Advances in Image Denoising

Nicola Pierazzo

NNT : 2016SACLN036

Thèse de doctorat de l'Université Paris-Saclay préparée à l'École Normale Supérieure de Cachan (École Normale Supérieure Paris-Saclay)

Ecole doctorale n°574
École doctorale de mathématiques Hadamard
Spécialité de doctorat : Mathématiques appliquées
par

M. Nicola Pierazzo

Quelque progrès en débruitage d'images

Thèse présentée et soutenue à Cachan, le 20 septembre 2016.

Composition du Jury :

M.	Sylvain Durand	Professeur Université Paris Descartes	(Président du jury)
M.	Matthias Zwicker	Professeur Universität Bern, Switzerland	(Rapporteur)
M.	Michael Elad	Professeur Technion City, Israel	(Rapporteur)
Mme	Julie Delon	Professeur Université Paris Descartes	(Examinatrice)
M.	Jalal Fadili	Professeur Institut Universitaire de France	(Examinateur)
M.	Jean-Michel Morel	Professeur École Normale Supérieure de Cachan	(Directeur de thèse)
M.	Gabriele Facciolo	Chercheur École Normale Supérieure de Cachan	(Directeur de thèse)

Acknowledgements

This PhD has been the central point of my life for the last four years. I wouldn't have made it alone, so now that it is coming to an end, I would like to thank all the people who helped me in this adventure.

First, I'd like to thank my advisors, Jean-Michel Morel and Gabriele Facciolo. The weekly meetings with Jean-Michel helped me organize my work, and the long discussions with Gabriele gave me insights I'd never had had alone. They helped me continue working when I was stuck, always supporting me and pushing me to do my best, with patience and enthusiasm. Without them this works would have been impossible.

I'd like to thank my reviewers, Sylvain Durand, Matthias Zwicker and Michael Elad, who accepted to read my thesis. Also, I'd like to thank Julie Delon and Jalal Fadili to have accepted to be part of my jury. I'd also like to thank the team of DxO, in particular Fréderic Guichard and Wolf Hauser. Meeting with them was a source of inspiration for the real-life problems that are tackled in this work.

The experience of the PhD wouldn't have been complete if it wasn't for my teaching experience. I'd like to thank my students, and hope that they managed to get some useful notion from my incoherent ramblings.

Of course, the people we work with become very important in our life. I had the luck of working at CMLA, that I believe is one of the best laboratories around. I spent four years between people who always helped me when they could, with their advice and their time. And an engine turns well when all its gears work properly. I'd like to thank in particular Véronique, Virginie, Micheline and Sandra for their continuous support, and for having the patience to deal with me in numerous occasions. Also, thanks to all the people who worked at CMLA in these years, especially Enric, Miguel, Marc, Rafael, José, Mauricio, Morgan, Nicolas, Irène, Barbara, Rachel, Tristan, Marie, Marie, Mariano and Axel. A heartfelt thank you goes to Ives, Carlo, Samy and Charles, with whom I shared food and good moments. Lara and Martin shared with me this journey since its beginning. They have been always there, and they helped me a lot with my work and my life; being at the same time colleagues, friends, psychologists and Spanish teachers. Thank you a lot. Gracias por la fiesta.

I also want to thank the friends I had in Paris during those years. Thanks to Luca, Marcello, Daniel, Laura, Martina, Hernan, Aleksandra, Simon, Claudia, Aina, Miquel, Victoria, Alberto, Gaia, Matias and Luis. Also, many thanks to Étienne, Alexandre, Dorine, Christophe, and all the Diving Club of Cachan, for making me re-discover scuba diving (and for the French lessons). I'd also like to thank all my Friends from Pisa, who were with me before coming here. Thank you Dario, Lorenzo, Beatrice, Luca, Vittorio, Mario, Giulio, Alessandra, Davide, Mara, Simone, Valerio, Ugo, Gabriele, Michele and all the oth-

ers. Many tanks to my friends from home. Thank you to Giulio, Giulio, Andrea, Elena, Alvise, Alvise, Gloria, Samantha, Erica, Federica, Marco, Bruno, Enrico and all the others.

I'd like to thank my family, in particular my Mother, my Father and my brother Alessandro. Thank you for always supporting me and for making me feel at home every time I come back.

Also, many thanks to Sandra's (big) family, who in these years have adopted me, for the fun and the good food.

A special thank you to Sandra, who supported me all this time.

Summary

This thesis explores the last evolutions on image denoising, and attempts to set a new and more coherent background regarding the different techniques involved. In consequence, it also presents a new image denoising algorithm with minimal artifacts and the best PSNR performance known so far.

A first result that is presented is DA3D, a frequency-based guided denoising algorithm inspired form DDID [Knaus-Zwicker 2013]. This algorithm achieves good results not only in terms of PSNR, but also (and especially) with respect to visual quality. DA3D works particularly well on enhancing the textures of the images and removing staircasing effects. DA3D works on top of another denoising algorithm, that is used as a guide, and almost always improve its results. In this way, frequency-based denoising can be applied on top of patch-based denoising algorithms, resulting on a hybrid method that keeps the strengths of both. The experiments demonstrate that, contarily to what was thought, frequency-based denoising can beat state-of-the-art algorithms without presenting artifacts. In this work DA3D is presented, and ad-hoc shrinkage curves are computed depending on the algorithm used as guide.

The second result presented is Multi-Scale Denoising, a multiscale framework applicable to any denoising algorithm. A qualitative analysis shows that current denoising algorithms behave better on high-frequency noise. This is due to the relatively small size of patches and search windows used in these single scale algorithms. Instead of enlarging those windows, that can cause other sorts of problems, the work proposes to decompose the image on a pyramid, with the aid of the Discrete Cosine Transformation. We introduce a new reconstruction scheme in the pyramid avoiding the appearance of ringing artifacts. This method removes most of the low-frequency noise, and improves both PSNR and visual results for smooth and textured areas.

