows from (4.6) by the Bayes rule. For each noisy $\tilde{P}$ one can define its MMSE

$$\text{MMSE}(\tilde{P}) = E[\|\hat{P} - \tilde{P}\|^2 \mid \tilde{P}] = \int \mathbb{P}(P \mid \tilde{P})(P - \hat{P})^2 dP. \tag{4.8}$$

The main interest of this formulation is that it permits to prove that the MMSE is, of all denoising algorithms, the one that minimizes the overall MSE. Indeed, differentiating (4.7) with respect to $\hat{P}$ yields back the MMSE estimator (4.5). The best overall MMSE achievable by any given denoising algorithm therefore is

$$\text{MMSE} = \int \mathbb{P}(\tilde{P}) E[\|\hat{P} - \tilde{P}\|^2 \mid \tilde{P}] = \int \mathbb{P}(\tilde{P}) \mathbb{P}(P \mid \tilde{P})(P - \hat{P})^2 dP d\tilde{P}. \tag{4.9}$$

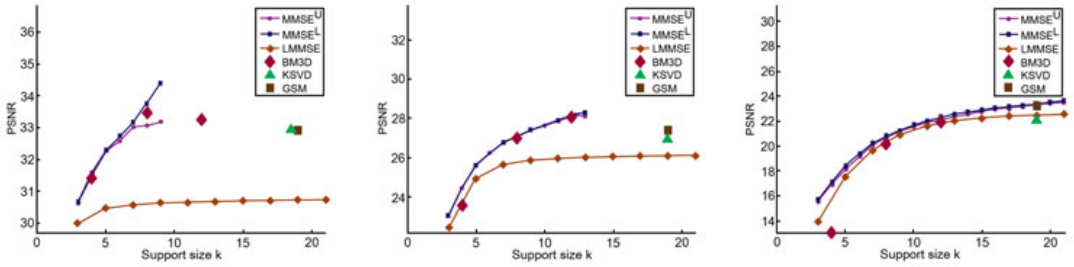
The goal of [86] is to bound the MMSE from below, ignoring of course the probability distribution $\mathbb{P}(P)$, but having enough samples of it. The main idea is to derive an upper and a lower bound on the MMSE from the two MSE formulations (4.6)-(4.7). Given a set of M clean and noisy pairs $\{(P_j, \tilde{P}_j)\}$, $j = 1 \ldots, M$ and another independent set of N clean patches $\{P_i\}$, $i = 1 \ldots, N$, both randomly sampled from natural images the proposed estimates are

$$\text{MMSE}^U = \frac{1}{M} \sum_j \|\hat{P}_j - P_j\|^2 \tag{4.10}$$

and

$$\text{MMSE}^L = \frac{1}{M} \sum_j \frac{\sum_i \mathbb{P}(\tilde{P}_j \mid P_i) \|\hat{P}_j - P_i\|^2}{\sum_i \mathbb{P}(\tilde{P}_j \mid P_i)}. \tag{4.11}$$

A striking feature of both estimates is that $MMSE^U$ uses the explicit knowledge of the original noise-free patch P_j, while MMSE^L does not involve it. Since MMSE^U simply measures the error for a given denoising algorithm, it obviously provides an upper bound for the MMSE of any other denoising algorithm. As the authors observe, MMSE^U and MMSE^L are random variables that depend on the choice of the samples. When the sample size approaches infinity, both converge to the exact MMSE. Nevertheless, [86] gives a simple proof that, for a finite sample, in expectation, MMSE^U and MMSE^L provide upper and lower bounds on the best possible MMSE. When both MMSE^U and MMSE^L coincide, they provide an accurate estimate of the optimal denoising possible with a given patch size.
For very high noise levels, the authors of [86] also tried to apply the linear minimum mean square error (LMMSE) estimator (or Wiener filter) using only the second order statistics of the data, by fitting a single k^2 dimensional Gaussian to the set of M image $k \times k$ patches. They conclude that even this simple approach is close to optimal for large noise.



$\sigma = 18$ $\sigma = 55$ $\sigma = 170$

Figure 4.4: Comparing [86] the PSNR ($= -10\log_{10}(\text{MMSE})$) of several denoising algorithms (K-SVD [92], BM3D [35], Gaussian Scale Mixture [114]) compared to the PNSR predicted by MMSE^U, MMSE^U. The performance of all algorithms is bounded by the MMSE^U estimate, but BM3D approaches this upper bound by fractional dB values. Nevertheless, the performance bounds consider more restrictive patch based algorithms than the class BM3D belongs to. Thus the actual gap to optimality may be higher.

4.5 The expected patch log likelihood (EPLL) method

The patch Gaussian mixture model This other shotgun method [145] is an almost literal application of the piecewise linear estimator (PLE) method [141], see section 4.9). But it is shotgun, namely applied to a huge set of patches instead of the image itself. A Gaussian mixture model is learnt from a set of 2.10^6 patches, sampled from the Berkeley database with their mean removed. The 200 mixture components with zero means and full covariance matrices are obtained using the EM (expectation maximization) algorithm. This training took about 30 hours with a public MATLAB code[2]. Thus were learnt: 200 means (actually they are all zero), 200 full covariance matrices and 200 mixing weights which constitute the Gaussian mixture model of this set of patches. Fig 4.5 shows some six bases extracted from the Gaussian mixture. Each one shows the patches that are eigenvectors of some of the covariance matrices, sorted by eigenvalue.
Once the Gaussian mixture is learnt, the denoising method maximizes the Expected Patch Log Likelihood (EPLL) while being close to the corrupted image in a way which is dependent on the (linear) corruption model. Given an image U (in vector form) the EPLL of U under prior $\mathbb{P}$ is defined by

$$EPLL_{\mathbb{P}}(U) = \sum_i \log \mathbb{P}(\mathbf{P}_i U)$$

[2] http://www.mathworks.com/matlabcentral/fileexchange/26184-em-algorithm-for-gaussian-mixture-model.

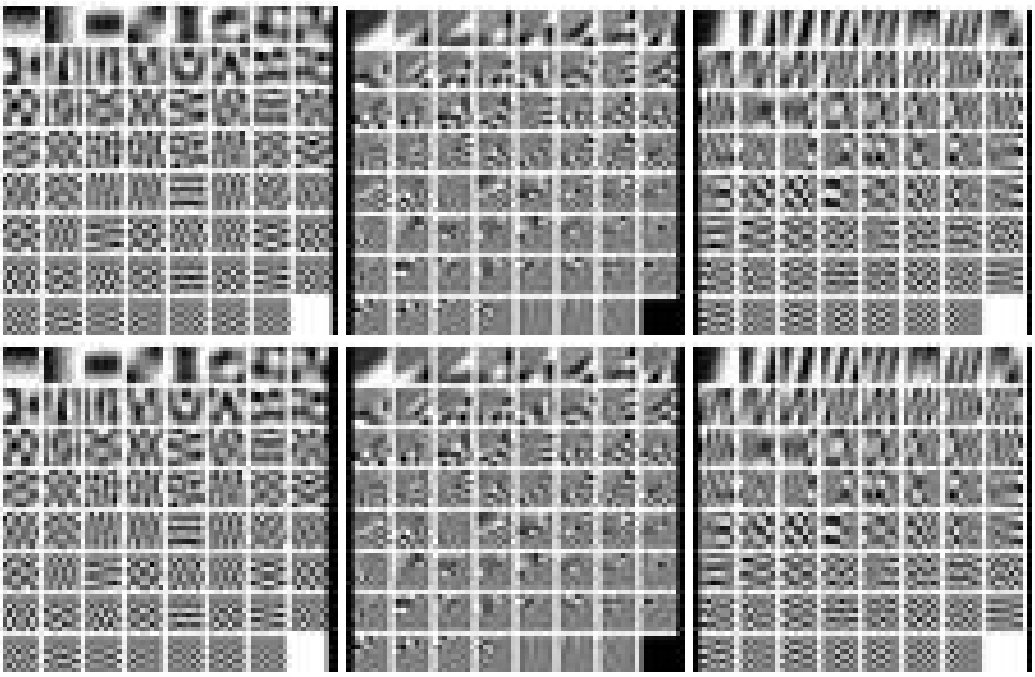


Figure 4.5: Eigenvectors of six randomly selected covariance matrices from the learnt Gaussian mixture model, sorted by eigenvalue from largest to smallest, from [145]. The authors notice the similarity of these basis elements to DCT, but also that many seem to model texture boundaries and edges at various orientations.

where $\mathbf{P}_i$ is a matrix which extracts the i-th patch P_i from the image U out of all overlapping patches, while $\log \mathbb{P}(\mathbf{P}_i X)$ is the likelihood of the i-th patch under the prior $\mathbb{P}$. Assuming a patch location in the image is chosen uniformly at random, EPLL can be interpreted as the expected log likelihood of a patch in the image (up to a multiplication by $1/M$). Given a corrupted image $\tilde{U}$ in vector form and a model of image corruption of the form $||\mathbf{A}U - \tilde{U}||^2$, the restoration is made by minimizing

$$f_{\mathbb{P}}(U|\tilde{U}) = \frac{\lambda}{2}||\mathbf{A}U - \tilde{U}||^2 - EPLL_{\mathbb{P}}(U).$$

According to the authors, "This equation has the familiar form of a likelihood term and a prior term, but note that $EPLL_{\mathbb{P}}(U)$ is not the log probability of a full image. Since it sums over the *log* probabilities of all overlapping patches, it "double counts" the log probability. Rather, it is the expected log likelihood of a randomly chosen patch in the image."
The optimization is made by "half quadratic splitting" which amounts to introduce auxiliary patch variables Z^i, $i = 1, \dots M$, one for each patch P_i, and to minimize alternatively the auxiliary functional

$$C_{\mathbb{P},\beta}(U, \{Z^i\}|\tilde{U}) := \frac{\lambda}{2}||\mathbf{A}U - \tilde{U}||^2 + \frac{\beta}{2}\sum_i ||\mathbf{P}_i U - Z^i||^2 - \log \mathbb{P}(Z^i).$$

Solving for U given $\{Z^i\}$ amounts to the inversion

$$U = \left(\lambda \mathbf{A}^T\mathbf{A} + \beta \sum_i \mathbf{P}_i^T \mathbf{P}_i\right)^{-1} \left(\lambda \mathbf{A}^T \tilde{U} + \beta \sum_i \mathbf{P}_i^T Z^i\right).$$

In the case of denoising, $\mathbf{A}$ is simply the identity, and the above formula boils down to computing for each pixel $\mathbf{j}$ a denoised value $U(\mathbf{j})$ as a weighted average over all patches P_i containing this given pixel $\mathbf{j}$ of the noisy pixel value $\tilde{U}(\mathbf{j})$ and of the patch denoised values $Z_i(\mathbf{j})$:

$$U(\mathbf{j}) = \frac{\lambda \tilde{U}(\mathbf{j}) + \sum_{P_i \ni \mathbf{j}} Z_i(\mathbf{j})}{\lambda + \beta k^2}, \tag{4.12}$$

where k^2 is the patch size.
Then, solving for $\{Z_i\}$ given U amounts to solving a MAP (maximum a posteriori) problem of estimating the most likely patch under the prior $\mathbb{P}$, given $\mathbf{P}_i U$ and parameter β.
The Gaussian mixture model being known, calculating the log likelihood of a given patch is trivial:

$$\log \mathbb{P}(Q) = \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(Q|\mu_k, \mathbf{C}_k) \right),$$

where π_k are the mixing weights for each of the mixture component, μ_k and $\mathbf{C}_k$ are the corresponding mean and covariance matrix.
Given a noisy patch $\tilde{Q}$, the MAP estimate is computed with the following procedure:

Algorithm 7 Patch restoration once the Patch Gaussian mixture is known

for each noisy patch $\tilde{Q}$ **do**

Compute the conditional mixture weights $\pi'_k = \mathbb{P}(k \mid \tilde{Q})$ (given by EM);

Pick the component k with the highest conditional mixing weight: $k_{max} = \max_k \pi'_k$;

The MAP estimate $\hat{Q}$ is a Wiener solution for the k_{max}-th component:

$$\hat{Q} = \left(\mathbf{C}_{k_{max}} + \sigma^2 \mathbf{I}\right)^{-1} \left(\mathbf{C}_{k_{max}} \tilde{Q} + \sigma^2 \mu_{k_{max}}\right).$$

end for

The authors comment that this is one iteration of the "hard version" of the EM algorithm for finding the modes of a Gaussian mixture [21]. The method can be used for denoising and several experiments seems to indicate that it equals the performance of BM3D and LLSC [92].

4.6 The Portilla et al. wavelet neighborhood denoising (BLS-GSM)

The basic idea of this algorithm is modeling a noiseless "wavelet coefficient neighborhood", P, by a Gaussian scale mixture (GSM) which is defined as

$$P = \sqrt{z}U$$

where U is a zero-mean Gaussian random vector and z is an independent positive scalar random variable. The wavelet coefficient neighborhood turns out to be a patch of an oriented channel of the image at a given scale, complemented with a coefficient of the channel at the same orientation and the next lower scale. Thus, we adopt again the patch notation P. (Arguably, this method is the first patch-based method.) Using a GSM model for P estimated from the image itself, the method makes a Bayes least square (BLS) estimator. For this reason, the method will be called here BLS-GSM (Bayes least square estimate of Gaussian scale mixture; the authors called it simply BLS.) Without loss of generality it is assumed that $Ez = 1$ and therefore the random variables U and P have similar covariances. To use the GSM model for wavelet patch denoising, the noisy input image is first decomposed into a wavelet pyramid, and each image of the pyramid will be separately denoised. The resulting denoised image is obtained by the reconstruction algorithm from the wavelet coefficients. To avoid ringing artifacts in the reconstruction, a redundant version of the wavelet transform, the so-called steerable pyramid, is used. For a $n_1 \times n_2$ image, the pyramid, $\mathcal{P}$, is generated on $\log_2(\min(n_1, n_2) - 4)$ scales and eight orientations using the following procedure. First the input image is decomposed into one low-pass and eight oriented high-pass component images using two polar filters in quadrature in the Fourier domain (the sum of their squares is equal to 1). The Fourier domain being represented in polar coordinates (r, θ), the low pass and high pass isotropic filters are

$$l(r) = \begin{cases} 1 & 0 \le r < 0.5; \\ \cos(\frac{\pi}{2}(-\log_2 r - 1)) & 0.5 \le r < 1; \\ 0 & 1 \le r \le \sqrt{2}; \end{cases} \tag{4.13}$$

and

$$h(r) = \begin{cases} 0 & 0 \le r < 0.5; \\ \cos(\frac{\pi}{2}(\log_2 r)) & 0.5 \le r < 1; \\ 1 & 1 \le r \le \sqrt{2}. \end{cases} \tag{4.14}$$

The high pass filter h is decomposed again into eight oriented components,

$$a_k(r, \theta) = h(r)g_k(\theta), \quad k \in [0, K-1], \tag{4.15}$$

where $K = 8$, and

$$g_k(\theta) = \frac{(K-1)}{\sqrt{K[2(K-1)]}} \left[2\cos\left(\theta - \frac{\pi k}{K}\right)\right]^{K-1}. \tag{4.16}$$

Then the steerable pyramid is generated by iteratively applying the a_k filters to the result of the low-pass filter to obtain bandpass images, and calculating the residual using the l filter followed by sub-sampling. For example in the case of a 512×512 image we have a 5 scales pyramid consisting of 49 sub-bands: 8 high-pass oriented sub-bands, from $\mathcal{P}^1$ to $\mathcal{P}^8$, 8 bandpass oriented sub-bands for each scale, from $\mathcal{P}^9$ to $\mathcal{P}^{48}$, in addition to one lowpass non-oriented residual subband, $\mathcal{P}^{49}$. (WLOG we shall keep this 49 number as landmark, but this number depends of course on the

image size). Assume now that the image has been corrupted by independent additive Gaussian noise. Therefore, a typical neighborhood of wavelet coefficients can be represented as

$$\tilde{P} = P + N = \sqrt{z}U + N, \tag{4.17}$$

where noise, N, and P are considered to be independent. Define $p_s(i,j)$ as the sample at position (i,j) of the sub-band $\mathcal{P}^s$, the subbands being enumerated as (e.g.) $s = 1, \ldots, 49$. The neighborhood of the wavelet coefficient $p_s(i,j)$ is composed of its spatial neighbors for the same sub-band s. It could have contained also coefficients from other sub-bands at the same scale as $p_s(i,j)$ but with different orientations, and could finally also contain sub-band coefficients from the adjacent scales, up and down. Surprisingly, the final neighborhood is quite limited: The authors sustain that the best efficiency is reached with a 3×3 spatial block around $p_s(i,j)$, supplemented with one coefficient at the same location and at the next coarser scale (considering its up-sampled parent by interpolation) with the same orientation. Hence, the neighborhood size is 10 and contains only $\{p_s(i-1,j-1), \ldots, p_s(i+1,j+1), p_{s+8}(i,j)\}$. There are two exceptions for this: first is the neighborhood of coarsest scale coefficients (without any coarser scale) has necessarily only 9 surrounding coefficients. Second, the boundary coefficients are processed using special steps described below. Using the observed noisy vector, $\tilde{P}$, an estimation of P, can be obtained by

$$E(P \mid \tilde{P}) = \int_0^\infty \mathbb{P}(z \mid \tilde{P}) E(P \mid \tilde{P}, z) dz.$$

This estimation is the Bayesian denoised value of the reference coefficient. The integral is computed numerically on experimentally obtained sampled intervals of z. Here, only 13 equally spaced values of z in the interval $[\ln(z_{min}), \ln(z_{max})] = [-20.5, 3.5]$ are used. Therefore $E(P \mid \tilde{P})$ is computed as

$$E(P \mid \tilde{P}) = \sum_{i=1}^{13} \mathbb{P}(z_i \mid \tilde{P}) E(P \mid \tilde{P}, z_i). \tag{4.18}$$

The only question left is: how to compute the conditional probability and the conditional expectation, $\mathbb{P}(z_i \mid \tilde{P})$ and $E(P \mid \tilde{P}, z_i)$. For each sub-band, $\mathcal{P}^s$ except the low-pass residual, $\mathcal{P}^{49}$ which remains unchanged, define $\mathbf{C}_N^s$ and $\mathbf{C}_{\tilde{P}}^s$, respectively the noise and the observation covariance matrices of the wavelet neighborhood. If n^s denotes the size of neighborhood at sub-band $\mathcal{P}^s$ (n^s therefore is 10 or 9 as explained above), $\mathbf{C}_N^s$ is a $n^s \times n^s$ matrix which can be experimentally generated by first decomposing a delta function $\sigma\delta$ on the steerable pyramid. Here σ is the known noise variance and δ is an $n_1 \times n_2$ image defined by

$$\delta(i,j) = \begin{cases} 1 & (i,j) = (\frac{n_1}{2}, \frac{n_2}{2}), \\ 0 & \text{otherwise.} \end{cases}$$

(This covariance matrix is equal to the covariance of the white noise defined as a band-limited function obtained by randomizing uniformly the phase of the Fourier coefficients of the discrete Dirac mass δ.) Using the steerable pyramid decomposition of $\sigma\delta$, define $\mathbf{N}_s$ as the matrix which has for rows all neighborhoods of the sub-band $\mathcal{P}_s$. This is a matrix with n_s columns and $(n_1-2)(n_2-2)$ rows. (Subtracting 2 is for eliminating the boundary coefficients). The covariance $\mathbf{C}_N^s$ matrix of the neighborhood samples for each sub-band is computed as

$$\mathbf{C}_N^s = \frac{\mathbf{N}_s^T \mathbf{N}_s}{(n_1-2)(n_2-2)},$$

where $(.)^T$ stands for matrix transposition. Since all the noise removal steps are calculated for each sub-band separately, in the following we skip the superscript s to simplify the notation. Similarly but using the pyramid of observed noisy samples, $\mathbf{C}_{\tilde{P}}$ can be computed. Using (4.17) and the assumption $Ez = 1$, for each sub-band s we have

$$\mathbf{C}_U = \mathbf{C}_{\tilde{P}} - \mathbf{C}_N.$$

$\mathbf{C}_U$ can be forced to be positive semi-definite by setting to zero all of its negative eigenvalues. We can now calculate $E(P \mid \tilde{P}, z_i)$. Using the fact that P and $\underline{\mathrm{N}}$ are Gaussian independent variables and also that the noise is additive, $E(P \mid \tilde{P}, z_i)$ is simply a local Wiener estimate:

$$E(P \mid \tilde{P}, z) = \frac{z\mathbf{C}_U}{z\mathbf{C}_U + \mathbf{C}_N}\tilde{P},$$

where the matrix fraction notation is understood as $\frac{\mathbf{C}}{\mathbf{W}} := \mathbf{C}\mathbf{W}^{-1}$. Clearly it would be cumbersome to compute as many matrix inversions as z_i's. Fortunately, with a bit of linear algebra this computation can be rendered common to all z_i. Define $\{\mathbf{Q}, \mathbf{\Lambda}\}$ as the eigenvectors and eigenvalues of $\mathcal{S}^{-1}\mathbf{C}_U\mathcal{S}^{-T}$, where $\mathcal{S}_{n^s \times n^s}$ is the symmetric square root of $\mathbf{C}_N$, $\mathbf{C}_N = \mathcal{S}\mathcal{S}^T$. So we have $\mathcal{S}^{-1}\mathbf{C}_U\mathcal{S}^{-T} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. Furthermore, set $\mathbf{M} = \mathcal{S}\mathbf{Q}, \quad \mathbf{v} = \mathbf{M}^{-1}\tilde{P}$. Then we have

$$\begin{aligned}
E(P \mid \tilde{P}, z) &= \frac{z\mathbf{C}_U}{z\mathbf{C}_U + \mathbf{C}_N}\tilde{P} \\
&= \frac{z\mathbf{C}_U}{z\mathbf{C}_U + \mathcal{S}\mathcal{S}^T}\tilde{P} \\
&= \frac{z\mathbf{C}_U}{\mathcal{S}(z\mathcal{S}^{-1}\mathbf{C}_U\mathcal{S}^{-T} + \mathbf{I})\mathcal{S}^T}\tilde{P} \\
&= \frac{z\mathbf{C}_U}{\mathcal{S}\mathbf{Q}(z\mathbf{\Lambda} + \mathbf{I})\mathbf{Q}^T\mathcal{S}^T}\tilde{P} \\
&= z\mathbf{C}_U\mathcal{S}^{-T}\mathbf{Q}(z\mathbf{\Lambda} + \mathbf{I})^{-1}\mathbf{Q}^T\mathcal{S}^{-1}\tilde{P} \\
&= z\mathcal{S}\mathcal{S}^{-1}\mathbf{C}_U\mathcal{S}^{-T}\mathbf{Q}(z\mathbf{\Lambda} + \mathbf{I})^{-1}\mathbf{Q}^T\mathcal{S}^{-1}\tilde{P} \\
&= z\mathcal{S}\mathbf{Q}\mathbf{\Lambda}(z\mathbf{\Lambda} + \mathbf{I})^{-1}\mathbf{Q}^T\mathcal{S}^{-1}\tilde{P} \\
&= z\mathbf{M}\mathbf{\Lambda}(z\mathbf{\Lambda} + \mathbf{I})^{-1}\mathbf{v}.
\end{aligned}$$

The interest is that one can calculate $\mathbf{M}$, $\mathbf{\Lambda}$ and $\mathbf{v}$ once for each subband. The scalar final formulation of the above equation is

$$E(P \mid \tilde{P}, z_i)_c = \sum_{j=1}^{n^s} \frac{z_i m_{c,j}\lambda_{j,j}v_j}{z_i\lambda_{j,j} + 1}, \tag{4.19}$$

where $m_{c,j}$, $\lambda_{j,j}$ and v_j are the elements of $\mathbf{M}$, $\mathbf{\Lambda}$ and $\mathbf{v}$ respectively, and c is the index of the reference coefficient in the neighborhood.
The second component of (4.18) is $\mathbb{P}(z_i \mid \tilde{P})$, which can be obtained using the Bayes rule ($p_z(z)$ denotes the density function of the random variable z:

$$\mathbb{P}(z_i \mid \tilde{P}) = \frac{\mathbb{P}(\tilde{P} \mid z_i)p_z(z_i)}{\int_0^\infty \mathbb{P}(\tilde{P} \mid \alpha)p_z(\alpha)d\alpha}$$

or its discrete form

$$\mathbb{P}(z_i \mid \tilde{P}) = \frac{\mathbb{P}(\tilde{P} \mid z_i)p_z(z_i)}{\sum_{j=1}^{13} \mathbb{P}(\tilde{P} \mid z_j)p_z(z_j)}. \tag{4.20}$$

where the density of observed noisy neighborhood vector $\tilde{P}$ conditioned on z_i is a zero-mean Gaussian with covariance

$$\mathbf{C}_{\tilde{P}|z_i} := z_i\mathbf{C}_U + \mathbf{C}_N,$$

so that

$$\mathbb{P}(\tilde{P} \mid z_i) = \frac{e^{-\frac{-\tilde{P}^T(z\mathbf{C}_U + \mathbf{C}_N)^{-1}\tilde{P}}{2}}}{\sqrt{|z\mathbf{C}_U + \mathbf{C}_N|}} \quad .$$

Using the above definitions of $\mathbf{v}$ and $\mathbf{\Lambda}$ and the same simplifications as for $E(P \mid \tilde{P}, z_i)$ we obtain

$$\mathbb{P}(\tilde{P} \mid z_i) = \frac{e^{-\frac{1}{2}\sum_{j=1}^{n^s} \frac{v_j^2}{z_j \lambda_{j,j}+1}}}{\sqrt{\Pi_{j=1}^{n^s}(z_i\lambda_{j,j}+1)}}, \tag{4.21}$$

The only question left is the form of $p_z(z)$. The authors, after a somewhat puzzling discussion, adopt "a non-informative Jeffrey prior" $p_z(z) \simeq \frac{1}{z}$. Since this function cannot be a density, being non integrable, the function is actually cut off to zero near $z = 0$.
To summarize, the Portilla et al. algorithm is:

Algorithm 8 Portilla et al. wavelet neighborhood denoising (BLS-GSM)

Input: noisy image
Output: denoised image

Parameters: $n_1 \times n_2$ the image size, number of pyramid scales $\log_2(\min(n_1, n_2) - 4)$.
Parameter s, enumeration of all oriented channels at each scale (8 per scale).
Establish n^s, dimension of wavelet neighborhood coefficient (10 or 9).

Apply the wavelet pyramid (4.13)-(4.16), respectively to the noise image δ and to the observed image.
Regroup the obtained wavelet coefficients to obtain $\tilde{P}^s$, the wavelet coefficient neighborhoods of rank s and N^s the noise wavelet coefficient neighborhoods of rank s.
for each filter index s **do**
  Compute $\mathbf{C}_N^s$ and $\mathbf{C}_{\tilde{P}}^s$, noise and observation covariance matrices of N_s and $\tilde{P}_s$. (In the sequel the subscript s is omitted.) Deduce $\mathbf{C}_U = \mathbf{C}_{\tilde{P}} - \mathbf{C}_N$.
  Compute $\{\mathbf{Q}, \mathbf{\Lambda}\}$ the eigenvectors and eigenvalues of $\mathcal{S}^{-1}\mathbf{C}_U\mathcal{S}^{-T}$, where $\mathcal{S}$ is the symmetric square root of $\mathbf{C}_N$, $\mathbf{C}_N = \mathcal{S}\mathcal{S}^T$.
end for
for each wavelet coefficient neighborhood $\tilde{P}$ and $i \in \{1, \ldots, 13\}$ **do**
  Compute $\mathbf{M} = \mathcal{S}\mathbf{Q}, \quad \mathbf{v} = \mathbf{M}^{-1}\tilde{P}$
  Using (4.19) obtain $E(P \mid \tilde{P}, z_i)_c = \sum_{j=1}^{n^s} \frac{z_i m_{c,j} \lambda_{j,j} v_j}{z_i \lambda_{j,j}+1}$, where $m_{c,j}$, $\lambda_{j,j}$ and v_j are the elements of $\mathbf{M}$, $\mathbf{\Lambda}$ and $\mathbf{v}$ respectively, and c is the index of the reference coefficient in the neighborhood.
  Apply (4.20) to get $\mathbb{P}(z_i \mid \tilde{P}) = \frac{\mathbb{P}(\tilde{P}|z_i)p_z(z_i)}{\sum_{j=1}^{13}\mathbb{P}(\tilde{P}|z_j)p_z(z_j)}$, using the value obtained by (4.21) for $\mathbb{P}(\tilde{P} \mid z_i) = \frac{e^{-\frac{1}{2}\sum_{j=1}^{n^s} \frac{v_j^2}{z_j \lambda_{j,j}+1}}}{\sqrt{\Pi_{j=1}^{n^s}(z_i\lambda_{j,j}+1)}}$.
  By (4.18) finally obtain $E(P \mid \tilde{P}) = \sum_{i=1}^{13} \mathbb{P}(z_i \mid \tilde{P}) E(P \mid \tilde{P}, z_i)$ where $p_z(z) \simeq \frac{1}{z}$ and z_i are quantized uniformly on the interval $[\ln(z_{min}); \ln(z_{max})] = [-20.5, 3.5]$.
end for
Reconstruct the restored image from its restored neighborhood coefficients $E(P \mid \tilde{P})$ by the inverse steerable pyramid.

As we shall see in the synthesis, in spite of its formalism, this method is actually extremely similar to other patch-based Bayesian methods. It has received a more recent extension, reaching state of the art performance, in [89]. This paper proposes an extension of the above method modeling the wavelet coefficients as a global random field of Gaussian scale mixtures.

4.7 K-SVD

The K-SVD method was introduced in [1] where the whole objective was to optimize the quality of sparse approximations of vectors in a learnt dictionary. Even if this article noticed the interest of the technique in image processing tasks, it is in [53] that a detailed study has been led on the denoising of grey-level images. Then, the adjustment to colour images has been treated in [93]. Let us notice that this last article proved that the K-SVD method can also be useful in other image processing tasks, such as non-uniform denoising, demoisaicing and inpainting. For a detailed description of K-SVD the reader is referred to [90], [92], and the dedicated chapter 10.
The algorithm is divided in three steps. In the two first steps an optimal dictionary and a sparse representation is built for each patch in the image, using among other tools a singular value decomposition (SVD). In the last step, the restored image is built by aggregating the computed sparse representations of all image patches. The algorithm requires an initialization of the dictionary which is updated during the process. The dictionary initialization may contain usual orthogonal basis (discrete cosine transform, wavelets...), or patches from clean images or even from the noisy image itself.

The first step looks for sparse representations of all patches of size κ^2 in the noisy image in vector form $\tilde{U}$ using a fixed dictionary $\mathbf{D}$. A dictionary is represented as a matrix of size $\kappa^2 \times n_{dic}$, with $n_{dic} \geq \kappa^2$, whose columns (the "atoms of the dictionary") are normalized (in Euclidean norm). For each noisy patch $\mathbf{R_i}\tilde{U}$, (where the index $\mathbf{i}$ indicates that the top left corner of the patch is the pixel $\mathbf{i}$, and $\mathbf{R_i}$ is the matrix extracting the patch vector from $\tilde{U}$) a "sparse" column vector $\alpha_\mathbf{i}$ (of size n_{dic}) is calculated by optimization. This vector of coefficients should have only a few non-zero coefficients, the distance between $\mathbf{R_i}\tilde{U}$ and its sparse approximation $\mathbf{D}\alpha_\mathbf{i}$ remaining as small as possible. The dictionary allows one to compute a sparse representation $\alpha_\mathbf{i}$ of each patch $\mathbf{R_i}\tilde{U}$. These sparse vectors are assembled in a matrix α with κ^2 rows and N_p columns where N_p is the number of patches of dimension κ^2 of the image.
More precisely, an ORMP (Orthogonal Recursive Matching Pursuit) gives an approximate solution of the (NP-complete) problem

$$\underset{\alpha_\mathbf{i}}{\text{Arg min}} \, ||\alpha_\mathbf{i}||_0 \quad \text{such that} \quad ||\mathbf{R_i}\tilde{U} - \mathbf{D}\alpha_\mathbf{i}||_2^2 \leq \kappa^2 (C\sigma)^2 \tag{4.22}$$

where $\|\alpha_\mathbf{i}\|_0$ refers to the l^0 norm of $\alpha_\mathbf{i}$, i.e. the number of non-zero coefficients of $\alpha_\mathbf{i}$. The additional constraint guarantees that the residual has an l^2 norm lower than $\kappa C\sigma$. C is an user parameter.

The second step tries to update one by one the columns of the dictionary $\mathbf{D}$ and the representations α to improve the overall fidelity of the patch approximation. The goal is to decrease the quantity

$$\sum_\mathbf{i} \|\mathbf{D}\alpha_\mathbf{i} - \mathbf{R_i}\tilde{U}\|_2^2 \tag{4.23}$$

while keeping the sparsity of the vectors $\alpha_\mathbf{i}$. We will denote by $\hat{d}_l$ $(1 \leq l \leq n_{dic})$ the columns of the dictionary $\hat{\mathbf{D}}$. First, the quantity (4.23) is minimized without taking care of the sparsity. The atom $\hat{d}_l$ and the coefficients $\hat{\alpha}_\mathbf{i}(l)$ are modified to make the approximations of all the patches more efficient. For each $\mathbf{i}$, introduce the residue

$$e_\mathbf{i}^l = \mathbf{R_i}\tilde{U} - \hat{\mathbf{D}}\hat{\alpha}_\mathbf{i} + \hat{d}_l\hat{\alpha}_\mathbf{i}(l) \tag{4.24}$$

which is the error committed by deciding not to use $\hat{d}_l$ any more in the representation of the patch $R_\mathbf{i}\tilde{U}$. Thus $e_\mathbf{i}^l$ is a vector of size κ^2.

These residues are grouped together in a matrix $\mathbf{E}_l$ (whose columns are indexed by $\mathbf{i}$). The values of the coefficients $\hat{\alpha_\mathbf{i}}(l)$ are also grouped in a row vector denoted by $\hat{\alpha}^l$. Therefore, $\mathbf{E}_l$ is a matrix of size $\kappa^2 \times N_p$ (recall that N_p is the total number of patches in the image) and $\hat{\alpha}^l$ is a row vector of size N_p. We must try to find a new $\hat{d}_l$ and a new row vector $\hat{\alpha}^l$ minimizing

$$\sum_{\mathbf{i}} \|\hat{\mathbf{D}}\hat{\alpha_\mathbf{i}} - \hat{d}_l\hat{\alpha_\mathbf{i}}(l) + d_l\alpha^l - \mathbf{R_i}\tilde{U}\|_2^2 = ||\mathbf{E}_l - d_l\alpha^l||_F^2 \tag{4.25}$$

where the squared Frobenius norm $\|\mathbf{M}\|_F^2$ refers to the sum of the squared elements of $\mathbf{M}$. This Frobenius norm is also equal to the sum of the squared (Euclidean) norms of the columns, and one can be convinced that minimizing (4.25) amounts to reduce the approximation error caused by $\hat{d}_l$. It is well-known that the minimization of such a Frobenius norm consists in a rank-one approximation, which always admits a solution, practically given by the singular value decomposition (SVD). Using the SVD of $\mathbf{E}_l$:

$$\mathbf{E}_l = \mathcal{U}\mathbf{\Delta}\mathbf{V}^T$$

(where $\mathcal{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{\Delta}$ is non-negative and decreasing), the updated values of $\hat{d}_l$ and $\hat{\alpha}^l$ are respectively the first column of $\mathcal{U}$ and the first column of $\mathbf{V}$ multiplied by $\mathbf{\Delta}(1,1)$.

After K iterations of these two steps, a denoised patch $\hat{\mathbf{D}}\hat{\alpha_\mathbf{i}}$ is available for each patch position $\mathbf{i}$, where $\hat{\mathbf{D}}$ is the final updated dictionary. The third and last (aggregation) step consists in merging the denoised versions of all patches of the image in order to obtain a global estimate. This is achieved by solving the minimization problem

$$\hat{U} = \underset{U_0 \in \mathbb{R}^M}{\operatorname{Arg\,min}} \ \lambda||U_0 - \tilde{U}||_2^2 \ + \ \sum_{\mathbf{i}} ||\hat{\mathbf{D}}\hat{\alpha_\mathbf{i}} - \mathbf{R_i}U_0||_2^2 \ ,$$

by the closed formula

$$\hat{U} = \left(\lambda\mathbf{I} + \sum_{\mathbf{i}} \mathbf{R_i}^T\mathbf{R_i}\right)^{-1} \left(\lambda\tilde{U} + \sum_{\mathbf{i}} \mathbf{R_i}^T\hat{\mathbf{D}}\hat{\alpha_\mathbf{i}}\right) \ . \tag{4.26}$$

This amounts for each pixel to average its initial noisy value with the average of all estimates obtained with all patches containing it. The parameter λ controls the tradeoff between these two values and thus measures the fidelity to the initial noisy image.

Mairal et al. [93] proposed to directly extend the algorithm to vector valued images instead of converting the colour image to another colour system decorrelating geometry and chromaticity. The previous algorithm is applied on column vectors which are a concatenation of the R,G,B values. In this way, the algorithm, when updating the dictionary, takes into account the inter-channel correlation. We shall detail the algorithm for grey level images, the colour version simply requires an adaptation of the Euclidean norm to the colour space.

Algorithm 9 K-SVD algorithm for grey level images

Input: noisy image $\tilde{u}$, $\tilde{U}$ in vector form, noise standard deviation σ.
Input: κ^2, dimension of patches (number of pixels).
Input n_{dic}, dictionary size, K iteration number of the dictionary optimization.
Input: initial patch dictionary $\mathbf{D}_{init}$ as matrix with n_{dic} columns and κ^2 rows.
Output: output image in vector form $\hat{U}$.

Collect all noisy patches of dimension κ^2 in column vectors $\mathbf{R_i}\tilde{U}$
Set $\hat{\mathbf{D}} = \mathbf{D}_{init}$.

for k=1 to K **do**

An ORMP is applied to the vectors $\mathbf{R_i}\tilde{U}$ in a such way that a vector of sparse coefficients $\hat{\alpha}_\mathbf{i}$ is obtained verifying $\mathbf{R_i}\tilde{U} \approx \mathbf{D}\hat{\alpha}_\mathbf{i}$.

Introduce $\omega_l = \{\, \mathbf{i} \,|\, \hat{\alpha}_\mathbf{i}(l) \neq 0 \,\}$;
For $\mathbf{i} \in \omega_l$, obtain the residue

$$e_\mathbf{i}^l = \mathbf{R_i}\tilde{U} - \hat{\mathbf{D}}\hat{\alpha}_\mathbf{i} + \hat{d}_l\hat{\alpha}_\mathbf{i}(l) \;;$$

Put these column vectors together in a matrix $\mathbf{E}_l$. Values $\hat{\alpha}_\mathbf{i}(l)$ are also assembled in a row vector denoted by $\hat{\alpha}^l$ for $\mathbf{i} \in \omega_l$;
Update $\hat{d}_l$ and $\hat{\alpha}^l$ as solutions of the minimization problem :

$$(\hat{d}_l, \hat{\alpha}^l) = \underset{d_l, \alpha^l}{\mathrm{Arg\,min}} \|\mathbf{E}_l - d_l\alpha^l\|_F^2 \;.$$

A truncated SVD is applied to the matrix $\mathbf{E}_l$. It provides partially $\mathcal{U}$, $\mathbf{V}$ (orthogonal matrices) and $\mathbf{\Delta}$ (filled in with zeroes except on its first diagonal), such that $\mathbf{E}_l = \mathcal{U}\mathbf{\Delta}\mathbf{V}^T$. Then $\hat{d}_l$ is defined again as the first column of $\mathcal{U}$ and $\hat{\alpha}^l$ as the first column of $\mathbf{V}$ multiplied by $\mathbf{\Delta}(1,1)$.
end for

Aggregation: for each pixel the final result $\hat{U}$ in vector form is obtained thanks to the weighted aggregation:

$$\hat{U} = \left(\lambda\mathbf{I} + \sum_\mathbf{i} \mathbf{R_i^t}\mathbf{R_i}\right)^{-1} \left(\lambda\tilde{U} + \sum_\mathbf{i} \mathbf{R_i^t}\hat{\mathbf{D}}\hat{\alpha}_\mathbf{i}\right) \;.$$

4.8 BM3D

BM3D is a sliding window denoising method extending DCT denoising and NL-means. Instead of adapting locally a basis or choosing from a large dictionary, it uses a fixed basis. The main difference with DCT denoising is that a set of similar patches are used to form a 3D block which is filtered by using a 3D transform, hence its name *Collaborative filtering*. The method has four steps: a) finding the image patches similar to a given image patch and grouping them in a 3D block b) 3D linear transform of the 3D block; c) shrinkage of the transform spectrum coefficients; d) inverse 3D transformation. This 3D filter therefore filters out simultaneously all 2D image patches in the 3D block. By attenuating the noise, collaborative filtering reveals even the finest details shared by the grouped patches. The filtered patches are then returned to their original positions and an adaptive aggregation procedure is applied by taking into account the number of kept coefficients per patch during the thresholding process (see chapter 3 for more details on aggregation).
The first collaborative filtering step is much improved in a second step using an oracle Wiener filtering. This second step mimics the first step, with two differences. The first difference is that it compares the filtered patches instead of the original patches like described in chapter 3. The second difference is that the new 3D group (built with the unprocessed image samples, but using the patch distances of the filtered image) is processed by an oracle Wiener filter using coefficients from the denoised image obtained at the first step to approximate the true coefficients given by Theorem 1. The final aggregation step is identical to those of the first step.
The algorithm is extended to colour images through the $Y_oU_oV_o$ colour system. The previous strategy is applied independently to each channel with the exception that similar patches are always selected by computing distances in the channel Y_o.

Algorithm 10 BM3D first iteration algorithm for grey images.

Input: noisy image $\tilde{u}$, σ, noise standard deviation.
Output: output basic estimation $\hat{u}_1$ of the denoised image.

Set parameter $\kappa \times \kappa = 8 \times 8$: dimension of patches.
Set parameter $\lambda \times \lambda = 39 \times 39$: size of search zone in which similar patches are searched.
Set parameter $N_{max} = 16$: maximum number of similar patches retained during the grouping part.
Set parameter $s = 3$: step in both rows and columns between two reference patches.
Set parameter $\lambda_{3D} = 2.7$: coefficient used for the hard thresholding.
Set parameter $\tau = 2500$ (if $\sigma > 40, \tau = 5000$): threshold used to determine similarity between patches.

for each pixel **i**, with a step s in rows and columns **do**

Select a square reference patch $\tilde{P}$ around **i** of size $\kappa \times \kappa$.

Look for square patches $\tilde{Q}$ in a square neighborhood of **i** of size $\lambda \times \lambda$ having a distance to $\tilde{P}$ lower than τ.

if there are more than N_{max} similar patches **then**
keep only the N_{max} closest similar patches to $\tilde{P}$ according to their Euclidean distance.
else
keep 2^p patches, where p is the largest integer such that 2^p is smaller than the number of similar patches
end if

A 3D group $\mathcal{P}(\tilde{P})$ is built with those similar patches.

A bi-orthogonal spline wavelet (Bior 1.5) is applied on every patch contained in $\mathcal{P}(\tilde{P})$.

A Walsh-Hadamard transform is then applied along the third dimension of the 3D group $\mathcal{P}(\tilde{P})$.

A hard thresholding with threshold $\lambda_{3D}\sigma$ is applied to $\mathcal{P}(\tilde{P})$. An associated weight $w_{\tilde{P}}$ is computed :

$$w_{\tilde{P}} = \begin{cases} (N_{\tilde{P}})^{-1} & N_{\tilde{P}} \geq 1 \\ 1 & N_{\tilde{P}} = 0 \end{cases}$$

where $N_{\tilde{P}}$ is the number of retained (non-zero) coefficients.

The estimate $\hat{u}_1^{\tilde{Q},\tilde{P}}$ for each pixel **i** in similar patches $\tilde{Q}$ of the 3D group $\mathcal{P}(\tilde{P})$ is then obtained by applying the inverse of the Walsh-Hadamard transform along the third dimension, followed by the inverse of the bi-orthogonal spline wavelet on every patches of the 3D group.
end for

for each pixel **i do**
Aggregation: recover the denoised value at **i** by averaging all estimatesof all patches $\tilde{Q}$ in all 3D groups $\mathcal{P}(\tilde{P})$ containing **i**, the weights being given by the $w_{\tilde{P}}$.
end for

Algorithm 11 BM3D second iteration algorithm for grey images.

Input: noisy image $\tilde{u}$, σ, noise standard deviation.
Input: basic estimation $\hat{u}_1$ obtained at the first step.
Output: final denoised image $\hat{u}$.

Set parameter $\kappa \times \kappa = 8 \times 8$ (up to 12 for high noise level): dimension of patches.
Set parameter $\lambda \times \lambda = 39 \times 39$: size of search zone in which similar patches are searched.
Set parameter $N_{max} = 32$: maximum number of similar patches retained during the grouping part.
Set parameter $s = 3$: step in both rows and columns between two reference patches.
Set parameter $\tau = 400$ (if $\sigma > 40, \tau = 3500$): threshold used to determinate similarity between patches.

for each pixel **i**, with a step s in rows and columns **do**

Take the square reference patches $\tilde{P}$ and $\hat{P}_1$ centered at **i**, of size $\kappa \times \kappa$ in the initial and basic estimation images.

Look for square patches $\hat{Q}_1$ in a square neighborhood of **i** of size $zsize \times zsize$ having a distance lower than τ in the basic estimate image $\hat{u}_1$.

if there are more than N_{max} similar patches **then**
keep only the N_{max} closest similar patches to $\hat{P}_1$ according to their Euclidean distance.
else
keep 2^p patches, where p is the largest integer such that 2^p is smaller than the number of similar patches
end if

Two 3D groups $\mathcal{P}(\tilde{P})$ and $\mathcal{P}(\hat{P}_1)$ are built with those similar patches, one from the noisy image $\tilde{u}$ and one from the basic estimate image $\hat{u}_1$.

A 2D DCT is applied on every patch contained in $\mathcal{P}(\tilde{P})$ and $\mathcal{P}(\hat{P}_1)$.

A Walsh-Hadamard transform is then applied along the third dimension of $\mathcal{P}(\tilde{P})$ and $\mathcal{P}(\hat{P}_1)$.

Denoting by τ_{3D} the 3D transform (2D DCT followed by the Walsh-Hadamard transform) applied on the 3D group, compute the Wiener coefficient $\omega_{\tilde{P}} = \frac{|\tau_{3D}(\mathcal{P}(\hat{P}_1)|^2}{|\tau_{3D}(\mathcal{P}(\hat{P}_1)|^2 + \sigma^2}$.

The Wiener collaborative filtering of $\mathcal{P}(\tilde{P})$ is realized as the element-by-element multiplication of the 3D transform of the noisy image $\tau_{3D}(\mathcal{P}(\tilde{P}))$ with the Wiener coefficients $\omega_{\tilde{P}}$.

An associated weight $w_{\tilde{P}}$ is computed :

$$w_{\tilde{P}} = \begin{cases} (\|\omega_{\tilde{P}}\|_2)^{-2} & \|\omega_{\tilde{P}}\|_2 > 0 \\ 1 & \|\omega_{\tilde{P}}\|_2 = 0 \end{cases}$$

The estimate $\hat{u}_2^{\tilde{Q},\tilde{P}}$ for each pixel **i** in similar patches $\tilde{Q}$ of the 3D group $\mathcal{P}(\tilde{P})$ is then obtained by applying the inverse of the 1D Walsh-Hadamard transform along the third dimension, followed by the inverse of the 2D DCT on every patch of the 3D group.
end for

for each pixel **i do**
Aggregation: Recover the denoised value $\hat{u}(\mathbf{i})$ at **i** by averaging all estimates of patches $\tilde{Q}$ in all 3D groups $\mathcal{P}(\tilde{P})$ containing **i**, using the weights $\omega_{\tilde{P}}$.
end for

Here we described the basic implementation given in its seminal paper, and which will also be used in the comparison section. Yet, BM3D has several more recent variants that improve its performance. Like for NL-means, there is a variant with shape-adaptive patches [36]. In this algorithm denominated BM3D-SAPCA, the sparsity of image representation is improved in two aspects. First, it employs image patches (neighborhoods) which can have data-adaptive shape. Second, the PCA bases are obtained by eigenvalue decomposition of empirical second-moment matrices that are estimated from groups of similar adaptive-shape neighborhoods. This method improves BM3D especially in preserving image details and introducing very few artifacts. The anisotropic shape-adaptive patches are obtained using the 8-directional LPA-ICI techniques [69]. The very recent development of BM3D is presented in [68], [39], where it is generalized to become a generic image restoration tool, including deblurring.

4.9 The piecewise linear estimation (PLE) method

The ambitious Bayesian restoration model proposed in [140] and [141] is a general framework for restoration, including denoising, deblurring, and inpainting. An image is decomposed into overlapping patches $n_i = \mathbf{A}_i P_i + N_i$ where $\mathbf{A}_i$ is the degradation operator restricted to the patch i, P_i is the original patch, n_i the degraded one, and N_i the noise restricted to the patch. Since we are studying only the denoising problem, we shall take for $\mathbf{A}_i$ the identity. The (straightforward) extension including a linear perturbation operator is out of our scope.
The patch density law is modeled as a mixture of Gaussian distributions $\{\mathcal{N}(\mu_k, \mathbf{C}_k)\}_{1\leq k\leq K}$ parametrized by their means μ_k and covariance matrices $\mathbf{C}_k$. Thus each patch n_i is assumed independently drawn from one of these Gaussians with an unknown index k and a density function

$$p(P_i) = \frac{1}{(2\pi)^{\frac{\kappa^2}{2}} |\mathbf{C}_{k_i}|^{\frac{1}{2}}} e^{-\frac{1}{2}(P_i-\mu_k)^T \mathbf{C}_{k_i}^{-1}(P_i-\mu_k)}.$$

Estimating all patches P_i from their noisy observations n_i amounts to solve the following problems:

- to estimate the Gaussian parameters $(\mu_k, \mathbf{C}_k)_{1\leq k\leq K}$ from the degraded data n_i;
- to identify the index k_i of the Gaussian distribution generating the patch P_i;
- to estimate P_i from its corresponding Gaussian distribution $(\mu_{k_i}, \mathbf{C}_{k_i})$ and from its noisy version n_i.

In consequence PLE [141]) has two distinct steps in the estimation procedure. In an E-step (E for Estimate), the Gaussian parameters $(\mu_k, \mathbf{C}_k)_k$ are known and for each patch the maximum *a posteriori* (MAP) estimate $\hat{P}_i^k$ is computed with each Gaussian model. Then the best Gaussian model k_i is selected to obtain the estimate $\hat{P}_i = \hat{P}_i^{k_i}$.
In the M-step (M for Model), the Gaussian model selection k_i and the signal estimates $\hat{f}_i$ are assumed known for all patches i, and permit to estimate again the Gaussian models $(\mu_k, \mathbf{C}_k)_{1\leq k\leq K}$. According to the terminology of section 3.2, this section gives the *oracle* permitting to estimate in the E-step the patches by a Wiener type filter.
For each image patch with index i the patch estimation and its model selection is obtained by maximizing the *log* a-posteriori probability $\mathbb{P}(P_i \mid n_i, k)$,

$$\begin{aligned}
(\hat{P}_i, k_i) &= \arg\max_{P,k} \log \mathbb{P}(P_i \mid n_i, \mathbf{C}_k) && (4.27)\\
&= \arg\max_{P,k} \left(\log \mathbb{P}(n_i \mid P_i, \mathbf{C}_k) + \log \mathbb{P}(P_i \mid \mathbf{C}_k)\right) && (4.28)\\
&= \arg\min_{P_i,k} \left(\|P_i - n_i\|^2 + \sigma^2 (P_i-\mu_k)^T \mathbf{C}_k^{-1}(P_i-\mu_k) + \sigma^2 \log|\mathbf{C}_k|\right) && (4.29)
\end{aligned}$$

where the second equality follows from the Bayes rule and the third one assumes a white Gaussian noise with diagonal matrix $\sigma^2\mathbf{I}$ (of the dimension of the patch) and $P_i \simeq \mathcal{N}(\mu_k, \mathbf{C}_k)$. This minimization can be made first over P_i, which amounts to a linear filter, and then over k, which is a simple comparison of a small set of real values. The index k being fixed, the optimal P_i^k satisfies

$$P_i^k = \arg\min_{P_i} \left(\|P_i - n_i\|^2 + \sigma^2 (P_i - \mu_k)^T \mathbf{C}_k^{-1} (P_i - \mu_i) + \log |\mathbf{C}_k| \right)$$

and therefore

$$P_i^k = \mu_k + (\mathbf{I} + \sigma^2 \mathbf{C}_{k_i}^{-1})^{-1} (n_i - \mu_k),$$

which is the formula (4.3) already seen in section 4.2. Then the best Gaussian model k_i is selected as

$$k_i = \arg\min_{k} \left(\|P_i^k - n_i\|^2 + \sigma^2 (P_i^k - \mu_k)^T \sigma_k^{-1} (P_i^k - \mu_i) + \log |\mathbf{C}_k| \right).$$

Assuming now that for each patch P_i its model k_i and its estimate $\hat{P}_i$ are known, the next question is to give for each k the maximum likelihood estimate for $(\mu_k, \mathbf{C}_k)$ knowing all the patches assigned to the k-th cluster $\mathcal{C}_k$, namely,

$$(\mu_k, \mathbf{C}_k) = \arg\max_{\mu_k, \mathbf{C}_k} \log \mathbb{P}(\{\hat{P}_i\}_{i \in \mathcal{C}_k} \mid \mu_k, \mathbf{C}_k).$$

This yields the empirical estimate

$$\mu_k = \frac{1}{\#\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \hat{P}_i, \quad \mathbf{C}_k = \frac{1}{\#\mathcal{C}_k - 1} \sum_{i \in \mathcal{C}_k} (\hat{P}_i - \mu_k)(\hat{P}_i - \mu_k)^T,$$

which are the estimates (4.2) also used in section 4.2.
Finally the above MAP-EM algorithm is iterated and the authors observe that the MAP probability of the observed signals $\mathbb{P}(\{\hat{P}_i\}_i \mid \{n_i\}_i, \{\mu_k, \mathbf{C}_k\}_k)$ always increases. The clusters and the patch estimates converge. Nevertheless, this algorithm requires a good initialization. Noticing that having the adequate Gaussians describing the patch space amounts to have a good set of PCA bases for intuitive patch clusters, the authors create 19 orthogonal bases in the following way: one of them, say $k = 0$, is the classic DCT basis and corresponds to the "texture cluster". The others are obtained by fixing 18 uniformly sampled directions in the plane. For each direction, PCA is applied to a set of patches extracted from a synthetic image containing an edge in that direction. The PCA yields an oriented orthonormal basis. In short, the initial clusters segment the patch set in 18 classes of patches containing an edge or an oriented texture, and one class containing the more isotropic patches.
The study in this paper gives an interpretation of the patch dictionary methods such as K-SVD and fuses them with Bayesian methods and the Wiener method. In particular the paper shows how the K-SVD method actually learns patches that are quite similar to oriented patches obtained by the above procedure, as illustrated in Fig. 4.6. This analysis structures the synthetic view proposed in chapter 6.

4.10 Non-local Dual Denoising

4.10.1 Dual Domain Image Denoising

This section presents an alternative interpretation of DDID that differs from the one originally proposed by Knaus and Zwicker [72]. The original description of DDID splits the image into a low- and a high-contrast layer, which are treated respectively with a spatial and a frequency domain method. In this work instead, the spatial domain filtering is seen as a pre-processing to improve the frequency domain denoising.
DDID consists of three almost identical steps. The output of each step is used to guide the following one. Each step of the algorithm processes the noisy image y pixel-wise using the guide

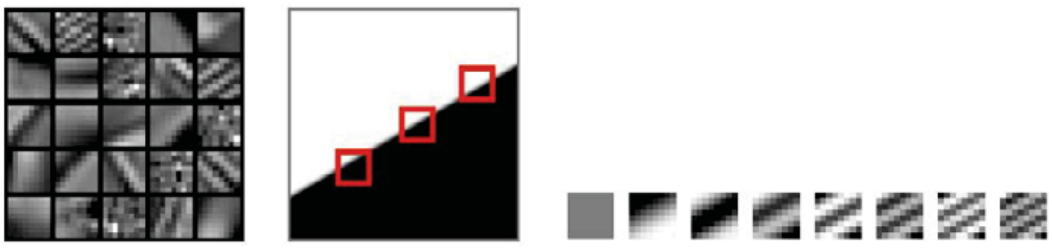
Algorithm 12 Piecewise linear estimation (PLE)

Input: noisy image $\tilde{u}$ given by the family of its noisy patches $(\tilde{P}_i)_i$, initial set of 19 Gaussian models $\mathcal{N}(\mu_k, \mathbf{C}_k)$ obtained as: a) the 18 PCAs of the patches of 18 synthetic edge images, each with a different orientation; b) a Gaussian model with a diagonal covariance matrix on the DCT basis.
Output: denoised image $\hat{u}$

E-STEP
for all patches $\tilde{P}_i$ of the noisy image **do**

  for each k **do**
    Estimate the MAP of P_i knowing k: $P_i^k = \mu_k + (\mathbf{I} + \sigma^2 \mathbf{C}_k^{-1})^{-1} n_i$.
  end for
  Select the best Gaussian model k_i for P_i as
  $k_i = \arg\min_k \left(\|P_i^k - n_i\|^2 + \sigma^2 (P_i^k - \mu_k)^T \mathbf{C}_k^{-1} (P_i^k - \mu_i) + \log |\mathbf{C}_k| \right)$.
  Obtain the best estimate of P_i knowing the Gaussian models $(\mu_k, \mathbf{C}_k)$, $\hat{P}_i = P_i^{k_i}$.
end for

M-STEP
for all k **do**
  Compute the expectation μ_k and covariance matrix $\mathbf{C}_k$ of each Gaussian by
  $\mu_k = \frac{1}{\#\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \hat{P}_i, \quad \mathbf{C}_k = \frac{1}{\#\mathcal{C}_k - 1} \sum_{i \in \mathcal{C}_k} (\hat{P}_i - \mu_k)(\hat{P}_i - \mu_k)^T$.
end for
Iterate **E-STEP** and **M-STEP**
Aggregation: Obtain the pixel value of the denoised image $u(\mathbf{i})$ as a weighted average of all values of all denoised patches P_i which contain $\mathbf{i}$.



(a) (b) (c)

Figure 4.6: Taken in [141], this figure shows : (a) typical dictionary atoms learnt from the classic image Lena with K-SVD; (b)-(d) the numerical procedure to create one of the oriented PCAs; (b) a synthetic edge image. Patches 8×8 touching the edge are used to calculate an initial PCA basis; (c) the first 8 patches of the PCA basis (ordered by the larger eigenvalue).

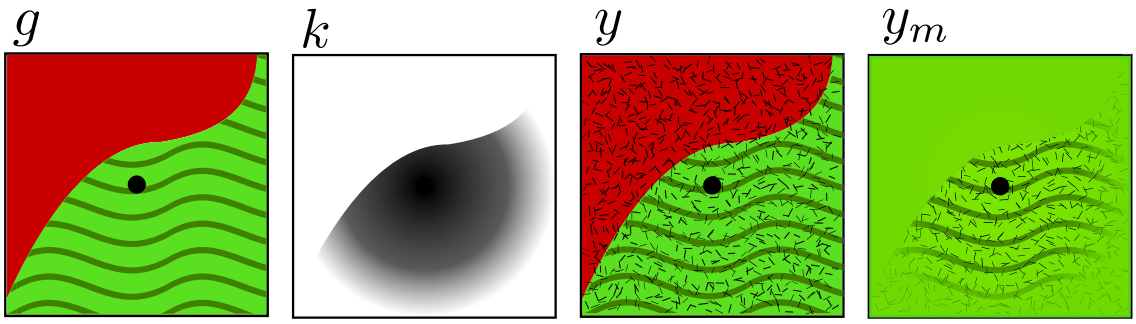


Figure 4.7: Illustration of DDID's preprocessing of a patch. The kernel k is computed using the guide g. In the modified patch y_m all object discontinuities have been removed, leaving only the texture information corresponding to the object selected by the kernel k. The removed pixels are replaced by $\tilde{s}$: the average of the *meaningful* portion of the patch.

image g. Each pixel p is denoised using the $d \times d$ neighborhood ($d = 31$) of both the noisy and the guide image.
Denoising in the frequency domain often results in the appearance of artifacts. To prevent it each patch is pre-processed to eliminate discontinuities corresponding to object's edges and patch's boundaries. To that end, a kernel k is created from g identifying the pixels belonging to the same object as its central pixel p. This kernel is the product of a *spatial* and *range* kernels, as used in the bilateral filter [137, 127]

$$k(q) = k_s(q) \cdot k_r(q). \tag{4.30}$$

- The *range* kernel is used to identify the pixels belonging to the same object. The idea is that, in g, pixels belonging to the same object as the central pixel will have similar values. The kernel is

$$k_r(q) = \exp\left(-\frac{|g(q) - g(p)|^2}{\gamma_r \sigma^2}\right), \tag{4.31}$$

  where γ_r is a parameter of the algorithm and σ is the standard deviation of the noise.

- The *spatial* kernel, identifies the pixels close to the central one and smooths periodization discontinuities associated to the frequency domain processing. To achieve that a Gaussian kernel of standard deviation σ_s is used, where σ_s is a parameter of the algorithm:

$$k_s(q) = \exp\left(-\frac{|q - p|^2}{2\sigma_s^2}\right). \tag{4.32}$$

Since denoising with Fourier coefficients has problems in presence of edges (due to the Gibbs phenomenon), the goal is to make the parts of the patch not relevant to the denoising as *regular* as possible. k is used to compute the average of the "relevant" part of both the noisy and the guide patches:

$$\tilde{s} = \frac{\sum k(q)y(q)}{\sum k(q)}, \qquad \tilde{g} = \frac{\sum k(q)g(q)}{\sum k(q)}, \tag{4.33}$$

where the sums are computed over $\mathcal{N}_p$, the domain of the $d \times d$ patch centered at p. After that, the parts of the patch not taken into account by k are set to the respective average. The resulting *modified* patch is

$$y_m(q) = k(q)y(q) + (1 - k(q))\tilde{s}. \tag{4.34}$$

As illustrated in Fig. fig:DDIDsketch the patch y_m is similar to y in the parts belonging to the same object as the central pixel (including the noise) and smooth in the rest. The same procedure

is applied to the guide patch

$$g_m(q) = k(q)g(q) + (1 - k(q))\tilde{g}. \tag{4.35}$$

At this point, y_m and g_m are two patches, built in the same way, in which discontinuities have been strongly reduced and only information "relevant" to denoise the central pixel has been kept. It is therefore safe to apply the Fourier transform and to continue the process in the frequency domain

$$G(f) = \sum_{q \in \mathcal{N}_p} \exp\left(-\frac{2i\pi(q-p)f}{d}\right) g_m(q), \tag{4.36}$$

$$S(f) = \sum_{q \in \mathcal{N}_p} \exp\left(-\frac{2i\pi(q-p)f}{d}\right) y_m(q). \tag{4.37}$$

Assuming that y contains an additive white Gaussian noise of variance σ^2, the amount of noise present in y_m depends on k. In particular, for a pixel q, $y_m(q)$ contains a noise equal to $\sigma^2 k(q)$. An interesting property of the Fourier transform is that the noise in every pixel is evenly distributed over all frequencies. Thus every frequency of S has Gaussian noise with the same variance

$$\sigma_f^2 = \sigma^2 \sum_{q \in \mathcal{N}_p} k(q)^2. \tag{4.38}$$

The patch is then denoised by shrinking its Fourier coefficients $S(f)$ by the factor

$$K(f) = \begin{cases} 1 & \text{if } f = 0, \\ \exp\left(-\frac{\gamma_f \sigma_f^2}{|G(f)|^2}\right) & \text{otherwise,} \end{cases} \tag{4.39}$$

where γ_f is a parameter of the algorithm. The denoised value of the central pixel is finally recovered by reversing the Fourier transform. Inverting equation (4.34) is unnecessary, since $k(p) = 1$. Since the inverse Fourier transform evaluated in the center of the patch is the average of the frequencies, the central pixel's value is computed as

$$x(p) = \frac{1}{d^2} \sum_f S(f)K(f). \tag{4.40}$$

Equations (eq:fourier- eq:res) are slightly different from the ones presented in [72]. In fact, it can be easily proved that $G(f)$ and $S(f)$ differ from the ones presented in the original paper only at the zero frequency. This frequency is then restored after the shrinkage. In the presented version, the zero frequency is left untouched by the shrinkage, by imposing $K(0) = 1$.
For color images, the kernel k_r is computed by using the Euclidean distance in the color space, while the Fourier thresholding is done independently on each channel in the YUV color space.