A third main issue addressed in this thesis is the evaluation of denoising algorithms. Experiences indicate that the PSNR is not always a good indicator of visual quality for denoising algorithms, since, for example, an artifact on a smooth area can be more noticeable than a subtle alteration in a texture. A new metric is proposed to improve on this matter. Instead of a single value, a "Smooth PNSR" and a "Texture PSNR" are presented, to measure the result of an algorithm for those two types of image regions. We claim that a denoising algorithm, in order to be considered acceptable, must at least perform well with respect to both metrics. Following this claim, an analysis of current algorithms is performed, and it is compared with the combined results of the Multi-Scale Framework and DA3D. We found that the optimal solution for image denoising is the application of a frequency shrinkage, applied to regular regions only, while a multiscale patch based method serves as guide. This seems to resolve a long standing question for which DDID gave the first clue: what is

the respective role of frequency shrinkage and self-similarity based methods for image denoising? We describe an image denoising algorithm that seems to perform better in quality and PSNR than any other based on the right combination of both denoising principles.

In our last contribution, a study on the feasibility of external denoising is carried out, where images are denoised by means of a big database of external noiseless patches. This follows a work of Levin and Nadler, in 2011, that claims that optimal Bayesian patch denoising is reachable by a simple integral formula provided it is applied to a gigantic and representative patch database. We prove by a mathematical argument combined with empirical observations on the space of patches that this space can be factorized, thereby reducing considerably the number of patches needed in order to compute the integral on the space of patches.

Résumé

Cette thèse explore les dernières évolutions du débruitage d'images, et elle essaie de développer une vision synthétique des techniques utilisées jusqu'à présent. Elle aboutit à un nouvel algorithme de débruitage d'image évitant les artefacts et avec un meilleur PSNR que tous les algorithmes que nous avons pu évaluer.

La première méthode que nous présentons est DA3D, un algorithme de débruitage fréquentiel avec guide, inspiré de DDID [Knaus-Zwicker 2013]. La surprise de cet algorithme, c'est que le débruitage fréquentiel peut battre l'état de l'art sans produire artefacts. Cet algorithme produit des bons résultats non seulement en PSNR, mais aussi (et surtout) en qualité visuelle. DA3D marche particulièrement bien pour améliorer les textures des images et pour enlever les effets de *staircasing*. DA3D, guidé par un autre algorithme de débruitage améliore presque toujours le résultat de son guide. L'amélioration est particulièrement nette quand le guide est un algorithme à patchs, et alors on combine deux principes différents : auto-similarité suivi de seuillage fréquentiel.

Le deuxième résultat présenté est une méthode universelle de débruitage multi-échelle, applicable à tout algorithme. Une analyse qualitative montre en effet que les algorithmes de débruitage à patchs éliminent surtout les hautes fréquences du bruit, à cause de la taille limitée des voisinages traités. Plutôt que d'agrandir ces voisinages nous décomposons l'image en une pyramide basée sur la transformée en cosinus discrète, avec une méthode de recomposition évitant le *ringing*. Cette méthode traite le bruit à basse fréquence, et améliore la qualité de l'image.

Le troisième problème sérieux que nous abordons est l'évaluation des algorithmes de débruitage. Il est bien connu que le PSNR n'est pas un indice suffisant de qualité. Un artefact sur une zone lisse de l'image est bien plus visible qu'une altération en zone texturée. Nous proposons une nouvelle métrique basée sur un *Smooth PSNR* et un *Texture PSNR*, pour mesurer les résultats d'un algorithme sur ces deux types des régions. Il apparaît qu'un algorithme de débruitage, pour être considéré acceptable, doit avoir des bons résultats pour les deux métriques. Ces métriques sont finalement utilisées pour comparer les algorithmes de l'état de l'art avec notre algorithme final, qui combine les bénéfices du multi-échelle et du filtrage fréquentiel guidé.

Les résultats étant très positifs, nous espérons que la thèse contribue à résoudre un vieux dilemme, pour lequel la méthode DDID avait apporté de précieuses indications : comment choisir entre le seuillage fréquentiel et les méthodes basées sur l'auto-similarité pour le débruitage d'images ? La réponse est qu'il ne faut pas choisir.

Cette thèse termine avec quelques perspectives sur la faisabilité du débruitage "externe". Son principe est de débruiter un patch en utilisant une grande base de données externe de patches sans bruit. Un principe bayésien démontré par Levin et Nadler en 2011 implique

que le meilleur résultat possible serait atteint avec cette méthode, à condition d'utiliser tous les patches observables. Nous donnons les arguments mathématiques prouvant que l'espace des patches peut être factorisé, ce qui permet de réduire la base de données de patches utilisés d'un facteur au moins 1000.

Contents

Chapter 0

Introduction

This thesis explores the last evolutions on image denoising, and attempts to set a new and more coherent background regarding the different techniques involved. In consequence, it also presents a new image denoising algorithm with minimal artifacts and the best PSNR performance known so far.

A first result that is presented is DA3D, a frequency-based guided denoising algorithm inspired form DDID [Knaus-Zwicker 2013]. This algorithm achieves good results not only in terms of PSNR, but also (and especially) with respect to visual quality. DA3D works particularly well on enhancing the textures of the images and removing staircasing effects. DA3D works on top of another denoising algorithm, that is used as a guide, and almost always improve its results. In this way, frequency-based denoising can be applied on top of patch-based denoising algorithms, resulting on a hybrid method that keeps the strengths of both. The experiments demonstrate that, contarily to what was thought, frequency-based denoising can beat state-of-the-art algorithms without presenting artifacts. In this work DA3D is presented, and ad-hoc shrinkage curves are computed depending on the algorithm used as guide.

The second result presented is Multi-Scale Denoising, a multiscale framework applicable to any denoising algorithm. A qualitative analysis shows that current denoising algorithms behave better on high-frequency noise. This is due to the relatively small size of patches and search windows used in these single scale algorithms. Instead of enlarging those windows, that can cause other sorts of problems, the work proposes to decompose the image on a pyramid, with the aid of the Discrete Cosine Transformation. We introduce a new reconstruction scheme in the pyramid avoiding the appearance of ringing artifacts. This method removes most of the low-frequency noise, and improves both PSNR and visual results for smooth and textured areas.

A third main issue addressed in this thesis is the evaluation of denoising algorithms. Experiences indicate that the PSNR is not always a good indicator of visual quality for denoising algorithms, since, for example, an artifact on a smooth area can be more noticeable than a subtle alteration in a texture. A new metric is proposed to improve on this matter. Instead of a single value, a "Smooth PNSR" and a "Texture PSNR" are presented, to measure the result of an algorithm for those two types of image regions. We claim that a denoising algorithm, in order to be considered acceptable, must at least perform well with respect to both metrics. Following this claim, an analysis of current algorithms is performed, and it is compared with the combined results of the Multi-Scale Framework and DA3D. We found

that the optimal solution for image denoising is the application of a frequency shrinkage, applied to regular regions only, while a multiscale patch based method serves as guide. This seems to resolve a long standing question for which DDID gave the first clue: what is the respective role of frequency shrinkage and self-similarity based methods for image denoising? We describe an image denoising algorithm that seems to perform better in quality and PSNR than any other based on the right combination of both denoising principles.

In our last contribution, a study on the feasibility of external denoising is carried out, where images are denoised by means of a big database of external noiseless patches. This follows a work of Levin and Nadler, in 2011, that claims that optimal Bayesian patch denoising is reachable by a simple integral formula provided it is applied to a gigantic and representative patch database. We prove by a mathematical argument combined with empirical observations on the space of patches that this space can be factorized, thereby reducing considerably the number of patches needed in order to compute the integral on the space of patches.

Finally, secondary results are presented. A brief study of how to apply denoising algorithms on real RAW images is performed. An improved, better performing version of the Non-Local Bayes algorithm is presented, together with a two-step version of DCT Denoising. The latter is interesting for its extreme simplicity and for its speed.

In this introduction we give a brief overview of the chapters of this work.

0.1 Chapter 1: DA3D and the Dual Domain Methods

Image denoising is one of the fundamental image restoration challenges [40]. It consists in estimating an unknown noiseless image $\mathbf{y}$ from a noisy observation $\mathbf{x}$. We consider the classic image degradation model

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{1}$$

where the observation $\mathbf{y}$ of the image $\mathbf{x}$ is contaminated by an additive white Gaussian noise $\mathbf{n}$ of variance σ^2.

All denoising methods assume some underlying image regularity. Depending on this assumption they can be divided, among others, into transform-domain and spatial-domain methods.

Transform domain methods work by shrinking (or thresholding) the coefficients of some transform domain [18, 34, 51]. The Wiener filter [60] is one of the first such methods operating on the Fourier transform. Donoho et al. [11] extended it to the wavelet domain.

Space-domain methods traditionally use a local notion of regularity with edge-preserving algorithms such as total variation [49], anisotropic diffusion [41], or the bilateral filter [56]. Nowadays however spatial-domain methods achieve remarkable results by exploiting the self-similarities of the image [2]. These patch-based methods are non-local as they denoise by averaging similar patches in the image. Patch-based denoising has developed into attempts to model the patch space of an image, or of a set of images. These techniques model the patch as sparse representations on dictionaries [10, 14, 35, 36, 64], using Gaussian Scale Mixtures models [45, 46, 67], or with non-parametric approaches by sampling from a huge database of patches [32, 33, 39, 43].

Current state-of-the-art denoising methods such as BM3D [7] and NL-Bayes [27] take advantage of both space- and transform-domain approaches. They group similar image

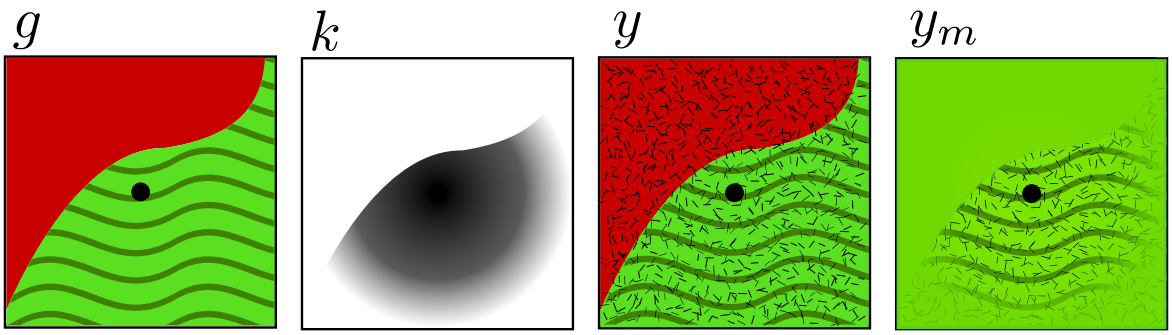


Figure 1 – Illustration of DDID's preprocessing of a patch. The kernel k is computed using the guide g. In the modified patch y_m all object discontinuities have been removed, leaving only the texture information corresponding to the object selected by the kernel k. The removed pixels are replaced by the average of the meaningful portion of the patch.

patches and jointly denoise them by collaborative filtering on a transformed domain. In addition, they proceed by applying two slightly different denoising stages, the second stage using the output of the first one as its guide.

Some recently proposed methods use the result of a different algorithm as their guide for a new denoising step. Combining for instance, nonlocal principles with spectral decomposition [55], or BM3D with neural networks [5]. This allows one to mix different denoising principles, thus yielding high quality results. DDID [22] is an iterative algorithm that uses a guide image (from a previous iteration) to determine spatially uniform regions to which Fourier shrinkage could be applied without introducing ringing artifacts. Several methods [21, 23, 42] use a single step of DDID with a guide image produced by a different algorithm. This yields much better results than the original DDID. The reason for their success is the use of large (31×31) and shape-adaptive patches. Indeed, the Fourier shrinkage works better on large stationary blocks.

Unfortunately, DDID has a prohibitive computational cost, as it paradoxically denoises a large patch to recover a single pixel. Moreover, contrary to other methods, aggregation of these patches doesn't improve the results since it introduces blur. We show in this paper that these two problems can be solved by introducing a new patch selection, accompanied by a weighted aggregation strategy.

Our first chapter presents DA3D (Data Adaptive Dual Domain Denoising), a new "last step" denoising method that performs frequency domain shrinkage on shape-adaptive and data-adaptive patches. DA3D consistently improves the results of state-of-the-art methods such as BM3D or NL-Bayes with little additional computation time.

Our second contribution in DA3D further improves the quality of the results by adapting the processing to the underlying data. The apparition of the staircasing is well known for non-local methods [3]. To mitigate the influence of such artifacts present in the guide image, we use a first order non-linear local kernel regression [53,54] to estimate, for each patch, an affine approximation of the image coherent with the data within the patch. The denoising is then performed with respect to this approximation. This data-adaptive approach is another innovation enabled by the use of large patches, and it noticeably improves the quality of the results on smooth regions of the image.