Artifacts in DDID The above description highlights that the denoising in DDID is accomplished in the frequency domain, while the spatial pre-processing is used to remove discontinuities from the image. The described procedure is applied three times with different parameters. Each time the result of the previous calculation is used as a guide, except in the first iteration where the noisy image itself is used. It is worth noting that the image is denoised in the last iteration only. The other two are only used to obtain a suitable guide.
Besides being slow to compute, the main drawback of DDID is that its results often present ringing artifacts (as seen in Fig. fig:DDIDartifacts). This is surprising since removing the strong edges, as in equation (4.34), should prevent it. Since the guide image used in the first iteration is noisy, and the kernel in (4.30) is computed from it, "parasite" information is retained and propagated in the following iterations (see Fig. fig:DDIDsteps). This yields a result that contains artifacts.

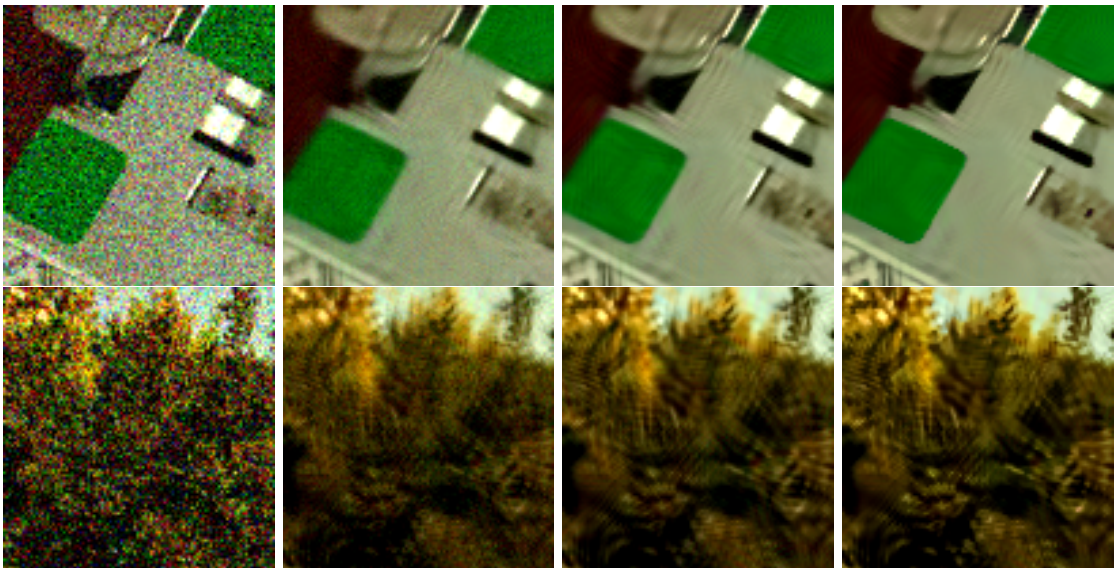


Figure 4.8: Artifacts in DDID. From left to right: the noisy image (with $\sigma = 30$), the result of the first, second, and last iteration of the algorithm.

4.10.2 Non-Local Dual Denoising

Since, as concluded in the previous section, most of the artifacts of DDID come from the guide image, a method to avoid them is to feed the algorithm with a cleaner image.
Non-Local Dual Denoising uses the Non-Local Bayes denoising algorithm as proposed in chapter ch:nlbayes and also in [76] to provide a clean guide, and then applies the last step of DDID to denoise the image (with parameters $\sigma_s = 7$, $\gamma_r = 0.7$ and $\gamma_f = 0.8$). NL-Bayes has been chosen over other state-of-the-art algorithms (such as BM3D) because it generally provides a smoother output). BM3D has been tested too as the guide. However the results, while still improving the ones of both BM3D and DDID, were slightly worse than the ones of NLDD. A pseudo-code for NLDD is listed in Algorithm alg:nldd.

Algorithm 13 Non-Local Dual Denoising

functionNLDD($\mathbf{y}$, σ)
$\mathbf{g} \leftarrow$ NL-Bayes($\mathbf{y}$, σ)
for all pixels $p \in \mathbf{y}$ **do**
  $y \leftarrow$ ExtractPatch($\mathbf{y}$, p)
  $g \leftarrow$ ExtractPatch($\mathbf{g}$, p)
  $k \leftarrow$ ComputeK($\mathbf{g}$, p) — Eq. eq:mask
  $y_m, g_m \leftarrow$ ModifyPatches(y, g, k) — Eq. eq:tildemeans- eq:g-reg
  $S \leftarrow \mathrm{FFT}(y_m)$
  $G \leftarrow \mathrm{FFT}(g_m)$
  $\mathbf{x}(p) \leftarrow$ Shrink(S, G, k, σ) — Eq. eq:Kshrink- eq:res
end for
return x
end function

This algorithm has several advantages over DDID. Since the guide image (provided by NL-Bayes) has less artifacts than the one computed in the first two iterations of DDID, it generally provides better results, as shown in chapter ch:comparisons. As expected, the results contain less artifacts. In addition, NLDD is faster than DDID since NL-Bayes is faster than DDID by an order of magnitude. So the overall computation time is about one third of DDID's. Moreover, since

Figure 4.9: Crops from the results of the images Alley, Flowers and Computer. From left to right: the original image, the noisy image ($\sigma = 30$), the outputs of BM3D, NL-Bayes, DDID, and NLDD. Full results are available in the article's website.

Image	BM3D	NL-Bayes	DDID	NLDD
Alley	29.32	29.12	29.33	**29.41**
Computer	30.66	30.68	31.00	**31.10**
Dice	38.02	37.97	38.45	**38.78**
Flowers	33.76	33.85	34.36	**34.48**
Girl	36.95	36.62	37.26	**37.33**
Traffic	28.83	29.00	29.20	**29.40**
Trees	24.62	25.02	24.85	**25.09**
Valldemossa	27.24	27.37	27.27	**27.48**
Mean	31.18	31.20	31.46	**31.64**

Table 4.1: Values of PSNR for $\sigma = 30$.

both DDID and NL-Bayes are heavily parallelizable, NLDD can also be implemented on a GPU architecture.

4.10.3 Experimental result

NLDD has been compared against DDID, BM3D and NL-Bayes with different amounts of noise. For the tests an heterogeneous set of noise-free images was used. All the results are evaluated using PSNR and SSIM [133], an alternative metric conceived to simulate the response of the Human Visual System.

The results for $\sigma = 30$, where DDID performs best, are summarized in Table tab:1. NLDD outperforms the other algorithms in terms of PSNR. The results for other levels of noise are summarized in Table tab:2. NLDD provides the best results for values of σ between 20 and 60. These coincide with the values of noise for which DDID has the best performance. Looking closely at Fig. fig:results fewer artifacts can be noticed for NLDD. However, the values of SSIM don't reflect the magnitude of this improvement, but the details in Fig. fig:DDIDartifacts suggest that the quality of the two reconstructions is significantly different.

It is worth noticing that when the guide image is inaccurate NLDD also performs relatively poorly. For example in the image "Flowers" NL-Bayes fails to recover the texture of the leaves. As a result, these areas of the image are not fully recovered by NLDD.

PSNR				
σ	BM3D	NL-Bayes	DDID	NLDD
2	45.75	**46.16**	45.31	45.42
5	40.62	**40.96**	40.37	40.45
10	36.84	**37.07**	36.92	36.92
20	33.22	33.42	33.52	**33.64**
30	31.18	31.20	31.46	**31.64**
40	29.70	29.62	29.99	**30.21**
60	27.45	28.35	27.96	**28.38**
80	26.53	**27.03**	26.50	26.89
100	24.95	**26.05**	25.42	25.82
SSIM				
σ	BM3D	NL-Bayes	DDID	NLDD
2	0.9973	**0.9975**	0.9971	0.9973
5	0.9872	0.9872	**0.9873**	0.9873
10	0.9684	0.9680	**0.9699**	0.9691
20	0.9303	0.9308	0.9331	**0.9346**
30	0.8938	0.8928	0.8957	**0.8995**
40	0.8614	0.8572	0.8600	**0.8671**
60	0.8064	0.8084	0.7994	**0.8100**
80	0.7592	**0.7595**	0.7500	0.7570
100	0.7132	**0.7215**	0.7096	0.7197

Table 4.2: Average values of PSNR and SSIM with different noise levels.

Chapter 5

Comparison of Denoising Algorithms

In this chapter we compare the following "state of the art" denoising algorithms introduced in chapters 2 and 4: the sliding DCT filter as specified in Algorithm 2, the wavelet neighborhood Gaussian scale mixture (BLS-GSM) algorithm, as specified in Algorithm 8, the classical vector valued NL-means as specified in Algorithm 3, the BM3D algorithm as specified in Algorithms 10 and 11, the K-SVD denoising method as described in Algorithm 17, the Non-local Bayes algorithm as specified in Algorithm 4 and the NLDD algorithm as described in Algorithm 13. These algorithms have been chosen for two reasons. First they have a public and completely transparent code available, which is in agreement with their present description. Second, they all represent distinct denoising principles and therefore illustrate the methodological progress and the diversity of denoising principles.
The comparison, using when possible the public IPOL algorithms `http://www.ipol.im/`, will be based on four quantitative and qualitative criteria: the visualization of the *method noise*, namely the part of the image that the algorithm has taken out as noise, the visual verification of the *noise to noise* principle, and the *mean square error* or *PSNR* tables. Last but not least the *visual quality* of the restored images must of course be the ultimate criterion. It is easily seen that a single criterion is not enough to judge a restoration method. A good denoising solution must have a high performance under all mentioned criteria.
This chapter uses the results of joint work with Antoni Buades, Gabriele Facciolo, Jean-Michel Morel, Nicola Pierazzo and Martin Rais.

5.1 "Method noise"

The difference between the original image and its filtered version shows the "noise" removed by the algorithm. This procedure was introduced in [13] and this difference was called *method noise* by the authors. The authors pointed out that the method noise should look like a noise, at least in case of additive white noise. A visual inspection of this difference tells us which geometrical features or details have been unduly removed from the original. Only human perception is able to detect these unduly removed structures in the "method noise". Furthermore for several classical algorithms like the Gaussian convolution, anisotropic filters, neighborhood filters or wavelet thresholding algorithms, a closed formula permits to analyse the method noise mathematically and thus gives an explanation of observed structured image differences when applying the denoising method [16]. Such an analysis is unfortunately not available and not easy for the state of the art algorithms which are compared in this section. The degree of complexity of each method does not allow for a mathematical study of the method noise. Therefore the evaluation of this criterion will be based only on visual inspection.
When the standard deviation of the added noise is higher than contrast in the original image, a

visual exploration of the method noise is nevertheless not reliable. Image features in the method noise may be hidden in the removed noise. For this reason, the evaluation of the method noise should not rely on experiments where a white noise with standard deviation larger than 5 or 10 has been added to the original.

Fig. 5.1 displays the method noise for the state of the art algorithms being compared in this chapter, when a Gaussian white noise of standard deviation $\sigma = 5$ has been added. The image differences have been rescaled from $[-4\sigma, 4\sigma]$ to $[0, 255]$ for visualization purposes, and values outside this range have been saturated. By a first visual inspection, it is noticed that all methods have a difference similar to a white noise. This is an outstanding properties of these algorithms, which is not shared by classical denoising techniques such as anisotropic filtering, total variation minimization or wavelet thresholding (see [16] for a more detailed study). It is also immediately observed that the magnitude of the method noise of NL-means and K-SVD is larger than for the rest of the methods. This is corroborated by the standard deviation of each residual noise (see Fig. 5.1), which is around 5.7 for NL-means and K-SVD, around 4.7 for DCT denoising and around 4.25 for the other algorithms. DCT-denoising, BLS-GSM, BM3D, NL-Bayes and NLDD keep the transform coefficients that are larger than the ones predicted by noise. This explains why they remove little noise in textured or edge regions. This fact can be easily noticed in Fig. 5.2 where a piece of the residual noise of Fig. 5.1 has been enlarged. The amplitude of the noise removed by NL-means and K-SVD is uniform all over the image, while it depends on the underlying image for the rest of the algorithms.

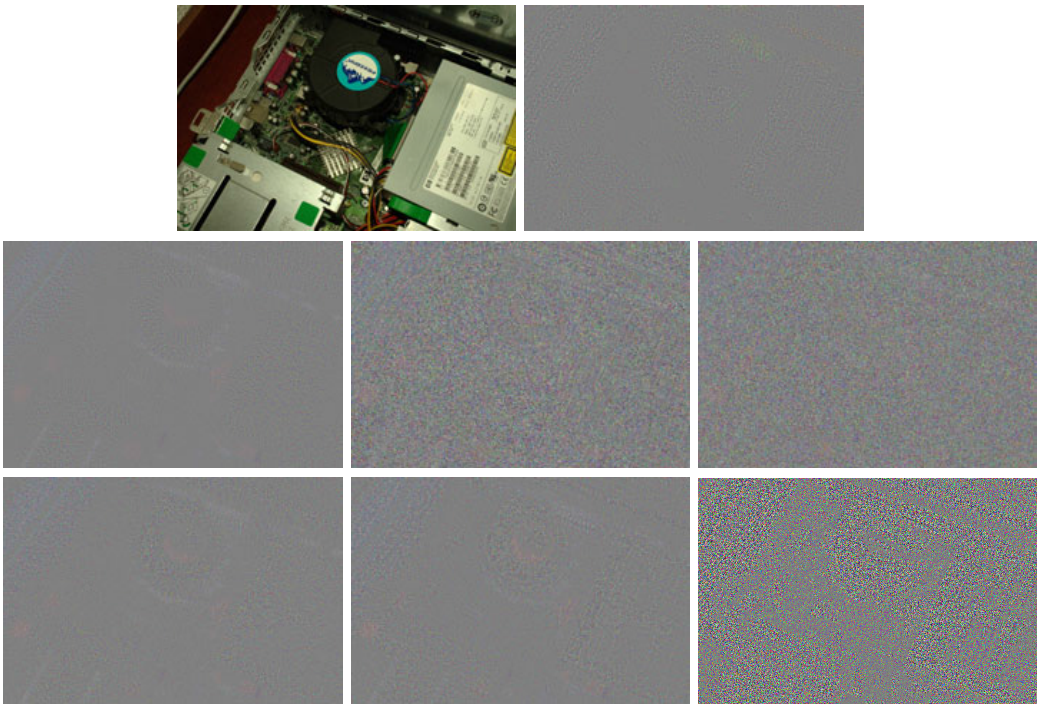


Figure 5.1: Display of method noise. The noisy image was obtained by adding a Gaussian white noise of standard deviation 5. From top to bottom and left to right: slightly noisy image, DCT sliding window (std = 4.69), BLS-GSM (std = 4.28), NL-means (std = 5.78), K-SVD (std = 5.67), BM3D (std = 4.25), Non-local Bayes (std = 4.28) and NLDD (std = 4.18). All methods have a difference similar to a white noise even if the magnitude of the NL-means and K-SVD differences is larger. This is corroborated by the standard deviation of each residual noise. Due to the thresholding nature of DCT, BLS-GSM, BM3D, NL-Bayes and NLDD, which alter little the coefficients larger than the ones predicted by noise, noise is not removed in textured and edge zones. This can be easily noticed in Fig. 5.2 where a piece of the residual noises has been enlarged.

5.2 The "noise to noise" principle

The *noise to noise* principle, introduced in [14], requires that a denoising algorithm transforms white noise into white noise. This paradoxical requirement seems to be the best way to characterize artifact-free algorithms. The transformation of a white noise into any correlated signal creates structure and artifacts. Only white noise is perceptually devoid of structure, as was pointed out by Attneave [5].
The noise to noise of classical denoising algorithms was studied in [14], where it was shown that neighborhood filters and, asymptotically, NL-means transform a white noise into a white noise. The convolution with a Gauss kernel keeps the low frequencies and cancels the high ones. Thus, the filtered noise actually shows big grains due to its prominent low frequencies. Noise filtered by a wavelet or DCT thresholding is no more a white noise. The few coefficients with a magnitude larger than the threshold are spread all over the image. The pixels which do not belong to the support of one of these coefficients are set to zero. The visual result is a constant image with superposed wavelets or cosines if the DCT is used. The mathematical analysis of the rest of algorithms is not feasible due to its degree of complexity. Thus, only a visual inspection of this filtered noise is possible.
The methodology adopted to process the *noise to noise* and to show it is the following:

- Since most recent methods process colour images (except BLS-GSM), the *noise to noise* is applied on a colour flat image, i.e. an image with three channels with slightly different[1] values: $RGB = (127, 128, 129)$;
- To reduce the variations due to the random nature of the noise, the tests are performed on relatively large noise images. The chosen size is 1024×1024. The PSNR and RMSE results become then fairly independent of the simulated noise;
- Noise is added on each channel independently. It therefore is a colour noise, and its standard deviation is equal to 30 on each channel of the flat original image;
- All compared algorithms are processed on this noisy image;
- The denoised image is displayed. The mean on every channel is set to 128, and the difference to this mean is enhanced by a factor 5. A small part with size 256×256 of the denoised image is shown after zoom in in 5.3.

The results in PSNR and RMSE are summarized in the following table:

Method	PSNR	RMSE
NLDD	45.48	1.34
NL-Bayes	45.45	1.36
BM3D	45.03	1.43
NL-means	41.45	2.16
TV denoising	41.06	2.26
DCT denoising	40.91	2.30
K-SVD	38.44	3.05

The "order" of performance of the methods is almost respected, except for TV denoising, which shows a really good result compared to K-SVD. Fig. 5.3 displays the filtered noise images by several state of the art algorithms.
As expected, threshold-based methods present noticeable artifacts, in particular DCT denoising and BM3D. The NL-means result reflects the size of the search zone, and therefore leaves behind a low-frequency oscillation. Despite its good results, TV denoising presents a lot of artifacts which

[1] Values are different on each channel in order to force the algorithm to consider this image as a colour image, and not a grey image with a single channel.

do not look like noise, and are uglier than the K-SVD artifacts. Only NL-Bayes has no artifacts. Indeed, it detects flat patches and replaces them by their mean. This trick could actually be applied to all algorithms. Last but not least, each method leaves a sizeable low-frequency noise, which could be removed with a multi-scale approach.

5.3 Comparing visual quality

The visual quality of the restored image is obviously a necessary, if not sufficient, criterion to judge the performance of a denoising algorithm. It permits to control the absence of artifacts and the correct reconstruction of edges, texture and fine structure. Figures 5.5-5.7 display the noisy and denoised images for the algorithms under comparison for noise standard deviations of 20, 30 and 40.

Figure 5.5 presents an image with straight edges and flat and fine structures with a noise of standard deviation 20. The main artifacts are noticeable in the DCT, BLS-GSM and K-SVD denoised images. These are the most local algorithms and therefore have more trouble in removing the low frequencies of the noise. As a consequence, the denoised images present many low frequency colour artifacts in flat and dark zones. These artifacts are noticeable for all these algorithms even if all use a different strategy to deal with colour images. DCT uses the $Y_oU_oV_o$, K-SVD a vector valued algorithm and BLS-GSM is applied independently to each RGB component. NL-means does not suffer of these noise low frequency problems, but it leaves some isolated noise points on non-repetitive structures, mainly on corners. These isolated noise points could be attenuated by using the $Y_oU_oV_o$ colour space instead of the vector valued algorithm. In this experience, BM3D and Non-Local Bayes give a similar performance and superior to the rest of algorithms.

Figures 5.6 and 5.7 illustrate again the low frequency colour artifacts of DCT, BLS-GSM and K-SVD. In these figures, DCT and BLS-GSM also suffer of a strong Gibbs effect near all image boundaries. This Gibbs effect is nearly not noticeable in the denoised image by K-SVD, since the use of the whole dictionary permits to better reconstruct edges when the right atoms are present in the dictionary. The NL-means denoised image has no visual artifacts but is more blurred than those given by BM3D and Non-Local Bayes, that have a clearly superior performance to the rest of the algorithms. The BM3D denoised image has some Gibbs effect near edges, which sometimes degrades the visual quality of the solution. Non-Local bayes image shows no artifacts. It preserves often better textures than BM3D, by which the trees and vegetation can be slightly blurred by the use of the linear transform threshold.

In short, the visual quality of DCT, BLS-GSM and K-SVD is inferior to that of NL-means, BM3D and NL-Bayes, because of strong colour noise low frequencies in flat zones, and of a Gibbs effect. NL-means does not show noticeable artifacts but the denoised image is more blurred than those of BM3D and Non-Local Bayes. BM3D still has some Gibbs effect due to the use of a single basis for all pixels and a slightly inferior noise reduction, compared to Non-Local Bayes.

5.4 Comparing by PSNR

The mean square error is the square of the Euclidean distance between the original image and its estimate. In the denoising literature an equivalent measurement, up to a decreasing scale change, is the PSNR,

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right).$$

These numerical quality measurements are the most objective, since they do not rely on any visual interpretation. Tables 5.1 and 5.2 display the PSNR of state of the art denoising methods using the images in Fig. 5.4 and several values of σ from 2 to 40.

Before jumping to conclusions, we would like to point out that such a PSNR comparison is just informative, and cannot lead to an objective ranking of algorithms. Indeed, what is really needed

is a comparison of denoising principles. To compare them, these denoising principles must be implemented in denoising recipes containing several ingredients. Since the PSNR difference between recipes is tight, the way such or such generic tool is implemented, and the degree of sophistication with which each principle is implemented do matter. For example, two of our readers have pointed out to us[2] that an experimental analysis carried out exclusively on color images does not permit a comparison between the different strategies devised to take advantage of spatial redundancy. They suggest to complement the denoising results on color images with experiments on grayscale images. Then it would be possible to: 1) compare the degree of success of these different denoising principles in exploiting spatial redundancy; 2) evaluate the effectiveness of the various ways in which these grayscale algorithms are extended to color data.

In short, these authors do not share our analysis herewith, and the way conclusions can be drawn from the experimental results, because these results are very much influenced by the way color data is treated while much of the conclusions are applied about the relative effectiveness in exploiting spatial redundancy.

For the same reasons, these authors also disagree with the taxonomy summarized in table 6.1, where it seems that the extension to color is to be considered as a feature of a particular algorithm. Some methods are applied to color data in a very simple non-adaptive way and thus cannot be expected to fully decorrelate the color channels. This is for instance the case of BM3D, which uses a YUV/Opp color transformation. Data-adaptive color transformations for multispectral data are considered in [38]. This adaptive method provides substantially better results than a standard color transformation.

Another reason for being cautious, is that all methods with some existence have actually variants, and we are using the basic algorithms as they were announced in their seminal paper. For example, it is shown in [63] that BM3D can be slightly improved for heavy noise > 40 by changing the method parameters.

In short, the following PSNR comparison on color images must be taken for what it is; it gives some hints and these hints depend on the particular implementation of the denoising principles. We observe in the results that DCT denoising, GLS-GSM, K-SVD and NL-means have a similar PSNR. The relative performance of the methods depends on the kind of image and on noise level σ. On average, K-SVD and BLS-GSM are slightly superior to the other two, even if this is not the case visually, where K-SVD and BLS-GSM have a poor visual quality compared to NL-means. In all cases, BM3D and Non-local Bayes have a better PSNR performance than the others. Because of a superior noise reduction in flat zones and the presence of less artifacts of Non-local Bayes, the PSNR of BM3D is slightly inferior to Non-local Bayes. BM3D seems to retain the best conservation of detail. Some ringing artefacts near boundaries can probably be eliminated by the same trick as Non-local Bayes, namely detecting and giving a special treatment to flat 3D groups.

[2] Alessandro Foi, Vladimir Katkovnik, personal communication.

	$\sigma = 2$						
	NLDD	NL-Bayes	BM3D	BLS-GSM	K-SVD	NL-means	DCT denoising
Alley	44.42	**45.42**	44.95	-	41.51	42.75	44.58
Computer	45.16	**45.96**	45.22	44.69	44.52	44.03	44.54
Dice	48.50	**49.00**	48.86	48.59	47.79	48.51	48.39
Flowers	46.68	**47.77**	47.31	47.12	47.09	46.36	47.05
Girl	47.20	**47.56**	47.40	47.14	47.28	46.96	46.76
Traffic	44.62	**45.33**	44.56	44.15	43.80	43.55	44.26
Trees	42.88	**43.51**	43.07	-	42.05	42.22	42.95
Valldemossa	44.23	**45.17**	44.68	44.41	40.08	43.33	44.50
Mean	45.46	**46.22**	45.76	-	44.27	44.71	45.37

	$\sigma = 5$						
	NLDD	NL-Bayes	BM3D	BLS-GSM	K-SVD	NL-means	DCT denoising
Alley	38.54	**39.24**	38.95	-	38.45	37.18	38.37
Computer	40.05	**40.69**	39.98	39.30	39.58	38.86	39.03
Dice	45.76	**46.09**	45.80	45.21	45.27	45.12	45.22
Flowers	42.78	**43.44**	42.99	42.76	43.09	42.05	42.78
Girl	44.20	**44.26**	44.03	43.70	43.59	43.44	43.36
Traffic	39.22	**39.70**	38.67	38.10	38.75	37.50	38.21
Trees	36.08	**36.70**	36.10	-	35.61	34.69	35.76
Valldemossa	37.90	**38.73**	38.33	38.02	37.87	35.94	37.94
Mean	40.56	**41.11**	40.61	-	40.28	39.35	40.08

	$\sigma = 10$						
	NLDD	NL-Bayes	BM3D	BLS-GSM	K-SVD	NL-means	DCT denoising
Alley	34.97	**35.05**	34.82	-	34.29	33.53	34.22
Computer	36.35	**36.58**	36.28	35.47	35.79	35.44	35.34
Dice	**43.30**	**43.30**	43.02	42.21	41.71	42.06	42.22
Flowers	39.48	**39.52**	39.49	39.10	39.31	38.49	39.03
Girl	**41.86**	41.69	41.45	41.14	40.29	40.42	40.55
Traffic	34.88	**34.93**	34.54	33.92	34.69	33.89	34.11
Trees	31.23	**31.70**	31.23	-	30.61	29.42	30.92
Valldemossa	33.35	**33.73**	33.33	33.02	32.87	32.02	32.45
Mean	36.92	**37.06**	36.83	-	36.31	35.66	36.23

Table 5.1: PSNR table for $\sigma = 2$, 5 and 10. Only the three first digits are actually significant; the last one may vary with different white noise realizations.

	$\sigma = 20$						
	NLDD	NL-Bayes	BM3D	BLS-GSM	K-SVD	NL-means	DCT denoising
Alley	**31.49**	31.36	31.23	-	30.55	29.94	30.21
Computer	**33.26**	33.08	32.71	31.89	31.96	31.59	31.45
Dice	**40.63**	40.19	39.93	39.00	37.23	38.17	38.67
Flowers	**36.24**	35.87	35.85	35.34	35.24	34.56	34.89
Girl	**39.39**	38.92	38.71	38.49	36.36	36.81	37.27
Traffic	**31.40**	31.14	30.83	30.14	30.70	30.12	29.98
Trees	27.15	**27.22**	26.92	-	26.88	26.28	26.27
Valldemossa	29.78	**29.81**	29.57	26.97	29.08	28.37	28.91
Mean	**33.66**	33.45	33.22	-	32.25	31.98	32.20

	$\sigma = 30$						
	NLDD	NL-Bayes	BM3D	BLS-GSM	K-SVD	NL-means	DCT denoising
Alley	**29.71**	29.42	29.33	-	28.60	27.58	28.25
Computer	**31.42**	31.00	30.67	29.90	29.84	28.98	29.20
Dice	**39.01**	38.20	37.88	37.05	36.52	37.18	35.89
Flowers	**34.30**	33.67	33.73	33.19	33.54	32.66	32.46
Girl	**37.83**	37.12	36.97	36.91	35.38	35.54	34.67
Traffic	**29.48**	29.08	28.87	28.20	28.60	27.40	27.87
Trees	**25.16**	24.95	24.64	-	24.52	23.29	23.83
Valldemossa	**27.62**	27.51	27.30	26.97	26.80	25.55	26.48
Mean	**31.82**	31.37	31.17	-	30.48	29.77	29.83

	$\sigma = 40$						
	NLDD	NL-Bayes	BM3D	BLS-GSM	K-SVD	NL-means	DCT denoising
Alley	**28.53**	28.16	28.08	-	27.29	26.30	27.14
Computer	**30.13**	29.55	29.15	28.52	28.25	27.31	27.44
Dice	**38.01**	36.91	36.28	35.50	34.49	35.31	33.06
Flowers	**32.76**	31.94	32.10	31.68	31.90	30.99	30.80
Girl	**37.04**	36.09	35.62	35.61	33.73	34.03	32.01
Traffic	**28.15**	27.67	27.50	26.93	27.19	26.01	26.49
Trees	**23.47**	23.35	23.17	-	23.06	21.91	22.46
Valldemossa	**27.74**	27.51	25.78	25.50	25.28	24.10	25.08
Mean	**30.73**	30.15	29.71	-	28.90	28.25	28.05

Table 5.2: PSNR table for $\sigma = 20$, 30 and 40.

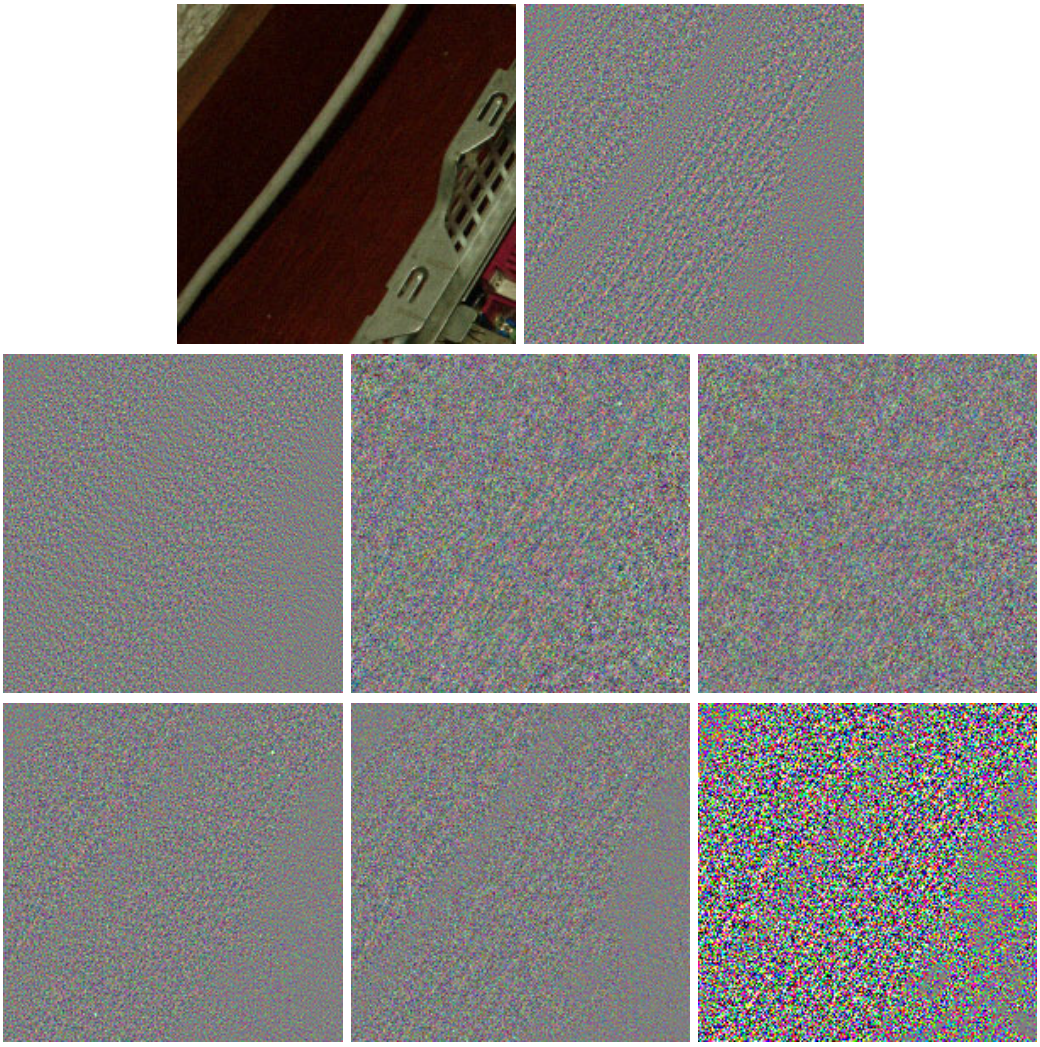


Figure 5.2: Enlargement of the method noise difference of Fig. 5.1. From top to bottom and left to right: slightly noisy image, and the method noise for DCT sliding window, BLS-GSM, NL-means, K-SVD, BM3D, Non-local Bayes and NLDD. The amplitude of the noise removed by NL-means and K-SVD is uniform all over the image while it is region dependent for the rest of the algorithms. Threshold based algorithms prefer to keep noisy values nearly untouched on highly textured or edge zones.

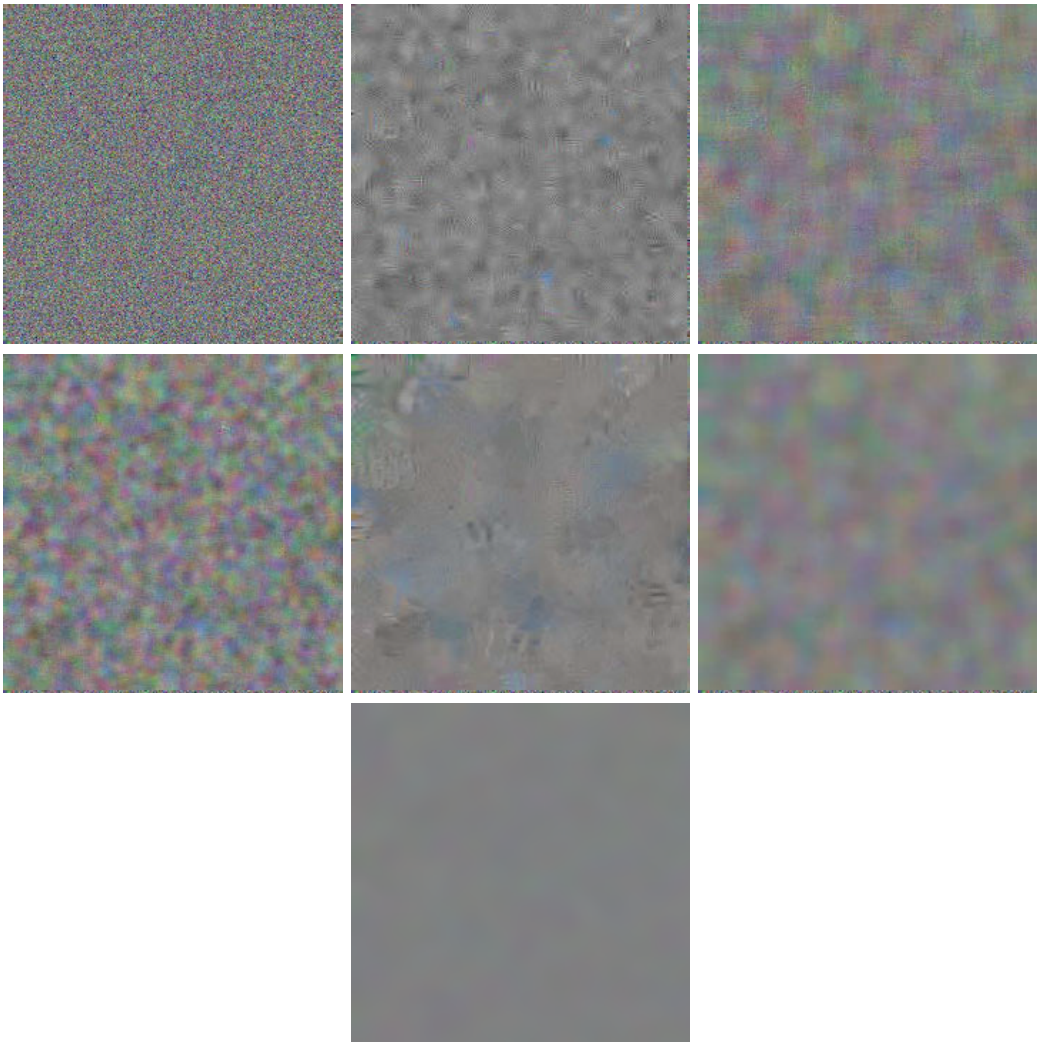
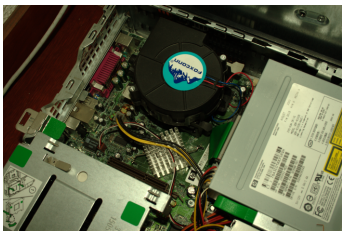
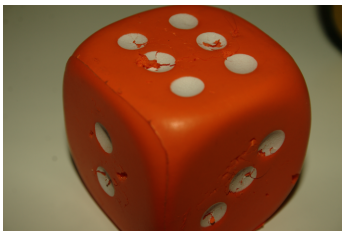


Figure 5.3: The noise to noise principle: a three-channels colour noise image filtered by the state of the art methods. From top to bottom and left to right: the noise image (flat, with independent homoscedastic noise added on each channel). Then, this same image denoised by DCT sliding window, NL-means, K-SVD, BM3D, Non-local Bayes and NLDD. The more the denoised image of a noise image looks like a noise image the better. Indeed, structured noise creates artifacts. BSM-GSM was not compared because we lack a colour version for this algorithm. None of the methods gives a satisfactory result: they all create a lower frequency oscillation or local artifacts for DCT and BM3D. Only multiscale version could cope with the low frequency remaining noise.

(a) Alley

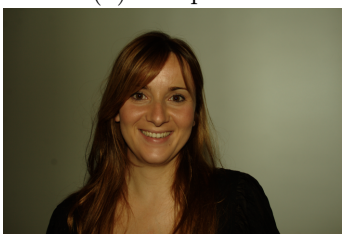


(b) Computer



(c) Dice

(d) Flowers



(e) Girl

(f) Traffic

(g) Trees

(h) Valldemossa

Figure 5.4: A set of noiseless images used for the comparison tests.

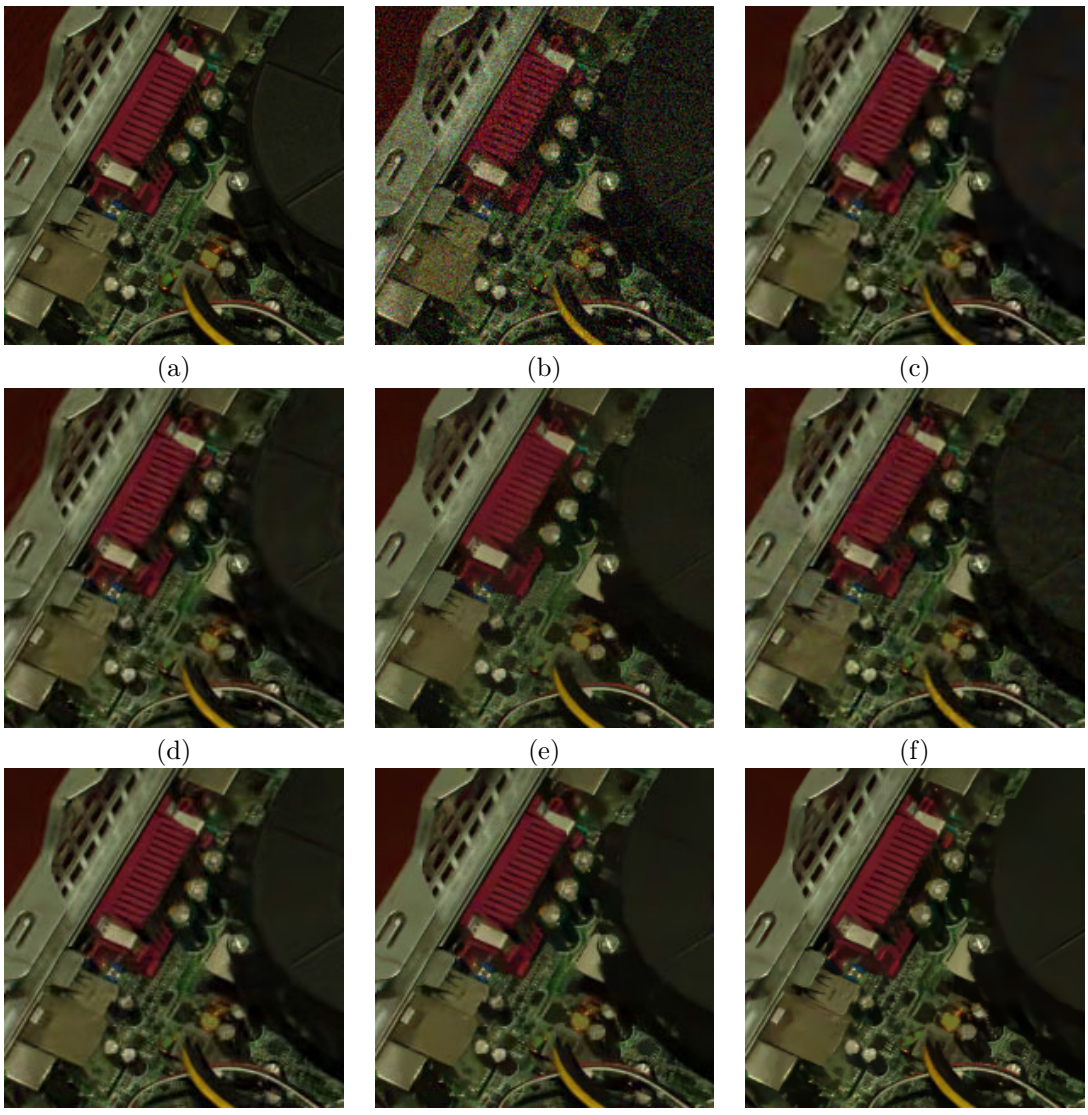


Figure 5.5: Visual quality comparison. The noisy image was obtained adding a Gaussian white noise of standard deviation 20. From top to bottom and left to right: original, noisy, DCT sliding window, BLS-GSM, NL-means, K-SVD, BM3D, Non-local Bayes and NLDD.

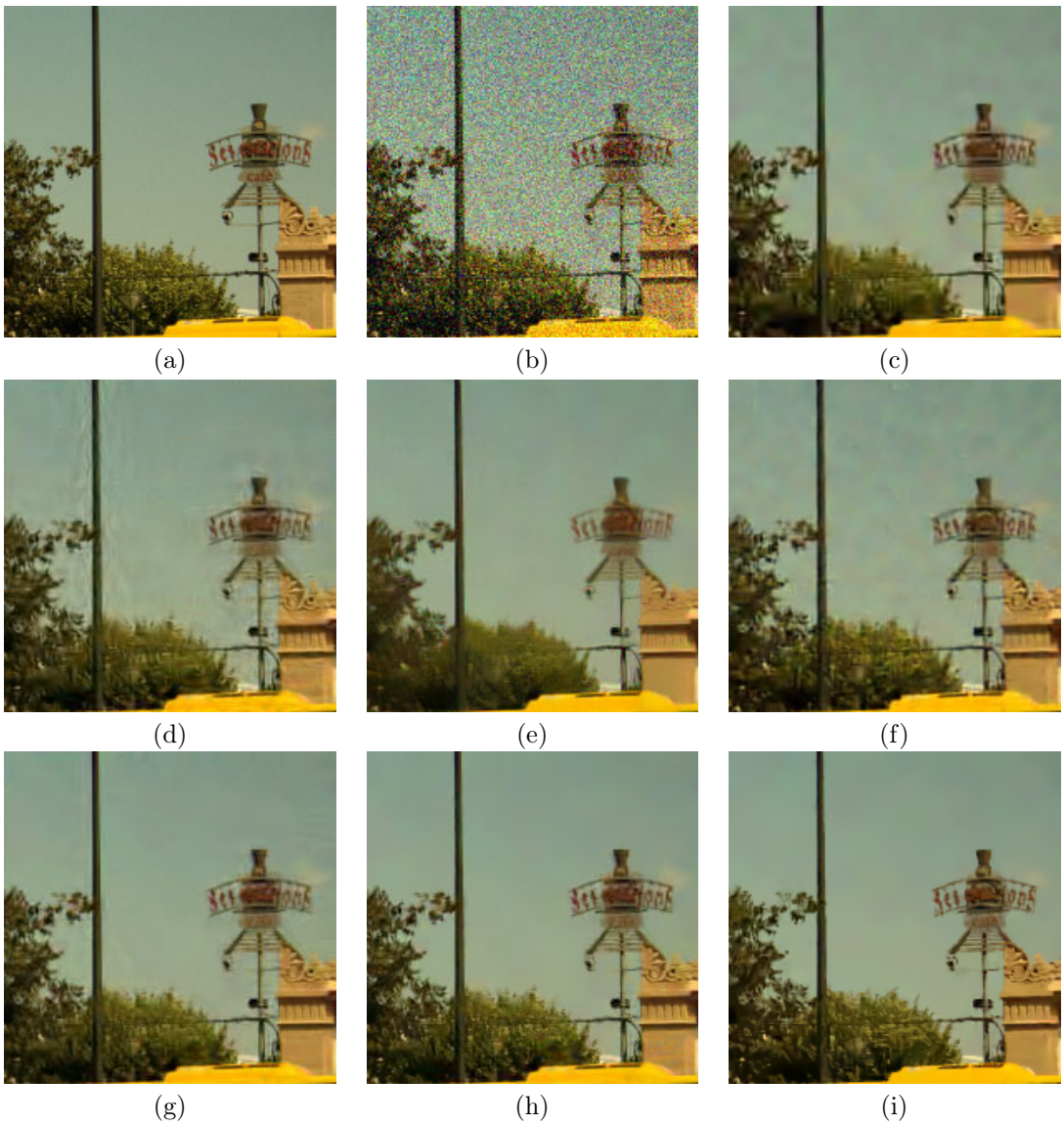


Figure 5.6: Comparison of visual quality. The noisy image was obtained adding a Gaussian white noise of standard deviation 30. From top to bottom and left to right: original, noisy, DCT sliding window, BLS-GSM, NL-means, K-SVD, BM3D, Non-local Bayes and NLDD.

Figure 5.7: Comparison of visual quality. The noisy image was obtained adding a Gaussian white noise of standard deviation 40. From top to bottom and left to right: original, noisy, DCT sliding window, BLS-GSM, NL-means, K-SVD, BM3D, Non-local Bayes and NLDD.

Chapter 6

Conclusion about Denoising Methods

After having presented the vast and prolific field of denoising (chapter 1), some common generic tools (chapter 3) shared by state of the art denoising algorithms (chapter 4) and compared them both visually and in term of PSNR (chapter 5), it is time now to summarize our knowledge and classify those methods according to their principles, their use of patches, the size of those patches, the tools they used, and their complexity. Therefore this chapter concludes this review of denoising methods in the case of white Gaussian noise, before starting to work in a more generic case in part II.
This chapter uses the results of joint work with Antoni Buades and Jean-Michel Morel.

6.1 Synthesis

We have shown that all methods either already use, or should adopt the same three *generic denoising tools* described in chapter 3. Since all methods denoise not just the pixel, but a whole neighborhood, they give several evaluations for each pixel. Thus, they all use an aggregation step. There is only one method for which the aggregation is not explicitly stated as such, the wavelet neighborhood (BLS-GSM) algorithm. Nevertheless, a closer examination shows that it denoises not one, but some 49 wavelet channels for a 512×512 image. The used wavelet transform is redundant. Thus, an aggregation is implicit in its final reconstruction step from all channels. BLS-GSM is also patch-based. Indeed, each "wavelet neighborhood" contains a 3×3 patch of a wavelet channel, complemented with one more sample from the down-scale channel sharing the same orientation. Thus, like the others, this algorithm builds Bayesian estimates of patches. The difference is that the patches belong to the wavelet channels. Each one of these channels is denoised separately, before the reconstruction of the image from its wavelet channels.
In short, even if the BLS-GSM formalization looks at first different from the other algorithms, it relies on similar principles: it estimates patch models to denoise them, and aggregates the results. But, it also is the only multiscale algorithm among those considered here. Indeed, it denoises the image at all scales. Furthermore, it introduces a scale interaction. These features are neglected in the other algorithms and might make a significant difference in future algorithms.
It may be asked why its performance is slightly inferior to that of the current state of the art algorithms. First of all, this algorithm, like many wavelet based algorithms, has not proposed a good solution to deal with colour. Applying the colour space tool of section 3.3 can probably bring a PSNR improvement. The paper does not specify if there is an aggregation step, but a first aggregation step is possible (the second aggregation being implicit in the reconstruction step from all channels, that are redundant). Indeed, each wavelet channel patch contains ten coefficients, and these coefficients are therefore estimated ten times. These estimates might be aggregated.

Table 6.1 shows a synopsis of the ten methods that have been thoroughly discussed. The classification criteria are: the **denoising principles**, the use of **patches**, the **size** of the patches, the use of **aggregation**, **oracle** or **color space transform**, and finally their **complexity**.

6.2 The denoising principles

Our task here is to show that, in spite of the different language used by each method, the underlying principles actually converge. The dominant principle is to compute a linear minimum least square estimator (LMMSE) after building a Bayesian patch model. As a matter of fact, even if this is not always explicit, *all* methods follow very closely the same LMMSE estimator principle. For example the DCT threshold is nothing but a Wiener thresholding version of the Bayesian LMMSE. This threshold is used because the DCT of the underlying noiseless image is actually unknown. The same argument applies for Nonlocal Means, which was interpreted as an LMMSE in section 4.1. A close examination of K-SVD can convince a practitioner that this algorithm is very close to EPL, PLOW or EPLL, and conversely. Indeed, the patch clustering performed in these three algorithms interprets the patch space as a redundant dictionary. Each cluster is treated by a Bayesian estimator as a Gaussian vector, for which an orthogonal eigenvector basis is computed. This basis is computed from the cluster patches by PCA. Thus, EPL, PLOW and EPLL actually deliver a dictionary, which is the union of several orthogonal bases of patches. EPL, PLOW and EPLL select for each noisy patch one of the bases, on which the patch will be sparse. In short, like K-SVD, they compute for each patch a sparse representation in an over-complete dictionary. In this argument, we follow the simple and intelligent interpretation proposed with the PLE method in [141], [140]. Their method was summarized by the authors as follows:

> An image representation framework based on structured sparse model selection is introduced in this work. The corresponding modeling dictionary is comprised of a family of learnt orthogonal bases. For an image patch, a model is first selected from this dictionary through linear approximation in a best basis, and the signal estimation is then calculated with the selected model. The model selection leads to a guaranteed near optimal denoising estimator. The degree of freedom in the model selection is equal to the number of the bases, typically about 10 for natural images, and is significantly lower than with traditional over-complete dictionary approaches, stabilizing the representation.

From the algorithmic viewpoint, EPLL is a variant of PLE, but used in a different setting. The comparison of these two almost identical Gaussian mixture models is of particular interest. EPLL is applied to a huge set of patches (of the order of 10^{10}) united in some 200 clusters. PLE is applied with 19 clusters learnt each from some 64 patches. Thus, the open question is: how many clusters and how many learning patches are actually necessary to obtain the best PSNR? The disparity between these figures is certainly too large to be realistic.
We must finally wonder if transform thresholding methods fit into the united view of all algorithms. The Bayesian-Gaussian estimate used by most mentioned algorithms can be interpreted as a Wiener filter on the eigenvector basis of the Gaussian. It includes sometimes a threshold (to avoid negative eigenvalues for the covariance matrix of the Gaussian vector). Thus, the only difference between Bayesian-Gaussian methods and the classic transform thresholding is that in the Bayesian methods the orthogonal basis is adapted to each patch. Therefore, they appear to be a direct extension of transform thresholding methods, and have logically replaced them. BM3D combines several linear transform thresholds (2D-bior 1.5, 2D-DCT, 1D-Walsh-Hadamard), applied to the 3D block obtained by grouping similar patches. Clearly, it has found by a rather systematic exploration the right 2D orthogonal bases, and therefore does not need to estimate them for each patch group.
We shall now reunite two groups of methods that are only superficially different. Non-local Means, Non-local Bayes, Shotgun-NL, and BM3D denoise a patch after comparing it to a group of similar patches. The other five patch-based Bayesian methods *do not perform a search for similar patches.*

These other patch methods, PLE, PLOW, EPLL, BLS-GSM and K-SVD, process globally the "patch space" and construct patch models. Nevertheless, this difference is easily reduced. Indeed, EPL, PLOW and EPLL segment the patch space into a sufficient number of clusters, each one endowed with a rich structure (an orthonormal basis). Thus, the patches contributing to the denoising of a given patch estimation are not compared to each other, but they are compared to the clusters. Similarly, the dictionary based methods like K-SVD propose over-complete dictionaries learnt from the image or from a set of images. Finding the best elements of the dictionary to decompose a given patch, as K-SVD does, amounts to classify this patch. This is what is suggested by the authors of PLE in [141]: the dictionary is a list of orthogonal bases which are initiated by sets of oriented edges. Each basis is therefore associated with an orientation (plus one associated with the DCT). Thus PLE is very similar to BLS-GSM, which directly applies a set of oriented filters. Another link between the Bayesian method and sparse modeling is elaborated in [143].

6.3 Patches

The second column in the classification table 6.1 indicates the number of patches used for the denoising method, and where they are found. The trivial DCT uses only the current patch to denoise it; Non-local Means, Non-local Bayes and BM3D compare the reference patch with a few hundred patches in its spatial neighborhood; PLE, PLOW, BLS-GSM and K-SVD compare each noisy patch to a learnt model of all image patches; finally Shotgun-NL and EPLL involve in the estimation a virtually infinite number of patches. Surprisingly enough, the performance of all methods are relatively similar. Thus, the huge numbers used to denoise in Shotgun-NL and EPLL clearly depend on the fact that the patches were not learnt from the image itself, and their number can arguably be considerably reduced.

6.4 Size of patches

The third column in our synoptic table compares the patch sizes. All methods without an exception try to deduce the correct value of a given pixel $\mathbf{i}$ by using a neighborhood of $\mathbf{i}$ called patch. This patch size goes from 3×3 to 8×8, with a strong dominance of 8×8 patches. Nevertheless, the size of the patches obviously depends on the amount of noise and should be adapted to the noise standard deviation. For very large noises, a size 8×8 can be insufficient, while for small noises small patches might be better. As a matter of fact, all articles focus on noise standard deviations around 30 (most algorithms are tested for σ between 20 and 80). There is little work on small noise (below 10). For large noise, above 50, most algorithms do not deliver a satisfactory result and most papers show denoising results for $20 \leq \sigma \leq 40$. This may also explain the homogeneity of the patch size.

6.5 Aggregation, Oracle, and Color Space Transform

A good sign of maturity of the methods is that the three generic improvement tools described in chapter 3 are used by most methods. When a "no" is present in the table on these three columns, this indicates that the method can probably be substantially improved with little effort by using the corresponding tool. Shotgun-NL and BLS-GSM can probably gain some decibels by aggregation and by the Oracle strategy.

6.6 Complexity and Information

Current research is focusing on getting the best ever, perhaps even the best denoising results, for ever. We have followed this track and have completely disregarded the complexity issue in this comparison. For example, the "shotgun" patch methods are not reproducible in acceptable time.

Method	Denoising principle	Patches	size	Aggr.	Oracle	Colour
DCT	transform threshold	one	8	yes	yes	yes
Non-local Means	average	neighborhood	3	yes	yes	no
Non-local Bayes	Bayes	neighborhood	3-7	yes	yes	yes
PLOW	Bayes, 15 clusters	image	11	yes	yes	yes
Shotgun-NL	Bayes	10^{10} patches	3-20	yes	no	no
EPLL	Bayes, 200 clusters	2.10^{10} patches	8	yes	yes	yes
BLS-GSM	Bayes in GSM	Image	3	yes	no	no
K-SVD	sparse dictionary	Image	8	yes	yes	yes
BM3D	transform threshold	neighborhood	8-12	yes	yes	yes
PLE	Bayes, 19 clusters	Image	8	yes	yes	yes
DDID	transform threshold	neighborhood	31	no	yes	yes

Table 6.1: Synoptic table of all considered methods.

Yet, "all is fair in love and war". The question of how to get the best acceptable results must be solved first, by every possible means, before fast algorithms are devised. On the other hand, the complexity does not seem to be a serious obstacle. Indeed, several of the mentioned algorithms are already realizable, and five of them are even functioning online at Image Processing online `http://www.ipol.im`. Among them, at least two give state of the art results. Thus, we hold the view that complexity is not a central issue in the current debate. Another question which emerged in this study is the *amount of information needed to achieve optimal denoising.* Here, we have observed that the methods do the splits. The simplest one (DCT denoising) uses only one image patch and get results only 1dB far away from optimal results. The classic nonlocal methods only use a larger neighborhood of a given pixel, in spite of their "nonlocal" epithet. Then, an intermediate class of methods uses simultaneously all image patches. The shotgun methods use virtually all existing image patches in the world. The fact that the performance gap between them is so small seems to indicate that all obtain a decent estimate of the "patch space" around each given image patch. This also means that, arguably, there is enough information for that in just one image.

Part II

Noise Clinic

As seen in part I, most papers on denoising methods assume a white Gaussian noise model. Yet in most images handled by the public or by scientific users, the noise model is unknown and is not white, because of the various processes applied to the image before it reaches the user: scanning, demosaicing, compression, deconvolution, etc. Therefore, the noise needs to be estimated, in order to obtain a signal-dependent noise model. As we will see in chapter 7, it is even possible to estimate it from the noisy image itself. However, as it is really rare to be able to work directly on raw images, and since demosaicing is the first step of the digital camera processing chain, most of the images contain correlated noise. Therefore, chapter 8 presents the interest of estimating not only signal-dependent noise, but also frequency-dependent noise. The method gives the noise model for patches and therefore it can be used as the input to a patch-based denoiser. This connection between the noise estimation and a patch-based denoiser is done through the covariance noise matrices in the case where the denoiser is NL-Bayes. Chapter 9 proposes a blind multiscale denoising algorithm working for noise which is simultaneously signal and frequency dependent. On noisy images coming from diverse sources (JPEG, scans of old photographs,. . .) we show perceptually convincing results. This algorithm is compared to the state of the art and it is also validated on images with white noise.

Chapter 7

White Noise Estimation

As shown in part I, denoising methods require a noise model and an image model. As explained in chapter 1, natural images have a signal-dependent noise. Therefore, the Gaussian noise model used in part I is no longer valid when it comes to work on real images. However, it is relatively easy to obtain a signal-dependent noise model. As will be explained in this present chapter, it is even possible to estimate it from a single noisy image.
This chapter uses the results of joint work with Miguel Colom and Jean-Michel Morel.

7.1 Can noise be estimated from (just) one image?

Compared to the denoising literature, research on noise estimation is a poor cousin. Few papers are dedicated to this topic. Among the recent papers one can mention [144], which argues that images are scale invariant and therefore noise can be estimated by a deviation from this assumption. Unfortunately this method is not easily extendable to estimate scale dependent or signal dependent noise, like the one observed in most digital images in compressed format. As a rule of thumb, the noise model is relatively easy to estimate when the raw image comes directly from the imaging system, in which case the noise model is known and only a few parameters must be estimated. For this, efficient methods are described in [56], [55] for Poisson and Gaussian noise.
In this short review we will focus on methods that allow for local, signal and scale dependent noise. Indeed, one cannot denoise an image without knowing its noise model. It might be argued that the noise model comes with the knowledge of the imaging device. Nevertheless, the majority of images dealt with by the public or by scientists have lost this information. This loss is caused by format changes of all kinds, which may include resampling, denoising, contrast changes and compression. All of these operations change the noise model and make it *signal and scale dependent.*
The question that arises is why so many researchers are working so hard on denoising models, if their corpus of noisy images is so ill-informed.
It is common practice among image processing researchers to add the noise themselves to noise-free images to demonstrate the performance of a method. This proceeding permits to reliably evaluate the denoising performance, based on a controlled ground truth. Nevertheless the denoising performance may, after all, critically depend on how well we are able to estimate the noise. Most world images are actually encoded with lossy JPEG formats. Thus, noise is partly removed by the compression itself. Furthermore, this removal is scale dependent. For example, the JPEG 1985 format divides the image into a disjoint set of 8×8 pixels blocks, computes their DCT, quantizes the coefficients and the small ones are replaced by zero. This implies that JPEG performs a frequency dependent threshold, equivalent to a basic Wiener filter. The same is true for JPEG 2000 (based on the wavelet transform).
In addition, the Poisson noise of a raw image is signal dependent. The typical image processing operations, demosaicking, white balance and tone curve (contrast change) alter this signal-dependency in a way which depends on the image itself.

Figure 7.1: Two examples of the ten noise-free images used in the tests: *computer* (left) and *traffic* (right).

In short:

- the noise model is different for each image;
- the noise is signal dependent;
- the noise is scale dependent;
- the knowledge of each dependence is crucial to denoise properly any given image which is not raw, and for which the camera model is available.

Thus, estimating JPEG noise is a complex and risky procedure, as well explained in [87] and [88]. It is argued in [37] that noise can be estimated by involving a denoising algorithm. Again, this procedure is probably too risky for noise and scale dependent signal.