Section 1.4 recalls the DDID post-process and develops an explanation of its efficiency,

Method	$\sigma = 5$	$\sigma = 10$	$\sigma = 25$	$\sigma = 40$	$\sigma = 80$
BM3D	39.55	36.01	31.79	29.23	**26.83**
	39.59	**36.23**	**32.15**	**29.95**	**27.00**
	+0.05	+0.22	+0.36	+0.72	+0.16
DDID	39.20	35.91	31.88	29.75	26.41
	39.11	35.87	**31.99**	**29.98**	**26.85**
	-0.09	-0.04	+0.11	+0.23	+0.43
NLB	**39.77**	**36.09**	31.71	29.35	26.64
	39.67	**36.26**	**32.18**	**30.09**	26.74
	-0.10	+0.16	+0.47	+0.74	+0.11
NLDD	39.26	35.90	31.98	29.90	26.60
	39.17	35.87	**31.99**	**29.97**	26.63
	-0.09	-0.04	+0.02	+0.07	+0.03
PID	39.03	35.81	**32.05**	**30.10**	**27.00**
	38.99	35.80	**32.03**	**30.02**	**26.83**
	-0.04	-0.02	-0.02	-0.08	-0.17

Table 1 – Average PSNR comparison between state-of-the-art methods for color images. The first line of each row shows the average PSNR obtained by denoising the test images. The second line shows the average PSNR of DA3D using the corresponding denoising algorithm to generate the guide. The third line shows the average improvement due to DA3D. The best result for each noise level is shown in **bold**, and the ones within a range of 0.2 dB are shown in **gray**.

illustrated in Figure 1, where it becomes apparent that the method manages to denoise separately each texture patch, while avoiding th einterference of edges and other neighboring textures. Sections 1.5 and 1.6 present the DA3D algorithm, first describing the proposed sparse aggregation, then the data-adaptive patches. The performance of DA3D is extensively validated in the experiments of section 1.7, where it becomes apparent that it improves significantly the result of most denoising algorithms, as shown in Table 1.

0.2 Chapter 2: A Multi-Scale Denoising Framework

In Chapter 1, we provided a new view on frequency transform thresholding, showing that it could perform excellently as the last step of any patch based denoising algorithm. In Chapter 2 we address another question left untreated by most state of the art denoising algorithms. We observed that most of them limit their action to a limited pixel neighborhood. This clearly means that low frequency noise remains untreated, as made clear in Figure 2. This is a drawback that yields very visible artifacts in flat areas, as we shall check in the experimental part of this chapter, and in our experimental study of Chapter 3. Thus, in this chapter we provide a new perspective of another principle: the multi-scale representation. This principle has already been explored in [4, 15]. Even though the results were only partially satisfying, the ideas presented are simple and promising. The main problem with their approach is that in higher scales, the noise becomes correlated, thus reducing the performance of standard algorithms. Another work that tries to use a multi-scale model is [47]. The difference with our work is that [47] does not use classical denoising algorithm in the process and that it does not avoid artifacts in the recomposition.

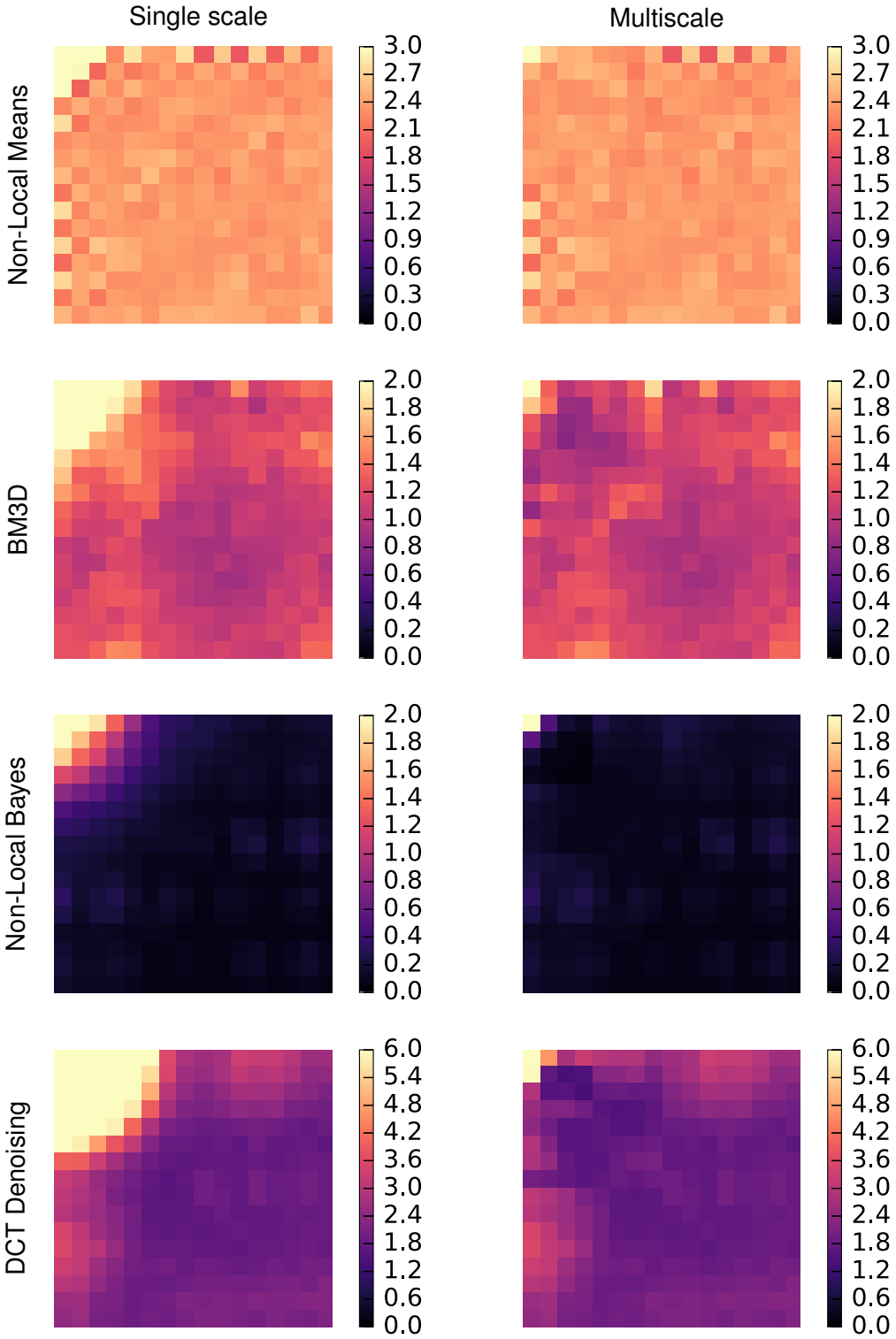


Figure 2 – DCT transform of the result of various denoising algorithms applied to an image of pure white noise, with and without the Multi-Scale Framework. Notice that there is still noise remaining in the upper left corner of the Single Scale version, that contains the low frequencies of the image. In the Multi-Scale version, the residual noise is much more uniform across frequencies.

Noise	Best single-scale		Best multi-scale		Gain
$\sigma = 10$	38.59dB	BM3D	38.71dB	NLB	+0.12dB
$\sigma = 20$	34.97dB	BM3D	35.09dB	NLB	+0.12dB
$\sigma = 30$	32.87dB	BM3D	33.07dB	NLB	+0.20dB
$\sigma = 40$	31.37dB	BM3D	31.65dB	NLB	+0.28dB
$\sigma = 50$	30.30dB	NLB IPOL	30.61dB	NLB	+0.31dB
$\sigma = 60$	29.32dB	NLB IPOL	29.75dB	NLB	+0.43dB
$\sigma = 70$	28.65dB	BM3D	29.02dB	NLB	+0.37dB
$\sigma = 80$	27.97dB	BM3D	28.40dB	NLB	+0.43dB
$\sigma = 90$	27.30dB	BM3D	27.86dB	NLB	+0.56dB
$\sigma = 100$	26.75dB	NLB IPOL	27.46dB	NLB IPOL	+0.71dB

Table 2 – Results with the best settings for every algorithm. The results are an average over the images of Figure 2.10.

The multi-scale representation is also intrinsically present in wavelet-based denoising algorithms [11, 18, 34, 45]. It is also present in [52], where the KSVD algorithm is applied on a wavelet decomposition of the image. The improvement over a single scale KSVD is important, especially for high PSNR, but since the wavelet decomposition does not allow for a conservative recomposition, the authors need what they call a fusion strategy, in order to reduce the artifacts.

Denoising a multi-scale pyramid is a way to process more information to reach a better result. Other efforts have been made to provide more input to denoising algorithm. In particular, it is worth citing [5, 65], that try to improve the results of a denoising algorithm by using an external database of similar images.

In Chapter 2 we present a new multi-scale framework that can be applied to any other existing denoising algorithm, consistently improving its results (see Table 2). The framework uses a simple DCT pyramid, and is not computationally demanding.

Simply using the DCT pyramid may lead to ringing artifacts. We solve this issue by introducing what we call *conservative recomposition*, which allows to keep the advantages of the pyramid while avoiding its problems.

The result is a way of transforming any denoising algorithm into a multi-scale one, with improvements both in visual quality and PSNR, and with little additional cost.

0.3 Chapter 3: Quality Criteria for Image Denoising

Usually the performance of denoising algorithms is measured in terms of PSNR. Another famous quality index is the SSIM [58]. Unfortunately, neither of the two methods is closely correlated with the visual quality, since they are not sensitive enough to artifacts.

Different denoising algorithms behave differently on textures and flat areas of images. Measuring the PSNR of the result may be misleading, since an algorithm can provide a good result on the former and a bad one on the latter, or vice versa. Therefore, we propose to decouple the quality measure of a result in two values: the flat PSNR (or fPSNR) and the texture PSNR (or tPSNR), using a smoothness map (Figure 3) to discriminate flat and textured regions of an image.

Figure 3 – Example of smoothness map.