This section, following [18], gives a concise review and a comparison of existing noise estimation methods. The classic methods estimate white homoscedastic noise only, but they can be adapted easily to estimate signal and scale dependent noise. To test the methods, a set of ten noise-free images was used. These noiseless images were obtained by taking snapshots with a reflex camera of scenes under good lighting conditions and with a low ISO level. This means that the number of photons reaching each captor was very high, and the noise level therefore small. To reduce further the noise level, the average of each block of 5×5 pixels was computed, reducing the noise by a 5 factor. Since the images are RGB, taking the mean of the three channels reduces the noise by a further $\sqrt{3}$ factor. The (small) initial noise was therefore reduced by a $5\sqrt{3} \simeq 8.66$ factor, and the images can be considered noise-free. Two images from this noiseless set can be seen in fig. 7.1. The size of each image is 704×469 pixels.
For the uniform-noise tests, seven noise levels were applied to these noise-free images: $\sigma \in \{1, 2, 5, 10, 20, 50, 80\}$. Fig. 7.2 shows the result of adding white homoscedastic Gaussian noise with $\sigma \in \{1, 2, 5, 10, 20, 50, 80\}$ to the noise-free image *traffic*.
This study on noise estimation proceeds as follows: we review in detail in section 7.2 the method proposed in [18]. This method has all the features of the preceding methods, so we shall be able to make a rash review of them (section 7.3), followed by an overall comparison of all methods, at all noise levels. It follows that the Percentile method is the most accurate. Nevertheless, the estimation of very low noises remains slightly inaccurate, with some 20% error for noises below 2.

7.2 The Percentile method

The Percentile method, introduced in [111], is based on the fact that the histogram of the variances of all blocks in an image is affected by the edges and textures, but this alteration appears mainly on its rightmost part. The idea of the *percentile method* is to avoid the side effect of edges and textures by taking the variance of a very low percentile of the block variance histogram, and

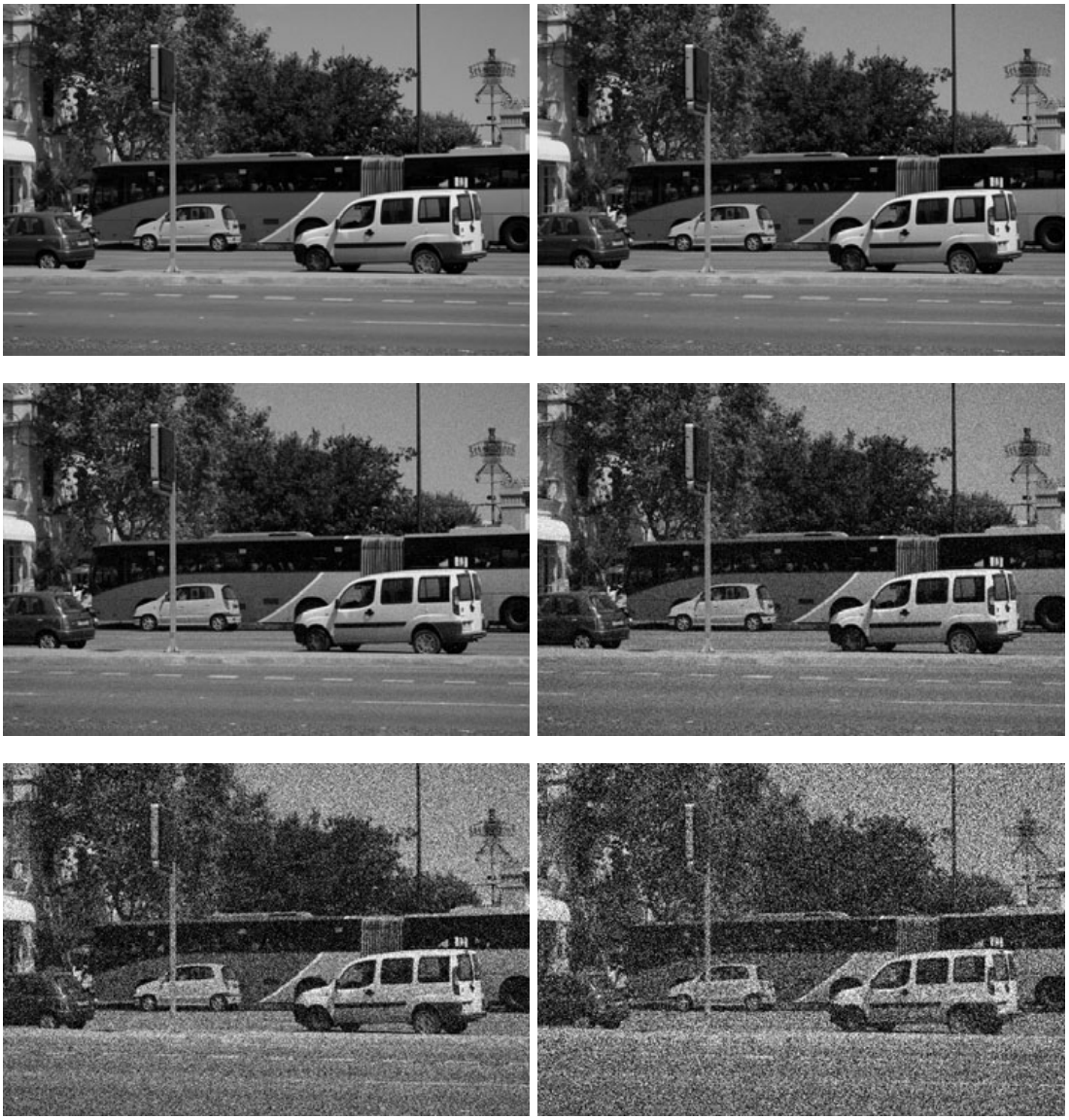


Figure 7.2: Result of adding white homoscedastic Gaussian noise with $\sigma \in \{2, 5, 10, 20, 50, 80\}$ to the noise-free image *traffic*. It may need a zoom in to perceive the noise for $\sigma = 2, 5$.

then to infer from it the real average variance of blocks containing only noise. This correction multiplies this variance by a factor that only depends on the choice of the percentile and the block size. As usual in all noise estimation methods, to reduce the presence of deterministic tendencies in the blocks, due to the signal, the image is first high passed. The commonly used high pass filters are differential operators or waveforms. The typical differential operators are directional derivatives, the Δ (Laplace) operator, its iterations $\Delta\Delta$, $\Delta\Delta\Delta$, ..., the wave forms are wavelet or DCT coefficients. All of them are implemented as discrete stencils. Filtering the image with such a local high pass filter operator removes smooth variations inside blocks, which increases the number of blocks where noise dominates and on which the variance estimate will be reliable. According to the performance tests, for observed $\hat{\sigma} < 75$ the best operator is the wave associated to the highest frequency coefficient of the transformed 2D DCT-II block with support 7×7 pixels. The coefficient $\tilde{X}(6,6)$ of the 2D DCT-II of a 7×7 block P of the image is:

$$DCT(6,6) = \sum_{n_1=0}^{6} \sum_{n_2=0}^{6} F_7(n_1) F_7(n_2) P(n_1, n_2) \cos\left[\frac{\pi}{7}\left(n_1 + \frac{1}{2}\right)6\right] \cos\left[\frac{\pi}{7}\left(n_2 + \frac{1}{2}\right)6\right].$$

where

$$F_7(n) = \begin{cases} \frac{1}{\sqrt{7}}, \text{if } n = 0 \\ \sqrt{\frac{2}{7}}, \text{if } n \in \{1, \ldots, 6\} \end{cases}$$

Therefore, the values of the associated discrete filter are

$$F_7(n_1) F_7(n_2) \cos\left[\frac{\pi}{7}\left(n_1 + \frac{1}{2}\right)6\right] \cos\left[\frac{\pi}{7}\left(n_2 + \frac{1}{2}\right)6\right], n_1, n_2 \in \{0, 1, \ldots, 6\}.$$

These values must of course be normalized in order to keep the standard deviation of the data, by dividing each value by the root of the sum of the filter squared values.

The Percentile method computes the variances of overlapping $w \times w$ blocks in the high-pass filtered image. The means of the same blocks are computed from the original image (before the high pass). These means are classified into a disjoint union of variable intervals, in such a way that each interval contains (at least) 42000 elements. These measurements permit to construct, for each interval of means, a histogram of block variance of at least 42000 samples having their means in the interval. In each such variance histogram the percentile value is computed. It was observed that, for observed $\hat{\sigma} < 75$ and large images, the percentile $p = 0.5\%$, a block size $w = 21$ and a 7×7 support for the DCT transform give the best results. If $\hat{\sigma} \geq 75$, the percentile that should be used is the median, the block is still 21×21, but the support of the DCT should be 3×3.

This percentile value is of course lower than the real average block variance, and must be corrected by a multiplicative factor. This correction only depends on the percentile, block size and on the chosen high pass filter. Nevertheless, the constant is not easy to calculate explicitly, but can be learnt from simulations. For the 0.5% percentile, 21×21 pixels blocks and the DCT pre-filter operator with support 7×7, this empirical factor learnt on noise images was found to be 1.249441884. In summary, to each interval of means, a standard deviation is associated. The association mean$\rightarrow$standard deviation yields a "noise curve" associated with the image. This noise curve predicts for each observed grey level value in the image its most likely underlying standard deviation due to noise. Optionally, the noise curve obtained on real images can be filtered. Indeed, it may present some peaks when variances measured for a given grey level interval belong to a highly-textured region. To filter the curve, the points that are above the segment that joins the points on the left and on the right are back-projected on that segment. In general, no more than two filtering iterations are needed. For the comparative tests presented here, the curves were not filtered at all.

The pseudo-code for the percentile method is given in Algo. 15 and the results for the white homoscedastic Gaussian noise in Table 7.1. When the image is tested for white homoscedastic Gaussian noise, only one interval for all grey level means is used, whereas in the signal-dependent noise case, the grey level interval is divided into seven bins.

Image / $\hat{\sigma}$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
bag	1.34	2.33	5.26	10.36	20.30	49.87	79.96
building1	1.12	2.17	5.24	10.14	20.48	50.19	80.45
computer	1.22	2.20	5.06	10.36	20.03	50.28	80.34
dice	1.11	2.00	5.01	10.03	20.02	49.95	79.79
flowers2	1.08	2.07	5.10	9.84	20.07	49.87	79.80
hose	1.15	2.13	5.10	10.15	20.06	49.99	79.99
leaves	1.51	2.43	5.38	10.29	19.82	50.07	80.04
lawn	1.57	2.50	5.57	10.48	20.42	50.05	79.92
stairs	1.42	2.27	5.19	10.15	19.96	49.92	79.93
traffic	1.25	2.35	5.33	10.61	20.64	50.10	80.29
Flat image	0.99	2.00	5.09	9.77	19.91	50.12	79.73

Table 7.1: Percentile method results on eleven noiseless images with white homoscedastic Gaussian noise added. The last image is simply flat. The real noise variance is σ. The estimated value is $\hat{\sigma}$. The noise estimation error is remarkably low on medium and large noise. It is nevertheless larger on very small noise (a $\sigma = 2$ noise is not visible with the naked eye). Indeed most photographed objects have everywhere some micro-texture (except perhaps sometimes in the blue sky which can be fully homogeneous). Such micro-textures are widespread and hardly distinguishable from noise. The parameters of the method are a 0.5% percentile, a 21×21 pixels block size, and the DCT has support 7×7. These parameters are valid if $\hat{\sigma} < 75$. If $\hat{\sigma} \geq 75$, the best parameters are: a 50% percentile, a 21×21 pixels block size and a DCT with support 3×3. Estimating the best parameters therefore requires a first estimation followed by a second one with the right parameters.

Algorithm 14 Percentile method algorithm.

PERCENTILE - Returns a list that relates the value of the image signal with its noise level.
Input $\tilde{u}$ noisy image.
Input b: number of bins.
Input $w \times w$: block dimensions.
Input p: percentile.
Input $filt$: filter iterations.
Output (M, S): list made of pairs (mean, noise standard deviation) for each bin of grey level value.

h = FILTER($\tilde{u}$). Apply high-pass filter to the image.
a, v =MEAN_FILTERED_VARIANCE($\tilde{u}, h, w$). Obtain the list of the block averages (in the original image $\tilde{u}$) and of the variances (of the filtered image h) for all $w \times w$ blocks.
Divide the block mean value list a into intervals (bins), having all the same number of elements. Keep for each interval the corresponding values in v.

$S = \emptyset$; $M = \emptyset$.
for each bin **do**
  v = Per(bin, p). Get the p-percentile v of the block variances whose means belong to this bin.

  m = Mean[Per(bin, p)]. Get the mean of the block associated to that percentile.
  $S \leftarrow \sqrt{v}$. Store the standard deviation $\hat{\sigma}$.
  $M \leftarrow m$. Store mean.
end for
$S_c = \emptyset$. Corrected values.
for $s \in S$ **do**
  Apply correction C according to p, w and filter operator used.
  $s = Cs$. Correct direct estimate.
  $S_c \leftarrow s$.
end for
for $k = 1 \ldots filt$ **do**
  $S_c[k]$ = FILTER($S_c[k]$, filt). Filter the noise curve $filt$ times.
end for

Figure 7.3: Mosaic used to learn the correction values in the Percentile method.

The Percentile method with learning The percentile method with learning is essentially the same algorithm explained in section 7.2, with the difference that it tries to compensate the bias caused by edges and micro-texture in the image by learning a relationship between observed values $\hat{\sigma}$ and noise real values σ. The difference value $f(\sigma) = \hat{\sigma} - \sigma$ is called the *correction*, that is, the value that must be subtracted from the direct estimate $\hat{\sigma}$ without correction to get the final estimate (which we shall still call $\hat{\sigma} \approx \sigma$). These corrections depend on the structure of real images. A mosaic of several noise-free images is shown in Fig. 9.4. Simulated noise of standard deviations $\sigma = 0, \ldots, 100$ was added to these noiseless images. These images were selected randomly from a large database, to be statistically representative of the natural world, with textures, edges, flat regions, dark and bright regions. The correction learnt with these images is intended to be an average correction, that works for a broad range of natural images. It should of course be adapted to any particular set of images. Furthermore, the correction depends on the size of the image, and must be learnt for each size.

When the observed noise level is high enough ($\hat{\sigma} > 10$ for pixel intensities $u \in \{0, 1, \ldots 255\}$), the image gets dominated by the noise, that is, most of the variance measured is due to the noise and not due to the micro-textures and edges. It is therefore convenient to avoid applying the learnt corrections to direct estimates $\hat{\sigma}$ when $\hat{\sigma} > 10$. Thus, for $\hat{\sigma} > 10$, only the percentile correction is applied. Table 7.2 shows the $\hat{\sigma}$ values estimated with the Percentile with learning method. The correction learnt with the mosaic is only applied for $\sigma \in \{1, 2, 5, 10\}$.

Image / $\hat{\sigma}$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
bag	1.15	2.11	5.05	10.26	20.06	49.68	80.05
building1	0.95	1.97	5.00	10.42	20.32	49.99	80.27
computer	1.04	2.00	4.88	10.39	20.13	50.29	80.16
dice	0.91	1.84	4.81	10.01	19.90	49.76	79.60
flowers2	0.92	1.88	4.87	9.47	20.00	49.48	79.67
hose	0.99	1.93	4.89	10.08	19.97	49.73	79.71
leaves	1.36	2.26	5.17	10.28	20.03	49.80	79.92
lawn	1.35	2.29	5.36	10.37	20.26	50.07	79.88
stairs	1.20	2.10	4.95	10.11	20.10	49.92	79.86
traffic	1.04	2.06	5.06	10.75	20.64	49.91	80.05
Flat image	0.84	1.82	4.84	10.02	20.13	50.13	79.44

Table 7.2: Percentile with learning method results with white homoscedastic Gaussian noise added. The correction learnt with the mosaic is only applied for $\sigma \in \{1, 2, 5, 10\}$. This method, being local on blocks, extends immediately to estimate signal dependent noise and the performance is similar [18].

7.3 A crash course on all other noise estimation methods

It is easier to explain the other methods after having explained in detail, as we did above, one method, namely the percentile method. Most noise estimation methods share the following features:

- they start by applying some high pass filter, which concentrates the image energy on boundaries, while the noise remains spatially homogeneous;
- they compute the energy on many blocks extracted from this high-passed image;
- they estimate the noise standard deviation from the values of the standard deviations of the blocks
- to avoid blocks contaminated by the underlying image, a statistics robust to (many) outliers must be applied. The methods therefore use the flattest blocks, which belong to a (low) percentile of the histogram of standard deviations of all blocks.

Table 7.3 shows a classification of the methods according the preceding criteria:

The first column is the choice of the high-pass filter, which can be a discrete differential operator of order two ($\frac{\partial^2}{\partial x \partial y}$) in the Estimation of Image Noise Variance (E.I.N.V.) method [118]). It is obtained as a composition of two forward discrete differences. Then we have a discrete Laplacian Δ [104] obtained as the difference between the current pixel value and the average of a discrete neighborhood, an order order three operator (a difference $\Delta_1 - \Delta_2$ of two different discretizations of the Laplacian [66]), a wave associated to a DCT coefficient [18], and sometimes a nonlinear discrete differential operator like in the Median method [104], which uses the difference between the image and its median value on a 3×3 block, thus equivalent to the curvature operator $curv$. The high-pass filter is previously applied to all pixels of the image. In the case of the DCT [110] the DCT is applied to a block centered on the reference pixel, and the highest frequency coefficients, for example $DCT(6, 7)$, $DCT(7, 6)$, $DCT(7, 7)$, are kept. The most primitive methods, the Block [82, 98], the Pyramid [99] and the Scatter method [84] do not apply any high pass filter. Nevertheless, since they compute block variances, they implicitly remove the mean from each block, which amounts to applying a high-pass filter of Laplacian type.

The second column gives the size of the block on which the standard deviation of the high-passed image is computed, which varies from 1 to 21. The pyramid method [99] uses standard deviations of blocks of all sizes and is unclassifiable. Two methods, F.N.V.E. [66] and the Gradient method

[9, 131] do not compute any block standard deviation of the high-passed image before the final estimation.

The last column gives the value of the (low) percentile on which the block standard deviation are computed. When the slot contains "all", this means that the estimator is taking into account all the values.

The third column characterizes the estimator, for which there are several variants. The three compared percentile methods [18] use a very low percentile 0.5% of the block standard deviations. The Average, Median [104] and Block method [82, 98] use an 1% percentile of the gradient to select the blocks which variance is kept, while the high pass image is a higher order differential operator. The Pyramid [99] is instead quite complex, but uses overall all standard deviations of all possible blocks in the image. We give up giving its detailed algorithm. The F.N.V.E. [66] method has actually no outlier elimination, taking simply the root mean square of all samples of the high-passed image.

Rather than using a percentile of the block variance histogram followed by a compensation factor, several methods extract a mode, considering that the mode (peak of the histogram variance) must correspond to the noise. The Gradient method [9, 131] takes for $\hat{\sigma}$ the peak of the modulus of the gradient histogram. The Scatter [84] method, which also computes a mode when estimating white homoscedastic noise, namely the value at which the peak of the block standard deviations histogram is attained. The E.I.N.V. [118] method does a sort of iterative deconvolution of the histogram of block variances and also extracts its mode.

All of the values obtained by these methods are proportional to the noise standard deviation when the image is a white noise. Thus the final step, not mentioned in the table, is to apply a correction factor to get the final estimated noise standard deviation, as explained in the percentile method (sec. 7.2).

The comparison of the methods which use the highest DCT coefficients, DCT-mean [110] and DCT-MAD [49] where MAD stands for median value of absolute deviations, shows clearly the win with a robust estimator: the estimation is obtained by averaging the three MAD (median of absolute deviation) of the three highest frequency DCT coefficients for all blocks.

The ultimate choice for the methods is of course steered by their RMSE, namely the root mean square error between the estimated value of σ and σ itself, taken over a representative set of images. As Table 7.4 shows the ordering of methods by their RMSE is coherent and points to the percentile method as the best one. This method is still improved by learning. A good point justifying all methods is that they perform satisfactorily for all large noise values, down to $\sigma = 20$. But, with the exception of the Percentile method with learning, no method performs acceptably for $\sigma < 5$.

Method	Hi-pass	Block	estimator	percentile
Perc. learn. [18, 111]	DCT 7×7	**21**	block dev. at perc.	0.5%
Percentile [18, 111]	DCT 7×7	**21**	block dev. at perc.	0.5%
Block [82, 98]	none	7	mean of block dev	1%
Average [104]	Δ	3	mean of block dev	1% of grad. hist.
Median [104]	*curv*	3	mean of block dev	1% of grad. hist.
Scatter [84]	none	8	block dev at	block dev mode
Gradient [9, 131]	∇	1	$\lvert\nabla\rvert$ mode	all
E.I.N.V. [118]	$\frac{\partial^2}{\partial x \partial y}$	3	deconv. of block dev.	all
F.N.V.E. [66]	$\Delta_1 - \Delta_2$	1	RMS	all
DCT-MAD [49]	3-DCT	8	MAD of 3 DCT coef	all
DCT-mean [110]	3-DCT	8	mean of variances	all
Pyramid [99]	none	2^L	block dev	complex

Table 7.3: Table summarizing all methods. The abbreviation"block dev." means standard deviation of block, "at perc 1%" means that the chosen value is the one at which the 1% percentile is attained. "3-DCT" means the three highest frequency coefficients, namely $DCT(6,7)$, $DCT(7,6)$, $DCT(7,7)$. "DCT 7×7" means the DCT wave associated to the highest frequency coefficient of the 7×7 pixels support of the DCT-II transform of the block. MAD stands for median of absolute deviation (it is applied to the three DCT coefficients for all blocks.) The methods belong to three classes. The first main class (rows 1 to 5) does: high pass+ standard deviation of blocks+ low percentile. The second class (rows 6-7) replaces the percentile by a mode of the high-pass filter histogram. The rows 8-9-10-11 are more primitive and do a simple mean of the block variances of the high-pass filtered image. The last method is unclassifiable, and performs poorly.

Method	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$	$\sigma = 80$
Percentile	**0.309**	**0.276**	**0.265**	**0.315**	**0.293**	**0.130**	**0.229**
Percentile learning	**0.182**	**0.152**	**0.157**	0.364	0.240	0.248	0.270
Block	1.093	0.961	0.949	1.056	0.984	0.922	0.840
Average	2.669	2.556	2.375	2.165	1.771	1.227	0.874
Median	2.841	2.762	2.640	2.460	2.110	1.684	1.502
Scatter	4.533	4.013	3.141	2.290	1.436	1.488	1.862
Gradient	1.887	1.851	1.474	1.393	1.354	1.234	2.949
E.I.N.V.	1.406	1.159	0.924	0.842	0.656	0.450	0.557
F.N.V.E.	2.738	2.231	1.357	0.767	0.397	**0.196**	**0.225**
DCT-MAD	0.858	0.721	0.533	**0.356**	**0.239**	0.296	0.583
DCT-mean	1.895	1.469	0.837	0.462	0.316	0.355	0.726

Table 7.4: White homoscedastic Gaussian noise RMSE results for all methods and for varying σ. The Pyramid tests were omitted, being incomplete. Being obtained as an average on many noiseless images, the differences have been checked to be statistically significant. It is also apparent that the ranking of the compared methods may vary with the amount of noise. Nevertheless, the ranks of methods for noises larger than 20 is irrelevant, because all of them work at an acceptable level of precision. Thus, this ranking is mainly relevant for low noise levels, $\sigma = 1, 2, 5, 10$.

Chapter 8

Generic Noise Estimation

Whereas it is possible to estimate a signal-dependent "white" noise from a single image as presented in chapter 7, this estimation is not accurate enough for blind denoising applications on any kind of images, especially JPEG images.
In this chapter we propose a non-parametric method to estimate both intensity and frequency dependent noise which obtains the noise model from the noisy image itself. Since demosaicing is the first step of the digital camera processing chain, most of the images contain correlated noise and hence the interest of estimating not only signal-dependent, but also frequency-dependent noise. The method gives the noise model for patches and therefore can be used as the input to a patch-based denoiser. The method applies to cases where no access is granted to the image noise model, in particular to scanned photographs and JPEG images. In order to evaluate the accuracy of the method, we validate it by comparing its estimations to the ground-truth noise curves for both raw and JPEG-encoded images and also by visual inspection of the denoising results of real images that present correlated noise, especially at the low frequencies. The proposed method overcomes the state-of-the-art.
This chapter uses the results of joint work with Miguel Colom and Jean-Michel Morel.

8.1 Introduction

Noise in a digital image comes from several sources and it is transformed at each step of the processing chain of the camera. When it is acquired at the focal plane in color filter array (the raw image), it is Poisson distributed, signal-dependent (SD) and frequency-independent. The noise at the CFA is possibly saturated and will not obey the simple linear dependency of the noise variance with the intensity [56]. Even without saturation, the variance of the noise may not follow the linear model, depending on the characteristics of the detector [7] At the very first step of the camera processing chain, to get a color image from the raw mosaic acquired at the CFA, a demosaicing algorithm is applied [67, 17], that causes the noise to be spatially correlated and therefore frequency-dependent (FD). Estimating FD noise is thus justified, since it is a required to get a color image from the CFA. Any noise estimation method assuming that the noise does not depend on the frequency or the intensity is in fact unrealistic and inadequate for denoising real images. The correlated noise after demosaicing gets saturated after white balance and specially after gamma-correction. Finally, after JPEG-encoding [132] it becomes strongly strongly frequency-dependent noise, since JPEG encoding applies a quantification matrix that quantizes the coefficients of the 8×8 blocks of the image according to their associated frequency: the higher the frequency, the higher the quantization. Therefore, the noise at the JPEG image is highly correlated, contains a low level is noise at the high-frequencies (since the quantization removes both noise and signal high frequencies) and strong low and medium frequency noise (correlated noise after demosaicing and with increased energy after gamma correction).
This profile is not only found in modern digital images, but also in scans of old photographs,

which contain chemical noise. Therefore, the assumption that the resulting noise is both signal and frequency dependent is a minimal model for proper denoising of noisy images. Our purpose here is to find a general method for estimating such complex noise, and to validate it by comparing the estimated results to the appropriate ground truths.

8.2 Noise estimation algorithm

Little has been written on frequency and signal dependent noise estimation from a digital image. A method estimating a "JPEG compression history" from a single image can be found in [101]. The noise estimation method for JPEG images proposed in [87] estimates a signal dependent noise level which is not frequency dependent and therefore only gives a "noise level". Probably the most complete attempt to estimate a general noise model is contained in the blind denoising method [113], which estimates multiscale noise covariances for noise wavelet coefficients. This model is nevertheless not signal dependent. To the best of our knowledge, no method has proposed so far to estimate a general frequency and signal dependent noise patch model. The situation is nonetheless favorable, as most homoscedastic noise estimation algorithms are actually block based [104, 115, 66, 118, 99], and can therefore be adapted to measure signal and frequency dependent noise models on patches. For a complete review on noise estimation and denoising, we refer the reader to chapter 7 and [77].
Our noise estimation method is inspired by the ideas of a paper by Ponamarenko et al. [109] to estimate frequency-dependent noise variance. However, there are fundamental differences between our method and the method by Ponamarenko et al:

1. The Ponomarenko et al. method is designed for the estimation of homoscedastic noise, but since it is block-based, it is relatively easy to adapt it to deal with SD noise.

2. The Ponomarenko et al. method uses a $w \times w$ (where $w = 8$ is the side of the block) random mask that determines which are the coefficients used for block selection and which are used for noise estimation (see Sec. 8.3 for more details). In our method, there is no such a random mask and we compute the SE (squared error) of a pair of blocks depending of the coefficients with the greatest absolute values.

3. The Ponomarenko et al. method considers the difference between pairs of blocks with the less MSE (mean squared error). Our method uses directly the blocks without subtraction. In general, this improves the estimation, as shown in Sec. 8.3.

Our proposed method follows is detailed in Algorithm 15.
The correction function in Eq. (8.5) is obtained by adding simulated homoscedastic noise to a set of noise-free images and afterwards adjusting a linear function that returns the theoretical STD given the biased estimate $\hat{\sigma}[I][i,j]$.

$$\hat{\sigma}[I][i,j] = \mathrm{MAD}(\mathcal{S}_p) = \underset{\tilde{n} \in \mathcal{S}_p}{\mathrm{median}} \left(\left| \tilde{n}[i,j] - \underset{\tilde{m} \in \mathcal{S}_p}{\mathrm{median}} \left(\tilde{m}[i,j] \right) \right| \right). \tag{8.6}$$

8.3 Discussion

As shown in previous works [77] [30], it is possible to adapt most block-based homoscedastic noise estimation methods [104] [115] [66] [118] [28] [29] to measure SD noise, by simply splitting the list of input blocks into sets of blocks disjoint in mean intensity (*bins*). In the proposed algorithm, this is done in step 14. If step is avoided or only a single bin is used, the algorithm falls back into a homoscedastic noise estimator.
In this paper, we propose a new function to compute the similarity between patches. Both the proposed method and [109] rely on the fact that the list of blocks used to measure the variance frequency-by-frequency are similar, to minimize the interference of image textures and geometry

Algorithm 15 Noise estimation algorithm

Input : Noisy image $\mathbf{u}$ of size $N_x \times N_y$ pixels.
Input : $w \times w$ size of the block in pixels.
Output : Noise curve $\tilde{\sigma}$.
Extract from the input image $\mathbf{u}$ of size $N_x \times N_y$ all possible $M = (N_x - w + 1)(N_y - w + 1)$ overlapping $w \times w$ blocks $\mathbf{B}_k$ and compute their 2D orthonormal DCT-II, $\tilde{\mathbf{B}}_k, k \in [0, M-1]$.
Set $\mathbf{L} = \emptyset$ Empty set.
for each DCT block $\tilde{m}_1 \in \tilde{\mathbf{B}}$, **do**
    Compute the absolute value of the DCT-II coefficients of $\tilde{m}_1$:

$$\mathbf{A}[i,j] := \{|\tilde{m}_1[i,j]| : [i,j] \in [0, w-1]^2\}. \tag{8.1}$$

    Sort $\mathbf{A}$ from the highest to the lowest value and put the sorting indices in

$$\mathbf{Q} = \text{argsort}(\mathbf{A}). \tag{8.2}$$

    Compute the SE between $\tilde{m}_1$ and $\tilde{m}_2$ along the first $w^2/4$ sorting indices in $\mathbf{Q}$:

$$\text{SE}_{\tilde{m}_1,\tilde{m}_2} := \sum_{q=0}^{w^2/4-1} (\tilde{m}_1[\mathbf{Q}_q] - \tilde{m}_2[\mathbf{Q}_q])^2. \tag{8.3}$$

    Find the block $\tilde{m}_2$ that minimizes $\text{SE}_{\tilde{m}_1,\tilde{m}2}$ (Eq. 8.3). Consider only those blocks whose horizontal and vertical distance with respect to $\tilde{m}_1$ belongs to the interval $[r_1, r_2] = [4, 14]$.
    Add block $\tilde{m}_1$ and its SE, $[\tilde{m}_1, \text{SE}_{\tilde{m}_1,\tilde{m}2}]$, to list $\mathbf{L}$.
end for
Extract [1] from $\tilde{m}_1$ the mean of m_1.
Classify the elements of list $\mathbf{L}$ into disjoint bins according the mean intensity of the blocks [77, 129]. Each bin contains (with the exception of the last) 42000 DCT blocks.
for each bin, **do**
    Obtain the set $\mathcal{S}_p$ made by the DCT blocks inside the current bin whose SE is below the p-quantile, with $p = 0.005$.
    Assign to the current bin the intensity I as

$$I = \underset{\tilde{m} \in \mathcal{S}_p}{\text{median}} \, (\tilde{m}[0,0]/w) \tag{8.4}$$

    for each frequency $[i,j]$ with $[i,j] \in [0, w-1]^2, [i,j] \neq [0,0]$, **do**
        Compute the (biased[2]) variance of the noise at the current bin and frequency $[i,j]$ using the MAD estimator (Eq. 8.6).
        Correct the biased variance and obtain the final standard deviation (STD) estimate:

$$\tilde{\sigma}[I][i,j] = \begin{cases} 1.775 \times \hat{\sigma}[I][i,j] & \text{if } w = 4; \\ 1.677 \times \hat{\sigma}[I][i,j] & \text{if } w = 8. \end{cases} \tag{8.5}$$

    end for
end for

in the estimation. Indeed, in the ideal case where all $w \times w$ blocks contain exactly the same signal contaminated with noise, the variations of the values at any coefficient $[i, j] \in [0, w^2 - 1]^2$ are explained only by the noise and therefore the its sample variance is exactly that of the noise. If the blocks are not similar the former does not hold, and therefore it is important to have a reliable similarity function.
The idea of using the redundancy of the visual information along different blocks to estimate the variance of the noise was already exploited by Danielyan el at. [37] in a method based on BM3D, but applied to noise estimation. In [109], the authors use a random matrix of $w \times w$ values, where a "1" at position $[i, j] \in [0, w^2 - 1]^2$ means that a coefficient must be used to compute the MSE and a "0" that it must be used to estimate the noise at that frequency. The mask is afterwards inverted and the noise variance is obtained for the rest of the coefficients. But this approach has some disadvantages:

- Only 50% of the values are used to compute the MSE between two blocks. Since the selection of the coefficients that will participate in the MSE computation is random, it may happen that the chosen coefficients contain less energy than those which are discarded in the random selection. If the discarded coefficients have more energy (for example, because they belong a to an edge or to a textured area) than the coefficients that are selected for the similarity computation, then computed MSE will not be reliable.

- In a natural image it is well known that most of the visual information is carried at the low and medium frequencies, and a very few high-frequency coefficients due to edges. Thus, most homoscedastic noise estimation methods [104] [28] [111] estimate the noise at the high-frequencies.

A function that computes the MSE between two blocks with a random selection that includes high-frequency coefficients is not reliable, since the information they bring is caused by the noise. If the random selection mask contains more high-frequencies than low and medium, the similarity function will be biased by the noise, since each coefficient has the same weight when computing the MSE. On the other hand, if only coefficients at the low and medium frequencies are used to compute the similarity, the similarity function would fail when the block contains a large number of edges or is highly textured, since in that case most of the visual information would be located at the high-frequencies.
Instead of using a random selection or fixing the frequencies used to compute the block similarity, we only consider a small set of blocks made by the coefficients with the greatest energy, regardless of their frequency. When comparing two blocks m_1 and m_2, and after computing their orthonormal DCT-II, the transformed blocks $\tilde{m}_1$ and $\tilde{m}_2$ are evaluated with the proposed similarity function. It sorts the coefficients in $\tilde{m}_1$ according to the absolute value of each coefficient (see Eqs. 8.1 and 8.2), along with their associated frequencies. Our similarity function (Eq. 8.3) computes the SE between $\tilde{m}_1$ and $\tilde{m}_2$ only with the $w^2/4$ coefficients in $\tilde{m}_1$ with the largest absolute value. This similarity function is reliable for both smooth and high-textured areas, since the coefficient selection is adapted to the block spectral characteristics. If on the contrary the noise is so high that it dominates over the geometry of the blocks, the proposed similarity function will also be reliable, since the geometry simple adds more energy at each frequency and still the coefficients with the largest energy are expected to carry information mainly from the geometry.
About the blocks that are used to compute the variance frequency-by-frequency, in [109] once the most similar DCT block $\tilde{m}_2$ as been found for block $\tilde{m}_1$, the difference block $(\tilde{m}_1 - \tilde{m}_2)$ and their MSE are stored. In the proposed method, we store block $\tilde{m}_1$ and the corresponding MSE instead of the difference. Theoretically, when computing the variance of a set of coefficients at a fixed frequency $[i, j] \in [0, w^2 - 1]$, the only difference should be a factor of 2 at the variance computed at the difference with respect to the variance without subtraction. However, since the blocks that are subtracted are assumed to be very similar (minimal MSE), the computation of the variance may be less accurate numerically when computed using a finite precision. Indeed, if we compute the variance of K coefficients $c_k[i, j]$ at frequency $[i, j]$ with $[i, j] \in [0, w^2 - 1], k \in [0, K - 1]$ we

have that

$$\mathrm{Var}\left(\{c_k\}_{k=0\ldots K-1}\right) = \frac{K}{K-1}\left[\frac{1}{K}\sum_{k=0}^{K-1} c_k^2 - \left(\frac{1}{K}\sum_{k=0}^{K-1} c_k\right)^2\right].$$

If the values of c_k are small, the values of c_k^2 might be too small to be represented accurately with the `float` and `double` data types commonly available in programming languages, causing numeric inaccuracy or even catastrophic cancellation. In fact, the accuracy when computing the variance frequency-by-frequency is not improved if the blocks are subtracted. Indeed, if the blocks considered for noise estimation are not similar enough, the effect of a slightly different underlying geometry in the set of blocks can be seen as a perturbation that is added to the noisy signal. But the variance of the the set of difference coefficients with the added perturbation is just the double of the variance of the set of coefficients with the added perturbation without subtraction. Therefore, it is always better to avoid subtracting the blocks (see step 11 of the algorithm in Sec. 8.2), since there is not any theoretical or practical improvement in doing so, but the results may be inaccurate because the values may be too small to be represented with the data types used to compute the variance.

Eq. 8.5 gives the correction factor for the STD depending on the size of the blocks, for percentile $p = 0.5\%$. A correction of the STD is needed because MAD is a biased estimator of the STD and also because the available number of coefficients to compute the (sample) variance is finite and thus biased. To obtain the correction factors, we added simulated homoscedastic noise of STD $\sigma = 5$ to a synthetic image of a calibration pattern with large flat zones of several different grayscale intensities. The biased STD $\hat{\sigma}$ is estimated with our algorithm and compared with $\sigma = 5$. The ratio $\sigma/\hat{\sigma}$ gives the correction factor.

The size of the block depends on the application of the noise estimation. A small block, say 4×4 ($w = 4$) is useful if it is already known in advance that the image is highly textured, since the probability of capturing float zones or at least with less edges decreases with the size of the block. However, a larger block is associated with an estimator of the STD with less standard error (since more samples are available), but on the other hand the probability of capturing edges and highly textured zones in a single block increases. A block smaller than 4×4 DCT coefficients is not informative enough and blocks larger that 8×8 are not robust to edges and textures. The 8×8 size has an special interest, since it is adapted to JPEG compression, which quantizes the 8×8 DCT blocks of the image.

8.4 Validation of the method

The above proposed method gives an estimation of the standard deviation (STD) of the noise that depends both on the intensity and frequency in a single image. It uses the observation of blocks at many spatial locations and is therefore called the *spatial* estimation. We can validate the spatial estimation method by taking raw and JPEG photographs with a given camera. The value of the spatially estimated STD *on a single image* should match the ground-truth STD for that camera for the configured ISO speed [56], obtained from multiple frames. Note that with *JPEG images* we do not refer to simply compressing the image according to the JPEG standard, but to a raw image that has gone throughout all the camera processing chain, from the raw image acquired at the focal plane of the camera at the CCD o CMOS detector, including demosaicing, white balance, gamma correction and finally JPEG encoding.

For that purpose, consider a sequence of images of the same scene taken with fixed camera position and constant lighting. Under these conditions, any variation of the intensity in any pixel through the sequence is only attributable to the effect of the noise. It is therefore possible to build a GT noise curve for both raw and JPEG-encoded images, associating with each observed mean signal value the corresponding standard deviation of its observed samples. Similarly, by **frequency noise curve** we mean a numerical function associating with each value of the block mean a standard deviation (STD) of the DCT coefficient of the noise at that frequency. Thus, there are as many noise curves as DCT coefficients. To obtain such curves, instead of measuring the variation

of the intensity of the pixels in a fixed position along the sequence, we consider all M overlapping $w \times w$ blocks in the image, compute their orthonormal DCT-II, and measure the variance at the intensity of the bin and frequency $[i, j] \in [0, w-1]^2, [i, j] \neq [0, 0]$ along the coefficients of the blocks at the same spatial position and with varying image index.

The noise curve obtained this way for *each DCT frequency* is called the *temporal* estimation and can be used as a ground truth (GT) to compare with the spatial estimation. Even if a noise model for JPEG images has never been proposed in the literature, it is therefore possible to obtain reliable empirical GT curves for JPEG images. To obtain them, it suffices to JPEG-encode each image of the snapshot with the same quality parameter, and to apply the described procedure. Note that the fixed pattern noise (FPN) is also measured with the spatial estimation, but not at the temporal. However, it can be neglected.

The objective of this section is to verify that the spatial standard deviation (STD) measured at any frequency $[i, j] \in [0, w-1]^2, [i, j] \neq [0, 0]$ using the algorithm in Sec. 8.2 coincides with the STD of the temporal series measured only at that frequency for all intensities. To build the temporal STD noise curve we used 100 snapshots of the same calibration pattern, for both raw and JPEG-encoded images. In principle, any image might be used to get the temporal STD of the noise, but it is preferable to use an object with large flat regions of different gray levels, in order to avoid the effect of textures in the temporal estimation. To be robust to outliers (the edges between the large flat zones), we consider only the 0.05-quantile [28] of the STD estimations that is corrected afterwards to obtain an unbiased estimate.

The procedure to compute the ground-truth curve for JPEG-encoded images for frequency $[i, j] \in [0, w-1]^2, [i, j] \neq [0, 0]$ from a set of H images is detailed in Algorithm 16.

In the sequel, we compare the results of the spatial estimation to the GT, for both raw and JPEG-encoded images taken with a Canon EOS 30D camera with exposure time $t = 1/30s$, ISO speed 1600, and blocks of $w \times w$ DCT coefficients with $w = 4$ and $w = 8$. Fig. 8.1 compares the temporal and the spatial STDs for raw images and Fig. 8.3 shows the same for JPEG-encoded images with compression factor $Q = 92$ for $w = 4$. Only coefficients $[1, 1]$, $[2, 2]$, and $[3, 3]$ are shown, but equivalent results are obtained with all 15 coefficients. Respectively, Fig. 8.2 and Fig. 8.4 for $w = 8$. Only coefficients $[2, 2]$, $[5, 5]$, and $[7, 7]$ are shown for $w = 8$. The average of the estimations along all coefficients $[i, j] \in [0, w-1]^2, [i, j] \neq [0, 0]$ is also given in both cases.

Despite small oscillations in the spatial estimation, there is an accurate match between both the spatial and temporal estimations in the case of raw and JPEG images. It can be concluded that the method is able to estimate reliably signal-dependent noise at each frequency.

Note that this test was performed with snapshots of the calibration pattern, which is not textured and contains large flat areas whose spatial variations are caused mainly by the noise. Thus, the final validation must use real natural images compressed with JPEG. Since a proper noise model for JPEG encoding has not been already described, a visual comparison of the quality of the images before and after denoising using the frequency-by-frequency estimation given by the proposed method is needed.

We also compared the accuracy of the proposed method by simulating colored noise and comparing the temporal STD (GT) with the spatial estimation given by our algorithm for images of pure noise, frequency by frequency. To obtain the temporal STD, we created a list of 210 blocks of size 8×8 pixels made of simulated Gaussian noise of mean 127 and $\sigma = 10$ after applying a convolution with the discrete Gaussian kernel $\mathcal{G}$ in Eq. 8.7. Fig. 8.5 shows a crop of the convolved noise image, where it can be seen that has some spatial structure, since the noise is correlated because of the Gaussian convolution.

$$\mathcal{G} = \frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}. \tag{8.7}$$

Table 8.1 compares for some frequencies (first column) the temporal STD obtained for the 210

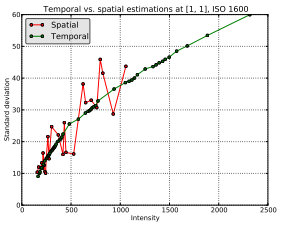
Algorithm 16 Algorithm to obtain the ground-truth curve for a sequence of images

Input : Sequence of JPEG (or raw) images.
Input : H number of images in the sequence.
Input : $Nx \times N_y$ the size of each image in the sequence, in pixels.
Output : GT noise curve $\tilde{\sigma}$.

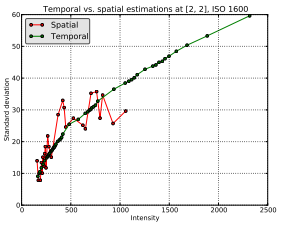
Set $M = (N_x - w + 1)(N_y - w + 1)$ the number of overlapping blocks.
Set $\text{E1} = \text{E2} = \text{E3} = \text{zeros}(M)$.
for For each JPEG (or raw) image of the series of H images, **do**
    Extract from the input image $\mathbf{u}$ of size $N_x \times N_y$ all possible M overlapping $w \times w$ blocks $\mathbf{B}_k$ and compute their 2D orthonormal DCT-II, $\tilde{\mathbf{B}}_k, k \in [0, M-1]$.
    for $k \in [0, M-1]$ **do**
        $\text{E1}[k] = \text{E1}[k] + (\tilde{\mathbf{B}}_k[i,j])^2$.
        $\text{E2}[k] = \text{E2}[k] + \tilde{\mathbf{B}}_k[i,j]$.
        $\text{E3}[k] = \text{E3}[k] + \tilde{\mathbf{B}}_k[0,0]/w$. The mean of $\mathbf{B}_k$
    end for
    $\text{E1}[k] = \text{E1}[k]/H$; $\text{E2}[k] = \text{E2}[k]/H$; $\text{E3}[k] = \text{E3}[k]/H$. Normalization
end for
Set $\mathbf{L} = \text{zeros}(M)$
for $k \in [0, M-1]$ **do**
    Set $\mathbf{L}[k] = \left[\frac{k}{k-1}\left(\text{E1}[k] - (\text{E2}[k])^2\right)\right]^{1/2}$. STD
end for
Classify the elements of list $\mathbf{L}$ into disjoint bins [77, 129] according the mean intensity $\text{E3}[k]$ of the blocks. Each bin contains (with the exception of the last) 42000 sample variance estimations.
for each bin b, **do**
    Set $\mathbf{X}$ the means of the blocks in bin b.
    Set $\mathbf{Y}$ the STDs of the blocks in bin b.
    Get the 0.05-quantile of $\mathbf{Y}$ and set $\hat{\mu}$ the mean in $\mathbf{X}$ associated with it.
    Assign the 0.05-quantile of $\mathbf{Y}$ to $\hat{\sigma}[\hat{\mu}][i,j]$.
    Set $\hat{\sigma}[\hat{\mu}][i,j]$ the 0.05-quantile of $\mathbf{X}$. Set $\hat{\mu}$ the mean at the quantile position in $\mathbf{X}$.
    Correct $\hat{\sigma}$ biased by the quantile and obtain the final control point of the GT for intensity $\hat{\mu}$ and frequency $[i,j]$:

$$\tilde{\sigma}[\hat{\mu}][i,j] = 1.22 \times \hat{\sigma}[\hat{\mu}][i,j].$$

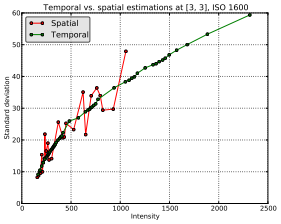
end for



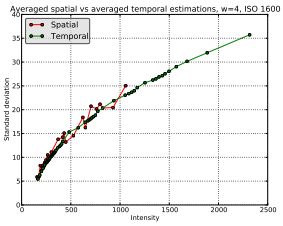
Frequency $[1, 1]$



Frequency $[2, 2]$

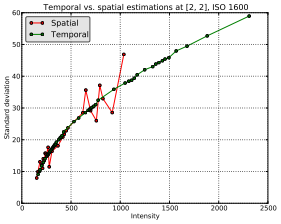


Frequency $[3, 3]$

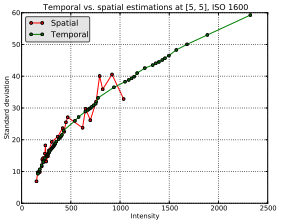


Average comparison

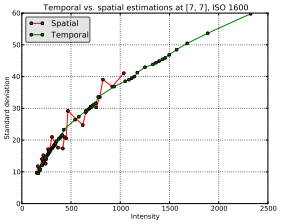
Figure 8.1: Comparison of the temporal GT (in green) and spatial STD (in red) for the Canon EOS 30D in raw images for ISO speed 1600 using blocks of 4×4 DCT coefficients. The temporal and spatial STD match despite some oscillation in the spatial estimation. The curve at the bottom right is the comparison between the averaged mean temporal STDs and the averaged mean spatial STDs (along all frequencies except DC), showing that in average both estimations match accurately.



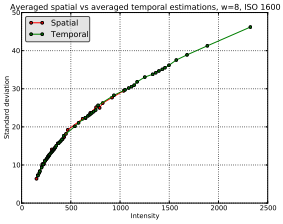
Frequency $[2, 2]$



Frequency $[5, 5]$

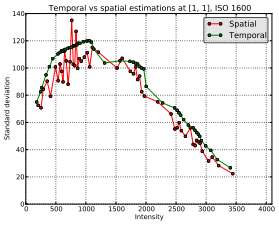


Frequency $[7, 7]$

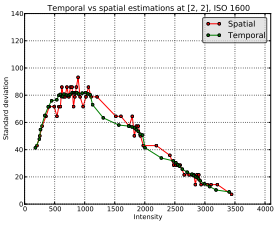


Average comparison

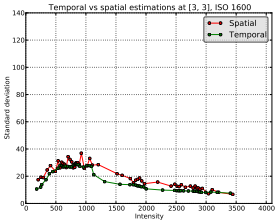
Figure 8.2: Comparison of the temporal GT (in green) and spatial STD (in red) for the Canon EOS 30D in raw images for ISO speed 1600 using blocks of 8×8 DCT coefficients. The temporal and spatial STD match despite some oscillation in the spatial estimation. The curve at the bottom right is the comparison between the averaged mean temporal STDs and the averaged mean spatial STDs (along all frequencies except DC), showing that in average both estimations match accurately.



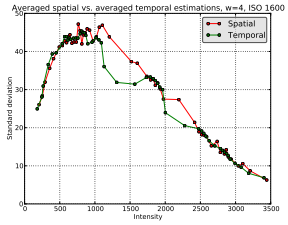
Frequency $[1, 1]$



Frequency $[2, 2]$

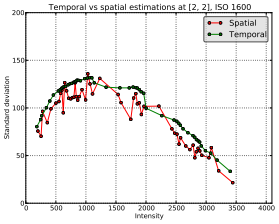


Frequency $[3, 3]$

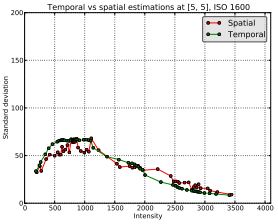


Average comparison

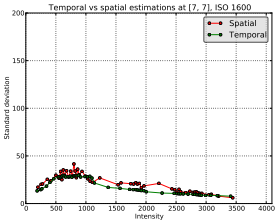
Figure 8.3: Comparison of the temporal GT (in green) and spatial STD (in red) for the Canon EOS 30D in JPEG-encoded images with quality factor $Q = 92$ for ISO speed 1600 using blocks of 4×4 DCT coefficients. The curve at the bottom right is the comparison between the averaged temporal STDs and the averaged mean spatial STDs (along all frequencies except DC), showing that in average both estimations match.



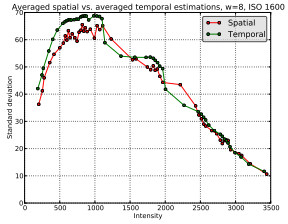
Frequency $[2, 2]$



Frequency $[5, 5]$



Frequency $[7, 7]$



Average comparison

Figure 8.4: Comparison of the temporal GT (in green) and spatial STD (in red) for the Canon EOS 30D in JPEG-encoded images with quality factor $Q = 92$ for ISO speed 1600 using blocks of 8×8 DCT coefficients. The curve at the bottom right is the comparison between the averaged temporal STDs and the averaged mean spatial STDs (along all frequencies except DC), showing that in average both estimations match.

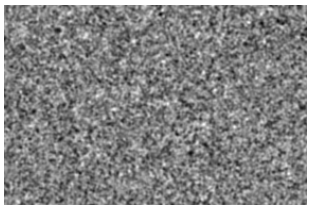


Figure 8.5: Crop of the image of pure Gaussian noise with mean 127 and $\sigma = 50$ after convolution with the kernel $\mathcal{G}$ in Eq. 8.7. The noise has spatial structure since it gets correlated after Gaussian convolution.

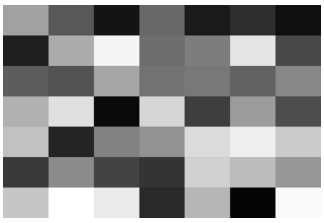
Table 8.1: This table compares, for some frequencies (first column), the temporal STD obtained for the 210 8×8 blocks of Gaussian noise of $\sigma = 50$ (second column), the spatial STD estimation obtained by our method for pure noise after convolution with the Gaussian kernel $\mathcal{G}$ in Eq. 8.7 (third column), and the spatial STD estimation given by our algorithm after adding homoscedastic Gaussian noise of $\sigma = 50$ to the noise-free test image *computer* and then convolving it with $\mathcal{G}$ (fourth column). Both STD estimation in pure noise and in a textured natural image match with small error the temporal STD. For the image of pure noise a single bin is used and 7 bins for the *computer* image.

Frequency	**Temporal STD**	**Spatial (pure noise)**	**Spatial (*computer*)**
$[1, 1]$	31.45	29.20	30.72
$[2, 2]$	21.77	19.93	21.06
$[3, 3]$	10.03	11.16	10.73
$[4, 4]$	3.44	3.72	3.76
$[5, 5]$	0.73	0.62	0.61
$[6, 6]$	0.15	0	0
$[7, 7]$	0.13	0	0

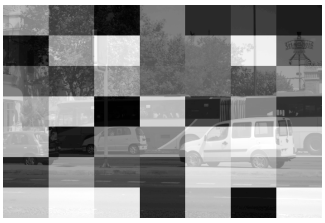
8×8 blocks of pure noise (second column), the spatial STD estimation obtained by our method for pure noise after convolution with the discrete Gaussian kernel $\mathcal{G}$ in Eq. 8.7 (third column), and the spatial STD estimation given by our algorithm after adding homoscedastic Gaussian noise of $\sigma = 50$ to the noise-free test image *computer* (in Fig. 8.7, top right) and then convolving it with $\mathcal{G}$ (fourth column). Despite an small error, the proposed method is able to measure accurately the STD of the noise for both pure noise and a textured natural image. If the STD of the noise is below 0.4, the method is unable to estimate it accurately and in some cases the MAD estimator gives negative values that the algorithm sets to zero afterwards.

8.4.1 Comparison

In this section we discuss the influence of in two decisions taken in the design of the algorithm: the subtraction or not of similar blocks in the list of DCT blocks under the 0.005-quantile and the performance of the new similarity function (Eq. 8.3) we propose in this paper. In order to measure the influence of textures in the performance of the compared noise estimation methods, we propose to use a synthetic noise-free calibration pattern (see Fig. 8.6, left). Since in the calibration pattern is very easy to find flat zones where any variation of the intensity is due to the noise and no interfering textures, most noise estimation methods are expected to perform optimally for this image. To simulate the effect of textures, we consider an image that combines both the calibration pattern and a noise-free image. For example, in Fig. 8.6 (right) we show the noise-free image made by the 80% of the intensity of the calibration pattern and the 20% of noise-free image *traffic*. Note that since both combined images are noise-free, the result is still



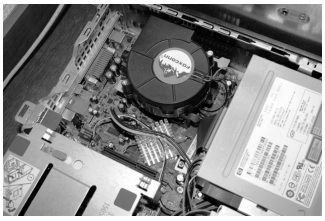
Noise-free calibration pattern



Combined with *traffic*

Figure 8.6: On the left, synthetic noise-free calibration pattern. On the right, the combination of the calibration pattern with the noise-free test image *traffic* (with a ponderation of the 80% of the calibration pattern and 20% of the *traffic* image. Since both combined images are noise-free, the result is still noise-free, but textured.

Bag



Computer

Flowers

Traffic

Figure 8.7: Noise-free images used to measure the robustness of the methods to the presence of textures. Each image is 704×469 pixels.

noise-free, but textured.

To show the influence of the textures in the noise estimation depending on the method, we added simulated signal-dependent noise of variance $\sigma^2 = 100 + 7\mathbf{u}$ to the combination of the calibration pattern image with several noise-free images and then estimated the RMSE along all frequencies and intensity bins ($\mathbf{u}$ is the pixel intensity of the combined image). The level of texture of the image analyzed is controlled by parameter α, since the combination is done as $\alpha\mathcal{P} + (1-\alpha)\mathcal{T}$, where $\mathcal{P}$ is the calibration pattern and $\mathcal{T}$ the noise-free image that has the role of the texture. We used the four noise-free images show in Fig. 8.7.

Fig. 8.8 shows the RMSEs obtained for the test images in Fig. 8.7. In the horizontal axis, the value of $\alpha \in [0, 1]$ (the texture level) and in the vertical axis, the RMSE along all frequencies and intensity bins.

We compare the adaptation to SD noise of the Ponomarenko et al. method [109] (which can be considered the state-of-the-art in FD-noise estimation) and our method, with two variants for each method: subtracting the blocks under the MSE quantile and not subtracting them (look at the discussion in Sec. 8.3). It can be observed that with independence of the texture level, in general it is better not to subtract similar blocks before estimation. Compare the Ponomarenko method (labeled *Ponomarenko sub*) with the variant which does not subtract similar blocks (labeled *Ponomarenko no-sub*): the estimation obtained with subtraction has less RMSE than when performing

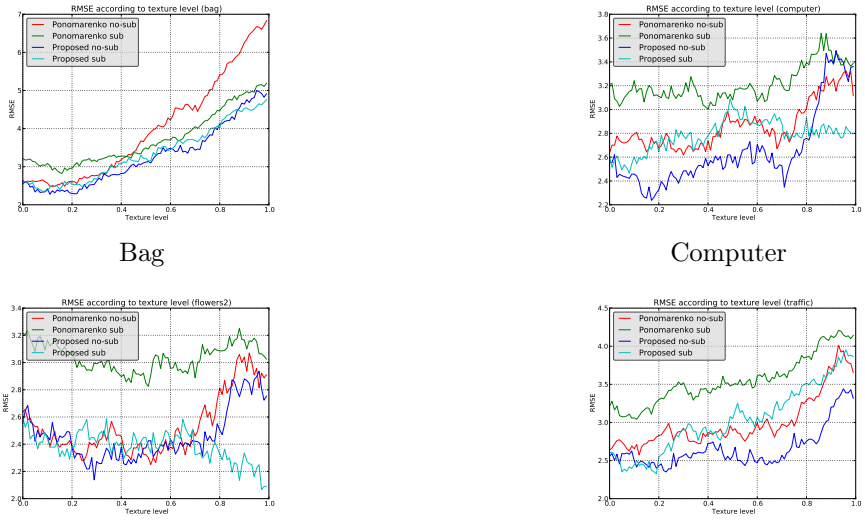


Figure 8.8: RMSEs obtained for the test images in Fig. 8.7. In the horizontal axis, the value of $\alpha \in [0, 1]$ (the texture level) and in the vertical axis, the RMSE along all frequencies and intensity bins. We compare the adaptation to SD noise of the Ponomarenko et al. method [109] and our method, and two variants for each method: subtracting the blocks under the MSE quantile and not subtracting them (look at the discussion in Sec. 8.3). In general, avoiding the subtraction is the best option and it is what we propose in the presented method.

subtraction. However, only in the case of extremely highly-textured images (as in the case of the *bag* image), the subtraction brings to a lower RMSE. The plots also compare the proposed method (labeled *Proposed no-sub*) with the variant that subtracts the blocks (labeled *Proposed sub*). It can be seen clearly that the proposed method gives a lower RMSE thanks to the use of a better similarity function (Eq. 8.3). The new similarity function is less affected by textures, since it measures the difference between blocks using only the coefficients with most energy. This coefficients are related to the geometry of the image and are not biased by other coefficients that carry information from the noise (as it happens in [109]). In the plot of the *Bag* image it can be seen that for high values of α, the RMSE of the variant that does not subtract the blocks is similar to the variant subtracting the blocks.
Depending on the image (*computer*, *flowers*), the variant subtracting the blocks is slightly better than the variant with subtraction, for high values of α, but only for highly-textured images, as shown. Nevertheless, the proposed method has a better RMSE that [109]. In general, avoiding the subtraction is the best option and it is what we propose in the presented method.

8.5 Conclusion

We presented a non-parametric noise estimation method for intensity-frequency dependent noise. It can be applied to images where the noise model is not available [7], as in the case of JPEG images. In general, a non-parametric estimation of intensity-dependent and frequency-dependent noise is needed for almost any kind of image, since the very first step in the camera processing chain is demosaicing the raw image, which correlates the noise. The exact algorithm used to demosaick the image is not made available by camera makers. However, the goal of blind noise estimation and denoising is to retrieve the noise model from the image itself, without relying at additional information such as metadata in the file format. Of course, for old photographs the metadata information might not exist at all (analogic photography) or may have been lost after

image manipulation and reencoding.
Instead of assuming a prefixed noise model and then obtaining the parameters that control it (as parametric models do), our non-parametric method obtains at the same time both the noise model for the patches and its characteristics, that is, the noise estimation according to the discovered model. The method was validated by showing that the STD obtained at the temporal series (the GT) coincides with the spatial STD given by the proposed algorithm, for both raw and JPEG images. The denoising results show that indeed the noise estimator is able to give an accurate estimation, since low frequency noise is removed and most of the fine details are kept. Our next endeavor would be to include an impulse noise estimator to the non-parametric noise estimation model. Old photographs can indeed present this sort of noise. Nevertheless our estimation algorithm cannot be applied to *any* noisy image. For example, it does not apply if the noise is space dependent (and not only signal dependent), as can be observed in some synthetic images.

Chapter 9

Noise Clinic

Arguably several thousands papers are dedicated to image denoising. As explained in part I, most papers assume a fixed noise model, mainly white Gaussian or Poissonian. This assumption is only valid for raw images.
Yet in most images handled by the public and even by scientists, the noise model is imperfectly known or unknown. End users only dispose of the result of a complex image processing chain effectuated by uncontrolled hardware and software (and sometimes by chemical means). For such images, we have shown in chapter 8 that recent progress in noise estimation permits to estimate from a single image a noise model which is simultaneously signal and frequency dependent.
As most of denoising algorithms mainly focus on Gaussian Noise (chapter 4) and then only work for signal-independent noise, they need to be adapted to be able to deal with signal-dependent noise. The NL-Bayes algorithm described in details in chapter 12 has both advantages to give really good results without providing any artefacts and to be simple enough to provide signal-dependent denoising with only small modifications.
Therefore we propose in this chapter a multiscale denoising algorithm -based on NL-Bayes- adapted to this broad noise model. This leads to a blind denoising algorithm which we demonstrate on real JPEG images and on scans of old photographs for which the formation model is unknown. The consistency of this algorithm is also verified on simulated distorted images. This algorithm is finally compared to the unique state of the art blind denoising method.
This chapter uses the results of joint work with Miguel Colom and Jean-Michel Morel.