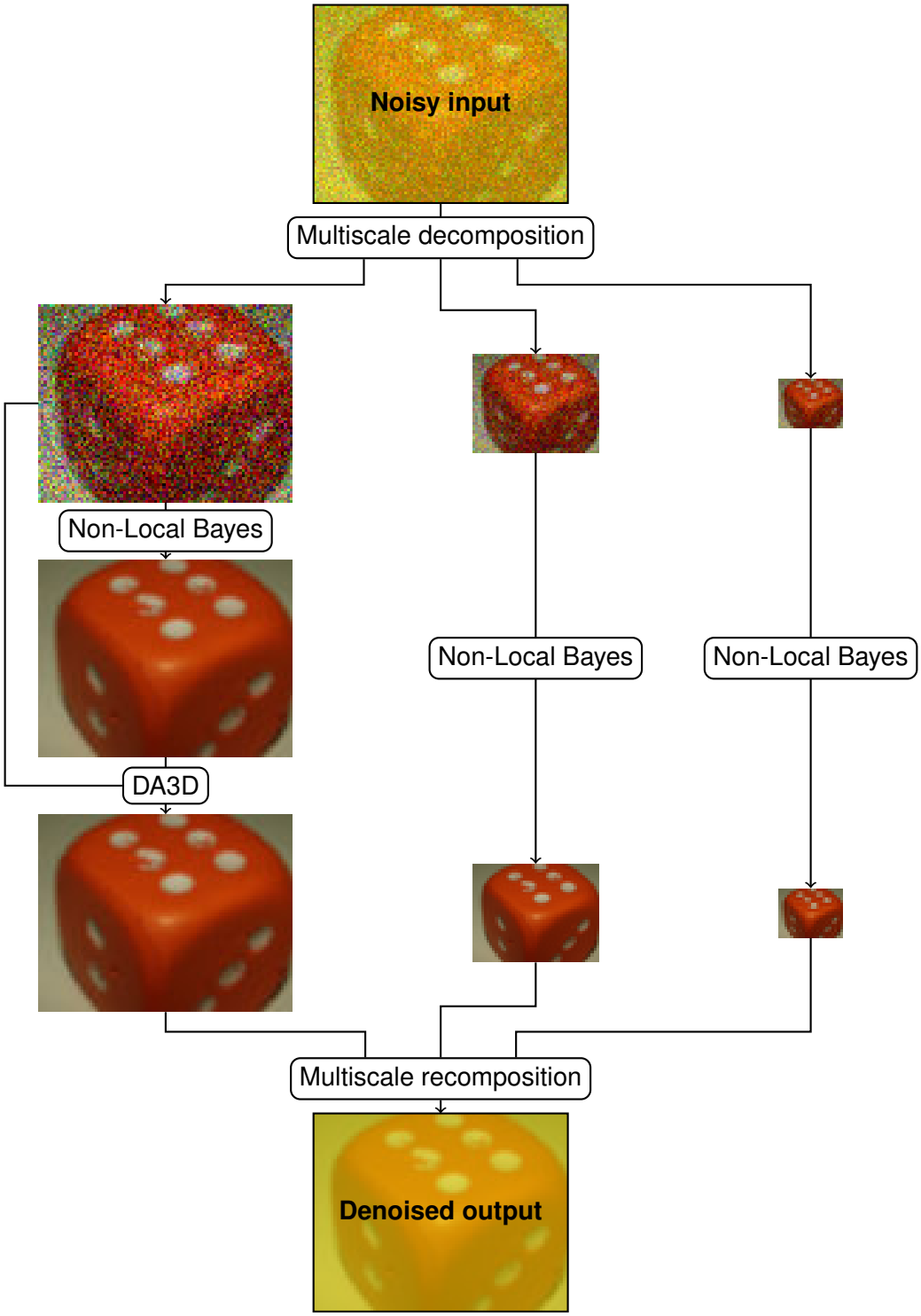


Figure 4 – Scheme of the *da3d_ms* denoising method that combines Non-Local Bayes, DA3D and the Multi-Scale Framework. The values for the multiscale recompositions are the one used for Non-Local Bayes in Chapter 2 (0.5 for $\sigma = 10$ and $\sigma = 20$, 0.6 for $\sigma = 40$ and 0.7 for $\sigma = 70$). DA3D is applied using the result of Non-Local Bayes as guide only on the finer scale. Applying DA3D to coarser scale degrades the results due to the presence of ringing in the DCT pyramid.

We applied the metrics discussed in this chapter to the results of the most common denoising algorithms.

Both Chapter 1 and Chapter 2 present a method to improve the result of other denoising algorithms. In this section, we present two different ways to combine both of them, in order to get a better performing denoising algorithm.

Since both DA3D and the Multiscale framework need a "baseline" algorithm to work, we chose to focus on Non-Local Bayes [27] and BM3D [7]. The first algorithm is particularly suited for both DA3D and the Multiscale framework, since it has relatively few artifacts and a smooth result, while we decided to included the second to offer a fair comparison with what is still considered the state-of-the-art.

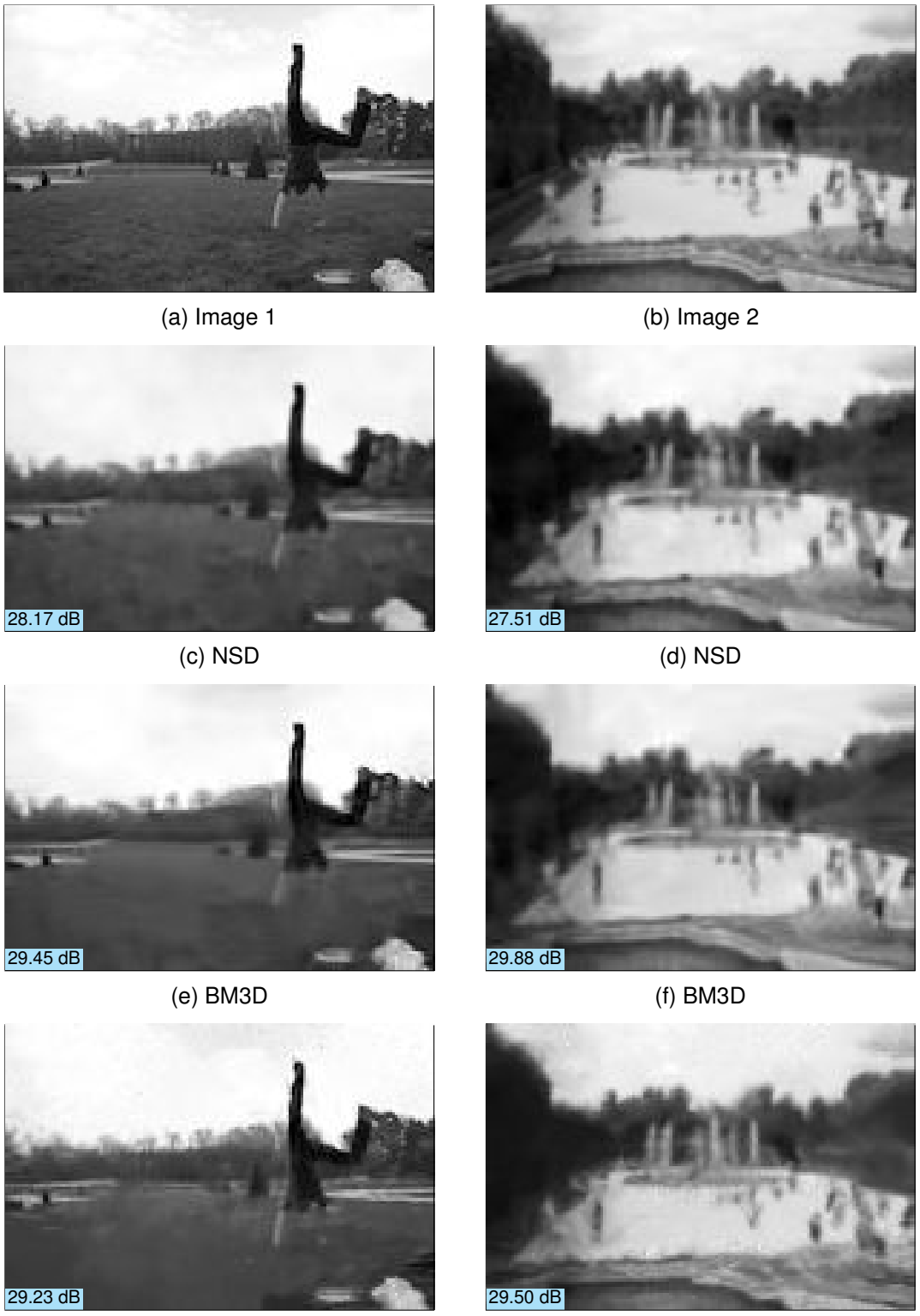
In order to combine DA3D and the Multiscale framework, there are fundamentally two possibilities: apply DA3D first, or apply the Multiscale framework first. We present them both. Figure 4 gives the diagram of the winning combination.

0.4 Chapter 4: External Denoising

A recent seminal paper on the absolute bounds of image denoising [32] proposes a patch denoising method effectively realizing the minimal mean square error, given all the known image patches. It is extremely important to reach these absolute limits, but they require processing a limitless database. In the above mentioned paper this database had 10 billion patches. In this paper we demonstrate that by factorizing the patch space the method can be sped up by a factor of more than a thousand, while maintaining the theoretical claim that the method is optimal. Using the method on real images demonstrates its potential to beat the state of the art, as it performs better on difficult patches.

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 *very large* patch set, assuming they represent "all possible natural image patches".

For example, Zoran and Weiss [67] introduced the EPLL algorithm, in which they learn the patch space distribution from a mixture of Gaussians trained with 2×10^6 patches. Then, they use this model as a prior for denoising patches by trying to maximize the *Expected Patch Log Likelihood*, instead of denoising each patch separately. To do this, they minimize a cost function using an optimization method called "Half Quadratic Splitting" [48]. Zontak and Irani [66] analyzed the denoising performance using patches extracted from the same image, or what they call internal patch searches, against extracting "external" patches from a huge database. They concluded that similar patches tend to recur much more frequently inside the same image than in any random external collection of natural images. Furthermore, internal patch redundancy rapidly decays with the growth of the spatial distance from the patch, and its gradient content. Finally, they observed that finding an equally good external representative patch for every patch of an image, requires a huge external database, which makes the denoising problem computationally intractable. Lee et al. [31] used the Bayesian framework to derive an MMSE non-local image denoising algorithm, that uses both internal image patches as well as an external codebook with noiseless patches. They perform denoising by normalizing the patch-space by the mean, an idea which will be further exploited here. Several "shotgun" methods have been recently applied in other image processing tasks, such as scene completion [19] (using 2.3 million images), or scene recognition [57] (using 80 million 32×32 patches).



(a) Image 1 (b) Image 2

(c) NSD (d) NSD

(e) BM3D (f) BM3D

(g) NL-Bayes (h) NL-Bayes

Figure 5 – Results and comparison of denoising methods (PSNR values in parenthesis).

DB Size	Shotgun method	NSD
10000	23.47 dB	26.79 dB
100000	25.92 dB	27.32 dB
4000000	27.66 dB	28.06 dB

Table 3 – Average PSNR obtained after denoising 3500 patches with a fixed Database size.

Let us now turn on the search for absolute bounds for the image denoising. The Cramer-Rao type lower bounds on the attainable RMSE performance given by Chaterjee and Milanfar [6] are optimistic: they allow for the possibility of a significant increase in denoising performance for a wide class of images at certain SNRs, in particular, for synthetic piecewise constant images. Yet, a recent paper by A. Levin and B. Nadler [32] provides a second answer to the question of absolute limits raised by [6]. This seminal paper uses a shotgun denoising method to approach the bounds, which is loosely based on the NL-means algorithm, with adequate parameters to account for a Bayesian linear minimum mean square estimation (LMMSE) of the noisy patch given a database of 10^{10} noiseless patches. The only and important difference is that similar patches are searched on a database of 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 estimate *the best average estimation error attainable by any patch-based denoising algorithm* (to be more specific, it is the best average estimation error knowing a single patch and all the observed; but most denoising algorithms actually also use the knowledge of overlapping patches in the image to denoise it). After performing experimentation, and comparing them with state-of-the-art denoising algorithms, they conclude that results might be within 0.1dB of the optimal possible denoising method, at least when using small patch sizes or under big amounts of noise, therefore leaving no further room for improvement. They propose increasing the support size to obtain better PSNR values. Yet, due to the curse of dimensionality, non-parametric techniques won't be able to find good similar patches. This leads the authors to give up a shotgun method, and to switch again to parametric approaches. However, a limitation of this work is that computational constraints restricted the experimentation to small patches. Hence real bounds independent of patch size are still unknown.

In [33], the same authors study the dependence of denoising error on patch size. Similarly to Zontak and Irani [66], they conclude that when increasing window size, patches with more detail require a significantly larger database, and that the gain from doing so is very small. At the same time, the opposite happens with smoother patches: increasing the window size is much more rewarded in the final denoising performance. This leads them to propose an adaptive strategy using variable window sizes depending on local patch complexity. Finally, they again explore the fundamental limits of denoising, with an infinite window size and assuming a perfect natural image prior. To do this, they use a simplified dead leaves image formation model and combine it with a scale invariance assumption for natural images, which implies a power-law convergence. Using this power-law to model the PSNR with respect to the window size, they fit the model and extrapolate its result to obtain optimal denoising bounds. They conclude that there is still some room for improvement (around 1dB).

In Chapter 4, we stick to the first shotgun approach [32] and intend to render it feasible by accelerating considerably the computation of their Monte Carlo integral. The proposed

algorithm attains similar results using less computational resources and in shorter time. To achieve this, a smaller probability space is used by factorizing the space of patches. This allows to work with entire equivalence classes of patches instead of single patches, thus permitting a significant decrease of the number of needed computations and a reduction of the required memory. Following this, a comparison is made between our algorithm and two state-of-the-art methods, the BM3D algorithm [7,26] and the Non Local Bayes method [27] showing how our method outperforms both for the task of denoising complex patches.

0.5 Publications related to this thesis

- Nicola Pierazzo and Martin Rais. Boosting Shotgun Denoising By Patch Normalization. *IEEE International Conference on Image Processing (ICIP)*, 2013.
- Nicola Pierazzo, Marc Lebrun, Martin Rais, Jean-Michel Morel, and Gabriele Facciolo. Non-Local Dual Image Denoising. *IEEE International Conference on Image Processing (ICIP)*, 2014.
- Nicola Pierazzo, Martin Rais, Jean-Michel Morel, and Gabriele Facciolo. DA3D: Fast and Data Adaptive Dual Domain Denoising. *IEEE International Conference on Image Processing (ICIP)*, 2015.
- Nicola Pierazzo, Jean-Michel Morel, and Gabriele Facciolo. Optimizing the Data Adaptive Dual Domain Denoising Algorithm. *IberoAmerican Congress on Pattern Recognition (CIARP)*, 2015.
- Nicola Pierazzo, Jean-Michel Morel, and Gabriele Facciolo. DA3D and the Dual Domain Methods for Image Denoising. *To be submitted*.
- Nicola Pierazzo, Jean-Michel Morel, and Gabriele Facciolo. Combining Dual Domain and Multiscale Methods for Image Denoising. *To be submitted*.
- Nicola Pierazzo. Multi-Scale DCT Denoising. *Image Processing On Line (IPOL), To be submitted*.