9.1 Introduction

9.1.1 Motivations

Blind denoising is the conjunction of a thorough noise estimation method followed by the application of an adapted denoising method. To cope with the broad variety of observed noises imaging, the noise model must be far more comprehensive than the usual white Gaussian noise. Our lead example will be JPEG images from digital CCD or CMOS cameras, where the initial signal dependent white Poisson noise has undergone nonlinear transforms, linear filters and a quantization of its DCT coefficients. After such alterations, a signal, frequency and scale dependency is a minimal assumption for the remaining noise. This requires dealing with a noise model depending on hundreds of parameters, in contrast with the usual one-parameter Gaussian white noise and the two-parameter Poisson noise. A flexible denoising method must also be conceived to cope with this signal, scale and frequency dependent noise model.
To be useful to all image users, who generally have only access to the end result of a complex processing chain, blind denoising must be able to cope with both raw and preprocessed images of all sorts. The archives of the online executions at the IPOL journal of six classic denoising methods, namely DCT denoising [139], TV denoising [59], K-SVD [80], NL-means [15], BM3D [74] and NL-Bayes [76] are replete with such puzzling noisy images. IPOL users are in principle

requested to upload noiseless images, to which the noise is added on line to test the performance of each algorithm. Yet, as one can observe in this public archive, the demand for a blind denoiser is so strong that more than 10000 noisy images have been unduly uploaded. This shows how necessary "blind" methods are, for diffusing image processing techniques in science and technology.

9.1.2 Antecedents

We found only a few references on blind denoising approaches: Portilla [113], [112], Rabie [116] and Liu, Freeman, Szeliski and Kang [87]. Portilla's method is an adaptation of the famous BLS-GSM algorithm, which models wavelet patches at each scale by a Gaussian scale mixture (GSM), followed by a Bayesian least square (BLS) estimation for wavelet patches. This method is in principle adapted to homogeneous, Gaussian or mesokurtotic noise. Yet, according to the author, the GSM model provides an automatic way to separate noise from signal. Indeed, for natural images, a GSM captures for the wavelet coefficients both high kurtosis marginals and a positive covariance between neighbor coefficient amplitudes. These coefficients are not shared by Gaussian or lower kurtosis noise sources. Then, for each wavelet subband a correlated Gaussian model can be used to estimate the noise and a correlated GSM is used for the signal. This algorithm is fully automatic, and will be compared to our results in Section 9.6.3. Our proposed solution shares many features with Portilla's method. Our noise model is nonetheless more general, being signal dependent, and our patch model is local, while the GSM wavelet patch model is global. (A recent local version of BLS-GSM [117] obtains a better performance than BLS-GSM.)
Liu, Freeman, Szeliski and Kang [87] proposed a unified denoising framework for JPEG images with two tasks in view: 1) automatic estimation and 2) removal of colored noise from a single image. These steps are performed by involving a piecewise smooth image model and a segmentation. The authors introduce the so called "noise level functions" (NLF) to estimate the noise level as a function of the image grey level. The obtained noise curve by their algorithm is an estimate of an upper bound of the real NLF, done by fitting a lower envelope to the standard deviations of per-segment image variances. In their denoising procedure, the chrominance of the colored noise is significantly removed by projecting pixel values onto a line fitted to the RGB values in each segment. Then, a Gaussian conditional random field is constructed to obtain the underlying clean image from the noisy input. Unfortunately no code is available for this complex procedure.
The method proposed by Rabie [116] seems less effective and works only for Gaussian noise. Here the blind denoising filter is based on the theory of robust statistics. The denoising part is done by minimizing a stationary cost function. For an adaptive window around the pixel of interest, noise pixels are seen as outlier pixels and rejected according to the Lorentzian robust estimator. The noise is basically estimated over a flat area of the noisy image. "Optimal-size" adaptive window are used to obtain the largest area containing relatively uniform structures around each pixel of interest. The uniformity is based on local signal variance estimate. This method seems less general than Portilla's method, since it can only deal with a signal-independent Gaussian noise. Observing the results shown in [116], indicates that this method mainly works on images with large homogeneous areas. An entropy-based noise level estimator has been proposed in [57], which may work for any sort of noise. Unfortunately it delivers a noise level but not a noise model. So we could not use it for noise estimation. Our denoising method will be based on a noise signal and frequency noise estimator as explained in chapter 8 and proposed by Colom et al. [32], relying on a Ponomarenko et al. general principle [109] to build a noise patch model. This method is proved in the aforementioned reference to estimate accurately the variances of DCT coefficients of noise patches in a JPEG image. We shall see that it can be easily extended to cope with a scale dependency.
Section 9.2 gives a brief account of the original NL-Bayes algorithm and details why and how it can be adapted to the current general noise framework. Section 9.3 gives the noise model and details the computation of the noise covariance matrix at each scale. Section 9.4 gives the multiscale denoising procedure and details the up-and down-sampling operations. Section 9.5 validates the method on simulated noisy JPEG images and filtered images. This section ends with a final synthetic description of the whole blind denoising method. Section 9.6 is the experimental

section, with experiments on real noisy images with unknown history. A thorough comparison is also performed with the reproducible method [113].

9.2 A Generalized Nonlocal Bayesian Algorithm

Most denoising methods in the literature focus on Gaussian white noise, which is a reasonable simplification of the problem, since for example Poisson noise can be transformed into approximately white Gaussian noise by the Anscombe transform [3]. In this section we show that one of them, the NL-Bayes method, designed for Gaussian white noise, can be extended to deal with a signal, scale and frequency dependent noise. NL-Bayes only requires the knowledge of a local Gaussian patch model and of a Gaussian noise model. It is therefore possible to extend the noise model to make obtain a denoising method compatible with a scale and signal dependent.
Like other patch based denoising methods, NL-Bayes denoises all noisy square patches extracted from the noisy image $\tilde{u}$ and then obtains the final denoised image $\hat{u}$ by replacing every image pixel value by an average of the denoised values obtained for this pixel in all denoised patches containing it. We shall denote by $\tilde{P}$ a reference patch extracted from the image, and by $\mathcal{P}(\tilde{P})$ a set of patches $\tilde{Q}$ similar to the reference patch $\tilde{P}$. Assuming that $\tilde{Q}$ follows a Gaussian model, a first basic estimation of the denoised patch P can be obtained (see chapter 12 and [75]) by

$$P^{\mathbf{basic}} = \overline{\tilde{P}} + [\mathbf{C}_{\tilde{P}} - \mathbf{C}_n]\, \mathbf{C}_{\tilde{P}}^{-1} \left(\tilde{P} - \overline{\tilde{P}}\right) \tag{9.1}$$

where

- $\overline{\tilde{P}}$ is the empirical average of the patches similar to $\tilde{P}$:

$$\overline{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \tilde{Q} \tag{9.2}$$

- $\mathbf{C}_n$ is the covariance matrix of the noise;
- $\mathbf{C}_{\tilde{P}}$ is the empirical covariance matrix of the patches similar to $\tilde{P}$, which may be obtained by

$$\mathbf{C}_{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P}) - 1} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \left(\tilde{Q} - \overline{\tilde{P}}\right) \left(\tilde{Q} - \overline{\tilde{P}}\right)^t. \tag{9.3}$$

For pure Gaussian signal-independent noise with variance σ^2, we simply have $\mathbf{C}_n = \sigma^2 \mathbf{I}$. The above estimate would the optimal Bayesian estimate, if $\mathbf{C}_{\tilde{P}}$ and $\overline{\tilde{P}}$ were the true covariance matrix and expectation of the patches similar to $\tilde{P}$. In a second step, all the denoised patches obtained after the previous first step estimation can be reused by a classic Wiener argument to obtain a better unbiased estimation $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ for the covariance of the 3D group containing P. Similarly, a new estimation $\overline{\tilde{P}}^{\mathbf{basic}}$ of the average of patches similar to P can be obtained. This leads to a second Wiener-Bayes estimate

$$P^{\mathbf{final}} = \overline{\tilde{P}}^{\mathbf{basic}} + \mathbf{C}_{\tilde{P}}^{\mathbf{basic}} \left[\mathbf{C}_{\tilde{P}}^{\mathbf{basic}} - \mathbf{C}_n\right]^{-1} \left(\tilde{P} - \overline{\tilde{P}}^{\mathbf{basic}}\right). \tag{9.4}$$

Adaptation to Signal-Dependent Noise As formulas (9.1) and (9.4) show, the above Bayesian principle is compatible with a patch noise model $\mathbf{C}_n$ depending on each patch $\tilde{P}$. The above formulas only require a good estimate of the covariance matrix of the noise associated with each group of similar patches. The algorithm computing this matrix is given in Section 9.3. The noise model being signal dependent, for each intensity $\mathbf{i}$ in the range intensity $[0, 255]$ of the image a noise covariance matrix $\mathbf{C}_{n\mathbf{i}}$ will be available. The noise model for each group of patches similar to $\tilde{P}$ will depend on $\tilde{P}$ through their mean $\mathbf{i}$. The reference intensity for the current 3D group

$\mathcal{P}(\tilde{P})$ must therefore be estimated to apply formulas (9.1) and (9.4) with the appropriate noise covariance matrix. This intensity is simply estimated as the average of all pixels contained in $\mathcal{P}(\tilde{P})$.

Local Correction of the Covariance Matrix The denoising performance strongly depends on the noise covariance matrices estimation. If the matrices $\{\mathbf{C}_{n\mathbf{i}}\}_{\mathbf{i}\in[0,255]}$ are not accurate enough, denoising can cause ugly artifacts, particularly in the first step. The noise estimation procedure from the image is always at risk of an overestimation, particularly when the image is small or when it contains a uniform texture which becomes undistinguishable from colored noise. If $\mathbf{C}_n$ is overestimated, then (9.1) risks adding "negative noise" to the image, because of the $-\mathbf{C}_n$ term in this equation. Thus, a conservative estimation strategy must be applied on the first Bayesian step to avoid noise overestimation artifacts. This strategy ensures that the noise variances are always smaller than the noisy patch variances. This sanity check based on the diagonal values of both $\mathbf{C}_{\tilde{P}}$ and $\mathbf{C}_n$ covariance matrices leads to the following more conservative estimate of the diagonal elements of the patch covariance matrix used in (9.1):

$$\forall p \in [\![0, \kappa^2 - 1]\!], \mathbf{C}_{\tilde{P}}(p,p) = \max\left(\mathbf{C}_{\tilde{P}}(p,p), \mathbf{C}_n(p,p)\right). \tag{9.5}$$

Homogeneous Area Detection The original NL-Bayes algorithm presented in chapter 12 and also in [75] has a statistical test to determine if a 3D group belongs to a homogeneous area, and in this case the estimation of all patches is replaced by the global mean over all pixels contained in the 3D group. This criterion is merely based on the comparison of the empirical standard deviation of all pixels of $\mathcal{P}(\tilde{P})$ with σ^2.
In our generalization of this algorithm, σ doesn't exist since $\mathbf{C}_n \neq \sigma^2\mathbf{I}$. So this criterion must be adapted to better take into account $\mathbf{C}_n$ in the following way:

- First, compute the difference of the traces of both covariance matrices for each channel c,

$$\delta_c = \mathrm{Tr}(\mathbf{C}_{\tilde{P}}) - \mathrm{Tr}(\mathbf{C}_n). \tag{9.6}$$

- Denote by $\hat{\tilde{Q}}$ a first estimation of $\tilde{Q}$ obtained by (9.1). Then the basic estimate is $\forall \tilde{Q} \in \mathcal{P}(\tilde{P})$,

$$Q^{\mathbf{basic}} = \begin{cases} \overline{\overline{\tilde{P}}} & \text{if } \delta_c < \alpha \mathrm{Tr}(\mathbf{C}_n) \\ \hat{\tilde{Q}} & \text{if } \delta_c > \beta \mathrm{Tr}(\mathbf{C}_n) \\ t\hat{\tilde{Q}} + (1-t)\overline{\overline{\tilde{P}}} & \text{otherwise.} \end{cases} \tag{9.7}$$

  where

$$t = \frac{\delta_c - \alpha \mathrm{Tr}(\mathbf{C}_n)}{\beta \mathrm{Tr}(\mathbf{C}_n) - \alpha \mathrm{Tr}(\mathbf{C}_n)}$$

  and

$$\overline{\overline{\tilde{P}}} = \frac{1}{\#\mathcal{P}(\tilde{P})\kappa^2} \sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})} \sum_{p=1}^{\kappa} \sum_{q=1}^{\kappa} \tilde{Q}(p,q)$$

The thresholds (α, β) are chosen equal to $\left(-\frac{1}{3}, \frac{1}{3}\right)$. This (optional) correction which generally increases the PSNR is only used for the first step of the finest scale of the multiscale algorithm.

9.3 Obtaining the Covariance Matrix of Noise Patches

As seen in chapter 8 and also [32], an adaptation of the Ponomarenko et al. [109] method estimating a frequency dependent noise is proposed to estimate noise in JPEG images. Given a patch size $\kappa\times\kappa$, the method extracts from the image a set with fixed cardinality of sample blocks with minimal

variance, which are therefore likely to contain only noise. These noise blocks are transformed by a DCT, and an empirical standard deviation of their DCT coefficients is computed. This gives a noise model that is proved in chapter 8 and also in [32] to be accurately consistent with noise observed in JPEG images. This algorithm computes for every intensity $\mathbf{i}$ with a multi-frequency noise estimate given by a $\kappa^2 \times \kappa^2$ matrix

$$\mathbf{M_i} := \mathbb{E}\left(\mathcal{D}N_{\mathbf{i}} \left(\mathcal{D}N_{\mathbf{i}}\right)^t\right) \tag{9.8}$$

where:

- $\mathcal{D}$ is the $\kappa^2 \times \kappa^2$ matrix of the discrete cosine transform (DCT) ;
- $N_{\mathbf{i}}$ denotes the $\kappa \times \kappa$ stochastic noise patch model at intensity $\mathbf{i}$.

9.3.1 Are Noise Covariances Negligible in the Block DCT Space?

The method of the preceding section only estimates the variances of the DCT coefficients of noise blocks and not their covariances. The covariance matrices are therefore assumed to be diagonal, which amounts to assume that the DCT decorrelates the noise. A formal argument can be given in favour of this assumption. Assume that the initial image noise was white Gaussian, and that the image has undergone a symmetric, real, periodic linear filter H. Then this filter corresponds to applying a diagonal operator to the image in the DCT frequency domain. Thus the noise covariance of the filtered noise remains diagonal in the DCT domain. Yet, this argument is only valid for a *global* image DCT. Here, because we need a signal dependent noise model, we are estimating it on *local* DCTs applied to each block. It is therefore no more true that the blocks have undergone a periodic convolution filter. Thus, it cannot be *exactly* true that after the application of a global linear filter, the noise block DCTs have a diagonal covariance. To check nonetheless the quantitative validity of this assumption, we tested three different filters applied to a white noise:

- $\mathbf{H}_1$ with coefficients $\frac{1}{4}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ supported by the pixels $(-\frac{1}{2}, -\frac{1}{2})$, $(\frac{1}{2}, \frac{1}{2})$, $(\frac{1}{2}, -\frac{1}{2})$, $(-\frac{1}{2}, \frac{1}{2})$;
- $\mathbf{H}_2$ the centered filter with coefficients $\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$;
- $\mathbf{H}_3$ the centered filter with coefficients $\frac{1}{88}\begin{pmatrix} 1 & 2 & 4 & 8 & 4 & 2 & 1 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 1 & 2 & 4 & 8 & 4 & 2 & 1 \end{pmatrix}$.

The noise image $\tilde{u}$ was a 256×256 Gaussian white noise with mean 128, and standard deviation $\sigma = 20$. After convolution, we extracted N distinct $\kappa \times \kappa$ patches $\{P_n\}_{n \in N}$ from the image and a 2D normalized DCT was applied on them. Finally, their empirical $\kappa^2 \times \kappa^2$ covariance matrix $\mathbf{C}$ was computed as

$$\begin{aligned} &\forall (p,q),(i,j) \in [\![0, \kappa-1]\!]^2, \\ &\mathbf{C}(p,q,i,j) = \frac{1}{N}\sum_{n=1}^{N} \hat{P}(p,q)\hat{P}(i,j) \\ &-\frac{1}{N^2}\left(\sum_{n=1}^{N} \hat{P}(p,q)\right)\left(\sum_{n=1}^{N} \hat{P}(i,j)\right) \end{aligned} \tag{9.9}$$

These covariances matrices can be visualized by the absolute value of their coefficients $|C_{i,j}|$, normalized in $[0,1]$ so that the largest coefficient is set equal to 1, and the smallest equal to 0. The following colour code is used in the visualization: a coefficient appears in blue if it is near 0; in green if it is near 0.5 and in red if it is near 1. The results for various patch sizes are shown in Figure 9.1. This illustration and the quantitative tables 9.1, 9.2 and 9.3 confirm that the block DCT noise covariance matrices are nearly diagonal.
So from now one, only variance coefficients will be considered in DCT space.

κ	4	6	8	16
mean $\{\|C_{i,j}\|\}_{i\neq j}$	0.83	0.48	0.31	0.10
mean $\{\|C_{i,j}\|\}_{i=j}$	24.89	25.31	24.95	24.73
median $\{\|C_{i,j}\|\}_{i\neq j}$	0.04	0.03	0.02	0.01
median $\{\|C_{i,j}\|\}_{i=j}$	19.42	17.14	16.28	14.41

Table 9.1: Statistics of the estimated DCT covariance matrix of noise filtered by $\mathbf{H}_1$.

κ	4	6	8	16
mean $\{\|C_{i,j}\|\}_{i\neq j}$	0.48	0.28	0.19	0.06
mean $\{\|C_{i,j}\|\}_{i=j}$	14.59	13.95	14.45	14.23
median $\{\|C_{i,j}\|\}_{i\neq j}$	0.010	0.008	0.005	0.002
median $\{\|C_{i,j}\|\}_{i=j}$	6.75	4.50	3.77	2.35

Table 9.2: Statistics of the estimated DCT covariance matrix of noise filtered by $\mathbf{H}_2$.

κ	4	6	8	16
mean $\{\|C_{i,j}\|\}_{i\neq j}$	0.22	0.15	0.10	0.04
mean $\{\|C_{i,j}\|\}_{i=j}$	9.28	8.94	9.04	8.51
median $\{\|C_{i,j}\|\}_{i\neq j}$	0.020	0.016	0.010	0.003
median $\{\|C_{i,j}\|\}_{i=j}$	3.32	2.86	2.44	1.73

Table 9.3: Statistics of the estimated DCT covariance matrix of noise filtered by $\mathbf{H}_3$.

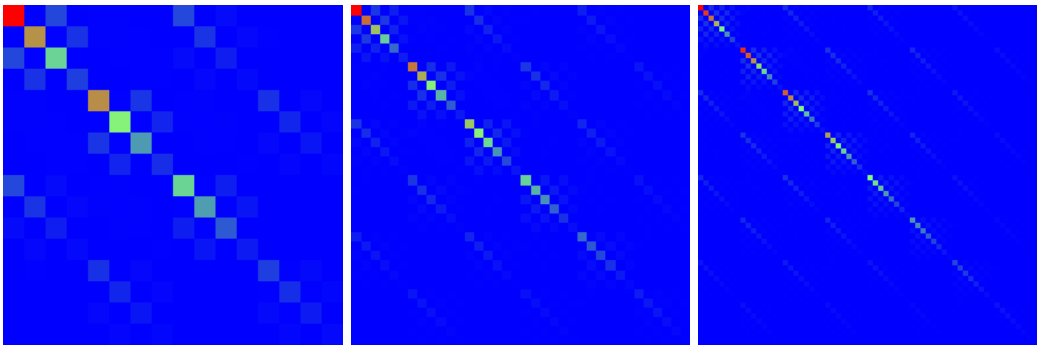


Figure 9.1: Visualization of the noise covariance matrices in DCT space after applying the filter $\mathbf{H}_1$ to illustrate that it is almost diagonal. From left to right, top to bottom patch size $\kappa = 4, 6, 8$.

9.3.2 Covariance Matrix Filtering

Since the noise covariance matrices can only be estimated for sparse bins in the intensity range, an interpolation must be applied to obtain a noise covariance matrix of the noise for each given intensity. The covariance matrices must be smoothed before such an interpolation. This can be obtained by a regularization of the covariance matrices in DCT space before applying the inverse DCT to get back a covariance matrix in the image domain. We found that a robust regularization could be performed in the following two steps:

1. For each frequency independently, perform a linear interpolation between the bin values to obtain a noise curve for this frequency, giving the variance as a function of the signal $\mathbf{i}$. Smooth this curve by applying a sliding average;

2. For every bin, replace each matrix coefficient by the median of its four neighbours and itself.

Since the filtering is channel independent, the pseudo-code only describes the filtering for one channel.

Getting Back to the Space Domain For a given intensity $\mathbf{i}$, the covariance matrix of the noise is by definition

$$Cov(N_\mathbf{i}) = \mathbb{E}\left(N_\mathbf{i} N_\mathbf{i}^t\right)$$

which leads to

$$\begin{aligned} \mathcal{D}Cov(N_\mathbf{i})\mathcal{D}^t &= \mathcal{D}\mathbb{E}\left(N_\mathbf{i} N_\mathbf{i}^t\right)\mathcal{D}^t \\ &= \mathbb{E}\left(\mathcal{D}N_\mathbf{i} N_\mathbf{i}^t \mathcal{D}^t\right) \\ &= \mathbb{E}\left(\mathcal{D}N_\mathbf{i} \left(\mathcal{D}N_\mathbf{i}\right)^t\right) \\ &= \mathbf{M_i} \end{aligned} \tag{9.10}$$

thanks to equation (9.8). Since $\mathcal{D}^{-1} = \mathcal{D}^t$, then from equation (9.10) we get

$$cov(N_\mathbf{i}) = \mathcal{D}^t \mathbf{M_i} \mathcal{D}. \tag{9.11}$$

9.4 The Multiscale Algorithm

9.4.1 Why a multiscale algorithm?

Classic denoising algorithms such as BM3D (Dabov et al. [35]), NL-means (Buades et al. [13]), K-SVD (Mairal et al. [93], [94]), Wiener filters applied on DCT (Yaroslavsky et al. [138], [137]) or on wavelet transform (Donoho et al. [126]) and the total variation minimization (Rudin et al. [122]) achieve good results for moderate noise ($\sigma \leq 20$). Yet for larger noise artifacts inherent to each method (and different for each method) start appearing. In particular all keep an often disturbing low frequency noise. A natural idea to deal with low frequency noise is to involve a coarse to fine multiscale procedure, which promises three improvements:

1. in the patch-based methods, it favors a better patch comparison, because the patch low frequencies are denoised *before* grouping them by similarity for denoising their higher frequencies;

2. at coarse scales the noise decreases by zoom out, and state-of-the-art algorithms work better;

3. subsampling the image before denoising amounts to enlarge the size of the neighborhood on which the denoising is performed, thus permitting to grab and remove low frequency noise on larger regions.

A still stronger argument in favour of a multiscale procedure is that in most images submitted by users, the main bulk of the noise is contained in the low frequencies. This is explainable by several factors. In accurately scanned old photographs, the chemical noise is over-sampled and its

Figure 9.2: A multiscale process is required to remove the low frequency noise. This is particularly apparent in the flat image regions. From left to right: Noisy image ($\sigma = 30$), result of the "Classic NL-Bayes", result of the multiscale (three scales) NL-Bayes.

grain has low frequency components. In JPEG images, compression has strongly attenuated high frequency noise components, but the low frequency components after the third octave are intact.

To define a coarse to fine multiscale structure, we proceed by a classic oversampled wavelet denoising strategy [27]. The image is convolved by a Haar "mother wavelet", which is nothing but a box-filter $\mathbf{F}$ where each lower scale pixel is the mean of four samples in the higher scale. This cumulates the advantage of dividing the noise standard deviation by two and of maintaining the independence of the samples after down-sampling. By this process a white noise remains white after subsampling. A classic objection to this wavelet method is that the sub-sampled image is aliased and cannot be up-sampled after denoising. The classic wavelet method avoid this obstacle by denoising simultaneously the three wavelet components obtained by convolving the image with the three Haar wavelets, before reconstructing the finer scale. Yet when dealing with patch based methods, it is better to keep all frequency components together to perform a better nonlocal patch comparison. For this reason the proposed multiscale algorithm keeps and processes four channels that are partly redundant. The four channels are obtained by moving the sub-sampling grid by respectively $(0,0)$, $(1,0)$, $(0,1)$, $(1,1)$. In that way there is enough information for up-sampling after denoising the denoised images at the lower scale.

The above method is multiscale but does not take advantage of the sub-sampling in the lower scales to increase the algorithm speed. A normal multiscale algorithm is only $1 + \frac{1}{4} + \frac{1}{16} + \cdots = \frac{4}{3}$ more complex than the single scale algorithm. Instead a multiscale algorithm keeping all sub-images when sub-sampling will be twice to five times slower, depending on the number of scales involved, (by default two). Yet, the redundancy of this denoising at lower scales notably increases the restoration quality. This is particularly important, as *any denoising error on a down-sampled image is amplified by a four-factor after upsampling.*

9.4.2 The Mean Sub-Sampling Method

We shall denote by s the current dyadic scale of the multiscale algorithm. For the particular case of white noise, the aim of the sub-sampling is to obtain from $\tilde{u}_s$ an image $\tilde{u}_{s+1}$ where the standard deviation of the noise has been divided by two compared to the noise contained in $\tilde{u}_s$. To get this result, one can use a filter $f(i,j)$ satisfying

$$\sum_{i,j} f(i,j) = 1 \quad \text{and} \quad \sum_{i,j} f(i,j)^2 = \frac{1}{4}.$$

The simplest filter coping with these conditions is the average filter $\mathbf{F}$, defined by

$$\mathbf{F}(i,j) = \begin{cases} \frac{1}{4} & \text{if } (i,j) \in [(0,0),(0,1),(1,0),(1,1)]\,, \\ 0 & \text{otherwise.} \end{cases}$$

which averages each group of four adjacent neighboring pixels. There are four different filter+sub-sample results, as shown in figure 9.3. Moreover if the image $\tilde{u}_s$ is well-sampled, so is $\tilde{u}_s * \mathbf{F}$. Thus, the difference image is not aliased. Since all sub-sampled images are available, the noise

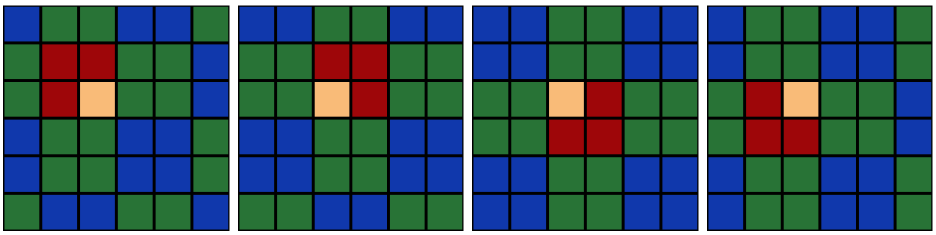


Figure 9.3: Four different ways to average red neighbors of the yellow reference pixel.

estimation can work with the same amount of samples at every scale, which favours a good precision on the noise estimation at lower scales. All sub-sampled images must also be denoised. To avoid handling them separately, we introduce here a new procedure to process them jointly in a single image, while avoiding creating artificial borders. The four sub-sampled images are regrouped in one mosaic image, as shown in figure 9.4. The boundaries of the sub-images are in that way better denoised, because they are included in a smooth larger image.

Figure 9.4: Left: mosaic of the scale 1 sub-images. Right: mosaic of the scale 2 sub-images, The input image has scale 0.

9.4.3 The Mean Up-Sampling Method

The aim of the up-sampling is to go back to the upper scale, after denoising the four sub-images obtained by sub-sampling as seen in Section 9.4.2. The four sub-images $\tilde{u}_1$, $\tilde{u}_2$, $\tilde{u}_3$ and $\tilde{u}_4$ have their pixel center (resp. in red, purple, green and blue in figure 9.5) located at the center of four pixels of $\tilde{u}$ (in black in figure 9.5). Thus they are shifted by $\pm\frac{1}{2}$ in both coordinate directions.

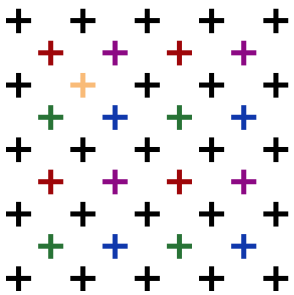


Figure 9.5: Position of the center of pixels in the original image $\tilde{u}$ in black, in the four sub-images $\tilde{u}_1$ in red, $\tilde{u}_2$ in purple, $\tilde{u}_3$ in green and $\tilde{u}_4$ in blue. The yellow pixel will be reconstructed by averaging the top left red pixel, the top right purple pixel, the bottom left green pixel and the bottom right blue pixel of its four pixel neighborhood.

The reconstruction of the pixels of $\tilde{u}$ (see the example of the pixel in yellow in figure 9.5) will be done by averaging their four neighbors, each one belonging to each sub-image.
Fig. 9.6 illustrates a multiscale denoising result, where it is apparent that noise remains mainly at lower scales.

9.4.4 Noise Estimation

If the input noisy image had pure Gaussian noise, then after each sub-sampling the noise should be divided by two and remain white. For raw images it is the case, since (almost) no alteration nor transformations are applied to the original noisy pixels. Then the noise is a Poisson random process, which can be approximated by a signal-dependent Gaussian noise.
However, the proposed algorithm must deal with all kinds of noisy images. A large majority of them are JPEG images where JPEG has killed most high-frequency coefficients. In such images the noise increases at lower scales, as illustrated in Figure 9.7, which are the noise curves of the image shown in Figure 9.20. This figure displays average noise curves for high and low frequencies respectively, in the three scales noise estimation from a JPEG image. The low-frequency noise is not altered by JPEG and becomes a high-frequency noise after three[1] subsampling operations.
In our redundant noise estimation, the noise covariance matrices are estimated at each dyadic scale. Section 9.4.2 explains how the noise estimation is applied on the mosaic image composed of all sub-images. Then for every scale the same number of samples is available, which allows the noise estimation to retain a decent accuracy even at coarse scales. At each given scale, all sub-images of the mosaic are denoised with the same set of noise covariance matrices.

9.5 Validation

Blind denoising is designed mainly for images where the image history is unknown and no ground truth available. But we can test the denoising performance of the Noise Clinic after simulating a whole image processing chain on a Poisson noisy image for which the ground truth is available. One of the worst possible noise distortion is provided by the image processing chain applied in the camera hardware and generally ending with JPEG compression. This chain includes nonlinear corrections on the raw image, followed by some denoising, demosaicking, gamma-correction, white balance and JPEG compression, namely the quantization of local block DCT coefficients. To see to which extent the method works, we started with perturbations consistent with our noise model and then simulated a typical camera image processing chain ending with JPEG compression. We

[1] JPEG transform is based on 8×8 DCT transform, then after three subsampling, when 8×8 pixels squares becomes a single pixel. Then at the third scale, the noise is only high-frequency and independent.

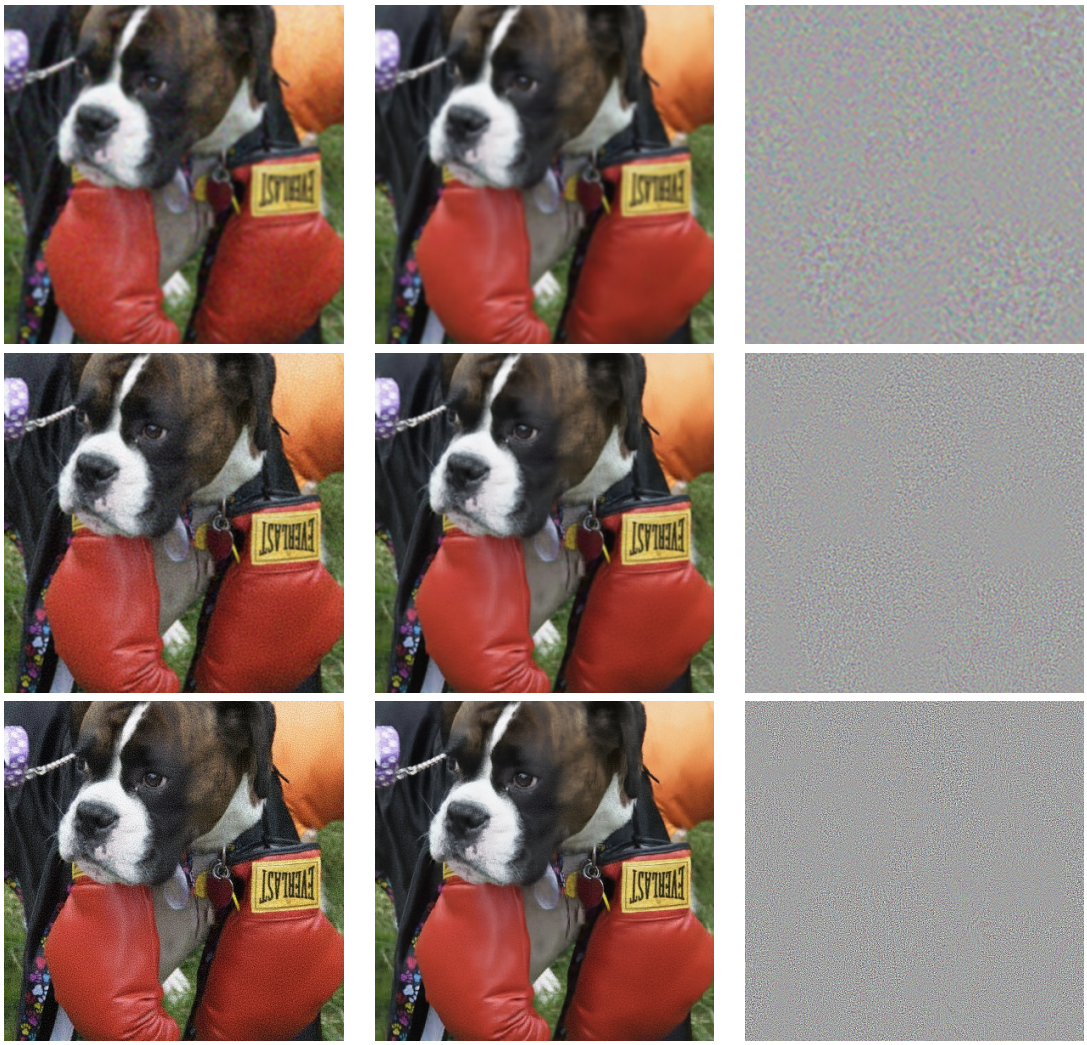


Figure 9.6: Result of the Noise Clinic at each scale. From top to bottom: scale 2, scale 1 and scale 0. From left to right: noisy image, denoised image, difference image. The noise in a JPEG image is mainly present in low scales.

first obtained a noise-free raw image u_{raw} by subsampling a high quality outdoor image. Then a Poisson noisy $\tilde{u}_{raw}$ was simulated from it. Four validation experiments were performed.
First, we computed a reference denoised version of the image:

- the Noise Clinic was directly applied on $\tilde{u}_{raw}$ to get $\hat{u}_{raw}$;
- a white balance and a gamma correction were applied on u_{raw}, $\tilde{u}_{raw}$ and $\hat{u}_{raw}$ to get u_{rgb}, $\tilde{u}_{rgb}$ and $\hat{u}_{rgb}$.

Those images will be used as reference, to see how other parts of the image processing chain (such as the demosaicking and the JPEG compression) impact the result of the denoising. Table 9.4 shows RMSEs between the noisy and denoised images and the reference one. One can also remark that the best result of the denoising (both in term of RMSEs and visual aspects) is obtained when the Noise Clinic is applied directly before any transformation.
Second, a demosaicking algorithm was added to the image processing chain before calling the denoising part:

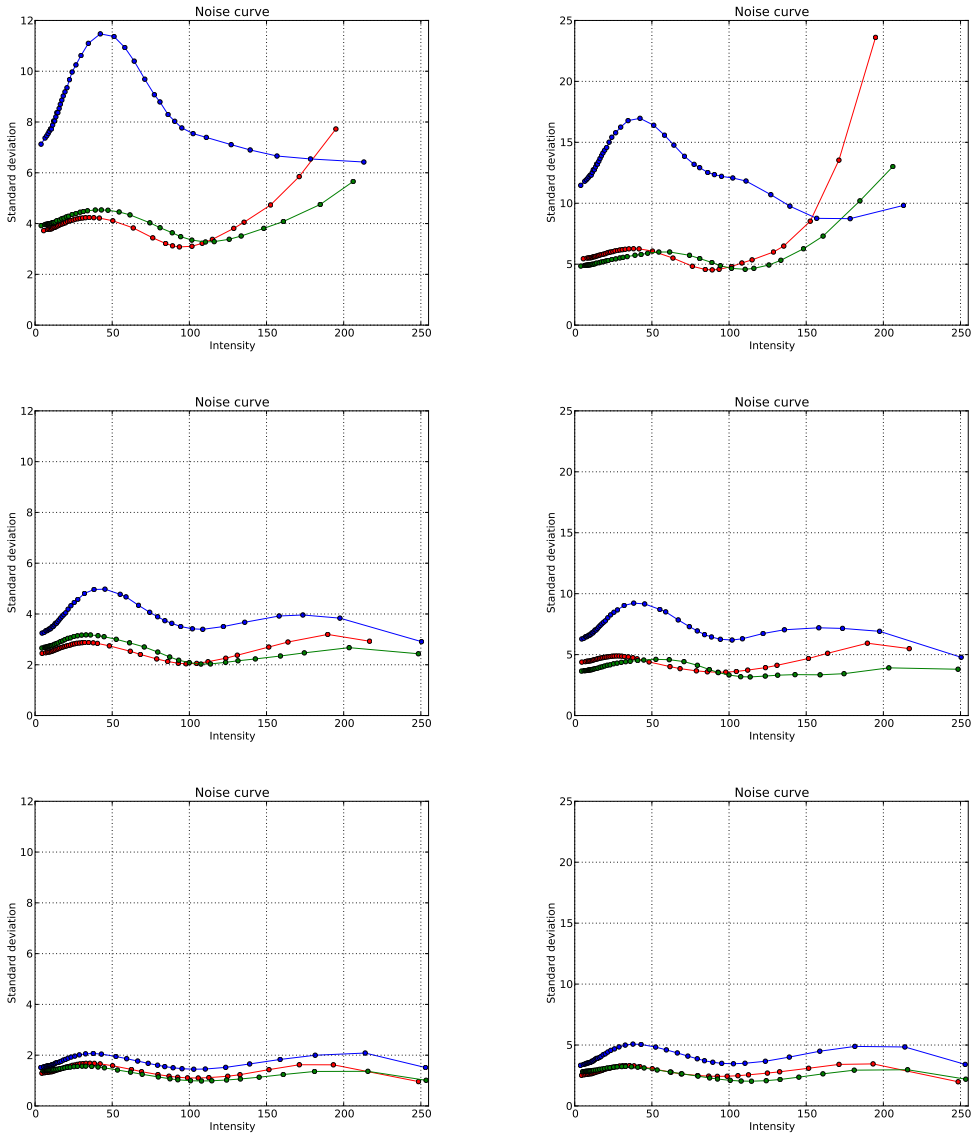


Figure 9.7: Average noise curves for a typical JPEG-encoded image (shown in Figure 9.20). From left to right: low frequencies, high frequencies. From top to bottom: scale 2, scale 1, scale 0. Instead of being divided by two at each scale (as it should happen with white noise), the noise grows in lower scales, where JPEG has not removed it.

$\tilde{u}_{rgb}$	$\hat{u}_{rgb}\|^{s2}$	$\hat{u}_{rgb}\|^{s3}$	$\hat{v}_{rgb}$
8.62	3.63	3.65	6.46

Table 9.4: RMSE between noisy/denoised images and corresponding reference image (u_{raw}) when two and three scales are used. $\hat{v}_{rgb}$ denotes the result of Blind BLS-GSM for this experiment.

$\tilde{u}_d$	$\hat{u}_d\|^{s2}$	$\hat{u}_d\|^{s3}$	$\hat{v}_d$
8.64	4.84	4.84	6.43

Table 9.5: RMSE between noisy/denoised images and corresponding reference image (u_d) when two and three scales are used for the demosaicking experiment. $\hat{v}_d$ denotes the result of Blind BLS-GSM for this experiment.

$\tilde{u}_{jpeg}$	$\hat{u}_{jpeg}\|^{s2}$	$\hat{u}_{jpeg}\|^{s3}$	$\hat{v}_{jpeg}$
8.70	5.34	5.53	6.30

Table 9.6: RMSE between noisy/denoised images and corresponding reference image (u_{jpeg}) for the *JPEG* experiment, with compression quality of 92. $\hat{v}_{jpeg}$ denotes the result of Blind BLS-GSM for this experiment.

- extract the mosaic[2] of the noise-free image: $u_m = Mosaic(u_{raw})$;
- do the same for the noisy image: $\tilde{u}_m = Mosaic(\tilde{u}_{raw})$;
- apply a classic demosaicking method[3] on both images, followed by a white balance and a gamma correction to get u_d and $\tilde{u}_d$;
- finally apply the Noise Clinic on $\tilde{u}_d$ to get $\hat{u}_d$.

Table 9.5 shows RMSEs for this experiment. One may notice that after a demosaicking the noise is no more white, and some structures appears in the noise. These structures are preserved and sometimes enhanced by the denoising algorithm, since it is seen as structure and not as noise. This explains why RMSEs are less favourable than when the denoising is directly applied on the raw images.
Third, a complete image processing chain was simulated to obtain a final JPEG compressed image:

- apply a JPEG compression of quality 92 over both u_d and $\tilde{u}_d$ to get u_{jpeg} and $\tilde{u}_{jpeg}$;
- apply the Noise Clinic to get $\hat{u}_{jpeg}$.

Table 9.6 shows RMSEs for this experiment. Of course, as JPEG compression creates more artifacts and structured noise, results are worse than with the first two experiments. This only means that the denoising should be applied as soon as possible in the whole image processing chain. However, results are not very far from the ideal case, which confirms the interest and the strength of the Noise Clinic.
Fourth, the filter $\mathbf{H}_2$ seen in section 9.3.1 was used to get:

- a reference filtered image: $u_f = \mathbf{H}_2 * u_{raw}$;
- a noisy filtered image: $\tilde{u}_f = \mathbf{H}_2 * \tilde{u}_{raw}$;
- the result of the Noise Clinic of the noisy filtered image: $\hat{u}_{f_1} = NC(\mathbf{H}_2 * \tilde{u}_{raw})$;
- the filtered result of the Noise Clinic of the noisy image: $\hat{u}_{f_2} = \mathbf{H}_2 * NC(\tilde{u}_{raw})$.

Table 9.7 shows RMSEs associated to this experiment. Of course after this filtering, there only remains low frequency noise, which explains why RMSEs values are better than in the ideal case. However, the Noise Clinic is still able to give good results.

[2]The mosaic image is obtain by keeping only the bayer (R Gr Gb B) over a group of four pixels instead of all RGB values.
[3]The demosaicking algorithm used in this experiment was Self-similarity Driven Demosaicking algorithm [17], available on IPOL.

$\tilde{u}_f$	$\hat{u}_{f_1}\|^{s2}$	$\hat{u}_{f_1}\|^{s3}$	$\hat{u}_{f_2}\|^{s2}$	$\hat{u}_{f_2}\|^{s3}$	$\hat{v}_{f_1}$	$\hat{v}f_2$
1.58	0.75	0.82	0.75	0.75	1.24	1.28

Table 9.7: RMSE between noisy/denoised images and corresponding reference image (u_f) for the *filtered* experiment. $\hat{v}_{f_1}$ and $\hat{v}_{f_2}$ denote results of Blind BLS-GSM for this experiment.

Figure 9.8 (resp. 9.9 and 9.10) shows results associated of the raw experiment (resp. demosaicking and JPEG).
Figure 9.11 (resp. 9.12 and 9.13) shows a comparison between the Noise Clinic and Blind BLS-GSM for the raw experiment (resp. demosaicking and JPEG).

9.6 Results

9.6.1 Detailed Results

In this section we applied the blind denoising to real noisy images for which no noise model was available. To illustrate the algorithm structure and its action at each scale, we present for each experiment the noisy input image and for each scale:

- the noisy image where noise has already been removed at coarser scales;
- the denoised image at this scale;
- the difference image = noisy - denoised at this scale;
- the average noise curve over high frequencies;
- the average noise curve over low frequencies.

For each scale larger than 1, the subsampled images are up-sampled to keep the original image size. Similarly, the noisy image shown at each scale is the sum of the upsampled version of the denoised sub-images of the previous scale and of the still noisy difference image kept in reserve. In other terms this image contains the remaining noise at the current scale; the noise at coarser scales has in principle already been removed. Visual results are shown in Figure 9.14 and 9.16. The corresponding noise curves are presented in Figure 9.15 and Figure 9.17.
The experiments made on JPEG photographs from unknown sources are obviously noisy but, as the noise curves illustrate, the noise is not white and is signal dependent. This is easily detected by the fact that the noise curves are not flat and that they are not divided by two from a scale to the next, as they should if the noise were white.
A typical fact of JPEG images is that the noise increases at the lower scales. This confirms the necessity of a multiscale algorithm.

9.6.2 Influence of the Number of Scales

Theoretically any number of scales could be used. Indeed at a very coarse scale the noise should be almost null and estimated as such, so that no denoising eventually would occur at very coarse scales. In practice however, some structure of the image may be confused with noise in the noise estimation step. Indeed the noise estimation method is tight on very large images on which pure noise samples in large numbers can be found [109]. After several subsamplings, the image becomes too small, and the risk of confusing texture with noise increases. In consequence applying a blind denoising on a small image is increasingly at risk of removing detail when the scale increases. Thus, it is almost always better to use a minimal number of scales, in most cases not more than two. However, we found that for some images with large low frequency noise it is sometimes better to use up to five scales. From that point of view our "blind denoising" is not fully blind and requires

Figure 9.8: Visual results of the reference (first) experiment. From top to bottom, and left to right: full noise-free image, crop of the noise-free image u_{rgb}, crop of the noisy image $\tilde{u}_{rgb}$, crop of the result of the Noise Clinic using two scales $\hat{u}_{rgb}|^{s2}$ and crop of the result of the Noise Clinic using three scales $\hat{u}_{rgb}|^{s3}$.

Figure 9.9: Visual results of the demosaicking (second) experiment. From top to bottom, and left to right: crop of the noise-free image u_d, crop of the noisy image $\tilde{u}_d$, crop of the result of the Noise Clinic using two scales $\hat{u}_d|^{s2}$ and crop of the result of the Noise Clinic using three scales $\hat{u}_d|^{s3}$.

Figure 9.10: Visual results of the JPEG (third) experiment. From top to bottom, and left to right: crop of the noise-free image u_{jpeg}, crop of the noisy image $\tilde{u}_{jpeg}$, crop of the result of the Noise Clinic using two scales $\hat{u}_{jpeg}|^{s2}$ and crop of the result of the Noise Clinic using three scales $\hat{u}_{jpeg}|^{s3}$.

Figure 9.11: Visual comparison of the reference (first) experiment. From left to right: crop of the result of the Noise Clinic using three scales $\hat{u}_{rgb}|^{s3}$ and crop of the result of the Blind BLS-GSM algorithm $\hat{v}_{rgb}$.

Figure 9.12: Visual comparison of the demosaicking (second) experiment. From left to right: crop of the result of the Noise Clinic using three scales $\hat{u}_d|^{s3}$ and crop of the result of the Blind BLS-GSM algorithm $\hat{v}_d$.

Figure 9.13: Visual comparison of the JPEG (third) experiment. From left to right: crop of the result of the Noise Clinic using three scales $\hat{u}_{jpeg}|^{s3}$ and crop of the result of the Blind BLS-GSM algorithm $\hat{v}_{jpeg}$.

an user evaluation of the number of scales involved. Nevertheless our default value is two, and works on a large majority of the images. Illustrations of the use of the "right" number of scales are presented in Figures 9.18 and 9.19 .
For the "Palace" image in Figure 9.18, five scales are needed to obtain a noise-free result because of the huge low-frequency noise. In the difference image using five scales one can see that some image structure has been included in the noise. Yet, this low frequency loss is harmless, being undetectable in the resulting denoised image.
For the "Postcard" image in Figure 9.19, one can see that although tiny details still remain when the number of scales increases, some large details such as the clouds are removed if too many scales are used. Those details are interpreted as noise by the algorithm. However, if only two scales are used, a slight low-frequency noise still remains. Despite these considerations, all results are independently visually pleasant.

Result on typical low-light JPEG image The amount of noise is directly related to the amount of light during the acquisition. Images as shown in Figure 9.20, taken in a bar with low light conditions are typically very difficult to denoise, even if we had directly access to the RAW image, due to the huge amount of noise. One can observe big coloured spots caused by the demosaicking. JPEG compression ends up creating structured noise. The big coloured spots are well attenuated by blind denoising, but the structure created by JPEG is partly left. This is easily explained. These artifacts present sharp recurrent structures which are necessarily confused with signal in an algorithm based on image self-similarity.

Results on Old Photographs Scanned old photographs form a vast image corpus for which the noise model can't be anticipated. The noise is chemical, generally with big grain and further altered by the scanning and JPEG encoding. Figures 9.21, 9.22 and 9.23 show results obtained by the Noise Clinic over this kind of noisy images.

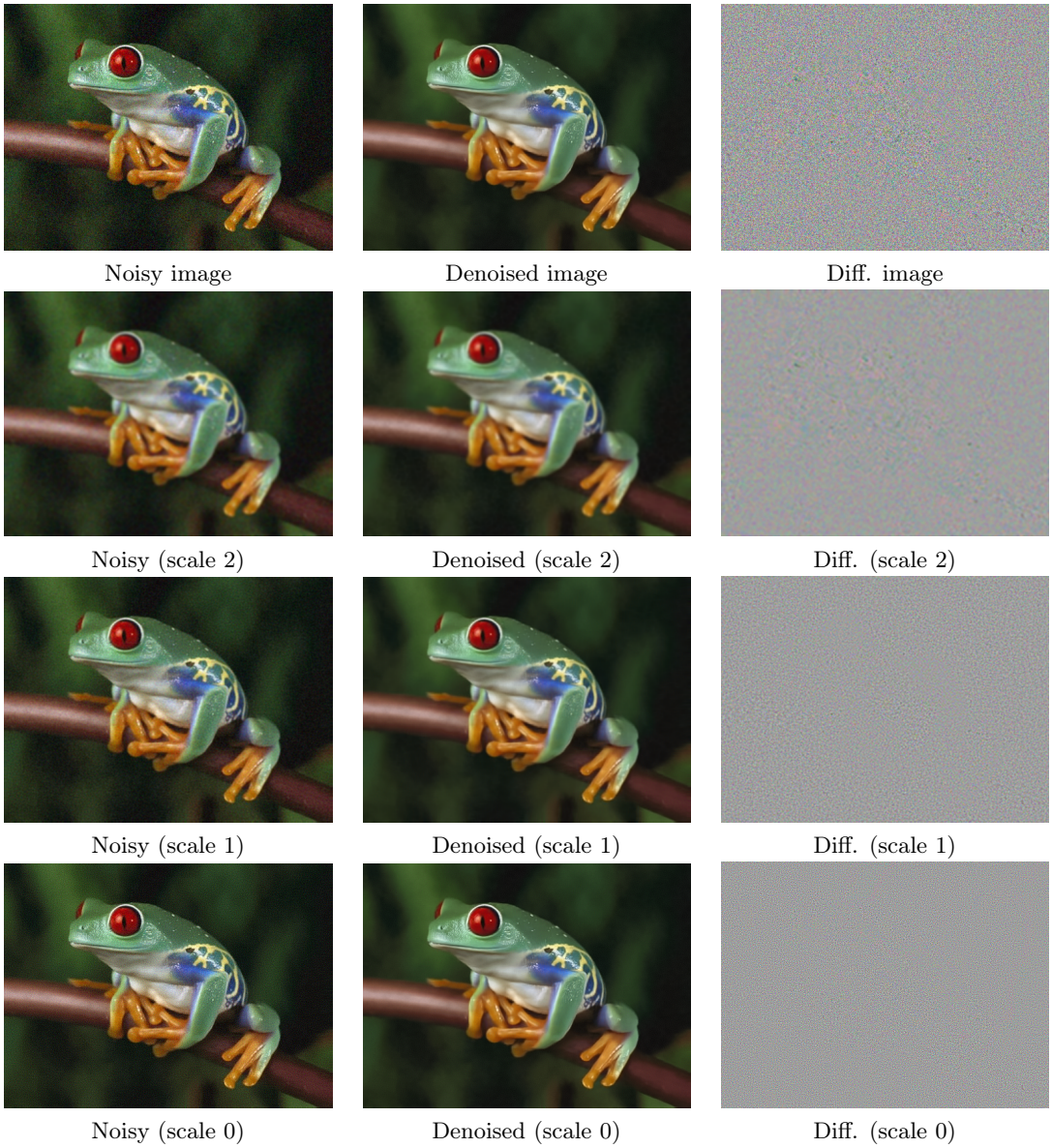
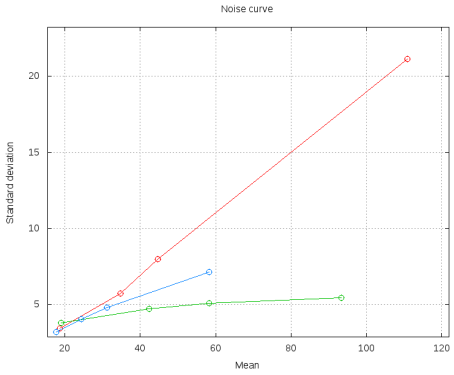
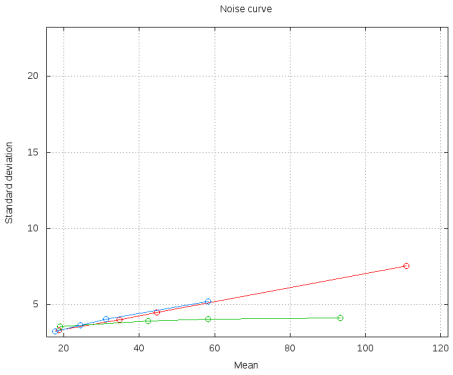


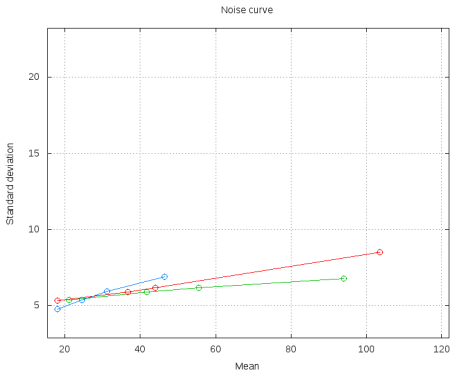
Figure 9.14: Illustration of blind denoising of a JPEG image, the "Frog" image. It is advised to zoom in the high quality .pdf to see detail.



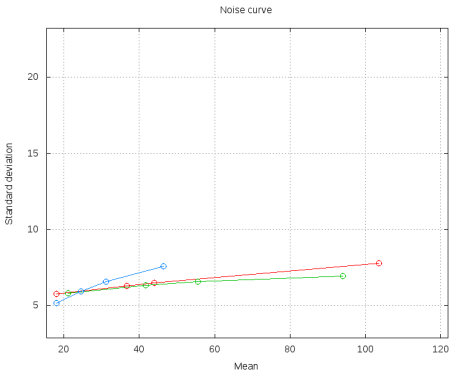
Low freq. av. curve (s. 2)



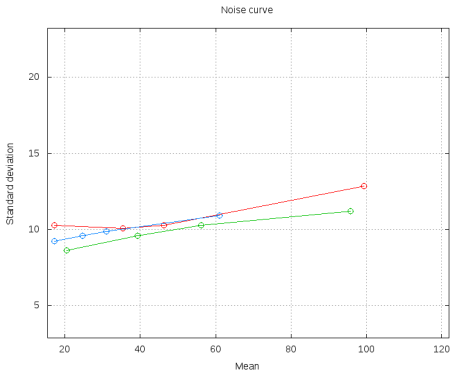
High freq. av. curve (s. 2)



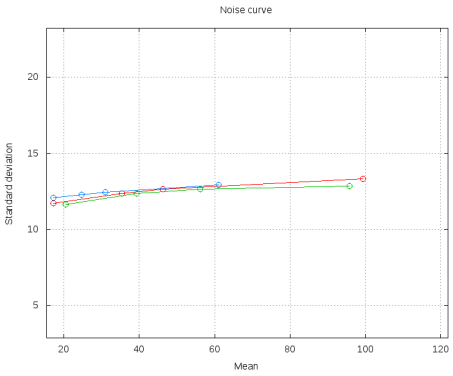
Low freq. av. curve (s. 1)



High freq. av. curve (s. 1)



Low freq. av. curve (s. 0)



High freq. av. curve (s. 0)

Figure 9.15: Noise estimation of the "Frog" image: The noise in this image is clearly colored: it increases with descending octaves instead of being divided by two, as it should if it were white.

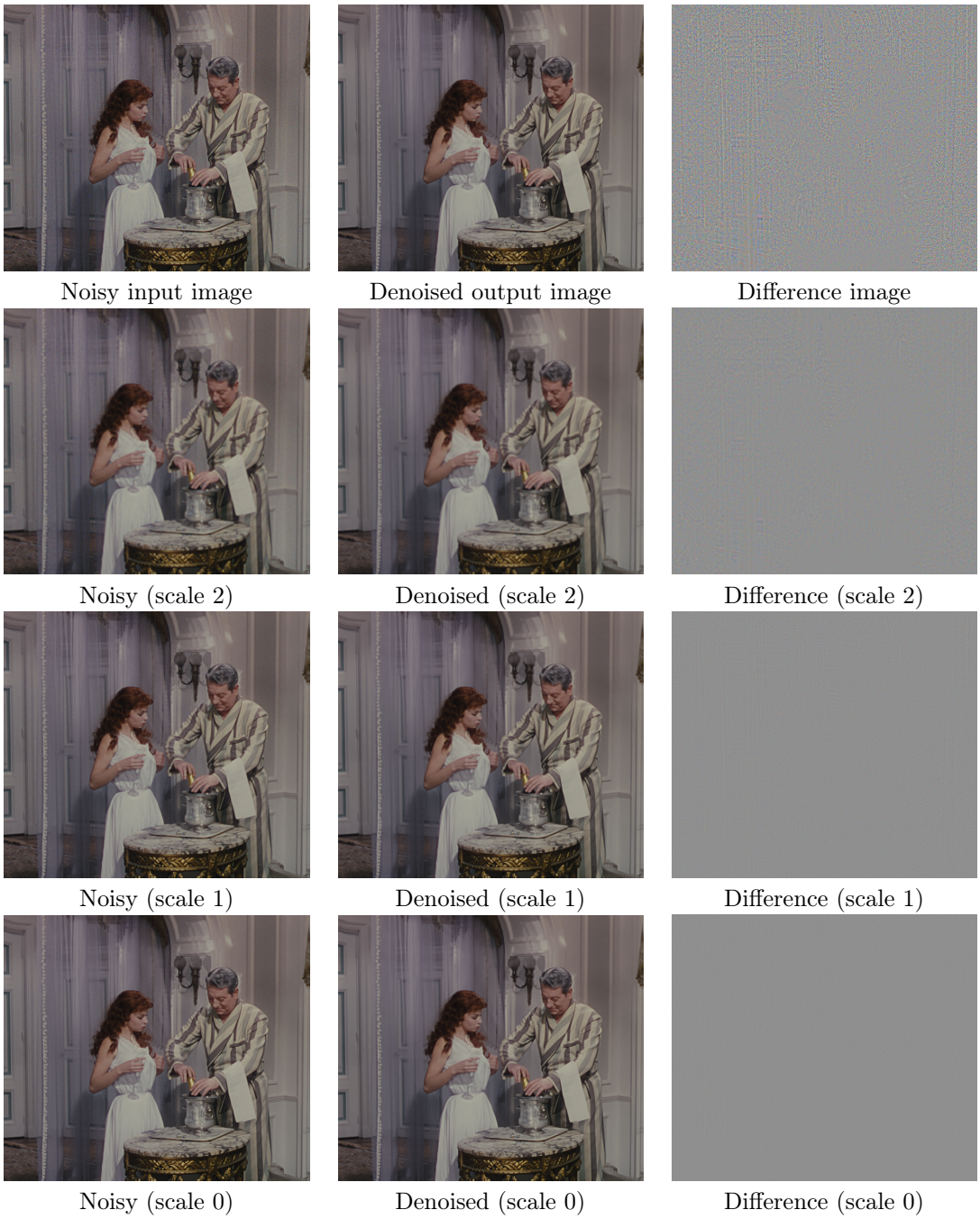
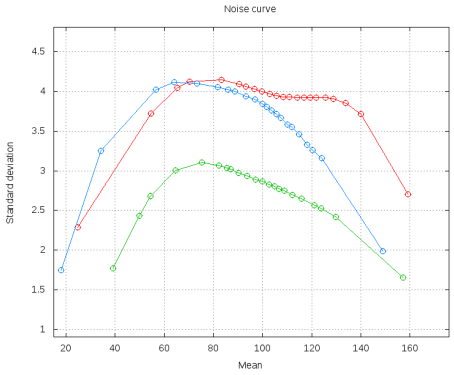
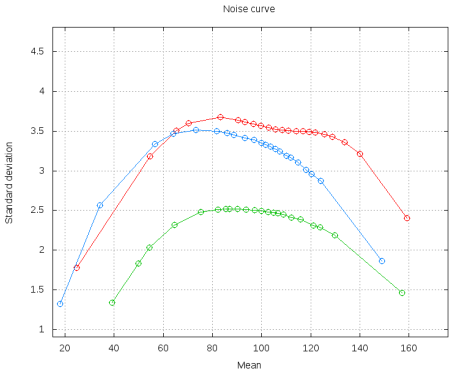


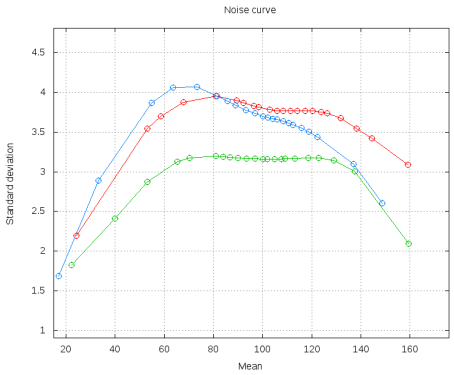
Figure 9.16: Blind denoising of the "Movie2" image.



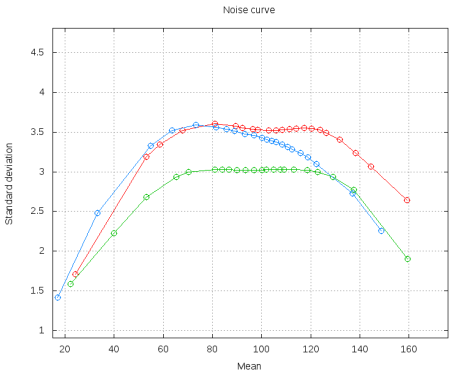
Low freq. av. curve (s. 2)



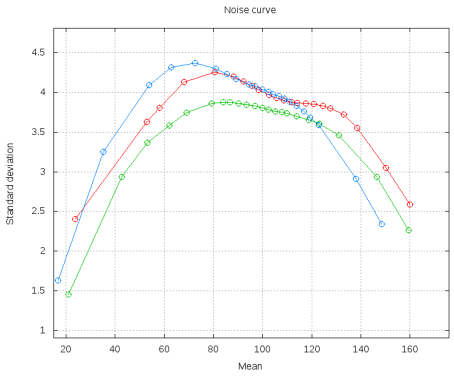
High freq. av. curve (s. 2)



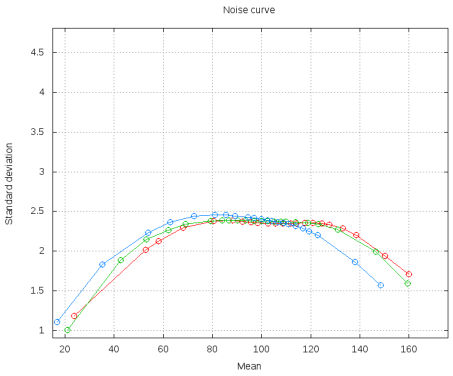
Low freq. av. curve (s. 1)



High freq. av. curve (s. 1)



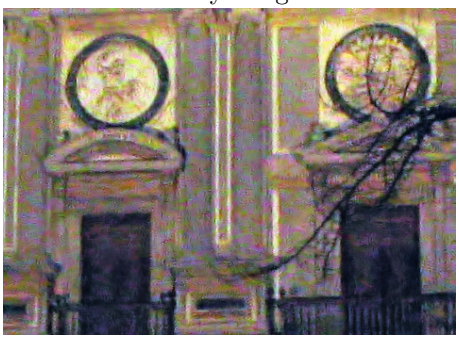
Low freq. av. curve (s. 0)



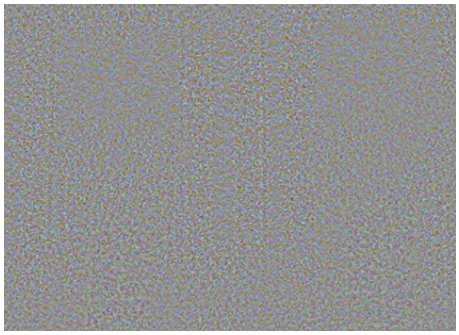
High freq. av. curve (s. 0)

Figure 9.17: Noise estimation of the "Movie2" image, with typical concave shapes for the noise curves probably caused by the gamma-correction applied previous to compression.

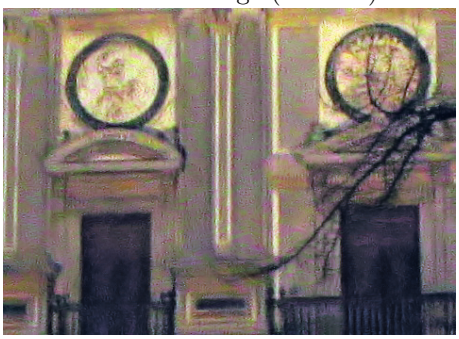
Noisy image



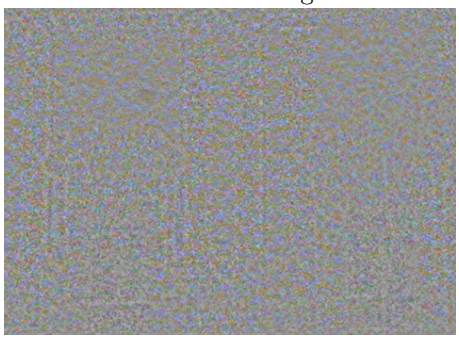
Denoised image (2 scales)



Difference image



Denoised image (3 scales)

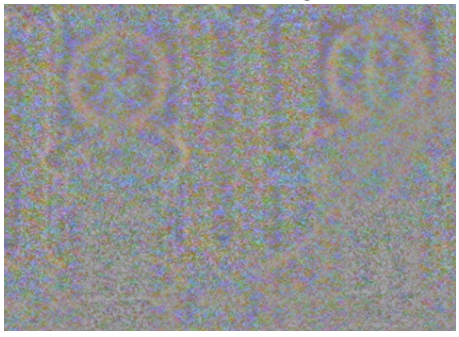


Difference image

Denoised image (4 scales)

Difference image

Denoised image (5 scales)



Difference image

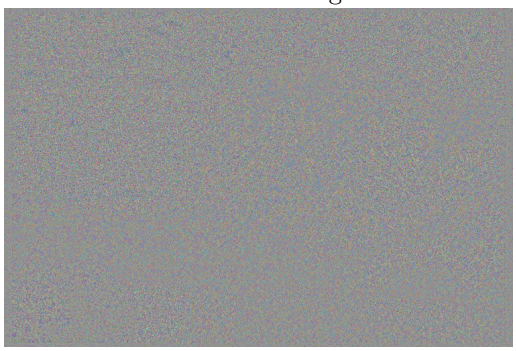
Figure 9.18: Blind denoising when varying the number of scales on “Palace”.

Noisy image

Denoised image (2 scales)

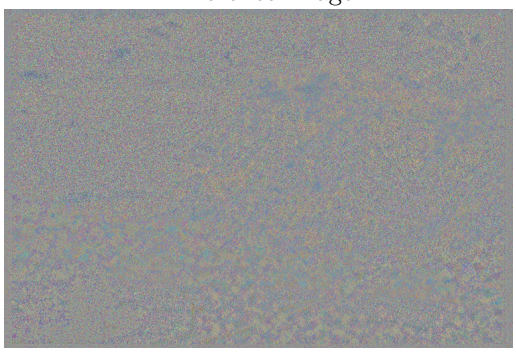
Difference image

Denoised image (3 scales)



Difference image

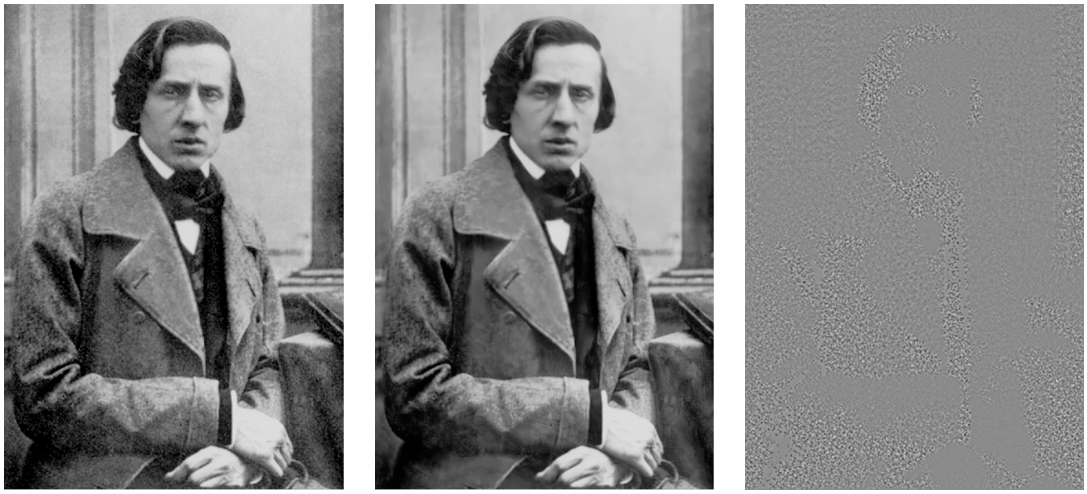
Denoised image (4 scales)



Difference image

Figure 9.19: Blind denoising when varying the number of scales on the "Postcard" image.

Figure 9.20: Blind denoising on “Bar”, using three scales. From left to right, top to bottom : input noisy image, crop of the noisy image, crop of the output denoised image, crop of the difference image.



Noisy image Denoised image Difference image

Figure 9.21: Blind denoising on “Chopin”, by using 3 scales.

Figure 9.22: Blind denoising on “Marilyn”, using two scales. From left to right, top to bottom : input noisy image, crop of the noisy image, crop of the output denoised image, crop of the difference image.

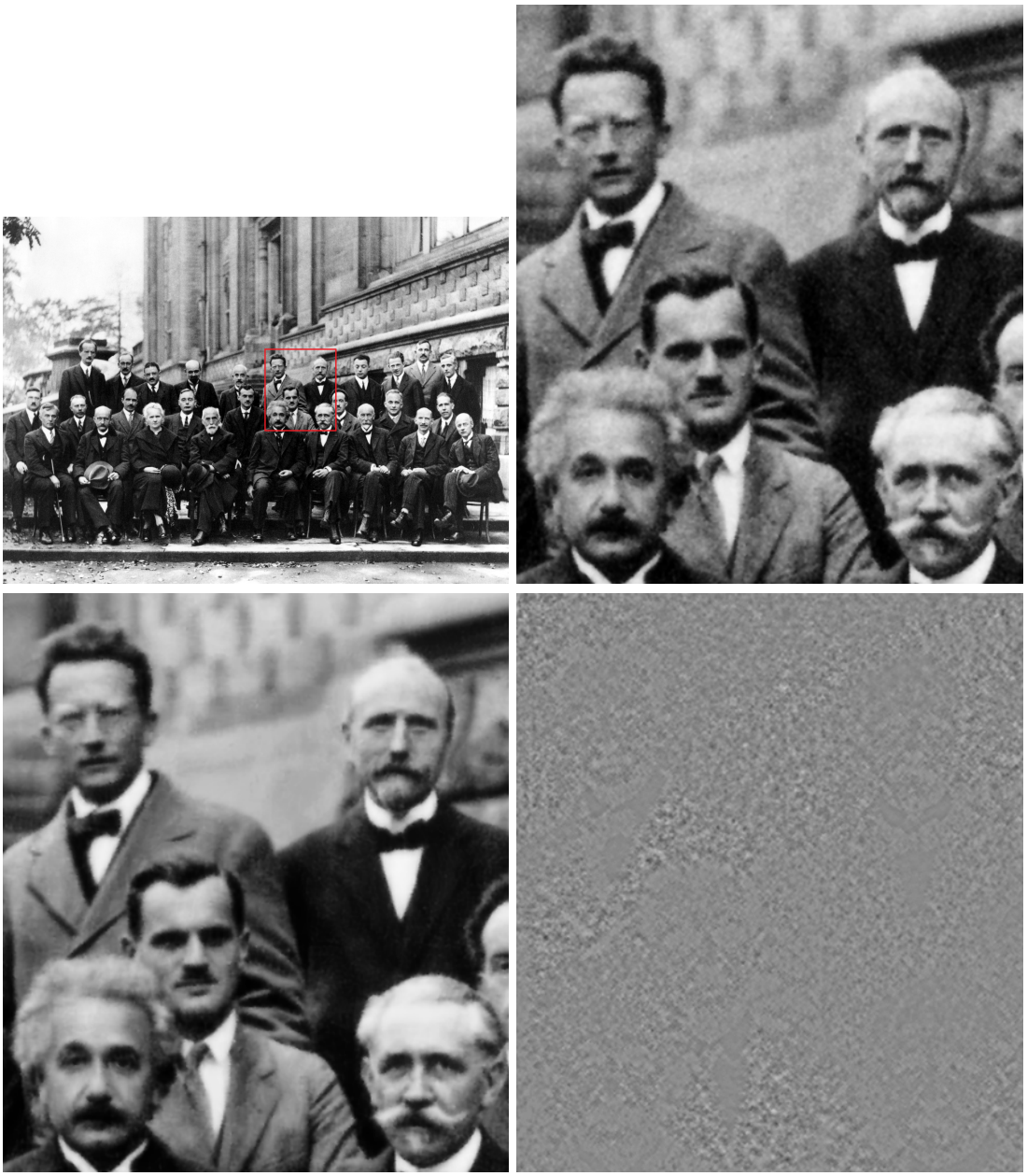


Figure 9.23: Blind denoising on "Solvay conference, 1927", using three scales. From left to right, top to bottom : input noisy image, crop of the noisy image, crop of the output denoised image, crop of the difference image.

9.6.3 Comparison to one of the very few available blind denoising algorithms

We end this experimental section with a comparison of the noise clinic with blind BLS-GSM introduced in [113] and [112], a state-of-the-art blind denoising algorithm. The comparison was performed on several images with various noise models. BLS-GSM also is a multiscale algorithm modeling wavelet coefficient patches at each scale and making a global sophisticated Bayesian estimation of them as a Gaussian mixture. NL-Bayes instead has a simpler, but local patch Gaussian model. The global patch model in BLS-GSM has to be more complex to cope with the global patch variability.

In Figure 9.24 noisy images present strongly structured periodic noise, which is remarkably removed by the blind BLS-GSM algorithm, whereas our blind denoising keeps it and even re-enforces it. However one can argue that this structured noise may be seen as a repetitive texture belonging to the image and therefore must be treated as detail and not as noise.

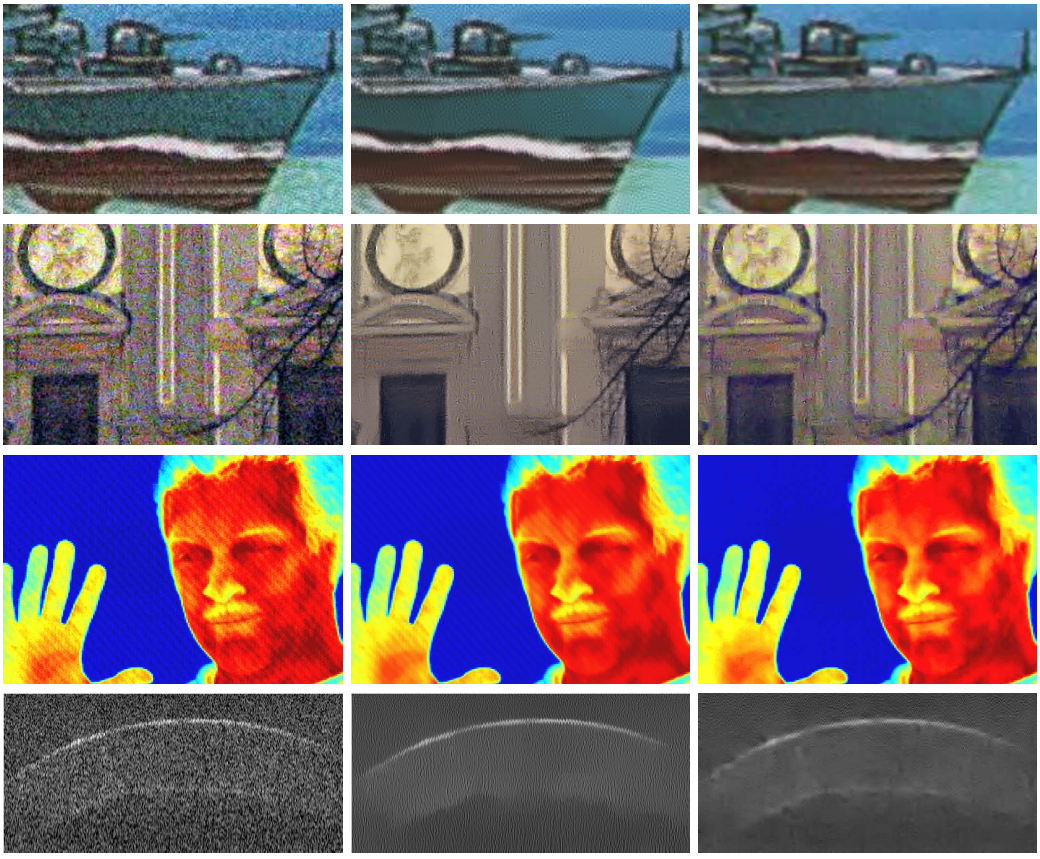


Figure 9.24: Results of our blind denoising and of Blind BLS-GSM on several images from [112]. From left to right: Noisy image, result of the Noise Clinic, result of the Blind BLS-GSM algorithm. It is advised to zoom in by a 300% factor the digital document to examine details.

In Figure 9.25 the noise is more "normal" and closer to what can be expected from a natural image, and our blind denoising performs better. Blind BLS-GSM manages to remove some noise, but a slightly structured noise still remains, appearing in horizontal strips.

Figures 9.26, 9.27 and 9.28 show comparisons for low-light JPEG image and old Photographs presented in section 9.6.2 and 9.6.2

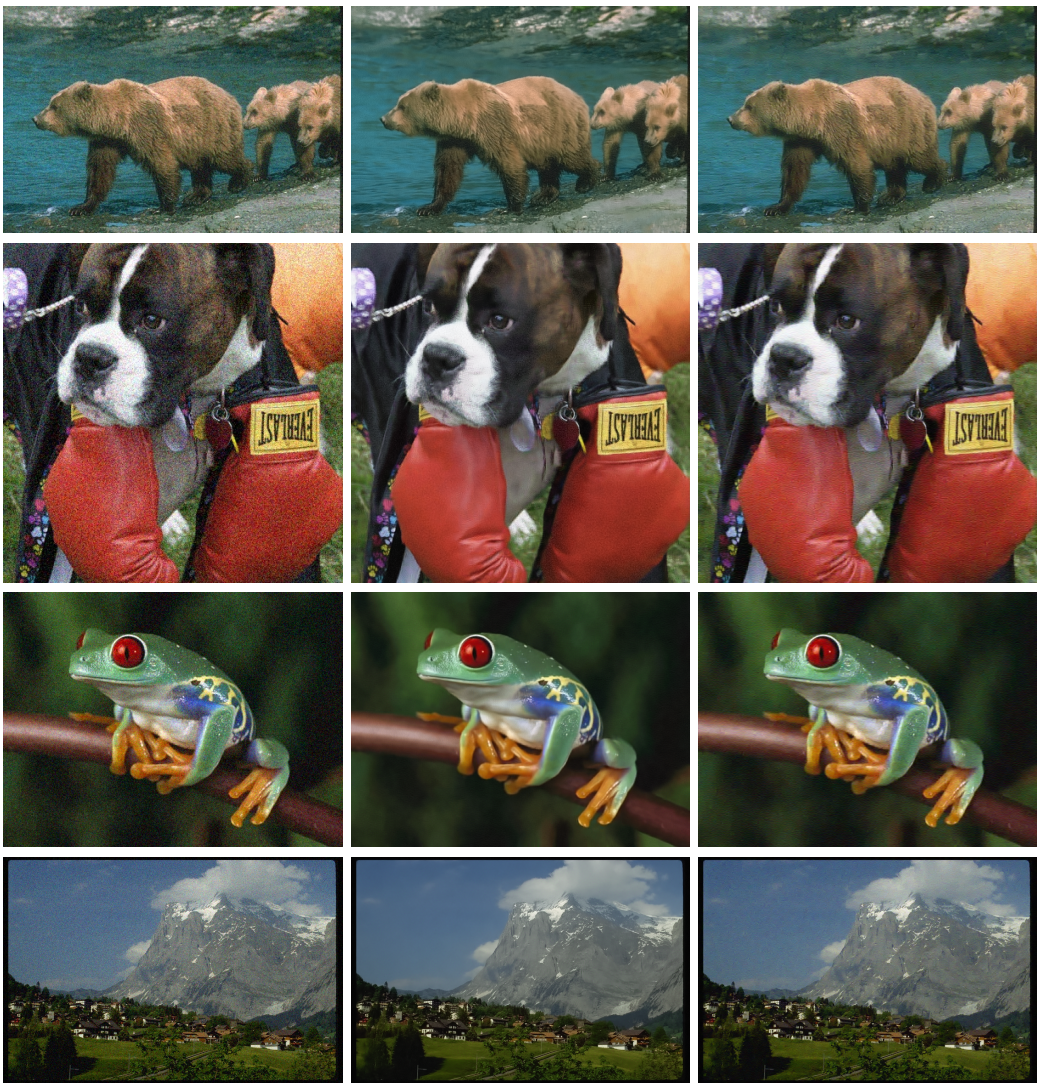


Figure 9.25: Comparing our blind denoising with Blind BLS-GSM on several images. It is advised to zoom in by a 400% factor the digital document to examine details. From left to right: Noisy image, result of the Noise Clinic, result of the Blind BLS-GSM algorithm.

Figure 9.26: Blind denoising on “Bar”. From left to right: crop of the result of the Noise Clinic by using three scales and crop of the result of the Blind BLS-GSM algorithm.

Figure 9.27: Blind denoising on Marilyn”. From left to right: crop of the result of the Noise Clinic by using two scales and crop of the result of the Blind BLS-GSM algorithm.

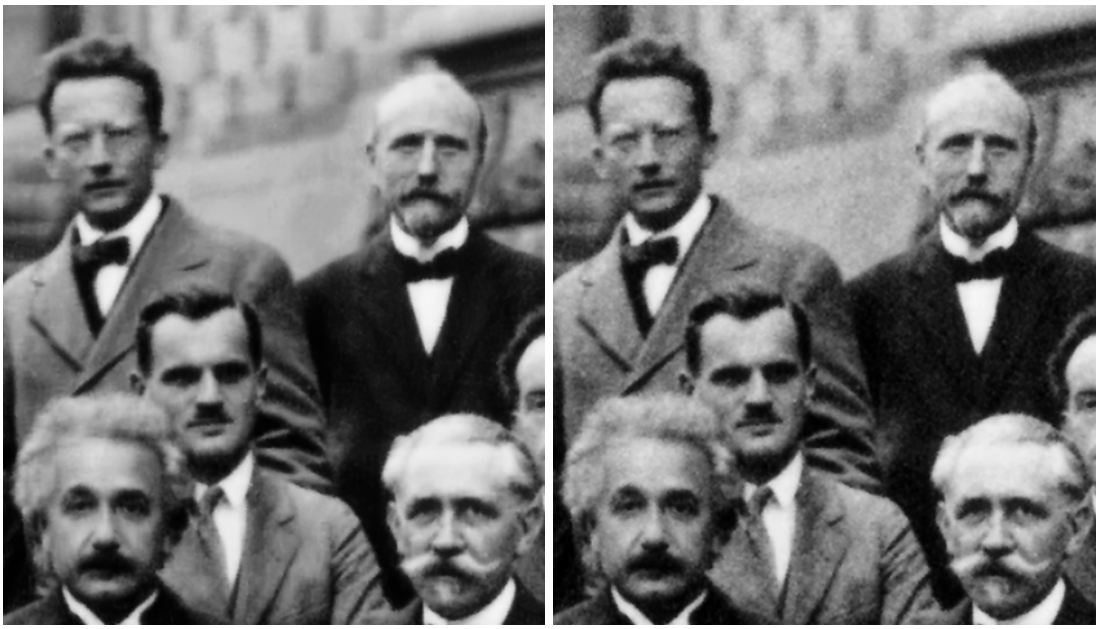


Figure 9.28: Blind denoising on "Solvay conference, 1927". From left to right: crop of the result of the Noise Clinic by using three scales and crop of the result of the Blind BLS-GSM algorithm.

9.7 Discussion

Blind denoising can be performed with minimal assumptions on the nature of the noise. We observed good results on almost any natural image, even if it had been modified by destructive applications such as JPEG compression or chemical processes. Particularly in old photographs, noise can acquire a thick grain which is only efficiently denoised at low scales. This method does not apply to impulse or multiplicative noise and should be extended to such alterations. Also our local noise estimation procedure did not detect the strength of the fully structured noise present in the third infrared image of Fig. 9.24. The case of a globally frequency dependent noise is of course better treated by Portilla's method which assumes a global noise model. We wrote that the proposed method was "signal, scale and frequency" dependent. In fact as indicated by the preceding caveat, the method estimates and processes noise frequencies in the DCT of small blocks. So these frequency coefficient are far less precise than global image frequencies. Furthermore they are scale dependent, since we applied a dyadic subsampling procedure. Since at each dyadic scale, frequencies are estimated for blocks with at least 4×4 size, it follows that these scale dependent frequencies overlap. This leads to a redundant denoising since left-over noise at a coarse scale can be estimated again, and removed again at the overlapping finer dyadic scale. This redundancy of estimators is particularly necessary for such a complex noise model. The fact that JPEG images can be denoised in that way was far from granted. Indeed, it is impossible to really model noise in JPEG images, which are the result of a chain of nonlinear operators. It can be argued that our noise signal, frequency and scale dependent noise estimation is not yet general enough to cope with such alterations. This objection is definitely valid for block artifacts apparent in strong JPEG compression. Thus, strongly compressed images where blocking effects dominate remain beyond our scope.

Part III

Reproducible research contributions

To be able to fairly compare denoising algorithms in both term of PSNR and visual quality is a really important issue. In order to make this kind of comparison, we need to apply algorithms on the same noisy images, which implies that the code source of the method is available. Moreover, we need to check that the used code perfectly corresponds to the method described in the original paper describing the method. Furthermore, if one want to understand well a method, to know how and why it works, some tests need to be done, and the best way to do it is to play with the parameters and the code.
For all these reasons, a detailed analysis of some emblematic methods is presented in this part, and reliable open source codes of the corresponding methods are available on IPOL. Chapter 10 presents an analysis of K-SVD, one of the best known sparse coding-based denoising methods. BM3D, the reference of transform thresholding, is analysed in chapter 11. Finally this part ends in chapter 12 with the analysis of NL-Bayes, the state of the art of Bayesian-based methods.

Chapter 10

A Detailed Analysis and Implementation of K-SVD

As described in chapter 4, K-SVD is a signal representation method which, from a set of signals, can derive a dictionary able to approximate each signal with a sparse combination of the atoms. As this method is emblematic of dictionary-based denoising methods, we present in this chapter a detailed description of its theory and a precise analysis of its parameters which ends with a reliable implementation.
This chapter uses the results of joint work with Arthur Leclaire.

10.1 Introduction

One class of such algorithms contains those which take profit of the analysis of the image in a (redundant) frame. For example, in this subset, we can mention the threshold of the image coefficients in an orthonormal basis, like the cosine basis [138, 137], a wavelet basis [47], or a curvelet basis [126]. In this category can also be included the methods which try to recover the main structures of the signal by using a dictionary (which basically consists of a possibly redundant set of generators). The matching pursuit algorithm [97] and the orthogonal matching pursuit [40] are of this type. The efficiency of these methods comes from the fact that natural images can be sparsely approximated in these dictionaries.
The variational methods form a second class of denoising algorithms. Among them let us mention the total variation (TV) denoising [122, 22] where the chosen regularity model is the set of functions of bounded variations.
In another class, one could include methods that take advantage of the non-local similarity of patches in the image. Among the most famous, we can name NL-means [13], BM3D [35], and NL-Bayes [76].
The K-SVD-based denoising algorithm merges some concepts coming from these three classes, paving the way of dictionary learning. Indeed, the efficiency of the dictionary is encoded through a functional which is optimized taking profit of the non-local similarities of the image. It is divided into three steps : a) sparse coding step, where, using the initial dictionary, we compute sparse approximations of all patches (with a fixed size) of the image; b) dictionary update, where we try to update the dictionary in such a manner that the quality of the sparse approximations is increased; and next, c) reconstruction step which recovers the denoised image from the collection of denoised patches. Actually, before getting to c), the algorithm carries out K iterations of steps a and b.
There is by now a thriving literature about dictionary learning. Here we will only quote the main articles that led to the design of the K-SVD algorithm for color images. The K-SVD method was introduced in [1] where the whole objective was to optimize the quality of sparse approximations of vectors in a learned dictionary. Even if this article noticed the interest of the technique in image

processing tasks, it is in [53] that a detailed study has been led on the denoising of gray-level images. Then, the adjustment to color images has been treated in [93]. Let us notice that this last article proved that the K-SVD method can also be useful in other image processing tasks, such as non-uniform denoising, demosaicing and inpainting.
Following these articles, dictionary learning has become a very active research topic. To go beyond the scope of this article, see [92] or [90].

10.2 Theoretical Description

To get a maximal coherence between the different documents about K-SVD, we use the same notations as in the article [93].

10.2.1 Algorithm for Grayscale Images

This paragraph explains the algorithm described in [53]. We work with images written in column vectors. In practice, in our C++ code, images are scanned one row at a time, these rows being next concatenated to make a single column vector. The same is done for patches.
Hence, let us denote by $\mathbf{x_0}$ a size N column vector containing the unknown clean grayscale image. Starting from $\mathbf{x}_0$, we assume that the noisy image is obtained as

$$\mathbf{y} = \mathbf{x_0} + \mathbf{w}$$

where $\mathbf{w}$ is a white Gaussian noise vector of zero mean and known standard deviation σ. Consequently, we look for an image $\hat{\mathbf{x}}$ that is close to the initial image, such that each of its patches admits a sparse representation in terms of a learned dictionary.
For every possible position (i, j) of a pixel in the image $\mathbf{x}$, we denote by $\mathbf{R}_{ij}\mathbf{x}$ the size n column vector formed by the grayscale levels of the squared $\sqrt{n} \times \sqrt{n}$ patch of the image $\mathbf{x}$ and whose top-left corner has coordinates (i, j). One can notice that, with the column notation, $\mathbf{R}_{ij}\mathbf{x}$ is precisely the multiplication of $\mathbf{x}$ (column vector of size N) by a matrix $\mathbf{R}_{ij}$ of size $n \times N$ whose columns are indexed by the image pixels. Each of the rows of $\mathbf{R}_{ij}$ allows to extract the value of one pixel of the image $\mathbf{x}$ and thus is zero except for the coefficient of index p, which is equal to 1.
In the following, the notation $\mathbf{D}$ refers to a dictionary. It is a matrix of size $n \times k$, with $k \geq n$ whose columns are normalized (in Euclidean norm). We take $k \geq n$ because otherwise, there is no chance that the columns of $\mathbf{D}$ can span $\mathbb{R}^n$. The algorithm will require an initialization of the dictionary : to this end, we may choose an usual orthogonal basis (discrete cosine transform, wavelets...), or we may collect patches from clean images or even from the noisy image itself (without forgetting the normalization). We give two examples of dictionaries in figure 10.1.
The dictionary allows to compute a sparse representation α_{ij} of each patch $\mathbf{R}_{ij}\mathbf{x}$. The representations α_{ij} will thus be column vectors of size k satisfying $\mathbf{R}_{ij}\mathbf{x} \approx \mathbf{D}\alpha_{ij}$. We put them together in a matrix $\boldsymbol{\alpha}$ with k rows and N_p columns where N_p is the number of patches of size $\sqrt{n} \times \sqrt{n}$ of the image.
With the above notation it is easy to detail each part of the algorithm. At first, $\hat{\mathbf{D}}$ is initialized with an initial dictionary denoted by $\mathbf{D}_{init}$. The initialization alternatives will be discussed later on.
The first step looks for sparse representations of the patches $\mathbf{R}_{ij}\mathbf{y}$ of $\mathbf{y}$ in the dictionary $\hat{\mathbf{D}}$. In other words, for each patch $\mathbf{R}_{ij}\mathbf{y}$, a column vector $\hat{\alpha_{ij}}$ (of size k) is built such that it has only a few non-zero coefficients and such that the distance between $\mathbf{R}_{ij}\mathbf{y}$ and its sparse approximation $\hat{\mathbf{D}}\hat{\alpha_{ij}}$ is small.
The second step updates one by one the columns of the dictionary $\hat{\mathbf{D}}$ and the representations $\hat{\alpha_{ij}}$ in such a way that all patches in the image $\mathbf{y}$ become more efficient. Therefore, the goal is to decrease the quantity

$$\sum_{i,j} \|\hat{\mathbf{D}}\hat{\alpha_{ij}} - \mathbf{R}_{ij}\mathbf{y}\|_2^2$$

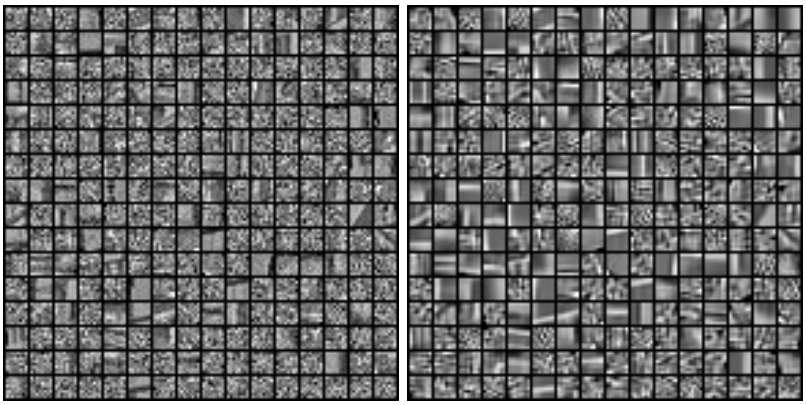


Figure 10.1: Left, a dictionary formed with random patches from the image "Castle" (converted in grayscale levels) after addition of a white Gaussian noise. Right, the dictionary obtained at the end of the K-SVD algorithm. For each atom, the contrast is enhanced differently.

while keeping the sparsity of the vectors $\hat{\alpha_{ij}}$.
K iterations of these two first steps are performed. Once finished, to each patch $\mathbf{R}_{ij}\mathbf{y}$ of the image $\mathbf{y}$ corresponds the denoised version $\hat{\mathbf{D}}\hat{\alpha_{ij}}$. The third and last step consists in merging the denoised versions of all patches of the image in order to obtain the final denoised image. A new parameter λ is introduced in this part, which blends a portion of the initial noisy image into the final result. To obtain a pixel p of the denoised image, a simple average is done on the values of p in the denoised patches to which it belongs (weighted by 1), and the value of p in the noisy image $\mathbf{y}$ (weighted by λ).
We will now take a closer look at each one of the three parts of the method.

Sparse Coding

This step allows, with a fixed dictionary $\hat{\mathbf{D}}$, to compute sparse representations $\hat{\boldsymbol{\alpha}}$ of the patches $\mathbf{R}_{ij}\mathbf{y}$ of the image in $\hat{\mathbf{D}}$. More precisely, an ORMP (Orthogonal Recursive Matching Pursuit) gives an approximate solution of the (NP-complete) problem

$$\underset{\alpha_{ij}}{\text{Arg min}}\, ||\alpha_{ij}||_0 \quad \text{such that} \quad ||\mathbf{R}_{ij}\mathbf{y} - \hat{\mathbf{D}}\alpha_{ij}||_2^2 \leq n(C\sigma)^2 \,. \tag{10.1}$$

where $\|\alpha_{ij}\|_0$ refers to the l^0 norm of α_{ij}, i.e. the number of non-zero coefficients of α_{ij}. We remind the reader that $\hat{\mathbf{D}}$ is a matrix whose size is $n \times k$, that α_{ij} is a size k column vector and that $\mathbf{R}_{ij}\mathbf{y}$ is a size n column vector. If it were perfect, this ORMP would find a patch with the sparsest representation in $\hat{\mathbf{D}}$ and which distance to $\mathbf{R}_{ij}\mathbf{y}$ is less than $n(C\sigma)^2$. This last constraint brings in a new parameter C. This coefficient multiplying the standard deviation σ guarantees that, with high probability, a white Gaussian noise of standard deviation σ on n pixels has an l^2 norm lower than $\sqrt{n}C\sigma$. We give details on the choice of C in Section 10.3. In fact, the ORMP is not perfect : indeed, it only allows one to find a patch having one sparse (not necessarily the sparsest) representation in $\hat{\mathbf{D}}$ and which distance to $\mathbf{R}_{ij}\mathbf{y}$ is lower than $n(C\sigma)^2$.
Let us give more details about how the ORMP can compute a sparse representation of a patch. A good reference to learn about ORMP is [34]. Nevertheless, we shall give here a complete explanation using the notation of our C++ code. In order to use lighter notations, we will rather explain how the ORMP finds a sparse representation $a \in \mathbb{R}^k$ of a vector $x \in \mathbb{R}^n$ in a dictionary formed by the normalized vectors $d_1, \ldots, d_k$ which span $\mathbb{R}^n$.
Let x be a vector of $\mathbb{R}^n$. We wand to find a sparse representation α of x in the dictionary $\mathbf{D}$ formed by the normalized vectors $d_0, \ldots, d_{k-1}$. Precisely, we are going to give an approximate

solution of the following optimization problem :

$$\underset{\alpha\in\mathbb{R}^k}{\text{Arg min}}\,||\alpha||_0 \quad \text{such that} \quad ||x-\mathbf{D}\alpha||_2^2 \leq \varepsilon\ . \tag{10.2}$$

We will detail the choice of the atoms in order to stick to our C++ code.
We denote by l_j the index of the element of the dictionary that we choose at the step $j \geq 0$. We also set $L_j = \{l_0, \ldots, l_j\}$.
Let us assume that we are at the beginning of the j-th loop ($j \geq 0$) (and thus $l_0, \ldots, l_{j-1}$ are already chosen).
We start by introducing the residue

$$r = x - \text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_{j-1}})}(x)$$

where Proj_F refers to the subspace F, and where $\text{Vect}(d_{l_0}, \ldots, d_{l_{j-1}})$ refers to the space spanned by the vectors $d_{l_0}, \ldots, d_{l_{j-1}}$. If $\|r\|_2 < \varepsilon$ then we stop and α is the representation of $\text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_{j-1}})}(x)$ in $(d_{l_0}, \ldots, d_{l_{j-1}})$ already obtained at the previous step, cf. its computation at the end of the loop [1]. We choose l_j in order to minimize the norm of the new potential residue :

$$l_j = \underset{i\notin L_{j-1}}{\text{Arg min}}\,\|x - \text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_{j-1}},d_i)}(x)\|^2\ .$$

Thanks to the Pythagorean theorem, this amounts to

$$l_j = \underset{i\notin L_{j-1}}{\text{Arg max}}\,\|\text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_{j-1}},d_i)}(x)\|^2\ .$$

Then we set $L_j = L_{j-1} \cup \{l_j\}$.
In order to compute the orthogonal projections

$$\text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_{j-1}},d_i)}(x)\ , \quad (i \notin L_{j-1})$$

we use the Gram-Schmidt process. We denote by $(t_{l_0}, \ldots, t_{l_{j-1}})$ the orthogonal family obtained after Gram-Schmidt orthogonalization of $(d_{l_0}, \ldots, d_{l_{j-1}})$, and by $(e_{l_0}, \ldots, e_{l_{j-1}})$ the orthonormal family obtained after Gram-Schmidt orthonormalization of $(t_{l_0}, \ldots, t_{l_{j-1}})$. For $i \notin L_{j-1}$, we denote by $(t_{l_0}, \ldots, t_{l_{j-1}}, t_i^{(j)})$ the family obtained after Gram-Schmidt orthogonalization of $(d_{l_0}, \ldots, d_{l_{j-1}}, d_i)$, and $(e_{l_0}, \ldots, e_{l_{j-1}}, e_i^{(j)})$ the (orthonormal) family obtained by normalizing of $(t_{l_0}, \ldots, t_{l_{j-1}}, t_i^{(j)})$. The reader have to be aware that this orthonormalization can be progressively computed : at the j-th step, the vectors $(t_{l_0}, \ldots, t_{l_{j-1}})$ and $(e_{l_0}, \ldots, e_{l_{j-1}})$ are already computed. It is thus sufficient to detail, at the j-th step, the computation of d_i and $t_i^{(j)}$ for $i \notin L_{j-1}$:

$$t_i^{(j)} = d_i - \sum_{p=0}^{j-1} \langle d_i, e_{l_p}\rangle e_{l_p}\ ,$$

$$\|t_i^{(j)}\|^2 = 1 - \sum_{p=0}^{j-1} \langle d_i, e_{l_p}\rangle^2\ ,$$

$$e_i^{(j)} = \frac{t_i^{(j)}}{\|t_i^{(j)}\|}\ .$$

We notice that

$$\text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_{j-1}},d_i^{(j)})}(x) = \langle x, e_{l_0}\rangle e_{l_0} + \ldots + \langle x, e_{l_{j-1}}\rangle e_{l_{j-1}} + \langle x, e_i^{(j)}\rangle e_i^{(j)}$$

[1] if we break when $j = 0$, then $\alpha = 0$

(where Proj_F refers to the orthogonal projection onto the subspace F) and, consequently,

$$\|\text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_{j-1}},d_i)}(x)\|^2 = \langle x, e_{l_0}\rangle^2 + \ldots + \langle x, e_{l_{j-1}}\rangle^2 + \langle x, e_i^{(j)}\rangle^2 \ .$$

Therefore, maximizing the norm of the projection is equivalent to maximize $\langle x, e_i^{(j)}\rangle$. This is why we choose

$$l_j = \underset{i \notin L_{j-1}}{\text{Arg max}} \ \langle x, e_i^{(j)}\rangle^2$$

and with this index comes the vector $t_{l_j} = t_{l_j}^{(j)}$ and the normalized vector $e_{l_j} = e_{l_j}^{(j)}$. The computation of $\langle x, e_i^{(j)}\rangle$ is done by replacing $e_i^{(j)}$ by its above given definition :

$$\langle x, e_i^{(j)}\rangle = \frac{\langle x, d_i\rangle - \sum_{p=0}^{j-1}\langle d_i, e_{l_p}\rangle\langle x, e_{l_p}\rangle}{\sqrt{1 - \sum_{p=0}^{j-1}\langle d_i, e_{l_p}\rangle^2}} \ . \tag{10.3}$$

To implement this computation efficiently, we notice that the denominator and the square of the numerator are nothing but the subtraction of those used at the previous step by respectively $\langle d_i, e_{l_{j-1}}\rangle\langle x, e_{l_{j-1}}\rangle$ and $\langle d_i, e_{l_{j-1}}\rangle$. Hence, at each step, we need $\langle d_i, e_{l_{j-1}}\rangle$ and $\langle x, e_{l_{j-1}}\rangle$ which correspond in the code to the variables `D_ELj[i][j]` and `x_elj`, which are updated at each loop. The computation of $\langle x, e_{l_{j-1}}\rangle$ is not a problem (it is only the formula 10.3 of the previous step !). However, we have to explain the update of $\langle d_i, e_{l_{j-1}}\rangle$. We will see thereafter that the computation of α requires the coordinates of $(e_{l_0}, \ldots, e_{l_{j-1}})$ on the basis $(d_{l_0}, \ldots, d_{l_{j-1}})$ and we will explain how we can obtain them progressively. Once these coordinates are computed, the scalar product $\langle d_i, e_{l_{j-1}}\rangle$ can be obtained by a linear combination of the scalar products $\langle d_i, d_{l_s}\rangle$, $(0 \leq s < j)$. The numerator $(\langle x, t_i^{(j)}\rangle)$ is then saved in the variable `x_T[i]`, and the square of the denominator in the variable `scores[i]`.
Once we have chosen l_j, we can go back to the beginning of the loop to stop or choose the next atom. Clearly, the algorithm terminates because the atoms $d_1, \ldots, d_k$ span $\mathbb{R}^n$.
At this point let us assume that we are at the end of the j-th loop (and thus, we have chosen $l_0, \ldots, l_j$). We still have to explain how the sparse representation α of x in $\mathbf{D}$ is computed.
As $(e_{l_0}, \ldots, e_{l_j})$ is orthonormal and span $\text{Vect}(d_{l_0}, \ldots, d_{l_j})$, we have

$$x \approx \text{Proj}_{\text{Vect}(d_{l_0},\ldots,d_{l_j})} = \sum_{p=0}^{j} \langle x, e_{l_p}\rangle e_{l_p} \ .$$

The coefficients $\langle x, e_p\rangle$, $(p < j)$ have already been computed in the preceding step. The last coefficient is given by the equality (10.3) for $i = l_j$.
Finally, we have to go back to the representation in terms of $d_{l_1}, \ldots, d_{l_j}$. To this aim, we introduce the coordinates of the $(e_{l_0}, \ldots, e_{l_{j-1}})$ on the basis $(d_{l_0}, \ldots, d_{l_{j-1}})$. Let us denote them by a_{pq}, $(p \leq q)$:

$$\forall p < j, \quad e_{l_p} = \sum_{q=0}^{p} a_{pq} d_{l_q} \ .$$

At the $j - 1^{th}$-step, the a_{pq} are computed for $p < j$ (and again $p \leq q$). It suffices to explain how we compute a_{jq} for $q \geq j$. From the definition e_{l_j}, replacing the e_{l_p}, $(p < j)$, we obtain

$$e_{l_j} = \frac{1}{\|t_{l_j}\|}\left(d_{l_j} - \sum_{p=0}^{j-1}\sum_{q=0}^{p}\langle d_{l_j}, e_{l_p}\rangle a_{pq} d_{l_q}\right) \ ,$$

from which we get (after inverting the sums)

$$a_{jj} = \frac{1}{\|t_{l_j}\|} \ , \tag{10.4}$$

$$\forall q < j, \quad a_{jq} = -\frac{1}{\|t_{l_j}\|} \left(\sum_{p=q}^{j-1} \langle d_{l_j}, e_{l_p} \rangle a_{pq} \right) . \tag{10.5}$$

Finally, we have

$$x \approx \sum_{p=0}^{j} \langle x, e_{l_p} \rangle e_{l_p} = \sum_{q=0}^{j} \left(\sum_{p=q}^{j} \langle x, e_{l_p} \rangle a_{pq} \right) d_{l_q} ,$$

and thus we set

$$\alpha_s = 0 , \quad \text{if } s \notin L_j, \quad \text{and}$$

$$\alpha_s = \sum_{p=q}^{j} \langle x, e_{l_p} \rangle a_{pq} , \quad \text{if } s = l_q .$$

We insist on the fact that the coordinates of $(e_{l_0}, \ldots, e_{l_{j-1}})$ on the basis $(d_{l_0}, \ldots, d_{l_{j-1}})$ are also required for the choice of the index l_j, as explained above. Subsequently, it is natural to compute these coordinates at each loop.

Correspondence with the Notations Used in the Code Now we link the notations used in the explanation above with the notations used in the code. First, in the code, let us warn the reader that we have used indexation in column order, that is, `D[i]` refers to the `i`-th column of the matrix `D`.
We have also used a convention : whenever a variable contains the matrix multiplication of the transpose of `B` by `A`, then the result is saved in the variable `A_B`. Therefore, `A_B` $= {}^T$ `B A`, and `A_B[p][q]` is the scalar product between `A[p]` and `B[q]`.
Let us add that `elj` (even if it is not a proper variable) will of course refer to e_{l_j}. Similarly, `DLj` (resp. `ELj`) will refer to the matrix whose columns are (in order) $d_{l_0}, \ldots, d_{l_j}$ (resp. $e_{l_0}, \ldots, e_{l_j}$). Last, `T` will refer to the matrix whose columns are $t_0^{(j)}, \ldots, t_{k-1}^{(j)}$.

- `Np` $= N_p$
- `n` $= n$
- `k` $= k$
- `epsilon` $= \varepsilon$
- `L` : maximal sparsity allowed for the representations (here we do not use this constraint, i.e. in our code, `L` $= \min(n, k)$)
- `norm[i]` $= \|t_i^{(j)}\|^2 = 1 - \sum_{p=0}^{j-1} \langle d_i, e_{l_p} \rangle^2$
- `x_T[i]` $= \langle x, t_i^{(j)} \rangle = \langle x, d_i \rangle - \sum_{p=0}^{j-1} \langle d_i, e_{l_p} \rangle \langle x, e_{l_p} \rangle$
- `scores[i]` $= \langle x, e_i^{(j)} \rangle^2 = \frac{\text{Rdn}[i]^2}{\text{norm}[i]}$
- `lj` $= l_j$
- `invNorm` $=$ `1/sqrt(norm[lj])` $= \frac{1}{\|t_{l_j}\|}$
- `x_elj` $=$ `x_T[lj]*invNorm` $= \langle x, e_{l_j} \rangle$
- `x_el[p]` $= \langle x, e_{l_p} \rangle$
- `delta` $=$ `x_elj*x_elj` $= \langle x, e_{l_j} \rangle^2$
- `normr` $= \|x\|^2 - \sum_{p=0}^{j} \langle x, e_{l_p} \rangle^2$
- `D_DLj[i][s]` $= \langle d_i, d_{l_s} \rangle$
- `A[p][q]` $= a_{pq}$, $(p \geq q)$
- `D_ELj[i][j]` is equal to $\langle d_i, e_{l_j} \rangle$ at the end of the j-th loop.
- `val` temporarily saves the variable $\langle d_i, e_{l_j} \rangle$
- `coord[q]` $= \alpha_{l_q} = \sum_{p=q}^{j} \langle x, e_{l_p} \rangle a_{pq}$: "coordinate" of x on d_{l_q}
- `s` : summing index

Some Remarks on the Implementation

update of A equations (10.4) and (10.5) suggest the update :

$$\texttt{A}[j][j] = \texttt{invNorm}$$

$$\forall i < j, \quad \texttt{A}[j][i] = \left(-\sum_{k=i}^{j-1} \texttt{D\_ELj[lj][k] * A[k][i]} \right) \cdot \texttt{invNorm} \, .$$

numerical stability an artificial break is added in the code. It happens if $\|t_j\| < 10^{-6}$. Thus the ORMP is stopped in order to avoid the division by $\|t_j\|$.

Dictionary Update

In this step, we will see that we will be able to update the columns of the dictionary one by one, to make the quantity

$$\sum_{i,j} \|\hat{\mathbf{D}}\hat{\alpha_{ij}} - \mathbf{R}_{ij}\mathbf{y}\|_2^2 \tag{10.6}$$

decrease, without increasing the sparsity penalty $\|\alpha_{ij}\|_0$. We will denote by $\hat{d}_l$ $(1 \leq l \leq k)$ the columns of the dictionary $\hat{\mathbf{D}}$.
First, let us try to minimize the quantity (10.6) without taking care of the sparsity. As explained above, we go through the columns of the dictionary, and the index of the current column will be denoted by l, $(1 \leq l \leq k)$. We are going to modify the atom $\hat{d}_l$ and the coefficients $\hat{\alpha_{ij}}(l)$ in order to improve the approximations in an L^2 distortion sense. In order to translate this objective into an optimization problem, for each (i, j), we introduce the residue

$$e_{ij}^l = \mathbf{R}_{ij}\mathbf{y} - \hat{\mathbf{D}}\hat{\alpha_{ij}} + \hat{d}_l\hat{\alpha_{ij}}(l) \tag{10.7}$$

which is the error committed by deciding not to use $\hat{d}_l$ any more in the representation of the patch $\mathbf{R}_{ij}\mathbf{y}$: e_{ij}^l is thus a size n vector.

These residues are grouped together in a matrix $\mathbf{E}_l$ (whose columns are indexed by (i, j)). The values of the coefficients $\hat{\alpha_{ij}}(l)$ are also grouped in a row vector denoted by $\hat{\alpha}^l$. Therefore, $\mathbf{E}_l$ is a matrix of size $n \times N_p$ and $\hat{\alpha}^l$ is a row vector of size N_p. We need to find a new $\hat{d}_l$ and a new row vector $\hat{\alpha}^l$ which minimize

$$\sum_{i,j} \|\hat{\mathbf{D}}\hat{\alpha_{ij}} - \hat{d}_l\hat{\alpha_{ij}}(l) + d_l\alpha^l - \mathbf{R}_{ij}\mathbf{y}\|_2^2 = ||\mathbf{E}_l - d_l\alpha^l||_F^2 \tag{10.8}$$

where the squared Frobenius norm $||M||_F^2$ refers to the sum of the squared elements of M. This Frobenius norm is also equal to the sum of the squared (Euclidean) norm of the columns, and it is easy to check that minimizing (10.8) amounts to reduce the approximation error caused by $\hat{d}_l$. It is well-known that the minimization of such a Frobenius norm consists in a rank-one approximation, which always admits a solution, practically given by the singular value decomposition (SVD). Using the SVD of $\mathbf{E}_l$:

$$\mathbf{E}_l = U\Delta V^T \tag{10.9}$$

(where U and V are orthogonal matrices and where Δ is the null matrix except from its first diagonal, where it is non-negative and decreasing), the updated values of $\hat{d}_l$ and $\hat{\alpha}^l$ are respectively the first column of U and the first column of V multiplied by $\Delta(1, 1)$. By the way, we will notice that the rank-one approximation does not require the computation of the whole matrices U, V, and Δ. In our implementation, it is sufficient to use a truncated SVD, which is much faster (especially if $\mathbf{E}_l$ is large). Let us explain the method we used to compute the truncated SVD.

To use lighter notations, we use, as in the code, the notation $X = \mathbf{E}_l$. Starting from the SVD (10.9), one can write

$$XX^T = U\Delta\Delta^T U^T \ ,$$

$$X^T X = V\Delta^T \Delta V^T \ .$$

As a result, $\Delta(1,1)$ is the square of the greatest eigenvalue of the symmetrical positive-definite matrix XX^T, and the first column of U is the corresponding eigenvector. The same observation is valid for V. Therefore, we can find these eigenvectors and $\Delta(1,1)$ thanks to the power method applied to the matrices XX^T and $X^T X$. Concerning the convergence of the power method, one could refer to [2]. One could notice that in the pseudo-code that we present below, the power method can be applied to the two matrices simultaneously.
The SVD function takes as arguments a matrix X of which we want the SVD, a maximal number of iterations `max_iter` (set to 100 in the code) and a tolerance threshold ε (set to 10^{-6} in the code). It gives back an approximation s of the greatest singular value of X, an approximation u of the first column of U, and an approximation v of the first column of V.
Here is the pseudo-code.
Initialization : we arbitrarily initialize v (in the code, we set $v = \hat{d}_l$); we also set $i = 0$, $s = 1$ and $s_{old} = 0$.
While ($i <$ `max_iter` and $\left|\frac{s - s_{old}}{s}\right| > \varepsilon$), we proceed to the following affectations :

$$u \leftarrow Xv, u \leftarrow \frac{u}{\|u\|}, v \leftarrow X^T u, s_{old} \leftarrow s, s \leftarrow \|v\|, v \leftarrow \frac{v}{s}$$

The values of s, u, and v obtained at the end of this loop are the return values of the truncated SVD.

Remark : At the end of this algorithm, we thus have

$$XX^T u \approx \lambda u$$

where λ is the greatest eigenvalue of XX^T. Taking the scalar product with u, and since u is normalized, we have

$$\|X^T u\|^2 = \langle XX^T u, u \rangle \approx \lambda \ ,$$

which yields

$$s \approx \sqrt{\lambda} \ .$$

This explains why s is an approximation of the largest singular value of X.
This way, for each $l = 1, \cdots, k$, the energy (10.6) never increases. But for now, the sparsity of the coefficients is not under control. In order to do that, a slight modification is brought in to the preceding process : for each l,the operations involved in the update of $\hat{d}_l$ and $\hat{\alpha}^l$ is restricted to the patches which already used the atom $\hat{d}_l$ before the update.
Setting

$$\omega_l = \{ \, (i,j) \, | \, \hat{\alpha_{ij}}(l) \neq 0 \, \} \ ,$$

the values that we will group together in $\mathbf{E}_l$ and $\hat{\alpha}^l$ will be only the values of e^l_{ij} and $\hat{\alpha_{ij}}(l)$ for indices $(i,j) \in \omega_l$. Hence, the indices (i,j) of the sum of the LHS of (10.8) will be restricted to $(i,j) \in \omega_l$; the matrix $\mathbf{E}_l$ is now of size $n \times \text{Card}(\omega_l)$ and $\hat{\alpha}^l$ is now a row vector of size $\text{Card}(\omega_l)$. Also, in (10.6), note that the terms of indices $(i,j) \notin \omega_l$ are not affected by this update. This proves that this modification decreases (10.6) without increasing $\|\alpha_{ij}\|_0$. This modification also implies a reduction of the matrix $\mathbf{E}_l$ which SVD is being computed.
Recall that the sparse coding computes sparse representations $\hat{\boldsymbol{\alpha}}$ and that the dictionary updates make $\hat{\mathbf{D}}$ change but also modify $\hat{\boldsymbol{\alpha}}$. After K iterations of these steps, we are in possession of a learned dictionary $\hat{\mathbf{D}}$ and of sparse representations $\hat{\alpha_{ij}}$ of the patches of the image.

Reconstruction

Now that the first two parts of the algorithm built a dictionary $\hat{\mathbf{D}}$ and sparse representations $\hat{\alpha_{ij}}$ which are well-adapted to our image, we can build the globally denoised image by solving the minimization problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^N}{\text{Arg min}} \ \lambda||\mathbf{x}-\mathbf{y}||_2^2 \ + \ \sum_{i,j} ||\hat{\mathbf{D}}\hat{\alpha_{ij}} - \mathbf{R}_{ij}\mathbf{x}||_2^2 \ . \tag{10.10}$$

The first term controls the global proximity to our reconstruction $\hat{\mathbf{x}}$ with the noisy image $\mathbf{y}$. It is thus a fidelity term that is weighted by the parameter λ. The second term controls the proximity of the patch $\mathbf{R}_{ij}\hat{\mathbf{x}}$ of our reconstruction to the denoised patch $\mathbf{D}\alpha_{ij}$. This functional is quadratic, coercive, and differentiable. Subsequently, this problem admits a unique solution that we can compute explicitly :

$$\hat{\mathbf{x}} = \left(\lambda\mathbf{I} + \sum_{i,j}\mathbf{R}_{ij}^T\mathbf{R}_{ij}\right)^{-1}\left(\lambda\mathbf{y} + \sum_{i,j}\mathbf{R}_{ij}^T\hat{\mathbf{D}}\hat{\alpha_{ij}}\right) \ . \tag{10.11}$$

This formula can appear a little bit complicated, but **it** is in fact very simple. The only thing to notice is that the matrix that has to be inverted is diagonal. In consequence, this formula only means that the value of a pixel in the denoised image is computed by averaging the value of this pixel in the noisy image (weighted by λ) and the values of this pixel on the patches to which he belongs (weighted by 1). We obtain the values of the pixels of $\hat{\mathbf{x}}$, one by one, without requiring any matrix inversion that (10.11) would perhaps suggest.

Comments

In the articles [53] and [93] the following minimization problem is mentioned :

$$(\hat{\mathbf{x}}, \hat{\mathbf{D}}, \hat{\boldsymbol{\alpha}}) = \underset{\mathbf{D},\mathbf{x},\boldsymbol{\alpha}}{\text{Arg min}} \lambda||\mathbf{x}-\mathbf{y}||_2^2 \ + \ \sum_{i,j}\mu_{ij}||\alpha_{ij}||_0 \ + \ \sum_{i,j}||\mathbf{D}\alpha_{ij} - \mathbf{R}_{ij}\mathbf{x}||_2^2 \tag{10.12}$$

which groups all the quantities that we have tried to minimize in the preceding paragraphs.
Let us briefly analyze this formula, even though the forthcoming comments are slightly redundant with the previous explanation :

- the first term controls the global proximity of $\hat{\mathbf{x}}$ to the noisy image $\mathbf{y}$ (fidelity term);
- the second term controls the sparsity of the representations of the patches;
- last, the third term controls for each (i,j), the proximity of the patch $\mathbf{R}_{ij}\hat{\mathbf{x}}$ of our reconstruction to the denoised patch $\mathbf{D}\alpha_{ij}$.

The coefficients λ and μ_{ij} set the balance between the importance given to the fidelity term and to the sparsity constraints of the representations of the patches.
This non-convex problem is too difficult to be addressed in this form. This explains why the article [53] suggests to break it down into parts, and to try to minimize separately the different terms of (10.12). This way, we are led to the K-SVD algorithm. Notice also a serious difference: the values of μ_{ij} are not required in the above implementation.
Without specifying values for μ_{ij}, we cannot really address the problems of linking the minimization of (10.12) and the suggested iterative method. Moreover, we do not understand why the authors did not set only one weight μ rather than weights μ_{ij} depending on the patches. We would have to explain why the sparsity of certain patches are more important than others. If the μ_{ij} are not equal, then their determination is still a crucial point of the method that remains to be analyzed.
The alternation of sparse coding step and dictionary update step makes difficult the analysis of the aforementioned energies. On the one hand, the ORMP is only an approximate solution. On

the other hand, in the sparse coding step, the constraints are formed by parts of the Frobenius norm that is minimized in the dictionary update. For this reason, we want to insist on the fact that the minimization of (10.12) is nothing but a possible interpretation of the K-SVD method. Of course, solving directly the problem (10.12) is appealing but seems for now out of reach.
The reader could notice that, at each of the K iterations of the first two steps, the algorithm uses a SVD, thus explaining the name K-SVD. As stated in [1], the reference to K-means is not just formal : in K-means, we do not allow sparse combinations of the atoms, but we try to optimize the dictionary in such a way that the error committed by representing each observation with a single atom in the dictionary is minimal.

10.2.2 Extending K-SVD to Color Images

It is now time to present the method proposed in [93] to adapt the grayscale algorithm to color images. To address this problem, a first suggestion would be to apply the K-SVD algorithm to each channel R, G and B separately. This naive solution gives color artifacts that are shown on the left image of the figure 10.2. They are due to the fact that in natural images there is an important correlation between channels. Another suggestion would be to apply a principal component analysis on channels RGB, which would uncorrelate them, and then to apply the first suggestion in this more appropriate environment. This solution has not been tried because the new proposition of [93] seems even more promising.

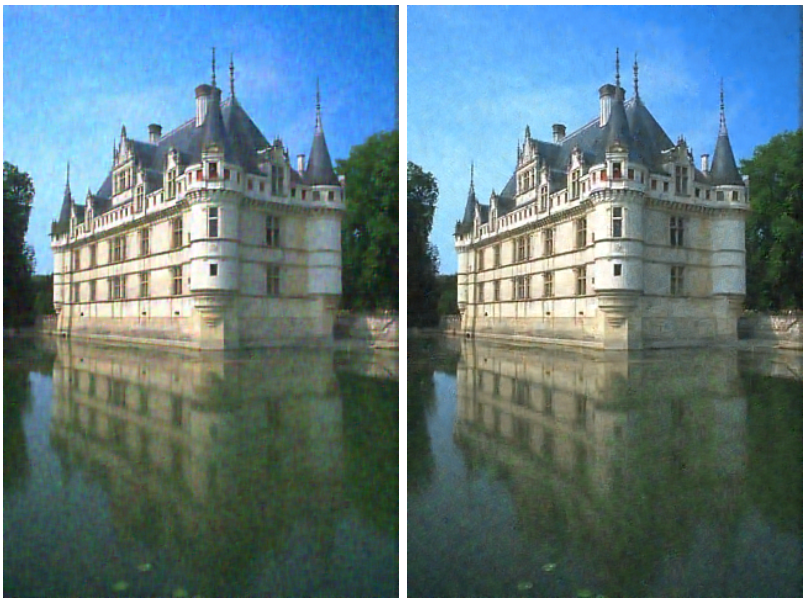


Figure 10.2: Denoised images with separated channels (left), and then concatenated channels (right). ($\sigma = 25$). The reader will notice that the denoising is better on the sky and the water surfaces.

In order to obtain the colors correctly, the algorithm previously described will be applied on column vectors which are the concatenation of the R,G,B values. In this way, the algorithm will better update the dictionary, because it is able to learn correlations which exist between color channels. An example of color dictionary is shown in figure 10.3.
One can see the difference in figure 10.2. We remind the reader that from now on the size of columns which represent images is $3N$, and the size of columns which represent patches is $3n$.
Unfortunately, even with this adaptation, non-negligible color artifacts are still present.
The authors of [93] justify these artifacts with the following statement : the previously described algorithm tries to adapt the dictionary to all patches contained in the image. This need of

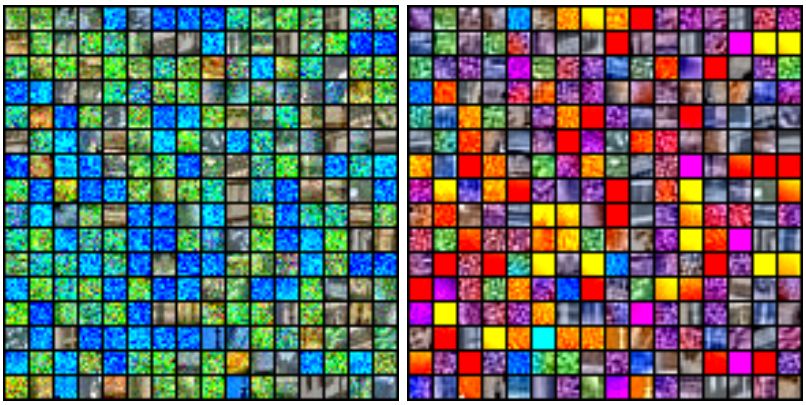


Figure 10.3: Left : dictionary composed by patches extracted randomly from the "Castle" image, on whom a white gaussian noise has been added. Right : dictionary obtained at the end of the color version of K-SVD. The contrast is enhanced independently for each atom.

universality implies that the atoms of the dictionary tend to look like grayscale atoms. To correct these color artifacts, [93] suggests to modify the metric used in the break condition of the ORMP. From now on we use the scalar product inferred metric

$$\langle y, x \rangle_\gamma = y^t x + \frac{\gamma}{n^2} y^t J^t J x \tag{10.13}$$

instead of the Euclidean metric, where we denote by J the matrix whose size is $3n \times 3n$ built from three diagonal blocks of size $n \times n$, full of 1, and where $\gamma \geq 0$ is parameter which needs to be fixed. In other words, the new norm can be written as

$$||x||_\gamma^2 = ||x||^2 + \gamma(m_R(x)^2 + m_G(x)^2 + m_B(x)^2) \tag{10.14}$$

where we denote by $m_{\mathcal{C}}(x)$ the average of x on the channel $\mathcal{C}$ (and where the Euclidean norm is denoted by $||\cdot||$).
Thus, the new metric, under the parameter γ, put more importance of the proximity of the mean value of the patches.
This color correction can be easily integrated in the ORMP thanks to the following equality :

$$I + \frac{\gamma}{n} J = \left(I + \frac{a}{n} J\right)^t \left(I + \frac{a}{n} J\right) \tag{10.15}$$

where $a > 0$ is chosen so that $\gamma = 2a + a^2$. Thus we can write for all vectors x,

$$||x||_\gamma = \left\| (I + \frac{a}{n} J)x \right\| \ . \tag{10.16}$$

Consequently, to work with the new metric, all columns have to be multiplied by $(I + \frac{a}{n} J)$ and we can work again with the Euclidean norm. Nevertheless we remind the reader that in the ORMP all columns of the dictionary are normalized, which is why a diagonal matrix $\mathcal{D}$ is introduced. Its elements are the inverses of the norm of the columns of $(I + \frac{a}{n} J)\mathbf{D}$. Its size is $k \times k$. Then $(I + \frac{a}{n} J)\mathbf{D}\mathcal{D}$ has normalized columns. Now the ORMP can be applied to obtain the $\hat{\beta_{ij}}$ such that

$$\left(I + \frac{a}{n} J\right) \mathbf{R}_{ij} \mathbf{y} \approx \left(I + \frac{a}{n} J\right) \mathbf{D}\mathcal{D}\hat{\beta_{ij}}$$

for the Euclidean norm. In the next session, if we denote

$$\hat{\alpha_{ij}} = \mathcal{D}\hat{\beta_{ij}} \ ,$$

we get

$$\mathbf{R}_{ij}\mathbf{y} \approx \mathbf{D}\hat{\alpha_{ij}}$$

for the norm $\|\cdot\|_\gamma$.
One can notice the contribution of this color version in the figure 10.4. Here again has appeared a new parameter γ which will be briefly discussed in the following part.

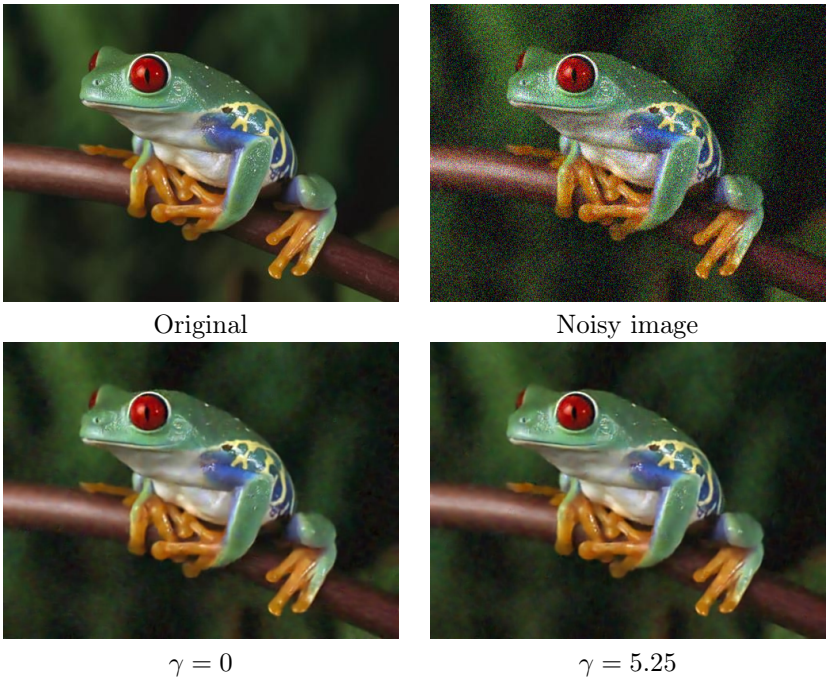


Figure 10.4: Denoising for $\sigma = 30$ with $\gamma = 0$ and $\gamma = 5.25$. Some color artifacts still remain, but the denoising is slightly better in some areas when $\gamma = 5.25$, cf figure 10.5.

10.2.3 Summary of the Algorithm

In this section all steps of the algorithm 17 are summarized in their right order.

10.3 Influence of the Parameters on the Performance

One can notice that the algorithm as described previously has plenty of parameters that can be tuned. Here is the exhaustive list :

- C : multiplier coefficient;
- λ : weight of the noisy image;
- K : number of iterations;
- k : size of the dictionary;
- γ : color correction parameter;
- $\sqrt{n}$: size of patches.

The question is to pick the right values for the various parameters listed above, and to evaluate their influence on the final result.

Algorithm 17 K-SVD algorithm

Input : noisy image $\mathbf{y}$, initial dictionary $\mathbf{D}_{init}$ and parameters listed in the next part
Output : denoised image $\hat{\mathbf{x}}$
All patches of the noisy image are collected in column vectors $\mathbf{R}_{ij}\mathbf{y}$ (channels R, G and B are concatenated).
Set initially $\hat{\mathbf{D}} = \mathbf{D}_{init}$.
for $k = 1, \cdots K$ **do**
Sparse coding
The inverses of the norms of the columns of $(I + \frac{a}{n}J)\hat{\mathbf{D}}$ are put in a diagonal matrix $\mathcal{D}$
An ORMP is applied to the vectors $(I + \frac{a}{n}J)\mathbf{R}_{ij}\mathbf{y}$ with the dictionary $(I + \frac{a}{n}J)\mathbf{D}\mathcal{D}$ in a such way that sparse coefficients $\hat{\beta_{ij}}$ for Euclidean norm are obtained, such that:

$$\left(I + \frac{a}{n}J\right)\mathbf{R}_{ij}\mathbf{y} \approx \left(I + \frac{a}{n}J\right)\mathbf{D}\mathcal{D}\hat{\beta_{ij}}.$$

Deduce $\hat{\alpha_{ij}} = \mathcal{D}\hat{\beta_{ij}}$ (sparse too) which then verify $\mathbf{R}_{ij}\mathbf{y} \approx \mathbf{D}\hat{\alpha_{ij}}$ for the norm $\|\cdot\|_\gamma$.
Dictionary update
for $l = 1, \cdots, k$ **do**
Introduce $\omega_l = \{\,(i,j)\,|\,\hat{\alpha_{ij}}(l) \neq 0\,\}$.
for $(i,j) \in \omega_l$ **do**
Obtain the residue $e^l_{ij} = \mathbf{R}_{ij}\mathbf{y} - \hat{\mathbf{D}}\hat{\alpha_{ij}} + \hat{d}_l\hat{\alpha_{ij}}(l)$.
end for
Put these column vectors together in a matrix $\mathbf{E}_l$. Values $\hat{\alpha_{ij}}(l)$ are also assembled in a row vector denoted by $\hat{\alpha}^l$ for $(i,j) \in \omega_l$.
Update $\hat{d}_l$ and $\hat{\alpha}^l$ as solutions of the minimization problem:

$$(d_l, \hat{\alpha}^l) = \underset{d_l, \alpha^l}{\text{Arg min}} \|\mathbf{E}_l - d_l\alpha^l\|^2_F.$$

In practice a truncated SVD is applied to the matrix $\mathbf{E}_l$. It provides partially U, V (orthogonal matrices) and Δ (filled in with zeroes except on its first diagonal), such that $\mathbf{E}_l = U\Delta V^T$. Then $\hat{d}_l$ is defined again as the first column of U and $\hat{\alpha}^l$ as the first column of V multiplied by $\Delta(1,1)$.
end for
end for
Then the final result $\hat{\mathbf{x}}$ is obtained thanks to a weighting aggregation (the formula has already been explained):

$$\hat{\mathbf{x}} = \left(\lambda\mathbf{I} + \sum_{i,j}\mathbf{R}^t_{ij}\mathbf{R}_{ij}\right)^{-1}\left(\lambda\mathbf{y} + \sum_{i,j}\mathbf{R}^t_{ij}\hat{\mathbf{D}}\hat{\alpha_{ij}}\right).$$

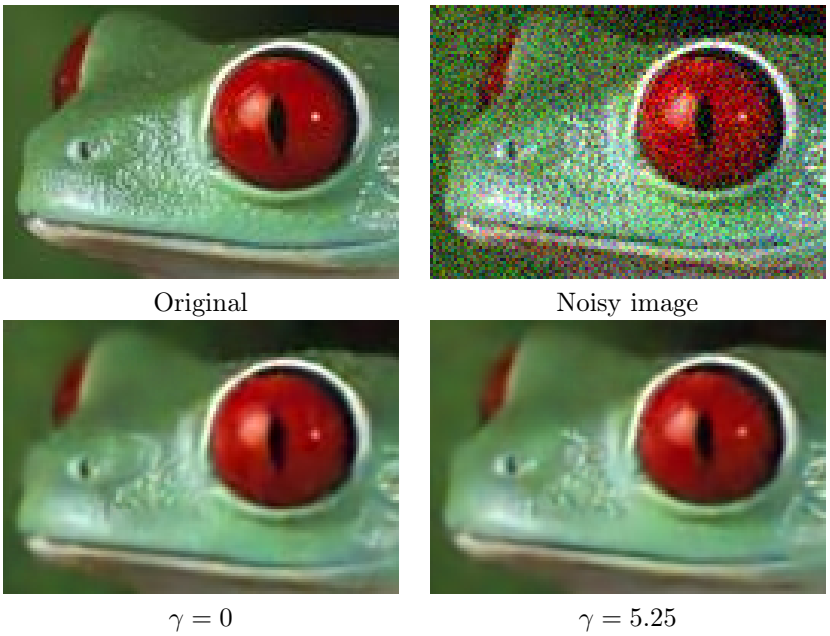


Figure 10.5: Denoising for $\sigma = 30$ with $\gamma = 0$ and $\gamma = 5.25$. Zooms.

10.3.1 Influence of C

This parameter is used in the stopping condition of the ORMP. In order to understand the chosen value, let us get started with a clean patch x_0 (where the length of the column is denoted by $\tilde{n} = n$ (resp. $\tilde{n} = 3n$) for grayscale (resp. color) images), on which a white Gaussian noise w is added to obtain a noisy patch x. Then the ORMP tries to find a vector α as sparse as possible such that

$$||x - D\alpha||_2 \leq \sqrt{\tilde{n}}C\sigma \ .$$

If the noise has norm lower than $\sqrt{\tilde{n}}C\sigma$, then x will be in the x_0-centered sphere, which radius is $\sqrt{\tilde{n}}C\sigma$. If we assume that x_0 is the only element of this sphere to have a sparse representation in the dictionary D, then one can suppose that the ORMP will be able to find this x_0. Then we will ensure that the noise has a large probability to belong to this sphere.
Thus the idea of [93] is to force

$$\mathbb{P}(||w||_2 \leq \sqrt{\tilde{n}}C\sigma) = 0.93 \ . \tag{10.17}$$

Practically, the corresponding value is obtained by using the inverse of the distribution function of $\chi^2(\tilde{n})$.

10.3.2 Influence of the Weighting *parameter* λ

The weighting parameter λ is used during the final reconstruction of the denoised image.
If for all $\lambda \geq 0$ we denote by $\hat{\mathbf{x}}_\lambda$ the final result of the algorithm as described previously using the parameter λ, then according to the definition of $\hat{\mathbf{x}}_\lambda$ (cf. formula (10.11)), one can notice that

$$\hat{\mathbf{x}}_\lambda = \frac{1}{\lambda + 1}(\lambda \mathbf{y} + \hat{\mathbf{x}}_0) \ .$$

If we want to remove from the noisy image the same quantity of energy than the one that was added by the noise, then it is natural to choose the parameter λ so that

$$\|\hat{\mathbf{x}}_\lambda - \mathbf{y}\| = \sqrt{\tilde{N}}\sigma \tag{10.18}$$

where $\tilde{N}$ is equal to N (resp. $3N$) for grayscale (resp. color) images. In other words the distance between $\hat{\mathbf{x}}_\lambda$ and $\mathbf{y}$ is forced to be exactly equal to $\sqrt{\tilde{N}}\sigma$. As $\hat{\mathbf{x}}_\lambda$ belongs to the segment $[\hat{\mathbf{x}}_0, \mathbf{y}]$, a such λ exists if and only if

$$d := \|\hat{\mathbf{x}}_0 - \mathbf{y}\| \geq \sqrt{\tilde{N}}\sigma \ .$$

In this case one can easily see that the only λ leading to the equality (10.18) is

$$\lambda = \frac{d}{\sqrt{\tilde{N}}\sigma} - 1 \ .$$

Despite this theoretical value, the algorithm has been tested with plenty of choices for λ. If λ is taken too large, then the contribution of the noisy image is too important and adds too much noise, which consequently reduces the PSNR, as one can see in the table 10.1.

σ	$\lambda = 0$	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.15$	$\lambda = 0.2$	$\lambda = 0.25$	$\lambda = 0.3$
2	44.43	**44.77**	44.70	44.52	44.32	44.13	43.97
5	39.03	**39.08**	38.97	38.78	38.57	38.34	38.12
10	34.55	**34.58**	34.53	34.42	34.28	34.13	33.96
20	30.47	**30.47**	30.44	30.38	30.30	30.20	30.08
30	**28.18**	**28.18**	28.15	28.10	28.03	27.96	27.86
40	**26.60**	26.58	26.55	26.50	26.45	26.38	26.30
60	**24.36**	24.33	24.29	24.25	24.20	24.14	24.07
80	**22.76**	22.73	22.69	22.64	22.59	22.54	22.48
100	**21.47**	21.43	21.38	21.34	21.28	21.23	21.17

Table 10.1: In **bold** the best result for a given σ. Other parameters are fixed to : $K = 15$; $\sqrt{n} = 5$; $\gamma = 5.25$; $k = 256$.

Some visual results are shown in figures 10.6, 10.7 and 10.8.
Table 10.2 shows the comparison between the empirically obtained parameter (λ_e) and the theoretically obtained parameter (λ_t).

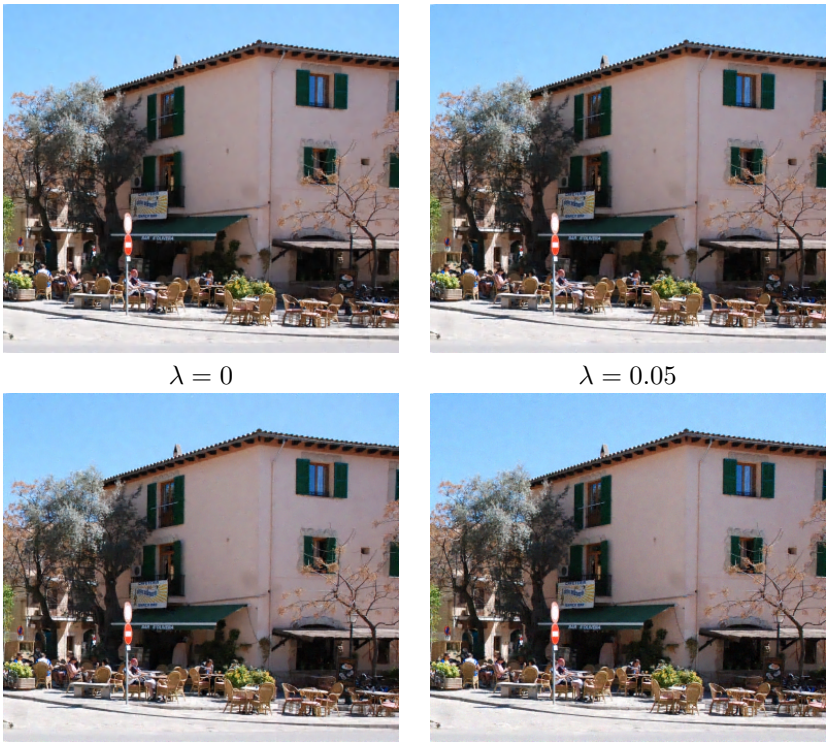
	λ_t			λ_e		
σ	PSNR	RMSE	value	PSNR	RMSE	value
5	**38.83**	**2.92**	0.0050	38.64	2.98	0.05
10	**34.24**	**4.95**	0.0078	34.10	5.03	0.05
20	**29.85**	**8.20**	0.012	29.84	8.21	0.05
30	27.64	10.58	0.013	**27.68**	**10.54**	0.05
40	26.08	12.66	0.014	**26.10**	**12.63**	0.0
60	**24.05**	**16.00**	0.018	24.00	16.08	0.0
80	22.78	18.52	0.017	**22.80**	**18.46**	0.0
100	**21.88**	**20.54**	0.019	21.84	20.63	0.0

Table 10.2: In **bold** the best result for a given σ. Other parameters are fixed to : $K = 15$; $\sqrt{n} = 5$; $\gamma = 5.25$; $k = 256$.

In the end the final kept value for λ is the one given by (10.18).

10.3.3 Influence of the Number of Iterations K

The iterative aspect of the method is important, because it allows the dictionary to be updated and then to obtain a better sparse representation of the patches of the image. Moreover it allows to show empirically the convergence of the method. Indeed when K is large enough further



$\lambda = 0$ $\lambda = 0.05$

$\lambda = 0.15$ $\lambda = 0.25$

Figure 10.6: $\sigma = 10$

iterations should improve the dictionary only marginally. Depending on the convergence of the method (which can change according to σ), a huge number of iterations is assumed to be needed in order to assure the best possible estimate. On an other side, each iteration is really expensive in terms of processing time. Thus avoiding spurious iterations allows one to obtain a faster algorithm. In consequence the main goal is to obtain a good compromise between having enough iterations to obtain a good result close to the optimum and having a correct processing time.
Table 10.3 shows the PSNR and RMSE evolutions depending on the number of iterations.
One can notice that for $\sigma \geq 5$ the PSNR converges, and the higher σ, the faster the convergence of the PSNR. Thus it is possible to keep few iterations for high values of noise.
In order to better illustrate the speed of the PSNR convergence in function of K and σ, figure 10.9 shows $f(PSNR(i))$ according to the number of iterations i, where f is defined by

$$f(x_i) = \frac{x_i - x_0}{x_m}$$

with $x_m = \max(x_i - x_0)$.
In the following, the number of iterations will therefore be fixed to $K = 15$, no matter what σ.

10.3.4 Influence of the Size of the Dictionary k

The only constraint on the size of the dictionary is to generate $\mathbb{R}^n$. As we want some redundancy, we set $k \geq n$. Table 10.4 contains a study about this parameter.
According to this table one can see that it might be interesting to choose larger sizes for the dictionary for relatively small noise ($\sigma \leq 30$), and smaller sizes for high noise ($\sigma \geq 60$). Although this parameter has an influence on the processing time, it remains relatively flexible according to PSNR results. In the following, this parameter will therefore be fixed to $k = 256$.

$\lambda = 0$ $\lambda = 0.05$

$\lambda = 0.15$ $\lambda = 0.25$

Figure 10.7: $\sigma = 30$

$K \backslash \sigma$	2	5	10	20	30	40	60	80	100
1	44.26	38.13	33.78	29.57	27.19	25.67	23.24	21.61	20.32
2	44.38	38.41	34.18	30.09	27.80	26.26	23.92	22.38	21.11
3	44.47	38.56	34.34	30.24	27.96	26.39	24.06	22.51	21.24
4	44.49	38.63	34.39	30.29	28.00	26.44	24.11	22.55	21.28
5	44.53	38.65	34.41	30.30	28.02	26.47	24.14	22.58	21.31
6	44.54	38.67	34.42	30.32	28.04	26.48	24.15	22.58	21.32
7	44.54	38.69	34.42	30.33	28.04	26.49	24.16	22.59	21.34
8	44.53	38.70	34.44	30.33	28.06	26.50	24.18	22.60	21.34
9	44.57	38.71	34.45	30.35	28.07	26.51	24.19	22.61	21.35
10	44.59	38.72	34.45	30.36	28.08	26.52	24.19	22.61	21.35
11	44.39	38.73	34.46	30.37	28.08	26.53	24.19	22.61	21.35
12	44.41	38.73	34.47	30.38	28.09	26.53	24.20	22.61	21.35
13	44.65	38.73	34.48	30.39	28.10	26.54	24.20	22.62	21.35
14	44.57	38.75	34.48	30.40	28.10	26.54	24.21	22.62	21.35
15	44.40	38.75	34.49	30.40	28.11	26.55	24.21	22.63	21.35
16	44.45	38.75	34.50	30.41	28.12	26.55	24.21	22.63	21.35
17	44.35	38.75	34.51	30.42	28.12	26.55	24.21	22.64	21.35
18	44.36	38.76	34.51	30.42	28.13	26.56	24.22	22.64	21.36
19	44.55	38.77	34.52	30.42	28.13	26.56	24.22	22.64	21.36
20	44.59	38.77	34.53	30.43	28.13	26.56	24.22	22.64	21.36

Table 10.3: Other parameters are fixed to : $\sqrt{n} = 5$; $\gamma = 5.25$; $\lambda = 0.15$; $k = 256$.

$\lambda = 0$ $\lambda = 0.05$

$\lambda = 0.15$ $\lambda = 0.25$

Figure 10.8: $\sigma = 80$

$\sigma \backslash k$	128	196	k=256	320
2	44.13	44.40	44.66	**44.70**
5	38.44	38.69	38.73	**38.76**
10	34.32	34.41	34.45	**34.50**
20	30.32	30.37	30.42	**30.44**
30	28.07	28.08	28.10	**28.10**
40	26.54	**26.55**	26.53	26.54
60	**24.37**	24.32	24.30	24.26
80	**22.73**	22.66	22.61	22.57
100	**21.48**	21.48	21.32	21.27

Table 10.4: In **bold** the best result for a given σ. Other parameters are fixed to : $K = 15$; $\sqrt{n} = 5$; $\gamma = 5.25$; $\lambda = 0.15$.

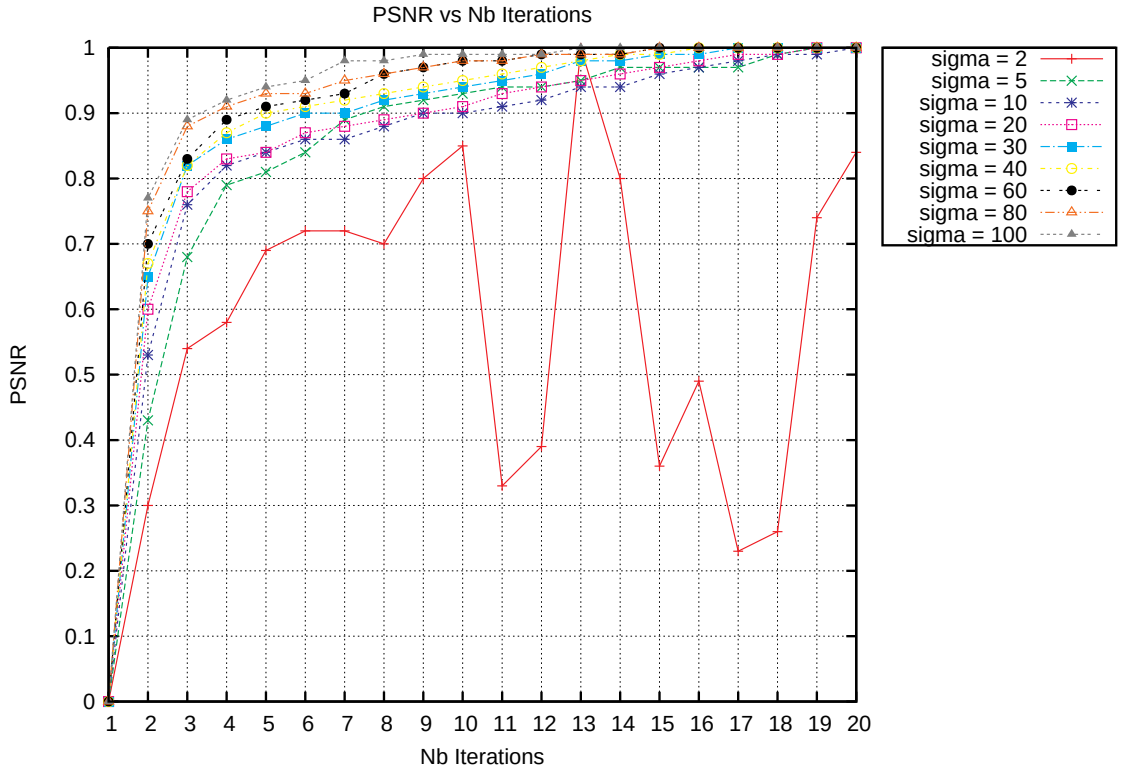


Figure 10.9: PSNR *vs* number of iterations

10.3.5 Influence of the Correction Parameter γ

The parameter γ is only used in the case of color images. We will see that this empirical parameter is quite flexible, because some low variations on its value have almost no consequences on the final result.

$\sigma \backslash \gamma$	3.5	4.5	4.75	5	5.25	5.5	5.75	6	7
2	44.24	**44.67**	44.66	44.64	44.66	44.58	**44.67**	44.64	44.66
5	38.75	**38.77**	38.75	38.75	**38.77**	38.76	**38.77**	**38.77**	38.74
10	34.48	34.50	34.47	34.49	34.50	34.49	34.49	34.49	**34.51**
20	**30.39**	**30.39**	**30.39**	30.37	30.35	30.37	30.36	30.37	30.36
30	**28.10**	28.08	28.09	**28.10**	28.09	28.07	28.08	28.09	28.07
40	26.53	26.52	26.51	26.52	26.50	**26.54**	26.50	26.51	26.51
60	24.29	24.27	24.28	24.27	24.26	24.24	24.27	**24.29**	24.26
80	22.68	**22.70**	22.68	22.64	22.69	22.67	22.69	22.66	22.68
100	**21.40**	21.36	21.37	21.38	21.37	21.38	21.37	21.37	21.36

Table 10.5: In **bold** the best result for a given σ. Other parameters are fixed to : $K = 15$; $\sqrt{n} = 5$; $\lambda = 0.15$; $k = 256$.

In the following and according to the original article the correction parameter will therefore be fixed to $\gamma = 5.25$.

10.3.6 Influence of the Size of the Patches $\sqrt{n}$

The size of the patches has a huge influence on the final result, and we can win several decibels in PSNR by choosing an appropriate n. As for most of the patch-based denoising method, best results are obtained by working with relatively big patches, as seen in table 10.6.

$\sigma \backslash \sqrt{n}$	3	5	7	9	11	13	15
5	38.52	**39.07**	38.87	38.55	38.42	38.12	36.80
10	33.87	**34.65**	34.45	34.18	33.89	33.62	33.37
20	29.27	**30.52**	30.32	30.04	29.74	29.48	29.26
30	26.46	**28.19**	28.11	27.78	27.48	27.20	26.99
40	24.40	**26.56**	26.55	26.24	25.90	25.60	25.35
60	21.51	24.42	**24.66**	24.37	24.06	23.69	23.41
80	19.30	22.72	**23.33**	23.15	22.87	22.59	22.28
100	17.56	21.47	**22.39**	22.38	22.13	21.84	21.56

Table 10.6: In **bold** the best result for a given σ. Other parameters are fixed to : $K = 15$; $k = 256$; $\gamma = 5.25$; $\lambda = 0$ if $\sigma > 0$, 0.05 otherwise.

Similarly to other patch-based denoising method (for example BM3D), it is necessary to increase the size of the patches when the noise increases.
Despite the fact that according to PSNR/RMSE results it seems better to take relatively small patches ($\sqrt{n} = 5$ or 7) for small values of noise, we have to take into consideration the visual result.
Visual results for several values of the noise and for all studied patch sizes are shown in figures 10.10, 10.11, 10.12, and 10.13.
One can notice that visually the choice is not so easy. Too small patches give huge artifacts, and leads to many low frequency fluctuations, but with big patches almost all details are lost. We get a visually nicer image, but completely blurred.

Noisy image

$\sqrt{n} = 3$

$\sqrt{n} = 5$

$\sqrt{n} = 7$

$\sqrt{n} = 9$

$\sqrt{n} = 11$

$\sqrt{n} = 13$

$\sqrt{n} = 15$

Figure 10.10: $\sigma = 10$

Noisy image

$\sqrt{n} = 3$

$\sqrt{n} = 5$

$\sqrt{n} = 7$

$\sqrt{n} = 9$

$\sqrt{n} = 11$

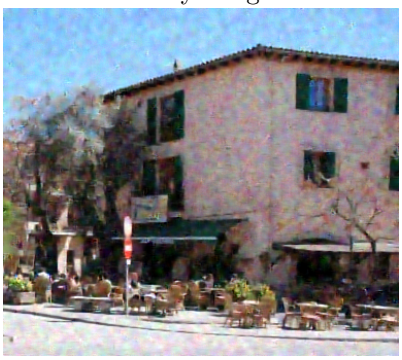
$\sqrt{n} = 13$

$\sqrt{n} = 15$

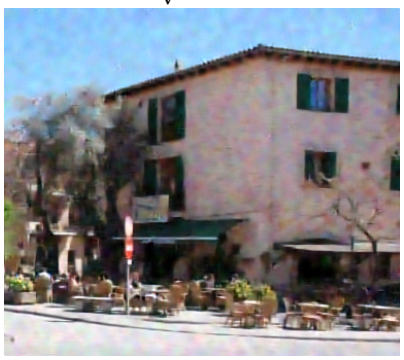
Figure 10.11: $\sigma = 30$

Noisy image

$\sqrt{n} = 3$



$\sqrt{n} = 5$



$\sqrt{n} = 7$

$\sqrt{n} = 9$

$\sqrt{n} = 11$

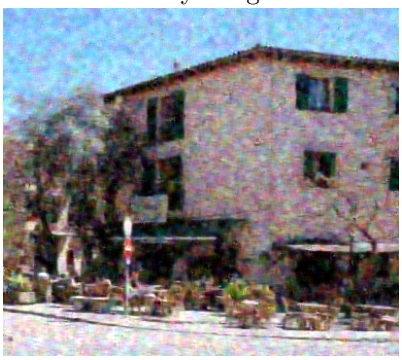
$\sqrt{n} = 13$

$\sqrt{n} = 15$

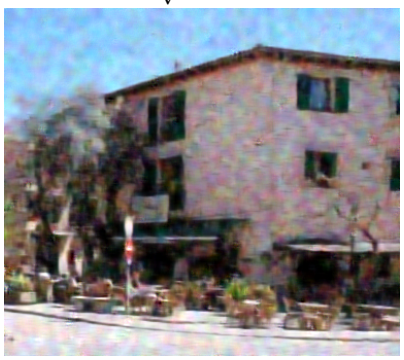
Figure 10.12: $\sigma = 60$

Noisy image

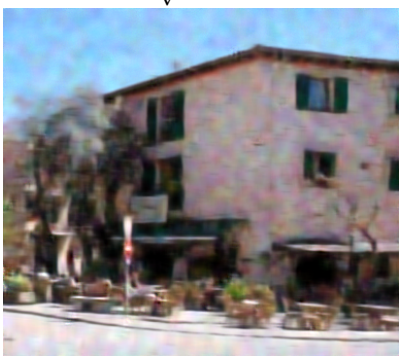
$\sqrt{n} = 3$



$\sqrt{n} = 5$



$\sqrt{n} = 7$



$\sqrt{n} = 9$

$\sqrt{n} = 11$

$\sqrt{n} = 13$

$\sqrt{n} = 15$

Figure 10.13: $\sigma = 100$

In conclusion a compromise has to be found, which cannot only be chosen according to the PSNR/RMSE results, but also taken into account the visual aspect. The values of n which will therefore be kept are

σ	$0 < \sigma \leq 20$	$20 < \sigma \leq 60$	$60 < \sigma$
$\sqrt{n}$	5	7	9

10.4 A Detailed Study of Possible Variants

10.4.1 Origin of the Initial Dictionary

In the original article, and in the previously described algorithm, the dictionary is initialized by taking randomly k patches in the noisy image.
Despite the fact that the method works well in this way, one can wonder whether there would be a better way to initialize the dictionary. For example by taking random patches from a noise-free image.
Let us denote by $\mathbf{Init}_0$ the dictionary initialized on the original noiseless image; by $\mathbf{Init}_1$ the dictionary initialized on the original noisy image, by $\mathbf{Init}_2$ the dictionary initialized on a noise-free reference image.

σ	$\mathbf{Init}_0$	$\mathbf{Init}_1$	$\mathbf{Init}_2$
5	**38.80**	38.76	38.65
10	**34.47**	**34.42**	34.29
20	**30.40**	30.35	30.28
30	28.05	**28.09**	27.87
40	26.44	**26.51**	26.27
60	24.21	**24.25**	24.21
80	22.64	**22.65**	22.61
100	21.29	**21.31**	21.28

Table 10.7: In **bold** the best result for a given σ. Other parameters are fixed to : $K = 15$; $\sqrt{n} = 5$; $\gamma = 5.25$; $k = 256$; $\lambda = 0.15$.

One can think that the initialization of the dictionary is quite important (because we run the algorithm with few number iterations, so the maximum is not reached), because depending on the initialization we have variations of more than 0.1dB. But when σ increases, one observes less variation in the results. An explanation might be that the number of iterations K is then more appropriate, so we are close to optimality, and the initialization is not really crucial.
In conclusion the initialization of the dictionary is not crucial, and the initialization by taking random patches from the noisy image is quite good.

10.4.2 Training of the Dictionary

Because the algorithm can hardly take into account parallel instructions (at least the update of the dictionary), the algorithm as previously described in this article is extremely slow. Its processing time is directly proportional to the size of the dictionary[2] as well as to the number of patches containing in the image (then to the size of the image itself) and to the size of the patches.
If obviously we cannot reduce the size of the image and if we cannot modify the size of the patches without highly damaging the final result, it is still possible to reduce the number of patches used during the training part of the dictionary, by applying the following principle :

1. The set of patches is built on the whole image;

[2]Although the study shows that the size of the dictionary can be reduced without affecting the result too much.

$\sigma \backslash T$	1	2	4	8	12	16	20
5	38.84	38.86	38.89	38.90	38.89	38.91	**38.92**
10	34.47	34.49	34.51	34.49	34.52	**34.55**	34.50
20	30.28	**30.31**	30.28	**30.31**	30.29	30.29	**30.31**
30	**28.05**	**28.05**	28.04	28.04	28.04	28.01	28.01
40	26.61	**26.63**	**26.63**	26.59	26.59	26.56	26.58
60	**24.69**	24.65	24.65	24.60	24.56	24.56	24.52
80	**23.28**	23.23	23.22	23.14	23.07	23.00	23.03
100	**22.28**	22.25	22.18	22.12	22.07	22.04	22.00

Table 10.8: In **bold** the best result for a given σ. Parameters are fixed to : $K = 15$; $\sqrt{n} = 7$; $\gamma = 5.25$; $k = 256$; $\lambda = 0.05$.

2. Keep one patch out of T to build a T times smaller patch set;

3. Apply the loop on the ORMP and the update of the dictionary by SVD K times on this sub-set, in order to obtain a final dictionary D_f;

4. Then apply with only one iteration the whole algorithm on the initial full set of patches, but with D_f as previously obtained.

With this simple trick it is then possible to divide the processing time by slightly less[3] than T. Before applying this trick, we have to determinate its impact on the final result, in order to find the more appropriate value of T for each σ.
We have seen during the study of the parameters that the result in PSNR for $\sigma = 2$ is highly chaotic depending on the number of iterations K. For that reason we do not present results for this particular value of noise.
Table 10.8 shows a summary for some values[4] of T.
This study shows that it is possible to highly reduce the processing time of this method whilst keeping a result close to the original method.
According to the obtained results, it seems reasonable to take $T = 16$ for $\sigma \leq 40$ and $T = 8$ for $\sigma > 40$.
In order to help the readers to make up their own idea concerning the gain in term of processing time with this trick, table 10.9 shows the processing time in seconds for a $512 \times 512 \times 3$ image on a i5 processor with 8Go of Ram[5].

σ	5	10	20	30	40	60	80	100
$T = 1$	1306	446	213	165	152	141	138	137
T tabulated	140	53	28	23	22	29	29	28

Table 10.9: Processing time.

Thanks to this trick, we obtain reasonable processing time for $\sigma \geq 10$. Moreover we can decrease this time to 112 seconds (resp. 42s.) for $\sigma = 5$ (resp. $\sigma = 10$) by taking $T = 32$, without decreasing the PSNR. But we cannot decrease the processing time more, because we have to process a single iteration of the full set of patches, which is mainly responsible for the processing time.
One can be surprised by the fact that the processing time is decreasing with respect to σ. But it can be easily explained :

[3] We have to apply at the end a single iteration on the whole set of patches, which can be slower than the previous 15 ones on the sub-set.
[4] $T = 1$ represents the initial algorithm as described in this article without any modifications.
[5] Moreover the process of the ORMP is fully parallelized.

- For very small values of noise, it is quite complex to get a sparse representation of the patches since they are very different from one another. Then at the end of the ORMP we have to process a large matrix;
- On the contrary for very high noise the signal is covered by the noise, then patches are very similar. Thus it is easier to get a sparse representation of them, and then at the end of the ORMP the matrix is even smaller.

10.5 Conclusion

In this chapter, we have proposed a detailed analysis of the K-SVD algorithm, already introduced in the articles [53] and [93]. Through this explanation, we showed why we could expect remarkable denoising results from this algorithm. But we also noticed immediately the difficulty of the related optimization problems.
In a numerical way, we have observed the stability of this method, but we also brought up its heavy computational cost. In spite of these drawbacks, our experiments have clarified the impact of the different parameters on the result, and thus we have proposed reliable values to tune some of them. Moreover, we showed some denoising experiments which prove that the K-SVD method leads to good results, both in terms of PSNR values and of visual quality. The skeptical reader can pursue our experiments by applying the proposed demonstration to the images of her choice. Finally, the suggested modification (taking into account only a subset of the patches of the image) seems to get similar results with an interesting reduction of the execution time.
In conclusion, the K-SVD method can be considered to be part of the state of the art. But, above all it has to be seen as a first successful use of dictionary learning to address an image processing task. The more recent algorithms of this field, in particular those which replace the l^0-sparsity constraint by a l^1 constraint (cf. [91]), seem very promising. They lead to a great gain in computational time, and therefore allow one to handle bigger images.

Chapter 11

A Detailed Analysis and Implementation of BM3D

A brief description of the well known denoising method BM3D was presented in chapter 4. This state of the art method is emblematic of patch-based methods. It was (and still is) the absolute reference for comparisons to any new denoising algorithm. For those reasons, this chapter will focus on a detailed description of its theory and practicable implementation. Its parameters will also be precisely analysed. Finally, we produced a reliable and open-source implementation of this method.

11.1 Introduction

BM3D is a recent denoising method based on the fact that an image has a locally sparse representation in transform domain [35]. This sparsity is enhanced by grouping similar 2D image patches into 3D groups. In this chapter, we discuss the choice of all parameter methods and confirm their actual optimality. The description of the method is rewritten with a a more transparent notation that in the original paper.
Collaborative filtering is the name of the BM3D grouping and filtering procedure. It is realized in four parts or substeps: a) finding the image patches similar to a given image patch and grouping them in a 3D block b) 3D linear transform of the 3D block; c) shrinkage of the transform spectrum coefficients; d) inverse 3D transformation. This 3D filter therefore filters out simultaneously all 2D image patches in the 3D block.
By attenuating the noise, collaborative filtering reveals even the finest details shared by the grouped patches. The filtered patches are then returned to their original positions. Since these patches overlap, many estimates are obtained which need to be combined for each pixel. *Aggregation* is a particular averaging procedure used to take advantage of this redundancy.
The first collaborative filtering sub-step is much improved by a a second step using Wiener filtering. This second step mimics the first step, with two differences. The first difference is that it compares the filtered patches instead of the original patches. The second difference is that the new 3D group (built with the unprocessed image samples, but using the patch distances of the filtered image) is processed by Wiener filtering instead of a mere threshold. The final aggregation sub-step is identical to those of the first step.
The proposed method improved on the *NL-means* [13] method which denoises jointly similar patches, but only by performing a patch average, which amounts to a 1D filter in the 3D block. The 3D filter in BM3D is performed on the three dimensions simultaneously.
The BM3D algorithm detailed here directly comes from the original article [35]. It is generally considered to achieve the best performance bounds in color image denoising. Nevertheless, the authors have pointed out to us more recent and sophisticated versions. Like for NL-means, there is a variant with shape-adaptive patches [36]. In this algorithm denominated BM3D-SAPCA,

the sparsity of image representation is improved in two aspects. First, it employs image patches (neighbourhoods) which can have data-adaptive shape. Second, the PCA bases are obtained by eigenvalue decomposition of empirical second-moment matrices that are estimated from group of similar adaptive-shape neighbourhoods. This method improves BM3D especially in preserving image details and introducing very few artifacts. The anisotropic shape-adaptive patches are obtained using the 8-directional LPA-ICI techniques [69]. The very recent development of BM3D is presented in [68], [39], where it is generalized to become a generic image restoration tool, including deblurring.
A previous analysis on BM3D has been made in [64], in order to resolve the problem that the denoising performance has a sharp drop when noise standard deviation reaches 40. On the contrary of this present study, only few parameters have been studied, and only for large value of noise. Moreover [64] presents a less detailed study of BM3D than the one of the present article. For large value of noise, we reach the same conclusions: the threshold value during the first step needs to be increased, and the best results are achieved by using a 2D bi-orthogonal spline wavelet (denoted by 2D-Bior 1.5 in the following) during the second step and by keeping a 8×8 size for patches in both steps.
It is furthermore interesting to notice that recent papers like [23] and [86] which try to evaluate the inherent bounds of patch-based denoising methods claims that BM3D is really close to those optimality bounds.

11.2 The Algorithm Step by Step

11.2.1 Architecture of the Algorithm

We shall first describe how to process grey level images. The extension to color images will be explained later on. In all the following we work in the case of white Gaussian noise where the variance is denoted by σ^2.
The algorithm is divided in two major steps:

1. The first step estimates the denoised image using hard thresholding during the collaborative filtering. Parameters in this step are denoted by the exponent **hard**;

2. The second step is based both on the original noisy image, and on the basic estimate obtained in the first step. It uses Wiener filtering. The second step is therefore denoted by the exponent **wiener**.

The algorithm is summarized in the two next figures 11.1 and 11.2.

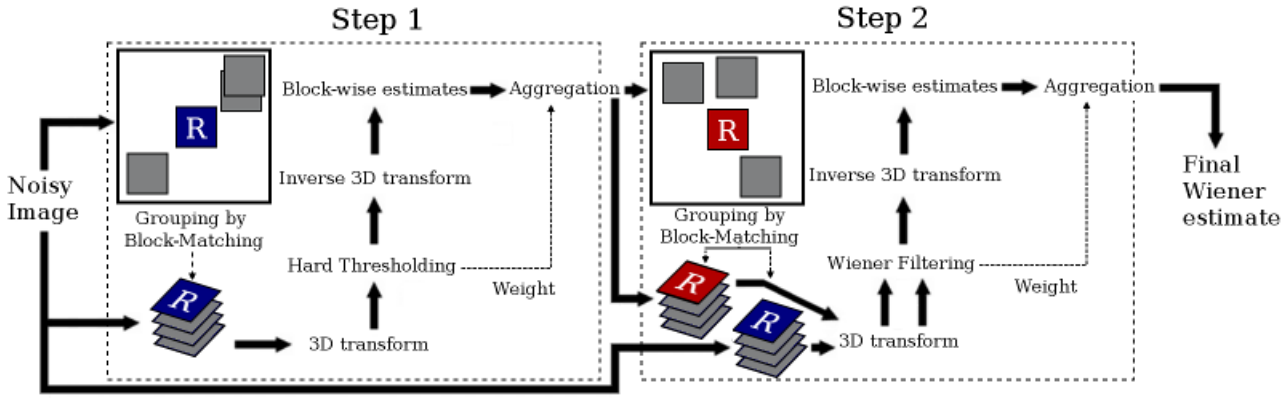


Figure 11.1: Scheme of the BM3D algorithm. [35]

Figure 11.3 shows the patches, a search window centered on reference patch P and illustrates the patch overlapping leading to multiple estimates.

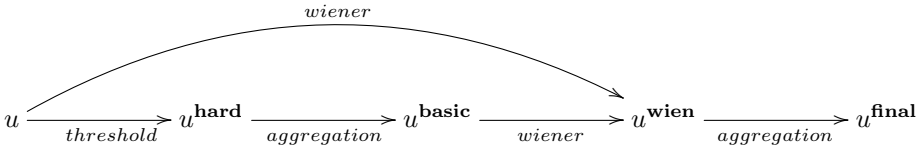


Figure 11.2: Notations in BM3D

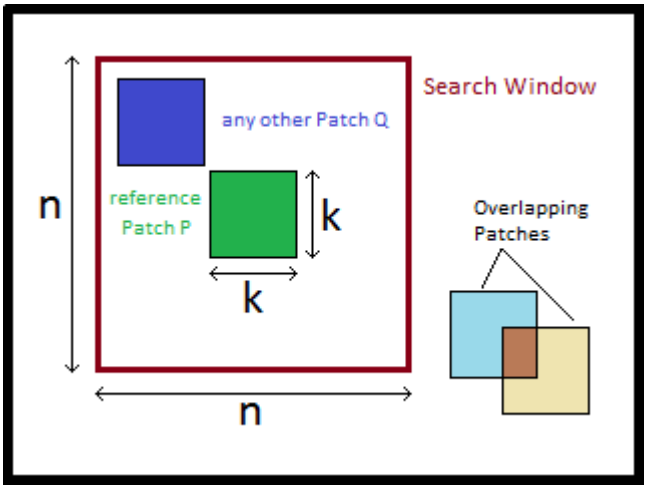


Figure 11.3: Patches, search window and overlapping.

11.2.2 The First Denoising Step

We denote by P the reference current patch whose size is $k^{\mathbf{hard}} \times k^{\mathbf{hard}}$ of the image loop.

1. *Grouping:*
   The first substep is grouping. The original noisy image u is searched in a P-centered $n^{\mathbf{hard}} \times n^{\mathbf{hard}}$ neighbourhood for patches Q similar to the reference patch P. The set of similar patches is simply defined by

$$\mathcal{P}(P) = \{Q : d(P,Q) \leq \tau^{\mathbf{hard}}\} \tag{11.1}$$

   where:

   - $\tau^{\mathbf{hard}}$ is the distance threshold for d under which two patches are assumed similar;
   - $d(P,Q) = \dfrac{\|\gamma'(P) - \gamma'(Q)\|_2^2}{(k^{\mathbf{hard}})^2}$ is the normalized quadratic distance between patches;
   - γ' is a hard thresholding operator with threshold $\lambda_{2D}^{\mathbf{hard}}\sigma$. For $\sigma \leq 40$ one has $\lambda_{2D}^{\mathbf{hard}}\sigma = 0$ (see detailed explanation). It simply puts to zero the coefficients of the patch with an absolute value below the threshold $\lambda_{2D}^{\mathbf{hard}}\sigma$. The other coefficients are unchanged. For $\sigma \leq 40$ all coefficients are unchanged;
   - σ^2 is the variance of the zero-mean Gaussian noise.

   The 3D group, denoted $\mathbb{P}(P)$, is then built by stacking up the matched patches $\mathcal{P}(P)$. In order to speed up the process, only[1] the $N^{\mathbf{hard}}$ patches in $\mathcal{P}(P)$ that are closest to the

[1]Moreover, when the applied 1D transform $\tau_{1D}^{\mathbf{hard}}$ is a Walsh-Hadamard transform it is necessary to have a power of 2 for the number of similar patches. Then $N^{\mathbf{hard}}$ is always chosen as a power of 2, and if there are less similar patches than $N^{\mathbf{hard}}$ for a given reference patch, only a power of 2 inferior or equal to this number of similar patches will be kept.

reference patch are kept in the 3D group. Patches in $\mathcal{P}(P)$ will be sorted according to their distance to P, in order to take the best $N^{\mathbf{hard}}$ similar patches easily. Thus naturally the first patch will be P because its distance to itself is zero. The order of the patches in the 3D group is not important : the results are similar no matter whether they are ordered according to their distance to the reference patch, or just randomly.

2. *Collaborative Filtering:*
Once the 3D-block $\mathcal{P}(P)$ is built the collaborative filtering is applied. A 3D isometric linear transform is applied to the group, followed by a shrinkage of the transform spectrum. Finally the inverse linear transform is applied to estimate for each patch

$$\mathbb{P}(P)^{\mathbf{hard}} = {\tau_{3D}^{\mathbf{hard}}}^{-1}(\gamma(\tau_{3D}^{\mathbf{hard}}(\mathbb{P}(P)))) \tag{11.2}$$

where γ is a hard thresholding operator with threshold $\lambda_{3D}^{\mathbf{hard}}\sigma$:

$$\gamma(x) = \begin{cases} 0 & \text{if} \quad |x| \leq \lambda_{\mathbf{3D}}^{\mathbf{hard}}\sigma \\ x & \text{otherwise} \end{cases}$$

For practical purposes, the 3D transform $\tau_{3D}^{\mathbf{hard}}$ of the 3D group $\mathcal{P}(P)$ is made up of two transforms : a 2D transform denoted by $\tau_{2D}^{\mathbf{hard}}$ applied on each patch of $\mathcal{P}(P)$, and a 1D transform denoted by $\tau_{1D}^{\mathbf{hard}}$ applied along the third dimension of the 3D group. The choice of these transforms will be carefully discussed later.

3. *Aggregation:*
When the collaborative filtering is done, we get an estimate for each used patch and then a variable number of estimates for every pixel. These estimates are saved in a buffer :

$$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + w_P^{\mathbf{hard}} u_{Q,P}^{\mathbf{hard}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + w_P^{\mathbf{hard}} \end{cases} \tag{11.3}$$

where:

- $\boldsymbol{\nu}$ (resp. $\boldsymbol{\delta}$) designates the numerator (resp. denominator) part of the basic estimate of the image obtained at the end of the step 1;
- $u_{Q,P}^{\mathbf{hard}}(x)$ is the estimate of the value of the pixel x belonging to the patch Q obtained during collaborative filtering of the reference patch P;
- $w_P^{\mathbf{hard}} = \begin{cases} \left(N_P^{\mathbf{hard}}\right)^{-1} & \text{if} \quad N_P^{\mathbf{hard}} \geq 1 \\ 1 & \text{otherwise} \end{cases}$
- $N_P^{\mathbf{hard}}$ is the number of retained (non-zero) coefficients in the 3D block after hard-thresholding: $\gamma\left(\tau_{3D}^{\mathbf{hard}}\left(\mathbb{P}(P)\right)\right)$.

The interest of this weighting is that it gives a priority to homogeneous patches (where there are many canceled coefficients). Patches containing an edge will be less taken into account than homogeneous ones on the border of the edge. Figure 11.4 illustrates this fact: priority is given to the green patches during the aggregation. The result is an artefact reduction around the edges. This results in an avoidance of the classic ringing effects observable with transform threshold methods. In order to reduce more the border effects which can appear, a $k^{\mathbf{hard}} \times k^{\mathbf{hard}}$ Kaiser window is used as part of the weights. It is simply done as a element-by-element multiplication between the Kaiser window and the estimated patch after the inverse 3D transformation. How the Kaiser window can be obtained will be discussed in the section 11.4.4.

The basic estimate after this first step is given by

$$u^{\mathbf{basic}}(x) = \frac{\displaystyle\sum_P w_P^{\mathbf{hard}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x) u_{Q,P}^{\mathbf{hard}}(x)}{\displaystyle\sum_P w_P^{\mathbf{hard}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x)} \tag{11.4}$$

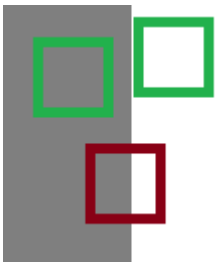


Figure 11.4: Green patches will have a weight superior to the red patch, because they are more sparse (have less nonzero transform coefficients).

which is simply obtained by dividing the two buffers (numerator and denominator) element-by-element, with $\chi_Q(x) = 1$ if and only if $x \in Q$, 0 otherwise.

11.2.3 The Second Denoising Step

In this second part of the algorithm a basic estimate $u^{\mathbf{basic}}$ of the denoised image is available. The second step performs a Wiener filter of the original image u, but uses as oracle the basic estimate $u^{\mathbf{basic}}$. It is observed in the experiments that this second step restores more details and improves the denoising performance, as will be clear in the experiments and tables below.

1. *Grouping:*
   The patch-matching is only processed on the basic estimate. When a set of similar patches

   $$\mathcal{P}^{\mathbf{basic}}(P) = \{Q : d(P,Q) \leq \tau^{\mathbf{wien}}\} \tag{11.5}$$

   has been obtained two 3D groups are formed:

   - $\mathbb{P}^{\mathbf{basic}}(P)$ by stacking up patches together from the basic estimation $u^{\mathbf{basic}}$ and;
   - $\mathbb{P}(P)$ by stacking up patches in the same order together from the original noisy image u.

   Once again, for optimization a maximum number of $N^{\mathbf{wien}}$ patches is kept in the two 3D groups. They have been chosen exactly as described in the first denoising step.

2. *Collaborative Filtering:*
   When the two 3D groups are obtained the collaborative filtering can be launched. To do so empirical Wiener coefficients are defined by

   $$\omega_P(\xi) = \frac{\left|\tau_{3D}^{\mathbf{wien}}\left(\mathbb{P}^{\mathbf{basic}}(P)\right)(\xi)\right|^2}{\left|\tau_{3D}^{\mathbf{wien}}\left(\mathbb{P}^{\mathbf{basic}}(P)\right)(\xi)\right|^2 + \sigma^2} \tag{11.6}$$

   The Wiener collaborative filtering of $\mathbb{P}(P)$ is realized as the element-by-element multiplication of the 3D transform of the noisy image $\tau_{3D}^{\mathbf{wien}}(\mathbb{P}(P))$ with the Wiener coefficients ω_P. (We denote this element-wise matrix multiplication by a dot.) Through this sub-step an estimate of the 3D group is obtained as

   $$\mathbb{P}^{\mathbf{wien}}(P) = {\tau_{3D}^{\mathbf{wien}}}^{-1}\left(\omega_P \,.\, \tau_{3D}^{\mathbf{wien}}(\mathbb{P}(P))\right) \tag{11.7}$$

3. *Aggregation:*
   When collaborative filtering is achieved, the estimates for every pixel are stored in a buffer:

   $$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + w_P^{\mathbf{wien}} u_{Q,P}^{\mathbf{wien}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + w_P^{\mathbf{wien}} \end{cases} \tag{11.8}$$

   where:

- $\boldsymbol{\nu}$ (resp. $\boldsymbol{\delta}$) designates the numerator (resp. denominator) part of the final estimation of the image obtained at the end of step 2;
- $u_{Q,P}^{\mathbf{wien}}(x)$ is the estimation of the value of the pixel x belonging to the patch Q obtained during the collaborative filtering of the reference patch P;
- $w_P^{\mathbf{wien}} = \|\omega_P\|_2^{-2}$.

As for the first step, a $k^{\mathbf{wien}} \times k^{\mathbf{wien}}$ Kaiser window is applied to reduce border effects.

The final estimate obtained after the second step is given by

$$u^{\mathbf{final}}(x) = \frac{\displaystyle\sum_P w_P^{\mathbf{wien}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x) u_{Q,P}^{\mathbf{wien}}(x)}{\displaystyle\sum_P w_P^{\mathbf{wien}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x)} \tag{11.9}$$

which is simply obtained by dividing both buffers (numerator and denominator) element-by-element. Here $\chi_Q(x) = 1$ if and only if $x \in Q$, 0 otherwise.

In the original article, during the patch aggregation sub-step, a Kaiser window is applied to each patch in order to slightly attenuate the patch borders. Nevertheless, experimental results show that the Kaiser windows do not improve the PSNR, and are visually less efficient than the weighting. For a sake of simplicity, these windows are not mentioned in this algorithmic description. For a sake of fidelity to the original article, they are nevertheless implemented in the provided code.

11.3 A Study of the Optimal Parameters

11.3.1 Comparison Criteria and Parameters Under Study

The results shown hereunder will be the result of the previously described algorithm applied to noiseless images to which a simulated white noise has been added. Many images have been tested, but for a sake of simplicity only one result by σ will be shown. All shown results has been obtained on the noise-free image shown in figure 11.5.
To describe quantitatively denoising results, two classic measures will be used :

- The *Root Mean Square Error* (RMSE) between the reference image (noiseless) u_R and the denoised image u_D. The RMSE is computed as:

$$RMSE = \sqrt{\frac{\displaystyle\sum_{x \in X} \left(u_R(x) - u_D(x)\right)^2}{|X|}} \tag{11.10}$$

  The smaller the RMSE, the better the denoising.

- The *Peak Signal to Noise Ratio* (PSNR) is evaluated in decibels (dB):

$$PSNR = 20 \log_{10}\left(\frac{255}{RMSE}\right) \tag{11.11}$$

  The larger the PSNR, the better the denoising.

Choosing the right values for the different parameters in the algorithm and their influence has to be discussed. The set of the parameters is:

- $k^{\mathbf{hard}}$ and $k^{\mathbf{wien}}$: size of patches;
- $N^{\mathbf{hard}}$ and $N^{\mathbf{wien}}$: maximum number of similar patches kept;

Figure 11.5: Right half of the Valldemossa noiseless image.

- $p^{\mathbf{hard}}$ and $p^{\mathbf{wien}}$: In order to speed up the processing, the loop over the pixels of the image is done with a (integer) step p in row and column. For example if $p = 3$ the algorithm is accelerated by a 9 factor;
- $n^{\mathbf{hard}}$ and $n^{\mathbf{wien}}$: search window size;
- $\tau^{\mathbf{hard}}$ and $\tau^{\mathbf{wien}}$: maximum thresholds for the distance between two similar patches;
- $\lambda_{2D}^{\mathbf{hard}}$ and $\lambda_{3D}^{\mathbf{hard}}$.

Several of these parameters have actually little influence on the final result, namely $n^{\mathbf{hard}}$ and $n^{\mathbf{wien}}$. Therefore the following values will be fixed and used throughout the study :

- $n^{\mathbf{hard}} = 39$;
- $n^{\mathbf{wien}} = 39$.

Moreover, when value of parameters are not explicitly given, the algorithm uses the default values shown in section 11.3.7.

11.3.2 Influence of $N^{\mathbf{hard}}$ and $N^{\mathbf{wien}}$

Parameters used for this study:

- $\tau^{\mathbf{hard}} = 2500$ if $\sigma < 40$, and 5000 otherwise. Moreover, if $N^{\mathbf{hard}} \geq 32$, this threshold is multiply by a 5 factor.
- $\tau_{2D}^{\mathbf{hard}}$ is a Bior1.5 transform whatever the value of σ;
- $\tau_{2D}^{\mathbf{wien}}$ is a 2D DCT transform, whatever the value of σ.

	$N^{\mathbf{hard}} = 8$							
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.54	1.35	**45.56**	**1.34**	**45.56**	**1.34**	45.55	1.35
5	39.41	2.73	39.44	2.72	**39.45**	**2.72**	39.44	2.72
10	35.03	4.52	35.07	4.50	**35.08**	**4.49**	35.07	4.50
20	30.90	7.27	30.94	7.24	**30.96**	**7.22**	30.95	7.23
30	28.70	9.37	28.76	9.30	**28.78**	**9.28**	28.78	9.28
40	27.18	11.16	27.22	11.11	27.25	11.07	**27.26**	**11.05**
60	25.28	13.88	25.35	13.77	25.39	13.71	**25.42**	**13.66**
80	24.05	16.00	24.14	15.83	24.22	15.69	**24.24**	**15.65**
100	22.90	18.26	23.04	17.97	23.14	17.76	23.20	17.64

	$N^{\mathbf{hard}} = 16$							
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.53	1.35	45.55	1.35	45.55	1.35	45.54	1.35
5	39.41	2.73	39.44	2.72	39.44	2.72	39.42	2.73
10	35.04	4.51	35.07	4.50	35.07	4.50	35.06	4.50
20	30.92	7.25	**30.96**	**7.22**	**30.96**	**7.22**	30.95	7.23
30	28.72	9.34	28.76	9.30	**28.78**	**9.28**	28.77	9.29
40	27.20	11.13	27.23	11.09	27.24	11.08	27.24	11.08
60	23.30	17.44	25.35	13.77	25.39	13.71	25.41	13.68
80	24.06	15.98	24.15	15.81	24.21	15.71	24.23	15.67
100	22.93	18.20	23.07	17.91	23.17	17.70	23.22	17.60

	$N^{\mathbf{hard}} = 32$							
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.50	1.35	45.51	1.35	45.51	1.35	45.50	1.35
5	39.37	2.74	39.39	2.74	39.39	2.74	39.37	2.74
10	35.01	4.53	35.03	4.52	35.01	4.53	35.01	4.53
20	30.91	7.26	30.93	7.25	30.92	7.25	30.90	7.27
30	28.72	9.34	28.75	9.31	28.75	9.31	28.73	9.33
40	27.19	11.14	27.21	11.12	27.22	11.11	27.20	11.13
60	25.34	13.79	25.39	13.71	25.41	13.68	**25.42**	**13.66**
80	24.07	15.96	24.13	15.85	24.17	15.78	24.16	15.80
100	23.14	17.76	23.21	17.62	23.27	17.50	**23.28**	**17.48**

	$N^{\mathbf{hard}} = 64$							
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.47	1.36	45.48	1.36	45.47	1.36	45.46	1.36
5	39.33	2.75	39.35	2.75	39.34	2.75	39.32	2.76
10	34.97	4.55	34.99	4.54	34.98	4.55	34.95	4.56
20	30.88	7.29	30.90	7.27	30.88	7.29	30.85	7.31
30	28.70	9.37	28.73	9.33	28.72	9.34	28.69	9.38
40	27.15	11.20	27.18	11.16	27.17	11.17	27.15	11.20
60	25.32	13.82	25.36	13.76	25.38	13.73	25.38	13.73
80	24.06	15.98	24.11	15.89	24.13	15.85	24.12	15.87
100	23.12	17.80	23.19	17.66	23.23	17.58	23.23	17.58

In **bold** the best result for a given σ.

One can see that those parameters have a very small influence on the result, and values given in the original paper ($N^{\mathbf{hard}} = 16$ and $N^{\mathbf{wien}} = 32$) are very close to the best result, whatever the

value of the noise. Then, according to this study, the final algorithm will kept original parameters, i.e.:

- $N^{\mathbf{hard}} = 16$;
- $N^{\mathbf{wien}} = 32$.

11.3.3 Influence of $\lambda_{3D}^{\mathbf{hard}}$

This parameter is important because it defines the coefficient thresholding level of the 3D group in the transform domain during the first filtering substep. The chosen value in the original article is 2.7 and it turns out to be the best choice. Here is a table evaluating its influence (the error results are given after application of the entire algorithm), in the case where $\tau_{2D}^{\mathbf{hard}}$ = Bior1.5 and $\tau_{2D}^{\mathbf{wien}}$ = DCT whatever the value of the noise:

$\lambda_{3D}^{\mathbf{hard}}$	2.5		2.7		3.0		3.2	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.51	1.35	**45.55**	**1.34**	**45.55**	**1.34**	**45.55**	**1.34**
5	39.39	2.73	**39.45**	**2.72**	**39.45**	**2.72**	39.41	2.73
10	34.96	4.55	**35.04**	**4.51**	35.02	4.52	34.96	4.55
20	30.88	7.29	**30.97**	**7.21**	30.93	7.25	30.82	7.34
30	28.63	9.44	**28.74**	**9.32**	28.65	9.41	28.52	9.56
40	27.12	11.23	**27.26**	**11.06**	27.19	11.14	27.07	11.30
60	25.15	14.10	**25.33**	**13.80**	25.26	13.92	25.07	14.23
80	24.00	16.09	**24.26**	**15.61**	24.24	15.64	24.08	15.95
100	23.00	18.05	**23.21**	**17.62**	23.00	18.05	22.77	18.53

In **bold** are the best results for a given σ.

One can see that small variations on this parameter have strong influence on the result. Then this parameter needs to be carefully chosen, and according to the original article, the final algorithm will kept original parameter, i.e. $\lambda_{3D}^{\mathbf{hard}} = 2.7$.

11.3.4 Influence of the Thresholds $\tau^{\mathbf{hard}}$ and $\tau^{\mathbf{wien}}$

The thresholds $\tau^{\mathbf{hard}}$ and $\tau^{\mathbf{wien}}$ are highly dependent on σ and very influential. One can lose many dBs in PSNR by choosing wrong values for these thresholds. Thus, their correct evaluation are crucial.
However, by picking them large enough, it is possible to keep a constant value for these thresholds for a wide range of σ's. In the original article a value of (2500, 400) is proposed for the thresholds pair $(\tau^{\mathbf{hard}}, \tau^{\mathbf{wien}})$ for $\sigma \in [0, 40]$. These values give good results.

$\tau^{\mathbf{hard}}$	PSNR	RMSE
100	37.02	3.59
200	37.43	3.43
300	37.59	3.36
400	37.61	3.35
600	37.63	3.35
1200	37.63	3.35
2500	37.63	3.35

$\tau^{\mathbf{wien}} = 20$

$\tau^{\mathbf{wien}}$	PSNR	RMSE
5	37.31	3.47
10	37.50	3.40
15	37.58	3.37
20	37.63	3.35
30	37.68	3.33
45	37.71	3.32
60	37.73	3.31

$\tau^{\mathbf{hard}} = 600$

Taking low values for the thresholds allows us to keep a limited number of similar patches during the patch-matching. But if this number is too limited, not enough denoising is being done. As it is preferable to work with few similar patches, for some values of the threshold this limit on the number of similar patches is reached for many reference patches. This explains the PSNR stagnation when the thresholds are increased. On the other hand, if these values increase too

much, patches significantly different from the reference patch will bring spurious details to the final result. Then, there will be a fall in the PSNR. A good balance must be found. Yet, the breathing space is quite substantial, and it is possible to get generic thresholds for a wide range of σ's.

In the original article, a distinction is done for high value of noise, i.e. $\sigma > 40$. For high noise, $\tau^{\mathbf{hard}}$ is increased to a value of 5000. Here is a study of this parameter for high value of noise, with $\tau_{2D}^{\mathbf{hard}}$ = Bior1.5 and $\tau_{2D}^{\mathbf{wien}}$ = 2D DCT:

$\tau^{\mathbf{hard}}$	2500		5000		10000		25000	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
40	**27.27**	**11.03**	**27.27**	**11.03**	**27.27**	**11.03**	**27.27**	**11.03**
50	26.27	12.38	**26.30**	**12.35**	**26.30**	**12.35**	26.29	12.35
60	25.40	13.69	25.43	13.64	**25.44**	**13.63**	**25.44**	**13.63**
70	24.61	15.00	**24.74**	**14.77**	24.73	14.79	24.73	14.79
80	23.77	16.52	**24.21**	**15.70**	24.19	15.73	24.19	15.73
90	23.18	17.67	23.68	16.69	23.71	16.64	**23.72**	**16.62**
100	22.81	18.45	23.26	17.51	23.34	17.35	**23.35**	**17.34**

In **bold** are the best results for a given σ.

One can see that an increasing of $\tau^{\mathbf{hard}}$ is necessary for a very high noise. But increasing a lot is useless because there is a stagnation of the enhancement of the result, due to the maximum number of of similar patches kept $N^{\mathbf{hard}}$. Then once again the value of the original article will be kept for high noise.

11.3.5 Influence of the Size of the Patches: $k^{\mathbf{hard}}$ and $k^{\mathbf{wien}}$

The window size of the patches influences the result and must be adapted to σ. Indeed, for low values of σ the window size must be relatively small to be well adapted to the details, whereas for large values of σ larger window sizes are better, because most of the details of the image are anyway lost in noise. Since much of the noise is canceled in the first step, smaller patches can now be worked with in the second step.

In the original article, a patch size of $k^{\mathbf{hard}} = k^{\mathbf{wien}} = 8$ (resp. $k^{\mathbf{hard}} = k^{\mathbf{wien}} = 12$, but only if the 2D DCT is chosen as τ_{2D}) is proposed for $\sigma \leq 40$ (resp. $\sigma > 40$).

As Bior1.5 can not be applied if the patch size is not a power of 2, τ_{2D} is always chosen equal to 2D DCT for this study.

	$k^{\mathbf{hard}} = 4$									
	$k^{\mathbf{wien}} = 4$		$k^{\mathbf{wien}} = 6$		$k^{\mathbf{wien}} = 8$		$k^{\mathbf{wien}} = 10$		$k^{\mathbf{wien}} = 12$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.58	1.34	**45.59**	**1.34**	**45.59**	**1.34**	45.54	1.35	45.51	1.35
5	39.42	2.73	39.46	2.71	39.45	2.72	39.40	2.73	39.36	2.74
10	34.97	4.55	35.05	4.51	35.05	4.51	35.00	4.53	34.96	4.56
20	30.65	7.48	30.83	7.33	30.86	7.30	30.82	7.34	30.80	7.35
30	28.25	9.86	28.52	9.56	28.58	9.50	28.55	9.53	28.54	9.54
40	26.67	11.83	27.05	11.33	27.13	11.22	27.12	11.23	27.10	11.26
60	24.32	15.51	24.81	14.66	24.96	14.41	24.97	14.39	24.97	14.39
80	22.79	18.49	23.47	17.10	23.60	16.85	23.61	16.83	23.59	16.87
100	21.51	21.43	22.31	19.55	22.51	19.10	22.52	19.08	22.53	19.06

	$k^{\mathbf{hard}} = 8$									
	$k^{\mathbf{wien}} = 4$		$k^{\mathbf{wien}} = 6$		$k^{\mathbf{wien}} = 8$		$k^{\mathbf{wien}} = 10$		$k^{\mathbf{wien}} = 12$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.54	1.35	45.56	1.34	45.55	1.35	45.51	1.35	45.48	1.36
5	**39.47**	**2.71**	39.45	2.72	39.43	2.72	39.39	2.74	39.35	2.75
10	**35.10**	**4.48**	35.09	4.49	35.06	4.50	35.01	4.53	34.98	4.55
20	**31.01**	**7.18**	31.00	7.19	30.96	7.22	30.93	7.25	30.89	7.28
30	28.78	9.28	**28.79**	**9.27**	28.77	9.29	28.74	9.32	28.72	9.34
40	27.32	10.98	**27.36**	**10.93**	27.34	10.95	27.34	10.95	27.31	10.99
60	25.19	14.03	25.31	13.84	25.36	13.76	**25.38**	**13.73**	25.37	13.74
80	24.11	15.89	**24.27**	**15.60**	**24.27**	**15.60**	**24.27**	**15.60**	24.24	15.65
100	22.67	18.75	23.06	17.93	23.17	17.71	23.21	17.62	**23.23**	**17.58**

	$k^{\mathbf{hard}} = 12$									
	$k^{\mathbf{wien}} = 4$		$k^{\mathbf{wien}} = 6$		$k^{\mathbf{wien}} = 8$		$k^{\mathbf{wien}} = 10$		$k^{\mathbf{wien}} = 12$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.45	1.36	45.50	1.35	45.50	1.35	45.46	1.36	45.43	1.36
5	39.38	2.74	39.39	2.74	39.36	2.74	39.31	2.76	39.28	2.77
10	35.01	4.53	35.00	4.53	34.97	4.55	34.91	4.58	34.87	4.60
20	30.90	7.27	30.87	7.30	30.83	7.33	30.78	7.37	30.74	7.41
30	28.65	9.42	28.64	9.43	28.60	9.47	28.56	9.52	28.52	9.56
40	27.14	11.21	27.16	11.18	27.11	11.25	27.09	11.27	27.04	11.34
60	25.05	14.26	25.19	14.03	25.24	13.95	25.24	13.95	25.22	13.98
80	23.92	16.24	24.07	15.96	24.05	16.00	24.01	16.07	23.95	17.93
100	22.51	19.10	22.92	18.22	23.02	18.01	23.05	17.95	23.06	17.93

In **bold** are the best results for a given σ.

An other influence of the size of the patch is on the processing time. Indeed smaller patches give faster algorithm. Then, as results for $\sigma > 40$ for $k^{\mathbf{wien}} = 8$ or $k^{\mathbf{wien}} = 12$ are really close, we will prefer to kept $k^{\mathbf{wien}} = 8$ because of the processing time.

11.3.6 Influence of $p^{\mathbf{hard}}$ and $p^{\mathbf{wien}}$

In order to speed up the algorithm, it is possible to use a step p in both rows and columns to go from one reference patch to the next. For example, a step of 3 theoretically divides by 9 the processing time. In the original article a step of 3 is proposed. The PSNR loss due to this step use is negligible for large values of noise. Nevertheless, for low noise, it is better to keep a step of 1 or 2. Moreover, since the second step works on a first basic estimate (which is assumed to be noiseless, or at least with lower noise) a bigger step for the first step than for the second can be used.

	$p^{\mathbf{hard}} = 1$					
	$p^{\mathbf{wien}} = 1$		$p^{\mathbf{wien}} = 3$		$p^{\mathbf{wien}} = 5$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**45.59**	**1.34**	45.56	1.34	45.50	1.35
5	**39.53**	**2.69**	39.49	2.70	39.41	2.73
10	**35.10**	**4.48**	35.07	4.50	34.89	4.59
20	**31.01**	**7.18**	30.97	7.21	30.88	7.29
30	**28.79**	**9.27**	28.76	9.30	28.66	9.41
40	**27.32**	**10.98**	27.31	10.99	27.26	11.05
60	25.37	13.74	25.35	13.77	25.31	13.84
80	**24.23**	**15.67**	**24.23**	**15.67**	24.21	15.71
100	22.87	18.32	22.87	18.32	22.86	18.35

	$p^{\mathbf{hard}} = 3$					
	$p^{\mathbf{wien}} = 1$		$p^{\mathbf{wien}} = 3$		$p^{\mathbf{wien}} = 5$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.58	1.34	45.55	1.35	45.49	1.36
5	39.51	2.70	39.47	2.71	39.39	2.74
10	35.09	4.49	35.05	4.51	34.96	4.56
20	30.99	7.20	30.92	7.25	30.85	7.31
30	28.76	9.30	28.73	9.33	28.64	9.43
40	27.31	10.99	28.29	9.82	27.24	11.08
60	**25.44**	**13.63**	25.42	13.66	25.36	13.76
80	24.19	15.74	24.19	15.74	24.17	15.78
100	**23.20**	**17.64**	23.19	17.66	23.19	17.66

	$p^{\mathbf{hard}} = 5$					
	$p^{\mathbf{wien}} = 1$		$p^{\mathbf{wien}} = 3$		$p^{\mathbf{wien}} = 5$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.54	1.35	45.51	1.35	45.44	1.36
5	39.47	2.71	39.43	2.72	39.34	2.75
10	35.04	4.51	35.00	4.53	34.90	4.59
20	30.93	7.25	30.89	7.28	30.78	7.37
30	28.70	9.37	28.67	9.40	28.55	9.53
40	27.22	11.11	27.20	11.13	27.15	11.20
60	25.31	13.84	25.29	13.87	25.22	13.98
80	24.09	15.92	24.08	15.94	24.07	15.96
100	23.03	17.99	23.02	18.01	23.01	18.03

As the loss in PSNR is negligible comparing to the gain in speed, and for a sake of fidelity to the original article, we keep $p = 3$ for both steps in the provided code.

11.3.7 Summary Table

Here is the summary table with the final chosen values for all parameters, depending on the value of the noise:

Parameter	$\sigma \leq 40$	$\sigma > 40$
$N^{\mathbf{hard}}$	16	16
$N^{\mathbf{wien}}$	32	32
$\lambda_{3D}^{\mathbf{hard}}$	2.7	2.7
$\tau^{\mathbf{hard}}$	2500	5000
$\tau^{\mathbf{wien}}$	400	3500
$k^{\mathbf{hard}}$	8	8
$k^{\mathbf{wien}}$	8	8
$p^{\mathbf{hard}}$	3	3
$p^{\mathbf{wien}}$	3	3

11.4 A Detailed Study of Possible Variants

This part examines some ambiguous choices of the original method and experimentally decides for the best choice in terms of PSNR. Unless otherwise specified, in the following $\tau_{2D}^{\mathbf{hard}} = \tau_{2D}^{\mathbf{wien}}$ refer to transform thresholds with the 2D DCT, and $\tau_{1D}^{\mathbf{hard}} = \tau_{1D}^{\mathbf{wien}}$ to the 1D DCT. Moreover shown results have been obtained on the image shown on figure 11.5, and when it is not explicitly said, default parameter values shown in section 11.3.7 have been used.

11.4.1 Variants of the First Denoising Step

1. *Grouping:*

   - Distance: To calculate the distance, we can choose between doing it on the patches obtained after τ_{3D} or directly on the patches of the image. If the 3D transform is an isometry, the distances are equal, so the distance will be calculated directly on the image;
   - Normalization: Several classic DCT definitions do not normalize the first coefficient. For the 1D case, the first coefficient is for example not divided by $\sqrt{2}$. If we do not take into account this fact during the hard-thresholding, the same threshold will be applied, whatever the coefficient. The table below shows that this makes a significant difference. Thus, some care must be taken to use a normalized DCT.

	non-normalized DCT		normalized DCT	
σ	PSNR	RMSE	PSNR	RMSE
2	41.70	2.10	41.53	2.14
5	38.02	3.20	38.00	3.21
10	35.10	4.48	35.21	4.42
20	31.65	6.67	32.36	6.14
30	29.13	8.91	30.29	7.80
40	27.21	11.11	28.70	9.36

Fixed parameters : Distance : image, Aggregation : with weighting, Step : 3, Patches : 8×8, Collaborative Filtering : DCT

   - Thresholding $\tau_{2D}^{\mathbf{hard}}$: In the original article, this threshold only appears from $\sigma > 40$ since we initialized $\lambda_{2D} = 0$ for $\sigma \leq 40$. Applying a threshold to the 2D transform coefficients for low values of σ is useless since there is no improvement after the second step. Moreover for a noise lower than 5 the results are degraded.

2. *Collaborative Filtering:*
   For the 3D group, a choice must be done between :

   - Simply averaging image patches along the third dimension, which would be a primitive version of NL-means (In NL-means there also is a weighting of the patches according to their distance). This simple average solution is denoted by *basic NL-means*;
   - Applying a hard-thresholding on a 1D transform along the third dimension (in other terms apply τ_{3D} as described before), which is denoted *Hard Thresholding*.

   The next table shows the considerable amelioration obtained by adding a thresholded 3D transform along patches compared to a simple average of the patches in the case where $\tau_{3D}^{\mathbf{hard}} = \tau_{3D}^{\mathbf{wien}} = DCT$:

	Collaborative Filtering			
	basic NL-Means		Hard Thresholding	
σ	PSNR	RMSE	PSNR	RMSE
2	40.96	2.28	41.36	2.18
5	36.40	3.86	37.80	3.28
10	33.54	5.36	34.72	4.68
20	29.54	8.50	31.41	6.86
30	27.01	11.36	28.96	9.09
40	25.04	14.26	27.14	11.21

Fixed parameters : Distance : image, Aggregation : with weighting, Step : 3, Normalized DCT, Patches : 8×8

3. *Aggregation:*
During the aggregation the weighting is based on the number of coefficients cancelled during the hard thresholding. However, as shown by the next table, this weighting works, but does not play such an important part in the PSNR improvement. Nevertheless, artifacts on edges are visually attenuated by using a weighted aggregation (see figure 11.6).

	Without weighting		With weighting	
σ	PSNR	RMSE	PSNR	RMSE
2	41.26	2.20	41.53	2.14
5	37.83	3.27	38.00	3.21
10	34.83	4.62	35.21	4.42
20	32.09	6.33	32.36	6.14
30	30.04	8.02	30.29	7.80
40	28.60	9.47	28.70	9.36

Fixed Parameters : Distance : image, Normalized DCT, Step : 3; Collaborative Filtering : *DCT*, Patches : 8×8

Because they have many coefficients that are thresholded, the aggregation weighting enforces the role of homogeneous patches compared to patches containing edges. The issue with this weighting is: how to choose the right homogeneity indicator?

Another natural indicator would be the standard deviation of the patches. Indeed, for homogeneous patches, the standard deviation would be very small, whereas for patches containing edges, the standard deviation would be much bigger. Then the weighting would be the inverse of the standard deviation, which would mean for a 3D group

$$w_P^{\mathbf{hard},\mathbf{wien}} = \left(\frac{1}{N-1} \sum_{k=1}^{N} \sum_{i=1}^{M} \left(\mathbb{P}(P)(x_{i,k}) \right)^2 \right)^{-\frac{1}{2}} \tag{11.12}$$

where:

- N is the number of similar patches to P;
- $M = k^{\mathbf{hard}} \times k^{\mathbf{hard}}$ or $M = k^{\mathbf{wien}} \times k^{\mathbf{wien}}$.

Then homogeneous patches would have more weights than patches containing edges.
One can also wonder if it would not be better to calculate this standard deviation on the 3D group rather than on its estimate. In the following we denote :

- *H.T.* the original weighting in step 1 (i.e., the inverse of the number of coefficients different from zero during the hard thresholding);
- *STD* the weighting using the standard deviation processed on the original image;
- *STD C.F.* the weighting using the standard deviation processed on the estimate of the 3D group.

The following table compares the new weighting results and shows that H.T. wins:

	Without weighting		*H.T.*		*STD*		*STD C.F.*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.13	1.41	45.13	1.41	**45.14**	**1.41**	45.13	1.41
5	40.68	2.36	**40.71**	**2.35**	**40.71**	**2.35**	40.69	2.35
10	37.26	3.50	**37.31**	**3.47**	37.29	3.48	37.27	3.49
20	33.58	5.34	**33.65**	**5.30**	33.58	5.34	33.57	5.35
30	31.10	7.10	**31.17**	**7.04**	31.10	7.11	31.08	7.12
40	29.35	8.68	**29.41**	**8.62**	29.35	8.69	29.32	8.71

In **bold** the best result for a given σ

Step 1 ,without weighting | Step 1 with weighting

Step 2, without weighting | Step 2 with weighting

Figure 11.6: Difference between with and without weighting in both steps.

Another question is: why is the weight computed for the whole 3D group when *a priori* every 2D patch of the 3D group could be given a different weight? This would yield more differentiated estimates for each pixel.

The answer is experimental. The next table shows that giving a single weight to the whole 3D group is better than giving a weight to each patch:

	2D weighting		3D weighting	
σ	PSNR	RMSE	PSNR	RMSE
2	45.08	1.42	45.13	1.41
5	40.44	2.42	40.71	2.35
10	36.94	3.63	37.31	3.47
20	33.16	5.61	33.65	5.30
30	30.64	7.49	31.17	7.04
40	28.86	9.19	29.41	8.62

Weighting results for each step of the algorithm:

11.4.2 Variants of the Second Denoising Step

1. *Wiener Filtering:*
   The second step adds details for large σ values, as illustrated in the experiments below.

Using a Wiener filter in this second step rather than a hard thresholding avoids losing details. Nevertheless the weighting does not visually improve much the image near edges, and the gain in PSNR is negligible.

	Step 1 only		Step 1 (Fixed) + Step 2 (differents sorts of Collaborative Filtering)					
			H. T. + weight.		Wiener Filtering		W. F. + weighting	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.13	1.41	43.18	1.77	45.21	1.40	45.22	1.40
5	40.70	2.35	39.12	2.82	40.86	2.31	40.90	2.29
10	37.31	3.47	35.97	4.06	37.61	3.36	37.66	3.34
20	33.65	5.30	32.75	5.87	34.23	4.96	34.27	4.92
30	31.17	7.04	30.64	7.49	32.01	6.40	32.03	6.38
40	29.41	8.62	29.08	8.96	30.44	7.66	30.47	7.64

Fixed Parameters :
Step 1 : Distance : image, Aggregation : With weighting
Collaborative Filtering : *DCT*, Patches : 8×8
Step : 3
Step 2 : Distance : image, Patches : 8×8
Step : 3

This table shows that the second step is worthwhile and important, especially for large noise values. However, the weighting in the second step is much less useful than the weighting in the first step. Using a Wiener filter during the second step and not having to repeat the hard thresholding shows the importance of this improvement. The gain is important both in PSNR and visually since it attenuates the noise again and improves/adds details at the same time.

2. *Ideal Wiener Filtering:*
It is also possible to compare the ordinary Wiener filter with an ideal Wiener filter, which is obtained when the original noise-free image is taken as oracle reference. This ideal Wiener filter is the best possible estimate for the second step of this algorithm. *It is therefore interesting to see how far we stand from this ideal estimate with the current one :*

	Step 1 + Step 2		Ideal Wiener	
σ	PSNR	RMSE	PSNR	RMSE
2	45.24	1.39	47.81	1.04
5	40.87	2.31	43.19	1.77
10	37.64	3.34	39.98	2.56
20	34.25	4.94	36.99	3.60
30	32.07	6.35	35.21	4.43
40	30.40	7.70	33.71	5.26

Fixed Parameters :
Step 1 : Distance : image, Aggregation : With weighting
Collaborative Filtering : *DCT*, Patches : 8×8
Step : 3
Step 2 : Distance : image, Patches : 8×8
Step : 3

As expected, the ideal Wiener filter gives a better result. A Wiener filter in the second step seems to be some 3 dB away from the ideal result.

3. *Aggregation:*
Wien denotes the original weighting in the second step (i.e., the inverse of the empirical Wiener coefficients norm). As for the first step, would it be possible to improve the weighting by using the standard deviation instead of the norm of the empirical Wiener coefficients?

	H.T. | *Wien*		*STD* | *STD*		*STD C.F.* | *STD C.F.*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**45.22**	**1.40**	45.12	1.41	**45.22**	**1.40**
5	**40.89**	**2.30**	40.71	2.35	40.88	2.30
10	**37.66**	**3.34**	37.45	3.42	37.64	3.35
20	**34.27**	**4.93**	34.10	5.03	34.26	4.94
30	**32.03**	**6.38**	31.92	6.46	**32.03**	**6.38**
40	30.47	7.64	30.36	7.73	**30.48**	**7.63**

In **bold** the best result for a given σ

Moreover it is possible to switch the weightings:

	H.T. | *STD*		*H.T.* | *STD C.F.*		*STD* | *Wien*		*STD C.F.* | *Wien*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.18	1.40	**45.23**	**1.40**	45.18	1.40	45.22	1.40
5	40.82	2.32	**40.89**	**2.30**	40.81	2.32	40.88	2.30
10	37.57	3.37	**37.66**	**3.34**	37.55	3.38	37.64	3.44
20	34.20	4.97	**34.28**	**4.93**	34.18	4.98	34.26	4.94
30	32.00	6.40	**32.06**	**6.36**	31.96	6.43	32.02	6.39
40	30.43	7.67	**30.50**	**7.61**	30.42	7.68	30.47	7.64

In **bold** the best result for a given σ

Even though the improvement is minor, it is still possible to improve the result a little bit by working on the aggregation weighting. The weighting in this second step seems useless. Yet, better results are obtained by modifying the weighting in the second step. Thus, this study underlines that it is still possible to gain a bit by working on the weighting in the aggregation sub-step.

11.4.3 Influence of the 3D Transform

The choice of the 3D transform (τ_{2D} and τ_{1D}) is crucial. Here are some choices:

- τ_{2D} : This 2D transform is applied on each patch of the 3D group. We have the choice between a normalized 2D DCT and a bi-orthogonal spline wavelet, where the vanishing moments of the decomposing and reconstructing wavelet functions are 1 and 5 respectively. We shall denote by Bior1.5 this bi-orthogonal spline wavelet;

- τ_{1D} : This 1D transform is applied along the third dimension of the 3D group, after the 2D transform has been applied. We have the choice between a normalized 1D DCT and a Walsh-Hadamard transform.

In order to clarify the use of those transforms, here is a brief explanation of the practical implementation of the bior1.5 and the Walsh-Hadamard transforms, as they are used in this implementation of BM3D.

Walsh-Hadamard Transform

This 1D transform is very simple. Since it recursively processes the sums and differences between pair of values of the vector it is applied to, the size of this vector must be a power of 2. This forces the number of similar patches in a 3D group to be a power of 2 itself. Let us denote for instance the first four coordinates of the processed vector by $V_0 = [a\ b\ c\ d]$. The first step computes the sums and the differences of the pairs of values and regroups them, which leads to

$$V_1 = [(a+b)\ (c+d)\ (a-b)\ (c-d)]$$

The process is iterated on each pair: $V_1^h = [(a+b)\ (c+d)]$ and $V_1^l = [(a-b)\ (c-d)]$, which yields the final vector

$$V_f = [(a+b+c+d)\ (a+b-c-d)\ (a-b+c-d)\ (a-b-c+d)].$$

Moreover, in order to keep the norm of the initial vector, it is necessary to normalize V_f by $\sqrt{N}$, where N is the vector dimension. In that case, the inverse Walsh-Hadamard transform is exactly the same as the forward transform.

Bior 1.5

To obtain this transform, it is necessary to have four filters, which values can be found in [46]:

1. The one of low frequencies for the forward transform :

$$lpd = \frac{\sqrt{2}}{256}[3;\ -3;\ -22;\ 22;\ 128;\ 128;\ 22;\ -22;\ -3;\ 3]$$

2. The one of high frequencies for the forward transform :

$$hpd = \frac{\sqrt{2}}{2}[0;\ 0;\ 0;\ 0;\ -1;\ 1;\ 0;\ 0;\ 0;\ 0]$$

3. The one of low frequencies for the backward transform :

$$lpr = \frac{\sqrt{2}}{2}[0;\ 0;\ 0;\ 0;\ 1;\ 1;\ 0;\ 0;\ 0;\ 0]$$

4. And the one of high frequencies for the backward transform :

$$hpr = \frac{\sqrt{2}}{256}[3;\ 3;\ -22;\ -22;\ 128;\ -128;\ 22;\ 22;\ -3;\ -3]$$

The 2D transform is separable, first done in column, then in row (or vice-versa). Thus it is enough to explain the 1D transform only. Let us denote by V_0 a size $N = 2^n$ vector. Then V_1^l and V_1^h -which size is 2^{n-1}- are obtained by the following steps.

1. First of all, V is periodically extended beyond its boundaries. Denote for instance $V_0 = [a\ b\ c\ d\ e\ f\ g\ h]$, of size $N = 8$. Then the extension is

$$\tilde{V_0} = [d\ e\ f\ g\ h\ a\ b\ c\ d\ e\ f\ g\ h\ a\ b\ c\ d\ e]$$

   The periodization is done symmetrically in order to have at any position from $M/2$ to $M/2 + N$ all values of V_0 and only them for the discrete convolution of V_0 with *lpd* or *hpd* (where M is the size of the transform, *lpd* or *hpd*). For example, at the first position $M/2$, the vector with which the discrete convolution is done has for values : $[d\ e\ f\ g\ h\ a\ b\ c]$. If the periodization was done anti-symmetrically, this same vector would be $[f\ e\ d\ c\ b\ a\ b\ c]$ and then the value g is not represented and the values b and c are over-represented.

2. The discrete convolution of $\tilde{V_0}$ with *lpd* and *hpd* is computed:

$$\forall i \in [0, 2^{n-1} - 1], V_1^l[i] = \sum_{k=0}^{9} \tilde{V_0}[2i + k] lpd[k]$$

$$\forall i \in [0, 2^{n-1} - 1], V_1^h[i] = \sum_{k=0}^{9} \tilde{V_0}[2i + k] hpd[k]$$

Then we get $V_1 = [V_1^h\ V_1^l]$. The process is only iterated on high frequencies part of the sub-vector V_1: V_1^h, which size is 2^{n-1}.
The backward transform is done in a similar way :

1. Let it be started for instance from $V_2 = [V_2^h\ V_2^l V_1^l]$. For a sake of clarity, we remind that the size of V_2^h and V_2^l is 2^{n-2} and the size of V_1^l is 2^{n-1} if the size of $V_2 = 2^n$.
2. V_2^h and V_2^l are periodically extended;
3. V_1^h is obtained by the following convolutions :

$$\forall i \in [0, 2^{n-2} - 1], V_1^h[i] = \sum_{k=0}^{4} \big(hpr[2k]V_2^h[k+i] + hpr[2k+1]V_2^l[k+i]\big)$$

$$\forall i \in [0, 2^{n-2} - 1], V_1^h[i + 2^{n-2}] = \sum_{k=0}^{4} \big(lpr[2k]V_2^h[k+i] + lpr[2k+1]V_2^l[k+i]\big)$$

4. The iteration is done on $V_1 = [V_1^h\ V_1^l]$.

In order to show the importance of the choice of the transforms τ_{2D} and τ_{1D}, here is the result of a comparative study led on the different possible choices :

	$\tau_{2D}^{\mathbf{hard}}$ | $\tau_{1D}^{\mathbf{hard}}$ | $\tau_{2D}^{\mathbf{wien}}$ | $\tau_{1D}^{\mathbf{wien}}$					
	DCT|DCT|DCT|DCT		DCT|Hadamard|DCT|Hadamard		DCT|Hadamard|Bior|Hadamard	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.11	1.41	45.49	1.35	38.76	2.94
5	38.79	2.93	39.29	2.77	36.68	3.74
10	34.21	4.97	34.83	4.62	33.94	5.12
20	29.89	8.17	30.69	7.44	30.59	7.53
30	27.78	10.41	28.44	9.65	28.57	9.51
40	26.27	12.39	26.57	11.96	26.96	11.45
60	24.52	15.15	24.75	14.75	25.09	14.19
80	23.46	17.12	23.67	16.71	24.03	16.02
100	22.45	19.22	22.61	18.88	22.74	18.60

	$\tau_{2D}^{\mathbf{hard}}$ | $\tau_{1D}^{\mathbf{hard}}$ | $\tau_{2D}^{\mathbf{wien}}$ | $\tau_{1D}^{\mathbf{wien}}$			
	Bior|Hadamard|DCT|Hadamard		Bior|Hadamard|Bior|Hadamard	
σ	PSNR	RMSE	PSNR	RMSE
2	**45.59**	**1.33**	38.58	3.00
5	**39.35**	**2.75**	36.48	3.82
10	**35.00**	**4.53**	33.80	5.21
20	**30.94**	**7.24**	30.51	7.61
30	**28.73**	**9.34**	28.50	9.58
40	**27.23**	**11.10**	27.06	11.31
60	**25.28**	**13.88**	24.91	14.49
80	**24.21**	**15.70**	24.20	15.71
100	**23.11**	**17.83**	22.96	18.14

In **bold the best PSNR for a given σ.**

In all cases the size of patches is 8×8, except when we use a 2D DCT for $\sigma \geq 40$: in this case the size is 12×12. One reason which could explain the utility of not using the same 2D transform for both steps is that the artifacts created by the chosen transform in the first step are not enhanced during the second step, which is the case if the same transform is used twice.
According to those results, the best combination of transforms is :

- Step 1: $\tau_{2D}^{\mathbf{hard}}$ = Bior1.5 and $\tau_{1D}^{\mathbf{hard}}$ = Hadamard;
- Step 2: $\tau_{2D}^{\mathbf{wien}}$ = 2D DCT and $\tau_{1D}^{\mathbf{wien}}$ = Hadamard;

11.4.4 Influence of the Kaiser Window

According to the original article, a Kaiser window (with parameter $\alpha = 2.0$) is applied during the weighting aggregation in order to reduce border effects which can appear when certain 2D transforms (typically 2D DCT) are used. Then an element-by-element multiplication is done between the Kaiser window and the estimated patch during the aggregation of block-wise estimates. Of course, the weight of each coefficient of the Kaiser window is taken into account.
The Kaiser window of length k is defined in 1-D by the formula:

$$K_n = \begin{cases} \dfrac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{2n}{k-1}-1\right)^2}\right)}{I_0\left(\pi\alpha\right)} & if\ 0 \leq n \leq k-1 \\ 0 & otherwise \end{cases}$$

where I_0 is the zeroth order modified Bessel function of the first kind:

$$I_0(x) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(m+1)} \left(\frac{x}{2}\right)^{2m}$$

where Γ is the gamma function, a generalization of the factorial function to non-integer values.
Practically $k \times k$ Kaiser windows are hard-coded for $k = 8$ and $k = 12$. If the size of patches is different from those values, Kaiser windows are not used (i.e. they are set to 1). Here an example of a 2-D Kaiser window for $k = 8$ and $\alpha = 2.0$ used in our algorithm:

$$\begin{pmatrix} 0.1924 & 0.2989 & 0.3846 & 0.4325 & 0.4325 & 0.3846 & 0.2989 & 0.1924 \\ 0.2989 & 0.4642 & 0.5974 & 0.6717 & 0.6717 & 0.5974 & 0.4642 & 0.2989 \\ 0.3846 & 0.5974 & 0.7688 & 0.8644 & 0.8644 & 0.7688 & 0.5974 & 0.3846 \\ 0.4325 & 0.6717 & 0.8644 & 0.9718 & 0.9718 & 0.8644 & 0.6717 & 0.4325 \\ 0.4325 & 0.6717 & 0.8644 & 0.9718 & 0.9718 & 0.8644 & 0.6717 & 0.4325 \\ 0.3846 & 0.5974 & 0.7688 & 0.8644 & 0.8644 & 0.7688 & 0.5974 & 0.3846 \\ 0.2989 & 0.4642 & 0.5974 & 0.6717 & 0.6717 & 0.5974 & 0.4642 & 0.2989 \\ 0.1924 & 0.2989 & 0.3846 & 0.4325 & 0.4325 & 0.3846 & 0.2989 & 0.1924 \end{pmatrix}$$

One can see that borders of the patch are smoothly decreased and then allow to reduce border effects during the aggregation part.
Here is a study of the influence of the use of a Kaiser window, by using 8×8 patches and $p^{\mathbf{hard}} = p^{\mathbf{wien}} = 3$:

	With		Without	
σ	PSNR	RMSE	PSNR	RMSE
2	**45.61**	**1.34**	45.59	1.34
5	**39.44**	**2.72**	39.42	2.72
10	**35.05**	**4.51**	35.04	4.52
20	**30.97**	**7.21**	30.96	7.22
30	**28.76**	**9.30**	28.75	9.32
40	27.27	11.04	**27.28**	**11.03**
60	25.43	13.65	**25.43**	**13.64**
80	24.23	15.66	**24.24**	**15.64**
100	23.13	17.78	**23.14**	**17.77**

One can see that the use of the Kaiser window has no influence of the PSNR result. Visually, there is no difference between using the Kaiser window or not for all values of noise, even near edges. However, according to original authors of [35], the use of the Kaiser window is useful for large values of $p^{\mathbf{hard}}$ and $p^{\mathbf{wien}}$. But as we are working with size of patches 8×8 this step between two references patches can not be greater than 4. Then a step greater than 3 has not been tested for studying the influence of the Kaiser window.
However, for a sake of fidelity with the original method, we keep the use of a Kaiser window in the provided code.

11.5 Extending BM3D to Color Images

Adapting the algorithm to color images is easy and can be done in the following steps:

1. First a transformation to a luminance-chrominance space from the RGB noisy image is applied. Denote by Y the luminance channel and by U and V the chrominance channels;
2. For each step :
   - Grouping is only performed with the Y channel;
   - The 3D block built on Y is used for all three channels;
   - Collaborative filtering is applied to each channel separately as well as the weighted aggregation.
3. Return to the RGB space by applying the inverse transformation.

Three classic transformations were tested : the YUV transform denoted by the matrix A_{YUV}, the YCbCr space transform denoted by A_{YCbCr} and, last but not least, a more intuitive transform introduced in the original article, A_{opp}:

$$\begin{pmatrix} 0.30 & 0.59 & 0.11 \\ -0.15 & -0.29 & 0.44 \\ 0.61 & -0.51 & -0.10 \end{pmatrix} \quad \begin{pmatrix} 0.30 & 0.59 & 0.11 \\ -0.17 & -0.33 & 0.50 \\ 0.50 & -0.42 & -0.08 \end{pmatrix} \quad \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

As those matrix are not normalized, the value of σ in each channel must be carefully adapted according to these transforms. The compared denoising performance with these different color transforms is shown in the table below:

	A_{YUV}		A_{YCbCr}		A_{opp}	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	43.43	1.72	43.58	1.69	43.73	1.66
5	36.63	3.76	36.84	3.67	37.14	3.54
10	31.81	6.54	32.03	6.38	32.44	6.09

The choice of the color space transform is therefore important and causes important variations. In conclusion, also verified on the experiments below, for JPEG images the A_{opp} transform is the best choice.

Chapter 12

A Detailed Analysis and Implementation of NL-Bayes

Chapter 4 has provided a brief description of the NL-Bayes algorithm by focusing on its genesis. This method is one of the final product of this thesis, and outperforms actual state of the art algorithms. This is why this chapter will present in details its implementation, as well as an analysis of the parameters of the method. As for the other methods presented in chapters 10 and 11, an open source reliable implementation is available on line on IPOL (Image Processing On Line).
This chapter uses the results of joint work with Antoni Buades and Jean-Michel Morel.

12.1 Introduction

As we will see in this chapter, the Bayesian approach can be merged with Fourier methods like BM3D, in a new method called NL-Bayes. A natural extension of this method, called NL-PCA in the following, can be seen as we described it as a fusion of BM3D and TSID (Two-Step Image Denoising), where NL-PCA begins and ends like BM3D, the only change being the use of the PCA instead of the DCT or Bi-orthogonal spline wavelet.
We shall take advantage of the similarity of the steps of the method with BM3D, and follow closely the description used for BM3D in chapter 11 and also in [74]. Like in BM3D, each step of the NL-Bayes algorithm is realized in three parts: a) finding the image patches similar to a given image patch and grouping then in a 3D block; b) collaborative filtering; c) aggregation. The collaborative filtering is realized in two parts: a) applying Bayes' formula on the 3D block; and b) repositioning the 3D block. This 3D filtering is applied simultaneously on a group of 2D image blocks. Since these filtered patches overlap, many estimates are obtained, which need to be combined for each pixel. Aggregation is a particular averaging procedure used to take advantage of this redundancy.

12.2 Theory

This section presents a short derivation of the main formulas used in the algorithm. For a detailed analysis, see [75]. Given u the noiseless ideal image and $\tilde{u}$ the noisy image corrupted with additive white Gaussian noise of standard deviation σ so that

$$\tilde{u} = u + n, \tag{12.1}$$

the conditional distribution $\mathbb{P}(\tilde{u} \mid u)$ reads

$$\mathbb{P}(\tilde{u} \mid u) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{||u-\tilde{u}||^2}{2\sigma^2}}, \tag{12.2}$$

where N is the total number of pixels in the image. Given a noiseless patch P of u with dimension $\kappa \times \kappa$ and $\tilde{P}$ an observed noisy version of P, the independence of noise realizations for the different pixels implies that

$$\mathbb{P}(\tilde{P} \mid P) = c.e^{-\frac{\|\tilde{P}-P\|^2}{2\sigma^2}}, \tag{12.3}$$

where P and $\tilde{P}$ are considered as vectors with κ^2 components and $\|P\|$ denotes the Euclidean norm of P. Knowing $\tilde{P}$, our goal is to deduce P by maximizing $\mathbb{P}(P \mid \tilde{P})$. Using Bayes' rule, we can compute this last conditional probability as

$$\mathbb{P}(P \mid \tilde{P}) = \frac{\mathbb{P}(\tilde{P} \mid P)\mathbb{P}(P)}{\mathbb{P}(\tilde{P})}. \tag{12.4}$$

$\tilde{P}$ being observed, this formula could be used to deduce the patch P maximizing the right term, viewed as a function of P. This is unfortunately not possible, unless we have a probability model for P. We shall now discuss how to proceed when all observed patches are noisy. Assume that the patches Q similar to P follow a Gaussian model where $\mathbf{C}_P$ denotes the covariance matrix of the patches similar to P and $\overline{P}$ the expectation of the patches similar to P. This means that

$$\mathbb{P}(Q) = c.e^{-\frac{(Q-\overline{P})^t \mathbf{C}_P^{-1}(Q-\overline{P})}{2}} \tag{12.5}$$

From (12.3) and (12.4) we obtain for each observed $\tilde{P}$ the following equivalence of problems:

$$\begin{aligned}
\underset{P}{\text{Arg max}}\ \mathbb{P}(P \mid \tilde{P}) &\Leftrightarrow \underset{P}{\text{Arg max}}\ \mathbb{P}(\tilde{P} \mid P)\mathbb{P}(P) \\
&\Leftrightarrow \underset{P}{\text{Arg max}}\, e^{-\frac{\|P-\tilde{P}\|^2}{2\sigma^2}} e^{-\frac{(P-\overline{P})^t \mathbf{C}_P^{-1}(P-\overline{P})}{2}} \\
&\Leftrightarrow \underset{P}{\text{Arg min}}\ \frac{\|P-\tilde{P}\|^2}{\sigma^2} + (P-\overline{P})^t \mathbf{C}_P^{-1}(P-\overline{P}).
\end{aligned}$$

This expression is not satisfactory, because the noiseless patch P and the patches similar to P cannot be observed directly. Yet, since we are observing the noisy version $\tilde{P}$, we can at least compute the patches $\tilde{Q}$ similar to $\tilde{P}$. An empirical covariance matrix can therefore be obtained with enough such observable samples $\tilde{Q}$. P and n being independent, we deduce from (12.1), under the assumption that $\mathbf{C}_{\tilde{P}}$ is a Gaussian vector that

$$\mathbf{C}_{\tilde{P}} = \mathbf{C}_P + \sigma^2 \mathbf{I}; \qquad E\tilde{Q} = \overline{P}. \tag{12.6}$$

These relations assume that patches similar to $\tilde{P}$ have been searched in a neighbourhood large enough to include all patches similar to P, but not too large either, to avoid containing outliers. A safe strategy for that is to search for similar patches in a distance slightly larger than the plausible distance caused by noise. If the above estimates are correct, the MAP (maximum *a posteriori* estimation) problem boils down by (12.6) to the feasible minimization problem:

$$\underset{P}{\text{Arg max}}\ \mathbb{P}(P \mid \tilde{P}) \Leftrightarrow \underset{P}{\text{Arg min}}\ \frac{\|P-\tilde{P}\|^2}{\sigma^2} + (P-\overline{\tilde{P}})^t (\mathbf{C}_{\tilde{P}} - \sigma^2 \mathbf{I})^{-1} (P-\overline{\tilde{P}}).$$

Differentiating this quadratic function with respect to P and equating to zero yields

$$P - \tilde{P} + \sigma^2 (\mathbf{C}_{\tilde{P}} - \sigma^2 \mathbf{I})^{-1} (P - \overline{\tilde{P}}) = 0$$

and therefore

$$P = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2 \mathbf{I}\right] \mathbf{C}_{\tilde{P}}^{-1} (\tilde{P} - \overline{\tilde{P}}) \tag{12.7}$$

Thus a restored patch $P^{\mathbf{basic}}$ can be obtained from the observed patch $\tilde{P}$ by the one step estimation

$$P^{\mathbf{basic}} = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2 \mathbf{I}\right] \mathbf{C}_{\tilde{P}}^{-1} (\tilde{P} - \overline{\tilde{P}}). \tag{12.8}$$

In a second step, where all patches have been denoised at the first step, all the denoised patches can be used again to obtain a better unbiased estimation $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ for $\mathbf{C}_P$, the covariance of the cluster containing P, and $\overline{\tilde{P}}^{\mathbf{basic}}$ a new estimation of $\overline{\tilde{P}}$, the average of patches similar to $\tilde{P}$. Indeed, the patch similarity is better estimated with the denoised patches, then sets of similar patches are more accurate. Then it follows from (12.6) and (12.8) that we can obtain a second better denoised patch,

$$P^{\mathbf{final}} = \overline{\tilde{P}}^{\mathbf{basic}} + \mathbf{C}_{\tilde{P}}^{\mathbf{basic}} \left[\mathbf{C}_{\tilde{P}}^{\mathbf{basic}} + \sigma^2 \mathbf{I}\right]^{-1} (\tilde{P} - \overline{\tilde{P}}^{\mathbf{basic}}) \tag{12.9}$$

The computation of $\mathbf{C}_{\tilde{P}}$ and $\overline{\tilde{P}}$ will be discussed in section 12.3.

12.3 Implementation

12.3.1 BM3D

The algorithm developed in this paper is very similar to BM3D and also has two successive steps. In order to use this similarity as much as possible our notation and exposition order will be as close as possible to those used in chapter 11 and in [74]. Here is a brief overview of BM3D:

1. **Step 1: first denoising**
   Loop on the image. We denote by $\tilde{P}$ the reference current noisy patch.

   (a) Grouping: Stacking up similar patches to the reference one, using a similarity threshold applied to the distance between patches in order to build the 3D block $\mathcal{P}(\tilde{P})(\tilde{P})$;

   (b) Collaborative Filtering: A 3D linear transform is applied to the 3D block, then a hard thresholding is applied to the coefficients, and finally the inverse 3D transform is applied. A weight is associated with the whole 3D block, depending on its sparsity;

   (c) Aggregation: A first basic estimate of the denoised image is obtained by doing a weighted aggregation of every estimate obtained in the preceding step for each pixel. This basic estimate will be denoted by $u^{\mathbf{basic}}$.

2. **Step 2: second denoising step using the result of the first as "oracle"**

   (a) Grouping: The distance between patches is computed on the basic estimate. Two 3D blocks are formed :

   - $\mathcal{P}(\tilde{P})^{\mathbf{basic}}(P^{\mathbf{basic}})$ by stacking up patches from the basic estimation $u^{\mathbf{basic}}$ and,
   - $\mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})$ by stacking up patches in the same order from the original noisy image $\tilde{u}$.

   (b) Collaborative Filtering: A 3D transform is applied on both 3D blocks, followed by a Wiener filtering of the group $\mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})$ using the empirical oracular coefficients obtained from the group $\mathcal{P}(\tilde{P})^{\mathbf{basic}}(P^{\mathbf{basic}})$, and finally by the inverse 3D transform. A weight is computed for the whole 3D block. It depends on the norm of the empirical Wiener coefficients;

   (c) Aggregation: A final estimate of the denoised image is obtained by using a weighted aggregation of every estimate obtained for each pixel. This final estimate will be denoted by $u^{\mathbf{final}}$.

12.3.2 Comparison of the Structure of NL-Bayes with BM3D

Both algorithms will be described for color images. They can also be applied on grey level images; the changes in that case will be indicated. Like BM3D, NL-Bayes is applied in two successive steps, the result of the first one serving as oracle for the second one:

1. the first step provides a basic estimate $u^{\mathbf{basic}}$ by using (12.8) during the *collaborative filtering*. Parameters in this step are denoted by the exponent **1**;
2. the second step is based both on the original noisy image $\tilde{u}$ and on the basic estimate obtained during the first step $u^{\mathbf{basic}}$ in order to apply (12.9) during the *collaborative filtering*. Parameters in this step are denoted by the exponent **2**.

The following table permits to compare steps between BM3D and NL-Bayes for color images.

Step 1		
	BM3D	NL-Bayes
Preprocessing	Transformation to the $Y_0U_0V_0$ color space	Transformation to the $Y_0U_0V_0$ color space
Scanning step between two patches	$p_{\mathbf{1}} = 3$	$p_{\mathbf{1}} = 1$
Processing already used patches	Yes	No
Grouping		
Distance between patches	Channel Y_0 Normalized quadratic distance	Channel Y_0 Normalized quadratic distance
Similarity threshold	Fixed, tabulated according to σ	-
Patches kept	$N_{\mathbf{1}}$ best	$N_{\mathbf{1}}$ best
Patches ordered	No	No
3D groups formed	One for each channel Y_0, U_0 et V_0	one for each channel Y_0, U_0 et V_0
Collaborative Filtering		
3D transform	2D Bior1.5 on each patch followed by 1D Hadamard transform along the third dimension	-
Filter	Hard thresholding on the coefficients of the DCT	Bayesian, based on (12.12)
Weighting	Depending on the number of non-zero coefficients after Hard thresholding	-
Aggregation	Identical part	
Post processing	-	Transform to the RGB color space
Step 2		
	BM3D	NL-Bayes
Step between two patches	$p_{\mathbf{2}} = 3$	$p_{\mathbf{2}} = 1$
Process of an already used patch	Yes	No
Grouping		
Distance	Channel Y_0 of $u^{\mathbf{basic}}$ Normalized quadratic distance	All channels of $u^{\mathbf{basic}}$ Normalized quadratic distance
Similarity threshold	fixed, tabulated according to the σ	Adaptive according to the distance of the $N_{\mathbf{2}}$-th best patch
Patches kept	$N_{\mathbf{1}}$ best	All
Patches ordered	No	No
3D groups formed	Two for each channel	Two
Collaborative Filtering		
3D transform	2D DCT then 1D Hadamard transform on both groups	-
Filter	Wiener filter using $u^{\mathbf{basic}}$ as oracle	Bayesian, based on (12.15)
Weighting	Depending on the norm of the empirical Wiener coefficients	-
Aggregation	Identical part	
Post processing	Transformation to the RGB color space	-

12.3.3 The First Step of NL-Bayes

Only for the first step, the noisy image $\tilde{u}$ in the usual RGB color space is converted in a different color space where an independent denoising of each channel will not create noticeable color arti-

facts. Most algorithms use the YUV system which separates the luminance and chromatic parts of the image. BM3D, uses a linear transform multiplying the RGB vector by the matrix

$$Y_oU_oV_o = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

We wrote the matrix above without normalization for readability, but the matrix is normalized to become orthonormal. In that way, σ still is the value of the standard deviation of the noise on each channel Y_0, U_0 and V_0. The transform increases the SNR of the geometric component, the Y_0 component being an average of the three colors. The geometric component is perceptually more important than the chromatic ones, and the presence of less noise permits a better performance of the algorithm in this component. The components U_0 and V_0 are differences of channels, which cancel or attenuate the signal. Thus a higher noise reduction on the chromatic components U_0 and V_0 is possible, due to their observable regularity.
We denote by $\tilde{P}$ the current reference patch with size $\kappa_{\mathbf{1}} \times \kappa_{\mathbf{1}}$ (seen as a column vector) of the noisy image $\tilde{u}$.

1. Grouping:
   The original noisy image $\tilde{u}$ is explored in a $\tilde{P}$-centered $\lambda_{\mathbf{1}} \times \lambda_{\mathbf{1}}$ neighbourhood for patches $\tilde{Q}$ similar to the reference patch $\tilde{P}$. The normalized quadratic distance between each patch $\tilde{Q}$ of the neighbourhood and the reference patch $\tilde{P}$ is computed as

   $$d^2\left(\tilde{P}, \tilde{Q}\right) = \frac{\|\tilde{P} - \tilde{Q}\|_2^2}{(\kappa_{\mathbf{1}})^2}.$$

   This distance is computed on the luminance channel Y_0 only. All patches $\tilde{Q}$ of the neighbourhood are sorted according to their distance to the reference patch $\tilde{P}$, and the $N_{\mathbf{1}}$ closest patches to $\tilde{P}$ are kept. Then three sets of similar patches -one for each channel- are built: $\mathcal{P}_{Y_0}(\tilde{P})$, $\mathcal{P}_{U_0}(\tilde{P})$ and $\mathcal{P}_{V_0}(\tilde{P})$. But the sets of similar patches for the chromatic channels are built with patches whose index are the same as for $\tilde{Q} \in \mathcal{P}_{Y_0}(\tilde{P})$, and in the same order. After this step, the same procedure is applied on each channel, but separately. For a sake of simplicity, we will describe the procedure for a generic channel.

2. Collaborative Filtering:
   Let $\mathcal{P}(\tilde{P})(\tilde{P})$ be the set of patches $\tilde{Q}$ similar to the reference patch $\tilde{P}$ obtained at the grouping step. We start by detecting if $\tilde{P}$ belongs to a homogeneous[1] area by processing the square of the standard deviation of $\mathcal{P}(\tilde{P})$:

   $$\sigma_{\tilde{P}}^2 = \frac{M_{\mathbf{1}}}{M_{\mathbf{1}} - 1}\left(\frac{1}{M_{\mathbf{1}}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\sum_{x\in\tilde{Q}}\left(\tilde{Q}(x)\right)^2 - \left(\frac{1}{M_{\mathbf{1}}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\sum_{x\in\tilde{Q}}\tilde{Q}(x)\right)^2\right) \tag{12.10}$$

   where $M_{\mathbf{1}} = N_{\mathbf{1}}(\kappa_{\mathbf{1}})^2$. Since a huge number ($M_{\mathbf{1}}$) of realizations of the variable $u(i)$ is taken into account, in a homogeneous area this random variable should be very concentrated around its mean. Thus, fixing a threshold γ close to 1,

   - if $\sigma_{\tilde{P}} \leq \gamma\sigma$, we can assume that with high probability $\tilde{P}$ belongs to a homogeneous area. In this case, the better result that can be obtained for the group is the average. Therefore, the estimate of all patches in the set of similar patches $\mathcal{P}(\tilde{P})$ is

     $$\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), \forall x \in Q^{\mathbf{basic}}, Q^{\mathbf{basic}}(x) = \frac{1}{M_{\mathbf{1}}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\sum_{y\in\tilde{Q}}\tilde{Q}(y)$$

[1] By homogeneous we mean a flat area, where there is no geometric detail.

- else, $\tilde{P}$ is assumed to contain some signal, for which a Gaussian model is built. By the law of large numbers we have

$$\mathbf{C}_{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P}) - 1} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \left(\tilde{Q} - \overline{\tilde{P}}\right)\left(\tilde{Q} - \overline{\tilde{P}}\right)^t, \qquad \overline{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \tilde{Q}. \tag{12.11}$$

  Once $\mathbf{C}_{\tilde{P}}$ and $\overline{\tilde{P}}$ have been computed, (12.8) yields an estimate for every patch in the set of similar patches,

$$\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), Q^{\mathbf{basic}} = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \beta_{\mathbf{1}} \sigma^2 \mathbf{I}\right] \mathbf{C}_{\tilde{P}}^{-1} \left(\tilde{Q} - \overline{\tilde{P}}\right). \tag{12.12}$$

  where $\beta_{\mathbf{1}}$ is a parameter of conservative attenuation close to 1.

  Remark: As $\mathbf{C}_{\tilde{P}}$ is in principle real and symmetric positive definite, it should be invertible. Nevertheless it may sometimes (but seldom) be ill-conditioned. If the computation of the inverse does not end well, the algorithm simply sets $\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), Q^{\mathbf{basic}} = Q$.

3. Aggregation:
   When the collaborative filtering is achieved, an estimate is associated with every used patch. This yields a variable number of estimates for each pixel. To take advantage of these multiple estimates an aggregation must be done. Contrary to BM3D, this aggregation is not weighted. The final estimate after this first step is given by

$$u^{\mathbf{basic}}(x) = \frac{\displaystyle\sum_{\tilde{P}} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \chi_{\tilde{Q}}(x) Q^{\mathbf{basic}}(x)}{\displaystyle\sum_{\tilde{P}} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \chi_{\tilde{Q}}(x)}$$

   with $\chi_{\tilde{Q}}(x) = 1$ if and only if $x \in \tilde{Q}$, 0 otherwise.

   Remark: For practical purposes, this computation is simplified by using two buffers $\boldsymbol{\nu}$ and $\boldsymbol{\delta}$, where respectively the numerator and the denominator are kept in memory

$$\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), \forall x \in \tilde{Q}, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + Q^{\mathbf{basic}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + 1 \end{cases}$$

   Thus the final estimate is simply obtained by dividing both buffers element-by-element.

4. Acceleration: To speed up the algorithm, each patch that has been used (and therefore denoised at least once) in a 3D group is no more considered as reference patch $\tilde{P}$. Nevertheless, it may be denoised several times, being potentially chosen in other groups.

Once $u_{Y_0}^{\mathbf{basic}}$, $u_{U_0}^{\mathbf{basic}}$ and $u_{V_0}^{\mathbf{basic}}$ have been obtained, inverting the color transform yields back $u^{\mathbf{basic}}$, the first estimate of the image in the *RGB* color space.

12.3.4 Second Step of NL-Bayes

In this second step of the algorithm a basic estimate $u^{\mathbf{basic}}$ of the denoised image is available. The second step follows exactly the same scheme as the first, but performs a Wiener filter of the original noisy image $\tilde{u}$, using as oracle the basic estimate $u^{\mathbf{basic}}$.

1. Grouping:
   The patch matching is processed on the basic estimate only. But this time the distance involves all channels, which are assumedly denoised by the first step:

$$\forall Q^{\mathbf{basic}}, d^2(P^{\mathbf{basic}}, Q^{\mathbf{basic}}) = \frac{1}{N_c} \sum_{c=1}^{N_c} \frac{\|P_c^{\mathbf{basic}} - Q_c^{\mathbf{basic}}\|_2^2}{(k_{\mathbf{2}})^2} \tag{12.13}$$

where N_c denotes the number of channels in the image. As a difference with the first step where only $N_{\mathbf{1}}$ patches were kept, here a threshold τ is used to obtain a set of similar patches

$$\mathcal{P}(\tilde{P})^{\mathbf{basic}}(P^{\mathbf{basic}}) = \{Q^{\mathbf{basic}} : d^2(P^{\mathbf{basic}}, Q^{\mathbf{basic}}) \leq \tau\},$$

with

- $\tau = \max(\tau_0, d_{N_{\mathbf{2}}})$;
- τ_0 is a fixed parameter;
- $d_{N_{\mathbf{2}}}$ is the distance between the reference patch and its $N_{\mathbf{2}}$-th best similar patches, sorted by their distance to $P^{\mathbf{basic}}$.

Thus, using τ, many more similar patches can be picked in homogeneous areas. A second set of similar patches from the noisy image $\tilde{u}$ is built

$$\mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P}) = \{\tilde{Q} : d^2(P^{\mathbf{basic}}, Q^{\mathbf{basic}}) \leq \tau\},$$

by stacking up patches together in the same order as $\mathcal{P}(\tilde{P})^{\mathbf{basic}}(P^{\mathbf{basic}})$.

2. Collaborative Filtering:
Once 3D-blocks are built the collaborative filtering is applied. Then by the law of large numbers,

$$\mathbf{C}_P^{\mathbf{basic}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})^{\mathbf{basic}}(P^{\mathbf{basic}}) - 1} \sum_{Q^{\mathbf{basic}} \in \mathcal{P}(\tilde{P})^{\mathbf{basic}}(P^{\mathbf{basic}})} \left(Q^{\mathbf{basic}} - \overline{\tilde{P}}^{\mathbf{basic}}\right)\left(Q^{\mathbf{basic}} - \overline{\tilde{P}}^{\mathbf{basic}}\right)^t,$$

$$\overline{\tilde{P}}^{\mathbf{basic}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})} \tilde{Q}. \tag{12.14}$$

Once $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ and $\overline{\tilde{P}}^{\mathbf{basic}}$ are computed, (12.9) yields an estimate for every patch in the set of similar patches

$$\forall \tilde{Q} \in \mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P}), Q^{\mathbf{final}} = \overline{\tilde{P}}^{\mathbf{basic}} + \mathbf{C}_{\tilde{P}}^{\mathbf{basic}} \left[\mathbf{C}_{\tilde{P}}^{\mathbf{basic}} + \beta_{\mathbf{2}}\sigma^2\mathbf{I}\right]^{-1} \left(\tilde{Q} - \overline{\tilde{P}}^{\mathbf{basic}}\right) \tag{12.15}$$

3. Aggregation:
When the collaborative filtering is achieved, an estimate is associated with every used patch and therefore a variable number of estimates for every pixel. Once again, contrarily to BM3D, this aggregation is not weighted. The final estimate after this second step is given by

$$u^{\mathbf{final}}(x) = \frac{\displaystyle\sum_{\tilde{P}} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})} \chi_{\tilde{Q}}(x) Q^{\mathbf{final}}(x)}{\displaystyle\sum_{\tilde{P}} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})} \chi_{\tilde{Q}}(x)}$$

with $\chi_{\tilde{Q}}(x) = 1$ if and only if $x \in \tilde{Q}$, 0 otherwise.

Remark: For practical purposes, this computation is simplified by using two buffers $\boldsymbol{\nu}$ and $\boldsymbol{\delta}$, where respectively the numerator and the denominator are kept in memory

$$\forall \tilde{Q} \in \mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P}), \forall x \in \tilde{Q}, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + Q^{\mathbf{final}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + 1 \end{cases}$$

Thus the final estimate is simply obtained by dividing both buffers element-by-element. Once again, in order to speed up the algorithm, each patch used once in a 3D group is no more processed as reference patch. It can be used anyway several times as secondary patch in other 3D blocks.

12.3.5 NL-PCA: a Particular Case of NL-Bayes

The Bayesian approach on which NL-Bayes is based can lead to formulate another algorithm, which we shall call NL-PCA. Its idea is to perform a Principal Component Analysis (PCA) on the 3D group $\mathcal{P}(\tilde{P})(\tilde{P})$. This idea was first proposed in [142] and is also studied in detail in [43]. NL-PCA is obtained by replacing in BM3D the fixed linear transform (DCT, bi-orthogonal spline wavelet) by a learnt basis for each patch, obtained by PCA on the patches of the 3D block. Indeed, applying a PCA to the 3D group amounts to diagonalize its covariance matrix, and the eigenvectors give the adaptive basis. In continuation, the collaborative filtering and the aggregation parts can be applied exactly like in BM3D.The algorithm developed in [43] is very close to the first step of the NL-PCA algorithm described hereafter.
Diagonalizing the covariance matrix $\mathbf{C}_{\tilde{P}}$, denoting the associated isometry by $R_{\mathbf{1}}$ and denoting by $S_{\mathbf{1}}(j)$ the squares of the associated eigenvalues, the restoration formula (12.8) becomes on the eigenfunction basis:

$$\left(R_{\mathbf{1}}(P^{\mathbf{basic}} - \overline{\tilde{P}})\right)_j = \frac{S_{\mathbf{1}}(j) - \sigma^2}{S_{\mathbf{1}}(j)} \left(R_{\mathbf{1}}(\tilde{P} - \overline{\tilde{P}})\right)_j .$$

The only (classic) variation with respect to this estimate is that $\mathbf{C}_{\tilde{P}}$ should **be** positive semi-definite. Thus $S_{\mathbf{1}}(j) - \sigma^2$ is replaced in the above formula by $\left(S_{\mathbf{1}}(j) - \sigma^2\right)^+$ and, instead of $-\sigma^2$, a more conservative attenuation is applied, $-\beta^2\sigma^2$ where an empirical β slightly larger than 1 accounts for the error of model. A still more conservative estimate is applied for large noises (typically if $\sigma \leq 40$), where the estimate becomes

$$\left(R_{\mathbf{1}}(P^{\mathbf{basic}} - \overline{\tilde{P}})\right)_j = \begin{cases} \left(R_{\mathbf{1}}(\tilde{P} - \overline{\tilde{P}})\right)_j & \text{if } S_{\mathbf{1}}(j) \geq \beta^2\sigma^2 \\ 0 & \text{otherwise.} \end{cases}$$

In the same way, in the second step, diagonalizing $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ and denoting the associated isometry by $R_{\mathbf{2}}$ and the squares of the associated eigenvalues by $S_{\mathbf{2}}(j)$, the restoration formula (12.9) becomes, without any alteration to the model,

$$\left(R_{\mathbf{2}}(P^{\mathbf{final}} - \overline{\tilde{P}}^{\mathbf{basic}})\right)_j = \frac{S_{\mathbf{2}}(j)}{S_{\mathbf{2}}(j) + \sigma^2} \left(R_{\mathbf{2}}(\tilde{P} - \overline{\tilde{P}}^{\mathbf{basic}}\right)_j$$

which retrieves exactly a classical Wiener filter based on the PCA of $\mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})$.

12.4 Influence of the Parameters on the Performance of NL-Bayes

We applied the previously described algorithm to noiseless images to which a simulated white noise had been added. Many images have been tested, but for the sake of simplicity only one result for each σ will be shown. To evaluate quantitatively the denoising results, two classic measurements have been used :

- The *Root Mean Square Error* (RMSE) between the reference image (noiseless) u_R and the denoised image u_D. The RMSE is

$$RMSE = \sqrt{\frac{\sum_{x \in X} \left(u_R(x) - u_D(x)\right)^2}{|X|}};$$

  the smaller it is, the better the denoising.

- the *Peak Signal to Noise Ratio* (PSNR) evaluated in decibels (dB) by

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE} \right);$$

  the larger it is, the better the denoising.

The question is: how to select the right values for the various parameters in the algorithm as described previously, and to evaluate their influence on the final result? The parameters of the method are:

- $\kappa_{\mathbf{1}}$, $\kappa_{\mathbf{2}}$: size of patches;
- $N_{\mathbf{1}}$, $N_{\mathbf{2}}$: maximum number of similar patches kept;
- $\lambda_{\mathbf{1}}$, $\lambda_{\mathbf{2}}$: search window size;
- γ: used to determine if a patch belongs to an homogeneous area;
- $\beta_{\mathbf{1}}$, $\beta_{\mathbf{2}}$: coefficient used during the collaborative filtering;
- τ_0: minimum threshold to determine similar patches during the second step.

It would be impossible to try all combinations of all parameters. Thus, the principle of the study is to assign to all parameters an optimal or robust value, while only one is varied. The parameters will therefore fixed in the following way:

- $\kappa_{\mathbf{1}}$ and $\kappa_{\mathbf{2}}$ as explained in Section 12.4.1;
- $N_{\mathbf{1}}$ and $N_{\mathbf{2}}$ as explained in Section 12.4.6;
- the homogeneous area trick is always used on the first step;
- $\lambda_{\mathbf{1}} = 5 * \kappa_{\mathbf{1}}$;
- $\lambda_{\mathbf{2}} = 5 * \kappa_{\mathbf{2}}$;
- $\gamma = 1.05$;
- $\beta_{\mathbf{1}} = 1.0$;
- $\beta_{\mathbf{2}} = \begin{cases} 1.2 & \text{if } \sigma < 50 \\ 1.0 & \text{otherwise.} \end{cases}$;
- $\tau_0 = 4$.

12.4.1 Influence of the Size of the Patches κ_1 and κ_2

The size of the patches influences the result, but unlike BM3D, NL-Bayes gives its best results for significantly smaller patch sizes. This fact may be the most surprising result of this comparison.

$\sigma = 2$							$\sigma = 5$					
κ_2	3		5		7		3		5		7	
κ_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	**46.03**	**1.27**	45.96	1.28	45.88	1.30	**39.89**	**2.58**	39.83	2.60	39.75	2.62
5	**46.03**	**1.27**	45.92	1.29	45.83	1.30	**39.89**	**2.58**	39.77	2.62	39.68	2.65
7	46.00	1.28	45.89	1.29	45.78	1.31	39.87	2.59	39.73	2.63	39.61	2.67
$\sigma = 10$							$\sigma = 20$					
κ_2	3		5		7		3		5		7	
κ_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	35.38	4.34	35.34	4.36	35.27	4.40	31.16	7.06	**31.20**	**7.02**	31.15	7.06
5	**35.39**	**4.34**	35.29	4.39	35.20	4.43	**31.20**	**7.02**	31.14	7.07	31.08	7.12
7	35.36	4.35	35.24	4.41	35.11	4.48	31.16	7.06	31.10	7.10	30.99	7.20
$\sigma = 30$							$\sigma = 40$					
κ_2	3		5		7		3		5		7	
κ_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	28.81	9.25	**28.92**	**9.13**	28.90	9.15	27.18	11.16	27.33	10.97	27.34	10.95
5	28.90	9.15	28.89	9.16	28.85	9.21	27.33	10.97	**27.36**	**10.93**	27.34	10.95
7	28.85	9.21	28.85	9.21	28.76	9.30	27.31	10.99	27.35	10.94	27.28	11.03
$\sigma = 60$							$\sigma = 80$					
κ_2	3		5		7		3		5		7	
κ_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	24.90	14.51	25.30	13.85	25.38	13.73	23.15	17.74	23.74	16.58	23.94	16.20
5	25.27	13.90	25.46	13.60	**25.47**	**13.58**	23.78	16.50	24.02	16.05	24.08	15.94
7	25.26	13.92	25.45	13.62	25.44	13.63	23.83	16.41	24.09	15.92	**24.10**	**15.91**
$\sigma = 100$												
κ_2	3		5		7							
κ_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE						
3	21.79	20.75	22.62	18.86	22.88	18.30						
5	22.65	18.79	22.98	18.09	23.06	17.93						
7	22.75	18.58	23.06	17.93	**23.11**	**17.83**						

In **bold** the best result for a given σ.

Unsurprisingly, from the examination of the above set of tables follows that the size of patches must increase with the noise value. According to these results, the following optimal values will be chosen for the size of patches:

σ	$0 \leq \sigma < 20$	$20 \leq \sigma < 50$	$50 \leq \sigma < 70$	$70 \leq \sigma$
κ_1	3	5	7	7
κ_2	3	3	5	7

12.4.2 Influence of γ

The parameter γ is used to determine if a set of similar patches belongs to a homogeneous area as defined in (12.10). This parameter must be fixed carefully. If its value is too big, small details in the image may be lost. If instead its value is too small, homogeneous area will not be denoised enough and artifacts can become conspicuous in these regions. Thus, although the PSNR gain is moderate, the visual impact of this step is important. The comparison table follows.

	$\gamma = 0.95$		$\gamma = 1.0$		$\gamma = 1.05$		$\gamma = 1.1$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**46.00**	**1.28**	45.99	1.28	**46.00**	**1.28**	**46.00**	**1.28**
5	39.88	2.58	**39.89**	**2.58**	**39.89**	**2.58**	**39.89**	**2.58**
10	**35.38**	**4.34**	**35.38**	**4.34**	**35.38**	**4.34**	35.37	4.35
20	31.20	7.02	31.23	7.00	**31.24**	**6.99**	31.23	7.00
30	28.83	9.22	28.90	9.15	28.91	9.14	**28.92**	**9.13**
40	27.25	11.07	27.35	10.94	**27.38**	**10.90**	27.37	10.91
60	25.33	13.80	25.44	13.68	**25.45**	**13.61**	25.40	13.69
80	24.12	15.86	**24.20**	**15.72**	24.16	15.79	24.04	16.01
100	23.15	17.75	**23.23**	**17.58**	23.19	17.65	22.99	18.08

In **bold** best result for a given σ.

One can deduce from the above table that small variations on γ lead to significant variations for high noise. According to this study, γ can be fixed to 1.05, whatever the value of noise.

12.4.3 Influence of β_1

This parameter is used in (12.12) and influences the filtering during the first step. The theoretical value is $\beta_1 = 1.0$, but a study of its influence needs to be done to learn its best empirical value.

	$\beta_1 = 0.8$		$\beta_1 = 0.9$		$\beta_1 = 1.0$		$\beta_1 = 1.1$		$\beta_1 = 1.2$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.92	1.29	45.98	1.28	46.01	1.28	46.02	1.27	**46.03**	**1.27**
5	39.71	2.64	39.80	2.61	39.85	2.59	**39.88**	**2.59**	39.86	2.59
10	35.22	4.42	35.34	4.36	35.39	4.33	**35.40**	**4.33**	35.37	4.35
20	30.96	7.22	31.12	7.08	**31.20**	**7.02**	31.18	7.03	31.09	7.11
30	28.64	9.43	28.85	9.21	**28.92**	**9.13**	28.89	9.16	28.75	9.31
40	27.07	11.30	27.27	11.05	**27.34**	**10.95**	27.30	11.01	27.13	11.22
60	25.07	14.23	25.38	13.72	**25.46**	**13.59**	25.36	13.75	25.08	14.21
80	23.80	16.45	24.13	15.94	**24.19**	**15.73**	24.08	15.93	23.78	16.50
100	22.85	18.37	23.09	17.86	**23.12**	**17.80**	23.00	18.06	22.70	18.67

In **bold** best result for a given σ.

The theoretical value is confirmed by this study. Thus β_1 is fixed to 1.0, whatever the value of the noise.

12.4.4 Influence of β_2

This parameter is used in (12.15) influences the filtering during the first step. The theoretical value is $\beta_2 = 1.0$, but a study of its influence needs to be done to learn its best empirical value.

	$\beta_2 = 0.8$		$\beta_2 = 0.9$		$\beta_2 = 1.0$		$\beta_2 = 1.1$		$\beta_2 = 1.2$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.91	1.29	45.95	1.28	45.98	1.28	46.00	1.28	**46.01**	**1.27**
5	39.75	2.62	39.79	2.61	39.82	2.60	39.84	2.60	**39.86**	**2.59**
10	35.30	4.38	35.34	4.36	35.36	4.35	35.38	4.34	**35.39**	**4.33**
20	31.13	7.08	31.17	7.05	31.19	7.03	31.19	7.03	**31.20**	**7.02**
30	28.85	9.20	28.87	9.18	28.89	9.16	**28.90**	**9.15**	28.89	9.16
40	27.27	11.04	27.28	11.02	**27.31**	**10.99**	27.30	11.00	27.30	11.00
60	25.48	13.57	**25.49**	**13.56**	25.47	13.58	25.46	13.59	25.45	13.61
80	24.16	15.79	**24.17**	**15.77**	24.16	15.79	24.15	15.82	24.12	15.86
100	**23.12**	**17.80**	23.12	17.81	23.10	17.84	23.09	17.87	23.07	17.91

In **bold** best result for a given σ.

Here again the theoretical value seems to work well, but a slight gain can be obtained if β_2 is tabulated according to σ. The chosen value for β_2 will be 1.2 if $\sigma > 50$, 1.0 otherwise.

12.4.5 Influence of the Size of the Search Window λ_1 and λ_2

The size of the search window influences the grouping part of the algorithm. Since the total number of patches $\tilde{Q}$ contained in the neighbourhood which needs to be sorted is proportional to the size of the search window, the computational time of the algorithm increases with λ_1 and λ_2. We would therefore like to minimize these numbers. Nevertheless, if the size of the search window is too small, the "similar" patches will not be that similar to the reference patch $\tilde{P}$. Thus it is necessary to find a good compromise between a good PSNR and a relatively small size for the search window. Moreover, the size of the search window is intuitively dependant on the size of patches κ_1 and κ_2. This is why λ_1 and λ_2 will be determined as a factor of κ_1 and κ_2, i.e. $\lambda_1 = \alpha\kappa_1$ and $\lambda_2 = \alpha\kappa_2$. For this comparison, the others parameters are fixed as usual.

	$\alpha = 3$		$\alpha = 4$		$\alpha = 5$		$\alpha = 6$		$\alpha = 7$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.95	1.28	46.00	1.28	46.01	1.28	**46.02**	**1.27**	46.01	1.28
5	39.74	2.63	39.82	2.60	39.84	2.59	**39.86**	**2.59**	**39.86**	**2.59**
10	35.19	4.43	35.33	4.37	35.36	4.35	35.40	4.33	**35.41**	**5.32**
20	30.98	7.20	31.17	7.05	31.23	6.99	31.30	6.94	**31.32**	**6.93**
30	28.63	9.44	28.85	9.20	28.92	9.13	28.99	9.06	**29.02**	**9.02**
40	27.05	11.33	27.30	11.00	27.38	10.90	27.46	10.80	**27.50**	**10.75**
60	25.13	14.12	25.34	13.79	25.42	13.66	25.52	13.51	**25.55**	**13.46**
80	23.85	16.37	24.09	15.93	24.18	15.77	24.25	15.63	**24.28**	**15.57**
100	22.85	18.35	23.04	17.97	23.13	17.79	23.19	17.67	**23.23**	**17.57**

In **bold** best result for a given σ.

It follows from the comparison table that increasing the size of the search window improves the result by finding more similar patches. Accordingly the parameters are fixed to:

- $\lambda_{\mathbf{1}} = 7\kappa_{\mathbf{1}}$;
- $\lambda_{\mathbf{2}} = 7\kappa_{\mathbf{2}}$.

12.4.6 Influence of the Minimal Number of Closest Neighbours, $N_{\mathbf{1}}$ and $N_{\mathbf{2}}$

As we have to invert a matrix in (12.12) and (12.15), a minimal number of similar patches is needed. Otherwise this matrix will not been invertible. Thus, $N_{\mathbf{1}}$ and $N_{\mathbf{2}}$ have to be determined empirically. Moreover, they depend on the noisy image. For this reason, only the final chosen values are given here

$$N_{\mathbf{1}} = \begin{cases} 30 & \text{if } \kappa_{\mathbf{1}} = 3 \\ 60 & \text{if } \kappa_{\mathbf{1}} = 5 \\ 90 & \text{if } \kappa_{\mathbf{1}} = 7 \end{cases} ; \qquad N_{\mathbf{2}} = \begin{cases} 30 & \text{if } \kappa_{\mathbf{2}} = 3 \\ 60 & \text{if } \kappa_{\mathbf{2}} = 5 \\ 90 & \text{if } \kappa_{\mathbf{2}} = 7 \end{cases} .$$

12.4.7 Influence of τ_0

This parameter is used only in the second step, to fix the minimum threshold between two similar patches. Indeed in the second step we got an estimate $u^{\mathbf{basic}}$ and the distances between patches are better estimated on $u^{\mathbf{basic}}$ than on $\tilde{u}$. Thus, in homogeneous areas we can allow for many more similar patches than $\lambda_{\mathbf{2}}$. The parameter τ_0 is voluntarily kept small, because otherwise patches which differ significantly from the reference patch would be considered similar.

	$\tau_0 = 0$		$\tau_0 = 2$		$\tau_0 = 4$		$\tau_0 = 8$		$\tau_0 = 16$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.99	1.28	**46.01**	**1.27**	46.00	1.28	45.94	1.29	45.81	1.30
5	39.88	2.58	**39.93**	**2.57**	39.91	2.58	39.85	2.59	39.68	2.64
10	35.36	4.35	35.44	4.31	**35.45**	**4.30**	35.39	4.34	35.21	4.42
20	31.17	7.05	31.28	6.96	**31.29**	**6.95**	31.25	6.99	31.08	7.12
30	28.82	9.23	29.01	9.03	29.02	9.02	**29.03**	**9.02**	28.89	9.16
40	27.18	11.16	27.40	10.87	27.44	10.82	**27.46**	**10.80**	27.34	10.95
60	25.43	13.64	25.55	13.46	**25.58**	**13.42**	25.54	13.47	25.44	13.64
80	24.08	15.94	24.18	15.75	**24.21**	**15.71**	24.17	15.78	24.00	16.10
100	23.16	17.72	23.28	17.48	**23.33**	**17.37**	23.31	17.42	23.08	17.88

In **bold** best result for a given σ.

Using the minimum threshold with a small value ($\tau_0 = 2$ or 4) is always better than ($\tau_0 = 0$). Moreover, as expected, a too large value ($\tau_0 = 16$) gives really worse results. According to this comparison, we shall set $\tau_0 = 4$.

12.4.8 Summary Table of the Best Parameters

Here is the summary table with the final chosen values for all parameters, depending on the value of the noise:

σ	$0 \leq \sigma < 20$	$20 \leq \sigma < 50$	$50 \leq \sigma < 70$	$70 \leq \sigma$
κ_1	3	5	7	7
κ_2	3	3	5	7
γ	1.05	1.05	1.05	1.05
β_1	1.0	1.0	1.0	1.0
β_2	1.2	1.2	1.0	1.0
λ_1	21	35	49	49
λ_2	21	21	35	49
N_1	30	60	90	90
N_2	30	30	60	90
τ_0	4	4	4	4

12.5 A Detailed Study of the Algorithm

This part discusses several sound variants for each step of the algorithm. It gives experimental evidence that the choices taken in the algorithm are the best in terms of PSNR.

12.5.1 Grouping

Color Space Transform

The $Y_0U_0V_0$ space will be compared to RGB, for both steps. When RGB is chosen in the algorithm,

- the distance is computed on all channels like in (12.13);
- the collaborative filtering is done on each channel separately;
- N_1 is increased to avoid having a non-invertible matrix.

Step 1 / Step 2	RGB / RGB		$Y_0U_0V_0$ / RGB		RGB / $Y_0U_0V_0$		$Y_0U_0V_0$ / $Y_0U_0V_0$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**46.10**	**1.26**	46.04	1.27	45.83	1.30	45.80	1.31
5	**39.89**	**2.58**	39.86	2.59	39.66	2.65	39.64	2.66
10	35.39	4.33	**35.42**	**4.32**	35.22	4.42	35.25	4.41
20	31.10	7.10	**31.30**	**6.94**	31.00	7.19	31.18	7.03
30	28.73	9.33	**28.93**	**9.11**	28.65	9.41	28.85	9.20
40	27.23	11.08	**27.47**	**10.78**	27.18	11.15	27.43	10.84
60	*23.87*	*16.34*	**25.54**	**13.48**	*23.82*	*16.43*	25.52	13.50
80	*22.96*	*18.13*	24.24	15.66	*22.92*	*18.23*	**24.24**	**15.65**
100	*22.14*	*19.93*	23.24	17.56	*22.10*	*20.02*	**23.25**	**17.54**

In **bold** the best PSNR for a given σ. In *italic* results with many observed non-invertible matrices in the first step.

Despite the fact that N_1 has been increased in the case where RGB is chosen for the first step, in practice the matrix $\mathbf{C}_{\tilde{P}}$ is often not invertible. This explains why the result is that bad for high values of the noise when RGB is chosen for the first step.

12.5.2 Collaborative Filtering

The Homogeneous Area Criterion

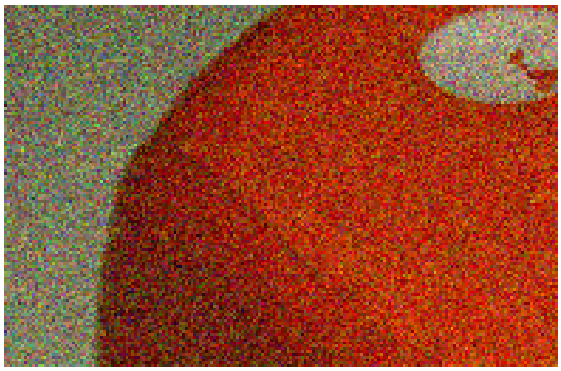
One of the innovations in NL-Bayes is the detection of the homogeneous areas in the first step. In order to show its relevance, here are some results with and without this criterion, in both steps:

Step 1 / Step 2	No / No		Yes / No		No / Yes		Yes / Yes	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**46.04**	**1.27**	**46.04**	**1.27**	**46.04**	**1.27**	**46.04**	**1.27**
5	39.89	2.58	**39.90**	**2.57**	39.88	2.58	39.89	2.58
10	35.36	4.35	**35.39**	**4.33**	35.37	4.34	35.35	4.35
20	31.05	7.14	31.27	6.96	**31.29**	**6.95**	31.26	6.98
30	28.68	9.38	**29.01**	**9.04**	**29.01**	**9.04**	28.97	9.08
40	27.05	11.32	**27.48**	**10.78**	27.47	10.79	27.42	10.85
60	25.39	13.70	**25.58**	**13.41**	25.55	13.45	25.45	13.62
80	24.14	15.83	**24.26**	**15.62**	24.16	15.80	23.98	16.12
100	23.09	17.86	**23.27**	**17.49**	23.15	17.73	22.93	18.20

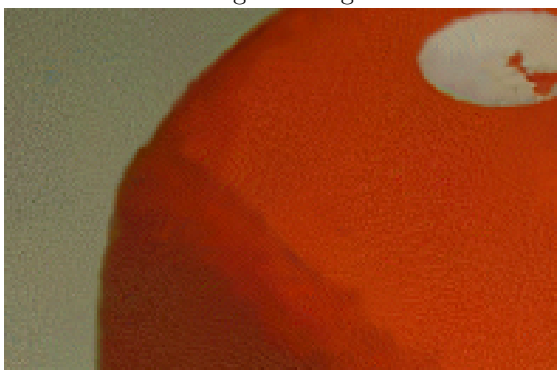
In **bold** the best PSNR for a given σ.

One can see that the homogeneity detection is useful for medium and high values of the noise variance. On an image with many homogeneous areas, the application of this detection avoids artifacts, as one can see on the following image for a noise standard deviation equal to 20.

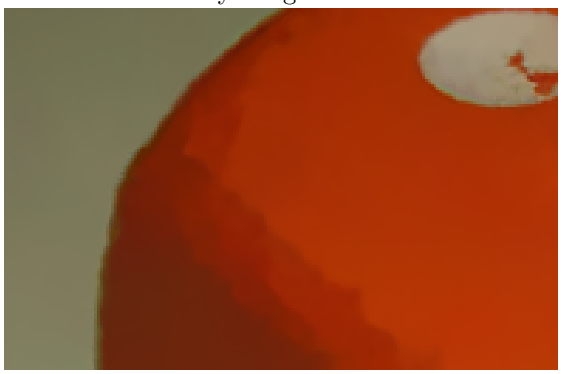
Original image



Noisy image $\sigma = 30$



No / No | PSNR = 33.93 / RMSE = 5.13



Yes / No | PSNR = 38.82 / RMSE = 2.92

No / Yes | PSNR = 38.69 / RMSE = 2.96

Yes / Yes | PSNR = 38.36 / RMSE = 3.08

Diagonalizing the Covariance Matrix

As presented in section 12.3.5, $\mathbf{C}_{\tilde{P}}$ and $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ can be diagonalized with the use of a PCA on the 3D block $\mathcal{P}(\tilde{P})(\tilde{P})$ and $\mathcal{P}(\tilde{P})^{\mathbf{basic}}(\tilde{P})$, which leads to an algorithm which we called NL-PCA. The principal difference between NL-PCA and NL-Bayes is in the collaborative filtering part, as detailed in 12.3.5. All the rest is exactly the same. We compare here the results of NL-PCA to NL-Bayes on three (noiseless) images, to which noise was added. The images are displayed below.

Dice

Flowers

Traffic

	Dice				Flowers				Traffic			
	NL-PCA		NL-Bayes		NL-PCA		NL-Bayes		NL-PCA		NL-Bayes	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	48.85	0.92	**49.08**	**0.90**	47.59	1.06	**47.77**	**1.04**	45.00	1.43	**45.25**	**1.39**
5	45.84	1.30	**45.92**	**1.29**	43.35	1.73	**43.41**	**1.72**	39.42	2.73	**39.59**	**2.67**
10	43.17	1.77	**43.31**	**1.74**	39.66	2.65	**39.83**	**2.60**	35.40	4.33	**35.49**	**4.29**
20	**40.68**	**2.36**	40.56	2.39	36.33	3.89	**36.42**	**3.85**	31.28	6.96	**31.48**	**6.80**
30	**38.83**	**2.92**	38.81	2.92	34.16	5.00	**34.20**	**4.97**	29.20	8.84	**29.34**	**8.70**
40	**37.35**	**3.46**	**37.35**	**3.46**	32.52	6.03	**32.59**	**5.98**	27.80	10.39	**27.87**	**10.30**
60	35.60	4.23	**35.62**	**4.22**	30.72	7.42	**30.84**	**7.32**	25.77	13.12	**26.01**	**12.77**
80	34.53	4.79	**34.60**	**4.75**	29.28	8.76	**29.47**	**8.57**	24.62	14.98	**24.75**	**14.76**
100	33.46	5.41	**33.48**	**5.40**	28.15	9.98	**28.26**	**9.85**	23.76	16.54	**23.85**	**16.37**

In **bold** the best PSNR for a given σ.

The table shows that NL-Bayes is slightly better than NL-PCA. Nevertheless, the results of NL-PCA could be improved by adapting the parameters, and adding a weight to the aggregation part, like BM3D does. Indeed, the values of $N_{\mathbf{1}}$ and $N_{\mathbf{2}}$ are not adapted to NL-PCA: this parameter has large values to avoid non-invertible matrices, but the PCA can be hedged if patches in $\mathcal{P}(\tilde{P})(\tilde{P})$ are not that similar. Moreover, $\beta_{\mathbf{1}}$ and $\beta_{\mathbf{2}}$ need to be adapted too. NL-Bayes is faster than NL-PCA by an average factor of 50%, due to the fact that there is only one matrix inversion, and not a PCA.

Ideal Wiener and Upper Bounds for the Performance

There is a significantly PSNR improvement by using the second step, because the covariance matrix is better estimated. To judge the contribution of this second step better, it is possible to compare it with an ideal Wiener filter, which is obtained when the original noise-free image is taken as oracle reference, i.e. as the output of the first step. This ideal Wiener filter is the best possible estimate for the second step of this algorithm. It is therefore interesting to see how far we stand from this ideal estimate with the current one:

	Algorithm		Ideal Wiener	
σ	PSNR	RMSE	PSNR	RMSE
2	45.98	1.28	**47.40**	**1.09**
5	39.89	2.58	**41.50**	**2.15**
10	35.45	4.30	**37.28**	**3.49**
20	31.26	6.97	**33.42**	**5.44**
30	28.96	9.09	**31.42**	**6.85**
40	27.46	10.81	**30.13**	**7.95**
60	25.58	13.41	**27.97**	**10.19**
80	24.29	15.57	**26.66**	**11.85**
100	23.25	17.55	**25.80**	**13.08**

In **bold** the best PSNR for a given σ.

Of course, the Wiener filter using the noise-free image as oracle is better than the filter using the basic estimate obtained after the first step. One can see that there is a large room for improvement for this first step, of 2 to 3 dBs. The images below compare the visual performance for different values of the noise.

Original image

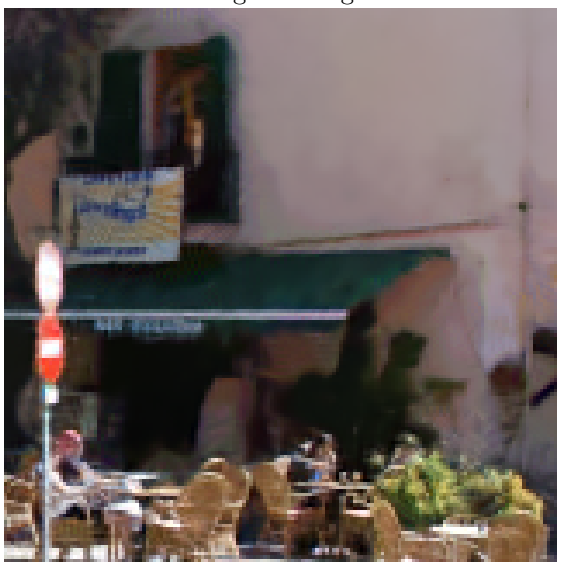
Noisy image ($\sigma = 10$)

NL-Bayes ($\sigma = 10$)

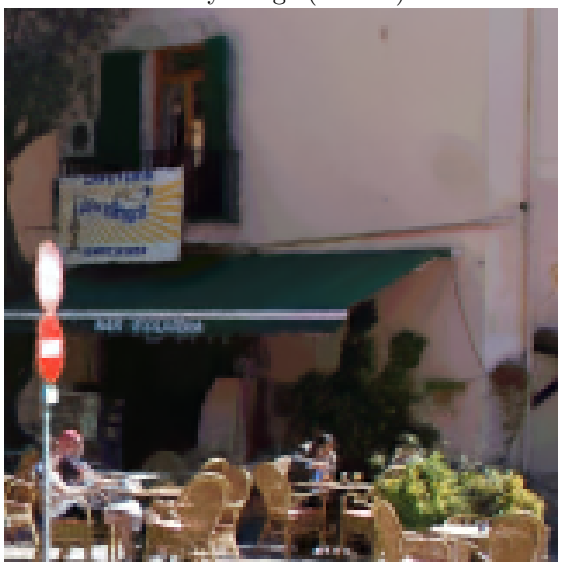
Ideal Wiener ($\sigma = 10$)

Original image

Noisy image ($\sigma = 30$)

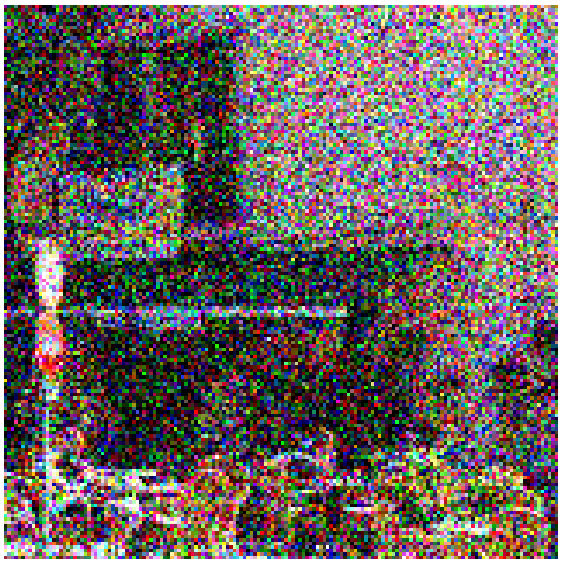


NL-Bayes ($\sigma = 30$)



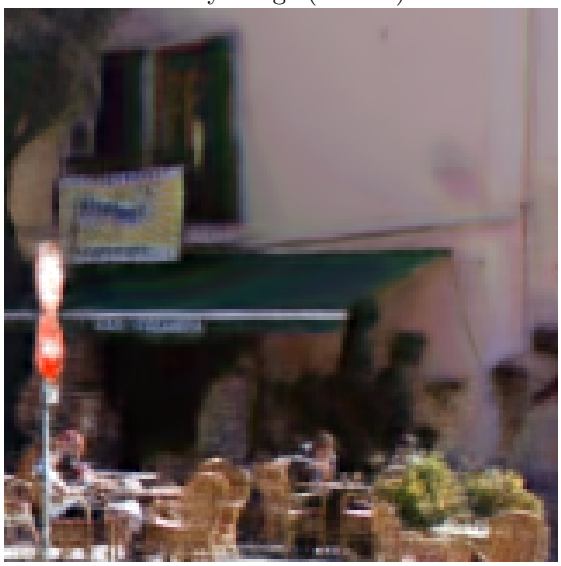
Ideal Wiener ($\sigma = 30$)

Original image



Noisy image ($\sigma = 80$)

NL-Bayes ($\sigma = 80$)



Ideal Wiener ($\sigma = 80$)

12.5.3 The "Paste" Option

To speed up the algorithm, a paste trick has been used. Whenever a patch $\tilde{Q} \in \mathcal{P}(\tilde{P})(\tilde{P})$ has an estimate, it is no more processed as a reference patch. One could think that this trick produces artifacts and has an impact on the PSNR. To check that, we will compare for both steps this paste option, denoted by "*paste*" in the following, and another trick used in BM3D. This other trick, denoted by "*step*" divides approximately the number of processed reference patches by nine, by taking a 3 pixels scanning step row and column.

Step 1 / Step 2	*Step* / *Step*		*Paste* / *Step*		*Step* / *Paste*		*Paste* / *Paste*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.99	1.28	46.00	1.28	**46.03**	**1.27**	**46.03**	**1.27**
5	39.81	2.61	39.84	2.59	**39.87**	**2.59**	**39.87**	**2.59**
10	35.33	4.36	35.40	4.33	35.41	4.32	**35.43**	**4.31**
20	31.25	6.98	31.27	6.96	31.30	6.94	**31.31**	**6.93**
30	28.97	9.08	28.98	9.06	29.01	9.03	**29.02**	**9.02**
40	27.35	10.94	27.39	10.89	27.40	10.87	**27.43**	**10.84**
60	25.47	13.58	25.52	13.51	25.51	13.53	**25.53**	**13.48**
80	24.16	15.79	24.22	15.68	24.20	15.73	**24.24**	**15.66**
100	23.24	17.55	**23.28**	**17.47**	23.26	17.52	23.27	17.50

In **bold** the best PSNR for a given σ.

There is no significant PSNR loss by using both acceleration tricks. The fact that the use of the "*paste*" option improves things could be explained by the fact that in a area with many details, patches are very different in a very small area. Then almost every patch will be treated as a reference patch, while with the "*step*" trick only 1 patch out of 9 will be treated, and then very small details will be processed worse. The use of this "*paste*" trick speeds up a bit more the algorithm than the "*step*" trick, by almost 30%, depending on the percentage of homogeneous areas in the noisy image.

12.5.4 Influence of the Second Step

Using a second step to obtain a better covariance matrix improves a lot the final result, as we can see in the following table:

	First step		Second step	
σ	PSNR	RMSE	PSNR	RMSE
2	45.65	1.33	46.03	1.27
5	39.41	2.73	39.88	2.58
10	34.84	4.62	35.45	4.30
20	30.47	7.64	31.24	6.99
30	28.15	9.97	28.99	9.06
40	26.58	11.95	27.46	10.80
60	24.15	15.80	25.50	13.53
80	22.87	18.32	24.25	15.64
100	21.92	20.43	23.27	17.51

12.6 Conclusion

This detailed study carried out on NL-Bayes had led us to the following conclusions:

- working on very small size of patches, this method is really fast for small and medium noise, even more thanks to the acceleration trick that used patches are no more processed again as reference patches;
- The main elements which allow for real improvements of the results are:
  - working in the $Y_0U_0V_0$ space color for the first step;
  - making a second step with the result of the first step as "oracle";
  - aggregating the estimates. This aggregation is improved significantly by working with a 3D group and by keeping all estimates obtained of the similar 2D-patches like BM3D;
  - using the homogeneous area criteria to remove almost all artifacts in homogeneous area.
- Nevertheless, there is still room for improvement, by, perhaps:
  - using a multi-scale approach to remove low frequency noise;
  - using a more complex model than the Gaussian patch model.

Conclusion

This tour of image denoising has shown us that this field is a "hot topic", really vast, extensively studied in the case of white Gaussian noise, and that some main principles start to arise. But surprisingly, when it comes to blind denoising, which is the direct application of denoising to real images, images on which each and every one daily works on, research comes up to be extremely rare. Moreover, even if the theory has seemed several times to converge to a final solution and performance, it appears that denoising results are still improving, despite the fact that at each major novelty in patch-based methods (BM3D, NL-Bayes, NLDD) one could think that it will be impossible (or really difficult) to achieve better results. However, in the light of the gap between methods that work only on the noisy image itself, and "shotgun" or neural network denoiser (that demands a lot of computational time), I think that the future of denoising will be "cloud denoising", where images will be denoised thanks to connected computers and huge image databases.
On the same spirit, whereas most articles and papers about denoising show PSNR (or SSIM or equivalent measures) table values, real results are rarely presented, are too often small images and most of the time in gray level. As far as we know, those measures are only indicative, and unfortunately do not reflect perfectly the visual result of the method. Therefore there is a real need to develop an objective measure which better fits the subjective (human) visual comparison. This is why an on line web-demo, such as the ones IPOL provides, and available open source codes are so important for the impact of a method. To be able to run an algorithm on a common set of images or on its own favourite images is crucial for a researcher to make its (fair) own idea about a method. Moreover, having an open source code allows everyone (researcher, engineer, ...) to save considerable time by avoiding to re-code entirely a method, and to be sure that everybody works on the same trustworthy algorithm. However, this kind of web demo is still not perfect yet: if we can test one particularly method on any image, it is still difficult when it comes to compare a bunch of algorithms.This has still to be done manually, but a web demo might play this benchmark role when the number of online algorithms is sufficient.
One can observe how IPOL's demonstration and publication style is far from a standard. Users even need to be educated: on IPOL archives about denoising algorithms, one can find many noisy input images. This is non-sense, since in these online demos noise has to be added by the demo itself to the input image before denoising. This fact clearly shows that blind denoising methods have a brilliant interest and future. Publishing blind algorithms will enable users to test on already noisy images, without having to first obtain noise-free image by their own. The best of both worlds is then very close, but has to be done: we should couple a white noise estimation to any white Gaussian noise denoiser algorithm. It will allow users to test and compare any method on any image. The only remaining problem would be to develop a reliable variance stabilizing transform, to transform the signal-dependent noise estimated by the noise estimater into a signal-independent noise that the denoiser could work on without any modifications of its algorithms.
As part of future work, adapting the Noise Clinic to movies should be relatively easy, as for any patch-based denoising algorithm: the considered neighborhood is then not only the current image, but the neighborhood of all neighboring frames. However, the best achievable result that can be obtained temporally still is a temporal aggregation applied before any denoising. This has to be done carefully, to avoid "ghosts" to appear due to movement in the scene. The HDR (High Dynamic Range) format is currently addressing this challenge. This also means that denoising will have to be adapted to video produced by HDR cameras.
Another enhancements can still be done in pure denoising: as we have seen in this thesis, it is really easy to deal with homogeneous areas, but the real problem is to detect those areas and to apply a different and intelligent treatment to areas with details. And as details become more difficult to retrieve in the noisy image itself, particularly when they are unique in the scene, we will be forced at the end to use "cloud" images and perform global denoising.

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on image processing*, pages 9–12, 2005. http://dx.doi.org/10.1109/TSP.2006.881199.

[2] G. Allaire and S. M. Kaber. *Algèbre Linéaire Numérique*. Ellipses, Paris, 2002. ISBN:2729810013.

[3] F. J. Anscombe. The transformation of Poisson, binomial and negative-binomial data. *Biometrika*, 35(3):246–254, 1948.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *Image Processing, IEEE Transactions on*, 1(2):205–220, 1992.

[5] F. Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183–193, 1954.

[6] S.P. Awate and R.T. Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):364–376, 2006.

[7] R. A. Boie and I. J. Cox. An analysis of camera noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):671–674, 1992.

[8] J.S. De Bonet. Noise reduction through detection of signal redundancy. *Rethinking artificial intelligence*, 1997.

[9] R. Bracho and A.C. Sanderson. Segmentation of images based on intensity gradient information. In *Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, pages 19–23, 1985.

[10] P. Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Verlag, 1999.

[11] A. Buades. *Image and film denoising by non-local means*. PhD thesis, 2006.

[12] A. Buades, B. Coll, and J.M. Morel. Image data process by image noise reduction and camera integrating the means for implementing this process. *French Patent 0404837*.

[13] A. Buades, B. Coll, and J.M. Morel. A non local algorithm for image denoising. *IEEE Computer Vision and Pattern Recognition*, 2:60–65, 2005. DOI: http://dx.doi.org/10.1109/CVPR.2005.38.

[14] A. Buades, B. Coll, and J.M. Morel. Nonlocal image and movie denoising. *International Journal of Computer Vision*, 76:123–139, 2008. DOI: http://dx.doi.org/10.1007/s11263-007-0052-1.

[15] A. Buades, B. Coll, and J.M. Morel. Non-Local Means Denoising. *Image Processing On Line*, 2011, 2011. http://dx.doi.org/10.5201/ipol.2011.bcm_nlm.

[16] A. Buades, B. Coll, J.M. Morel, et al. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2006. DOI: http://dx.doi.org/10.1137/040616024.

[17] A. Buades, B. Coll, J.M. Morel, and C. Sbert. Self-similarity Driven Demosaicking. *Image Processing On Line*, 1, 2011. http://dx.doi.org/10.5201/ipol.2011.bcms-ssdd.

[18] A. Buades, M. Colom, and J.M. Morel. Multiscale signal dependent noise estimation. *Image Processing On Line*.

[19] A. Buades, Y. Lou, J.M. Morel, and Z. Tang. A note on multi-image denoising. In *Local and Non-Local Approximation in Image Processing, 2009. LNLA 2009. International Workshop on*, pages 1–15. IEEE, 2009.

[20] E.J. Candès and M.B. Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30, 2008.

[21] M.A. Carreira-Perpinan. Mode-finding for mixtures of gaussian distributions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1318–1323, 2000.

[22] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20:89–97, 2004. http://dx.doi.org/10.1023/B:JMIV.0000011325.36760.1e.

[23] P. Chatterjee and P. Milanfar. Is denoising dead? *Image Processing, IEEE Transactions on*, 19(4):895–911, 2010. DOI: http://dx.doi.org/10.1109/TIP.2009.2037087.

[24] P. Chatterjee and P. Milanfar. Patch-based near-optimal image denoising. *IEEE Transactions on Image Processing*, 2012.

[25] C. Chevalier, G. Roman, and J.N. Niepce. *Guide du photographe.* C. Chevalier, 1854.

[26] A. Cohen, I. Daubechies, and J.C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 45(5):485–560, 1992.

[27] R. R. Coifman and D. L. Donoho. *Translation-invariant de-noising*, volume 103. Springer New York, 1995. http://dx.doi.org/10.1007/978-1-4612-2544-7_9.

[28] M. Colom and A. Buades. Analysis and extension of the percentile method, estimating a noise curve from a single image. *Image Processing On Line*, 2013:322–349, 2013. DOI: http://dx.doi.org/10.5201/ipol.2013.90.

[29] M. Colom and A. Buades. Analysis and Extension of the Ponomarenko et al. Method, Estimating a Noise Curve from a Single Image. *Image Processing On Line*, 3:173–197, 2013. DOI: http://dx.doi.org/10.5201/ipol.2013.45.

[30] M. Colom, A. Buades, and J.M. Morel. Nonparametric noise estimation method for raw images. *Journal of the Optical Society of America A*, 31, 2014.

[31] M. Colom, G. Facciolo, M. Lebrun, N. Pierazzo, M. Rais, Y.Q. Wang, and J.M. Morel. A mathematical perspective of image denoising. *ICM conference, 2014*, 2014. Submitted.

[32] M. Colom, M. Lebrun, A. Buades, and J.M. Morel. A non-parametric approach for the estimation of intensity-frequency dependent noise. *IEEE International Conference on Image Processing*, 2014. Submitted.

[33] M. Colom, M. Lebrun, A. Buades, and J.M. Morel. A non-parametric approach for the estimation of intensity-frequency dependent noise. *ICIP*, 2014. Submitted.

[34] S.F. Cotter, R. Adler, R.D. Rao, and K. Kreutz-Delgado. Forward sequential algorithms for best basis selection. In *Vision, Image and Signal Processing, IEE Proceedings*, volume 146, pages 235–244, 1999. http://dx.doi.org/10.1049/ip-vis:19990445.

[35] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(82):3736–3745, 2007. DOI: http://dx.doi.org/10.1109/TIP.2007.901238.

[36] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, et al. BM3D image denoising with shape-adaptive principal component analysis. *Proc. of the Workshop on Signal Processing with Adaptive Sparse Structured Representations, Saint-Malo, France*, April 2009.

[37] A. Danielyan and A. Foi. Noise variance estimation in nonlocal transform domain. In *Proceedings of International Workshop on Local and Non-Local Approximation in Image Processing, LNLA 2009*.

[38] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian. Denoising of multispectral images via nonlocal groupwise spectrum-pca. *Proc. of The fifth European Conference on Colour in Graphics, Imaging, and Vision and of the 12th International Symposium on Multispectral Colour Science held at University of Eastern Finland, Joensuu, Finland*, June 2010.

[39] A. Danielyan, V. Katkovnik, and K. Egiazarian. BM3D frames and variational image deblurring. *IEEE Transactions on Image Processing*, 2012. DOI: http://dx.doi.org/10.1109/TIP.2011.2176954.

[40] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Journal of constructive Approximation*, 13:57–98, 1997. http://dx.doi.org/10.1007/BF02678430.

[41] C.A. Deledalle, L. Denis, and F. Tupin. Nl-insar: Nonlocal interferogram estimation. *Geoscience and Remote Sensing, IEEE Transactions on*, 49(4):1441–1452, 2011.

[42] C.A. Deledalle, V. Duval, and J. Salmon. Non-local methods with shape-adaptive patches (nlm-sap). *Journal of Mathematical Imaging and Vision*, pages 1–18, 2010.

[43] C.A. Deledalle, J. Salmon, and A. Dalalyan. Image denoising with patch based PCA: local versus global. In *Proceedings of the British Machine Vision Conference*, pages 25.1–25.10. BMVA Press, 2011. DOI: http://dx.doi.org/10.5244/C.25.25.

[44] C.A. Deledalle, F. Tupin, and L. Denis. Poisson nl means: Unsupervised non local means for poisson noise. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 801–804. IEEE, 2010.

[45] C.A. Deledalle, F. Tupin, and L. Denis. Polarimetric sar estimation based on non-local means. In *Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International*, pages 2515–2518. IEEE, 2010.

[46] Donoho. Smooth wavelet decompositions with blocky coefficient kernels. *L. L. Schumaker and G. Webb Eds. Recent Advances in Wavelet Analysis. Academic Press, New York*, pages 259–308, 1993.

[47] D. Donoho and I. Johnstone. Ideal Spatial Adaptation by Wavelet Shrinkage. *Biometrika*, 81:425–455, 1993. http://dx.doi.org/10.1093/biomet/81.3.425.

[48] D.L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[49] D.L. Donoho and I.M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, pages 1200–1224, 1995.

[50] S. Durand and M. Nikolova. Restoration of wavelet coefficients by minimizing a specially designed objective function. In *Proc. IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 145–152, 2003.

[51] V. Duval, J.F. Aujol, and Y. Gousseau. A bias-variance approach for the nonlocal means. *SIAM Journal on Imaging Sciences*, 4:760, 2011.

[52] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, volume 2, pages 1033–1038. Corfu, Greece, 1999. DOI: `http://dx.doi.org/10.1109/ICCV.1999.790383`.

[53] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on image processing*, 15(12):3736–3745, 2006. `http://dx.doi.org/10.1109/TIP.2006.881969`.

[54] A. Foi. Noise estimation and removal in mr imaging: The variance-stabilization approach. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, pages 1809–1814. IEEE, 2011.

[55] A. Foi, S. Alenius, V. Katkovnik, and K. Egiazarian. Noise measurement for raw-data of digital imaging sensors by automatic segmentation of non-uniform targets. *IEEE Sensors Journal*, 7(10):1456–1461, 2007.

[56] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 17(10):1737–1754, 2008.

[57] S. Gabarda and G. Cristóbal. The generalized Rényi image entropy as a noise indicator. *Noise and Fluctuations in Photonics, Quantum Optics, and Communications*, 6603, 2007. `http://dx.doi.org/10.1117/12.725086`.

[58] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[59] P. Getreuer. Rudin-Osher-Fatemi Total Variation Denoising using Split Bregman. *Image Processing On Line*, 2012, 2012. `http://dx.doi.org/10.5201/ipol.2012.g-tvd`.

[60] O.G. Guleryuz. Weighted averaging for denoising with overcomplete dictionaries. *Transactions on Image Processing*, 16(12):3020–3034, 2007.

[61] J.L. Harris. Image evaluation and restoration. *Journal of the Optical Society of America*, 56(5):569–570, 1966.

[62] J. Hays and A.A. Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 4. ACM, 2007.

[63] Y. Hou, C. Zhao, D. Yang, and Y. Cheng. Comments on image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 20(1):268–270, 2011.

[64] Yingkun Hou, Chunxia Zhao, Deyun Yang, and Yong Cheng. Comments on "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering". *IEEE Transactions on Image Processing*, 20(1), January 2011. DOI: `http://dx.doi.org/10.1109/TIP.2010.2052281`.

[65] S.B. Howell. *Wavelets-Algorithms and applications*. Society for Industrial and Applied Mathematics, 1993.

[66] J. Immerkaer. Fast noise variance estimation. *Computer Vision and Image Understanding*, 64(2):300–302, 1996.

[67] J.F Hamilton Jr. and J.E. Adams Jr. Adaptive color plan interpolation in single sensor color electronic camera, may 1997. US Patent 5,629,734.

[68] V. Katkovnik, A. Danielyan, and K. Egiazarian. Decoupled inverse and denoising for image deblurring: variational BM3D-frame technique. In *Proceedings of IEEE International Conference on Image Processing (ICIP, 2011)*, 2011. DOI: http://dx.doi.org/10.1109/ICIP.2011.6116455.

[69] V. Katkovnik, K. Egiazarian, and J. Astola. *Local approximation techniques in signal and image processing*, volume PM157. Society of Photo Optical, 2006. DOI: http://dx.doi.org/10.1117/3.660178.

[70] S.M. Kay. Fundamentals of statistical signal processing: Estimation theory, 1993.

[71] C. Kervrann and J. Boulanger. Local Adaptivity to Variable Smoothness for Exemplar-Based Image Regularization and Representation. *International Journal of Computer Vision*, 79(1):45–69, 2008.

[72] C. Knaus and M. Zwicker. Dual-Domain Image Denoising. *IEEE ICIP*, 2013.

[73] E.D. Kolaczyk. Wavelet shrinkage estimation of certain Poisson intensity signals using corrected thresholds. *Statist. Sin.*, 9:119–135, 1999.

[74] M. Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2012, 2012. http://dx.doi.org/10.5201/ipol.2012.l-bm3d.

[75] M. Lebrun, A. Buades, and J.M. Morel. A Nonlocal Bayesian Image Denoising Algorithm. *SIAM Journal Image Science*, 6(3):1665–1688, 2013. http://dx.doi.org/10.1137/120874989.

[76] M. Lebrun, A. Buades, and J.M. Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. *Image Processing On Line*, 2013:1–42, 2013. http://dx.doi.org/10.5201/ipol.2013.16.

[77] M. Lebrun, M. Colom, A. Buades, and J.M. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21:475–576, 2012. http://dx.doi.org/10.1017/S0962492912000062.

[78] M. Lebrun, M. Colom, and J.M. Morel. Multiscale Image Blind Denoising. *Transaction on Image Processing*, 2014. Submitted.

[79] M. Lebrun, M. Colom, and J.M. Morel. The Noise Clinic, a Universal Blind Denoising Algorithm. *ICIP*, 2014. Submitted.

[80] M. Lebrun and A. Leclaire. An Implementation and Detailed Analysis of the K-SVD Image Denoising Algorithm. *Image Processing On Line*, 2012, 2012. http://dx.doi.org/10.5201/ipol.2012.llm-ksvd.

[81] A.B. Lee, K.S. Pedersen, and D. Mumford. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54(1):83–103, 2003.

[82] J.S. Lee. Refined filtering of image noise using local statistics. *Computer graphics and image processing*, 15(4):380–389, 1981.

[83] J.S. Lee. Digital image smoothing and the sigma filter. *Computer Vision, Graphics, and Image Processing*, 24(2):255–269, 1983.

[84] J.S. Lee and K. Hoppel. Noise modelling and estimation of remotely-sensed images. *Proceedings of the International Geoscience and Remote Sensing Symposium*, 2:1005–1008, 1989.

[85] S. Lefkimmiatis, P. Maragos, and G. Papandreou. Bayesian inference on multiscale models for poisson intensity estimation: Application to photo-limited image denoising. *IEEE Transactions on Image Processing*, 18(8):1724–1741, 2009.

[86] A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2833–2840. IEEE, 2011. DOI: http://dx.doi.org/10.1109/CVPR.2011.5995309.

[87] C. Liu, W. Freeman, R. Szeliski, and S. Kang. Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):299–314, February 2008. DOI: http://dx.doi.org/10.1109/TPAMI.2007.1176.

[88] C. Liu, W.T. Freeman, R. Szeliski, and S.B. Kang. Noise estimation from a single image. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 901–908. IEEE, 2006.

[89] S. Lyu and E.P. Simoncelli. Modeling multiscale subbands of photographic images with fields of Gaussian scale mixtures. *Transactions on Pattern Analysis and Machine Intelligence*, 31(4):693–706, 2009.

[90] J. Mairal. *Représentations parcimonieuses en apprentissage statistique, traitement d'image et vision par ordinateur*. PhD thesis, 2010.

[91] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[92] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV'09*, pages 2272–2279, 2009. http://dx.doi.org/10.1109/ICCV.2009.5459452.

[93] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on image processing*, 17(1):53–69, 2008. DOI: http://dx.doi.org/10.1109/TIP.2007.911828.

[94] J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Modeling and Simulation*, 7(1):214–241, 2008. DOI: http://dx.doi.org/10.1137/070697653.

[95] M. Makitalo and A. Foi. Optimal inversion of the Anscombe transformation in low-count Poisson image denoising. *Image Processing, IEEE Transactions on*, 20(1):99–109, 2011.

[96] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic press, 1999.

[97] S. Mallat and Z. Zhang. Matching Pursuits with Time-Frequency Dictionaries. *IEEE Transactions on signal processing*, 41(12), December 1992. http://dx.doi.org/10.1109/78.258082.

[98] G.A. Mastin. Adaptive filters for digital image noise smoothing: An evaluation. *Computer Vision, Graphics, and Image Processing*, 31(1):103–121, 1985.

[99] P. Meer, J.M. Jolion, and A. Rosenfeld. A fast parallel algorithm for blind estimation of noise variance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(2):216–223, 1990.

[100] P. Milanfar. A tour of modern image filtering. *IEEE Signal Processing Magazine*, 2, 2011.

[101] R. Neelamani, R. De Queiroz, Z. Fan, S. Dash, and R.G. Baraniuk. JPEG compression history estimation for color images. *Image Processing, IEEE Transactions on*, 15(6):1365–1378, 2006.

[102] A. Nemirovski. Topics in non-parametric statistics. *Lectures on probability theory and statistics (Saint-Flour, 1998)*, 1738:85–277, 2000.

[103] D.R. Nowak and R.G. Baraniuk. Wavelet-domain filtering for photon imaging systems. *IEEE Transactions on Image Processing*, 8(5):666–678, 1997.

[104] S.I. Olsen. Estimation of noise in images: An evaluation. *CVGIP: Graphical Models and Image Processing*, 55(4):319–323, 1993.

[105] E. Ordentlich, G. Seroussi, S. Verdu, M. Weinberger, and T. Weissman. A discrete universal denoiser and its application to binary images. In *International Conference on Image Processing*, volume 1, 2003.

[106] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. Using geometry and iterated refinement for inverse problems (1): Total variation based image restoration. *Department of Mathematics, UCLA, LA, CA*, 90095:04–13, 2004.

[107] E. Le Pennec and S. Mallat. Geometrical image compression with bandelets. In *Proceedings of the SPIE 2003*, volume 5150, pages 1273–1286, 2003.

[108] N. Pierazzo, M. Lebrun, M.E. Rais, J.M. Morel, and G. Facciolo. Non-Local Dual image Denoising. *ICIP*, 2014. Submitted.

[109] N. Ponomarenko, V. Lukin, K. Egiazarian, and J. Astola. A method for blind estimation of spatially correlated noise characteristics. In *IS&T/SPIE Electronic Imaging*, pages 753208–753208. International Society for Optics and Photonics, 2010. `http://dx.doi.org/10.1117/12.847986`.

[110] N.N. Ponomarenko, V.V. Lukin, S.K. Abramov, K.O. Egiazarian, and J.T. Astola. Blind evaluation of additive noise variance in textured images by nonlinear processing of block dct coefficients. In *Proceedings of SPIE*, volume 5014, page 178, 2003.

[111] N.N. Ponomarenko, V.V. Lukin, M.S. Zriakhov, A. Kaarna, and J.T. Astola. An automatic approach to lossy compression of AVIRIS images. *IEEE International Geoscience and Remote Sensing Symposium*, 2007.

[112] J. Portilla. Blind non-white noise removal in images using gaussian scale mixtures in the wavelet domain. *Benelux Signal Processing Symposium*, 2004.

[113] J. Portilla. Full blind denoising through noise covariance estimation using gaussian scale mixtures in the wavelet domain. *Image Processing, 2004. ICIP'04. 2004 International Conference on*, 2:1217–1220, 2004. DOI: `http://dx.doi.org/10.1109/ICIP.2004.1419524`.

[114] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *Image Processing, IEEE Transactions on*, 12(11):1338–1351, 2003. DOI: `http://dx.doi.org/10.1109/TIP.2003.818640`.

[115] S. Pyatykh, J. Hesser, and L. Zheng. Image noise level estimation by principal component analysis. *IEEE Transactions on Image Processing*, 2012.

[116] T. Rabie. Robust estimation approach for blind denoising. *IEEE Transactions on Image Processing*, 14(11):1755–1765, November 2005. DOI: `http://dx.doi.org/10.1109/TIP.2005.857276`.

[117] B. Rajaei. An analysis and improvement of the BLS-GSM denoising method. *Image Processing On Line*, 2013, 2013. Preprint.

[118] K. Rank, M. Lendl, and R. Unbehauen. Estimation of image noise variance. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 146, pages 80–84. IET, 1999.

[119] M. Raphan and E.P. Simoncelli. Learning to be bayesian without supervision. *Advances in neural information processing systems*, 19:1145, 2007.

[120] M. Raphan and E.P. Simoncelli. An empirical bayesian interpretation and generalization of nl-means. Technical report, Technical Report TR2010-934, Computer Science Technical Report, Courant Inst. of Mathematical Sciences, New York University, 2010.

[121] W.H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, 1972. DOI: `http://dx.doi.org/10.1364/JOSA.62.000055`.

[122] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60:259–268, 1992. DOI: `http://dx.doi.org/10.1016/0167-2789(92)90242-F`.

[123] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, 2008.

[124] C.E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[125] S.M. Smith and J.M. Brady. SUSAN-A new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[126] J.L. Starck, E.J. Candès, and D.L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on image processing*, 11:670–684, 2002. DOI: `http://dx.doi.org/10.1109/TIP.2002.1014998`.

[127] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *IEEE ICCV*, 1998.

[128] J.W. Tukey. Exploratory data analysis. 1977. *Massachusetts: Addison-Wesley*, 1976.

[129] M.L. Uss, B. Vozel, V.V. Lukin, and K. Chehdi. Image informative maps for component-wise estimating parameters of signal-dependent noise. *Journal of Electronic Imaging*, 22(1):013019–013019, 2013.

[130] D. Van De Ville and M. Kocher. Sure-based non-local means. *Signal Processing Letters, IEEE*, 16(11):973–976, 2009.

[131] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural image. In *Proceedings of the First International Conference on Computer Vision London*, pages 250–258. IEEE, Washington, DC, 1987.

[132] G.K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[133] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 13(4), 2004.

[134] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M.J. Weinberger. Universal discrete denoising: Known channel. *IEEE Transactions on Information Theory*, 51(1):5–28, 2005.

[135] L. Yaroslavsky and M. Eden. Fundamentals of Digital Optics, 2003.

[136] L.P. Yaroslavsky. *Digital Picture Processing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985.

[137] L.P. Yaroslavsky. Local adaptive image restoration and enhancement with the use of DFT and DCT in a running window. In *Proceedings of SPIE*, volume 2825, pages 2–13, 1996. DOI: http://dx.doi.org/10.1117/12.255218.

[138] L.P. Yaroslavsky, K.O. Egiazarian, and J.T. Astola. Transform domain image restoration methods: review, comparison, and interpretation. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 4304, pages 155–169, May 2001. DOI: http://dx.doi.org/10.1117/12.424970.

[139] G. Yu and G. Sapiro. DCT image denoising: a simple and effective image denoising algorithm. *Image Processing On Line*, 2011, 2011. http://dx.doi.org/10.5201/ipol.2011.ys-dct.

[140] G. Yu, G. Sapiro, and S. Mallat. Image modeling and enhancement via structured sparse model selection. In *2010 17th IEEE International Conference on Image Processing (ICIP)*, pages 1641–1644, 2010. DOI: http://dx.doi.org/10.1109/ICIP.2010.5653853.

[141] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity. *Transactions on Image Processing*, 21(5):2481–2499, 2012.

[142] L. Zhang, W. Dong, D. Zhang, and G. Shi. Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition*, 43(4):1531–1549, 2010. DOI: http://dx.doi.org/10.1016/j.patcog.2009.09.023.

[143] M. Zhou, H. Yang, G. Sapiro, D. Dunson, and L. Carin. Dependent hierarchical beta process for image interpolation and denoising. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[144] D. Zoran and Y. Weiss. Scale invariance and noise in natural images. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2209–2216. IEEE, 2009.

[145] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. *International Conference on Computer Vision*, 2011.