Chapitre 0

Introduction en Français

Cette thèse explore les dernières évolutions sur le débruitage d'images, et tente d'établir un fond nouveau et plus cohérent concernant les différentes techniques impliquées. En conséquence, il présente également un nouvel algorithme de débruitage d'image avec des artefacts minimes et la meilleure performance PSNR connue jusqu'ici.

Un premier résultat présenté est DA3D, un algorithme de débruitage guidé basé sur la fréquence inspiré de la forme DDID [Knaus-Zwicker 2013]. Cet algorithme atteint de bons résultats non seulement en termes de PSNR, mais aussi (et surtout) en ce qui concerne la qualité visuelle. DA3D fonctionne particulièrement bien en améliorant les textures des images et en supprimant les effets d'escalade. DA3D travaille sur un autre algorithme de débruitage, qui sert de guide, et presque toujours améliorer ses résultats. De cette façon, le débruitage basé sur la fréquence peut être appliqué au-dessus des algorithmes de débruitage basés sur des patchs, résultant d'une méthode hybride qui maintient les forces des deux. Les expériences démontrent que, à la suite de ce qui a été pensé, le débruitage basé sur la fréquence peut battre des algorithmes de pointe sans présenter d'artefacts. Dans ce travail, DA3D est présenté, et les courbes de retrait ad hoc sont calculées en fonction de l'algorithme utilisé comme guide.

Le deuxième résultat présenté est Denoising multi-échelle, un cadre multi-échelle applicable à tout algorithme de débruitage. Une analyse qualitative montre que les algorithmes de débruitage actuels se comportent mieux sur le bruit haute fréquence. Cela est dû à la taille relativement petite des correctifs et des fenêtres de recherche utilisés dans ces algorithmes à une seule échelle. Au lieu d'agrandir ces fenêtres, cela peut causer d'autres sortes de problèmes, le travail propose de décomposer l'image sur une pyramide, à l'aide de la Transformation de Cosinus Discrete. Nous introduisons un nouveau schéma de reconstruction dans la pyramide en évitant l'apparition d'artefacts de sonnerie. Cette méthode élimine la plupart des bruits de basse fréquence et améliore à la fois le PSNR et les résultats visuels pour les zones lisses et texturées.

Une troisième question principale abordée dans cette thèse est l'évaluation des algorithmes de débruitage. Les expériences indiquent que le PSNR n'est pas toujours un bon indicateur de qualité visuelle pour les algorithmes de débruitage, puisque, par exemple, un artefact sur une surface lisse peut être plus perceptible qu'une modification subtile d'une texture. Une nouvelle mesure est proposée pour améliorer la situation. Au lieu d'une seule valeur, un «PNSR lisse» et un «PSNR de texture» sont présentés pour mesurer le résultat d'un algorithme pour ces deux types de régions d'image. Nous prétendons qu'un al-

gorithme de débruitage, pour être considéré comme acceptable, doit au moins bien fonctionner en ce qui concerne les deux métriques. À la suite de cette affirmation, une analyse des algorithmes actuels est effectuée, et elle est comparée avec les résultats combinés du cadre multi-échelle et DA3D. Nous avons trouvé que la solution optimale pour le débruitage d'image est l'application d'un rétrécissement de fréquence appliqué uniquement aux régions régulières, alors qu'une méthode basée sur un patch multi-échelles sert de guide. Cela semble résoudre une question de longue date pour laquelle DDID a donné le premier indice : quel est le rôle respectif des méthodes de rétrécissement de fréquence et d'auto-similarité basées sur le débruitage d'image ? Nous décrivons un algorithme de débruitage d'image qui semble avoir un meilleur rendement en qualité et PSNR que tout autre basé sur la bonne combinaison des deux principes de débruitage.

Dans notre dernière contribution, une étude sur la faisabilité de débruitage externe est réalisée, où les images sont débruitées au moyen d'une grande base de données de patchs externes silencieux. Ceci suit un travail de Levin et Nadler, en 2011, qui prétend qu'un débruitage bayésien optimal est accessible par une formule intégrale simple à condition qu'il soit appliqué à une base de données de patchs gigantesque et représentatif. Nous démontrons par un argument mathématique combiné à des observations empiriques sur l'espace des patchs que cet espace peut être factorisé, réduisant ainsi considérablement le nombre de patchs nécessaires pour calculer l'intégrale sur l'espace des patchs.

Enfin, des résultats secondaires sont présentés. Une brève étude de la façon d'appliquer des algorithmes de débruitage sur de vraies images RAW est effectuée. Une version améliorée et performante de l'algorithme Non-Local Bayes est présentée, ainsi qu'une version en deux étapes de DCnoDenoising. Ce dernier est intéressant pour son extrême simplicité et pour sa rapidité.

Dans cette introduction, nous donnons un bref aperçu des chapitres de ce travail.

0.1 Chapitre 1 : DA3D et les méthodes à double domaine

Le débruitage d'image est l'un des défis fondamentaux de la restauration d'images [40]. Il consiste à estimer une image inconnue silencieuse $\mathbf{y}$ à partir d'une observation bruyante $\mathbf{x}$. Nous considérons le modèle classique de dégradation d'image

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{1}$$

où l'observation $\mathbf{y}$ de l'image $\mathbf{x}$ est contaminée par un bruit Gaussien blanc additif $\mathbf{n}$ de variance σ^2.

Toutes les méthodes de débruitage supposent une certaine régularité de l'image sous-jacente. En fonction de cette hypothèse, ils peuvent être divisés, entre autres, en domaines de transformation et de domaine spatial.

Les méthodes de domaine de transformation fonctionnent en réduisant (ou en limitant) les coefficients de quelque domaine de transformation [18, 34, 51]. Le filtre de Wiener [60] est l'une des premières méthodes de ce type opérant sur la transformée de Fourier. Donoho et al. [11] l'ont étendue au domaine d'ondelettes.

Les méthodes spatiales utilisent traditionnellement une notion locale de régularité avec des algorithmes de préservation des arêtes comme la variation totale [49], la diffusion anisotrope [41], ou le filtre bilatéral [56]. Aujourd'hui, cependant, les méthodes spatiales atteignent des résultats remarquables en exploitant les auto-similitudes de l'image [2]. Ces

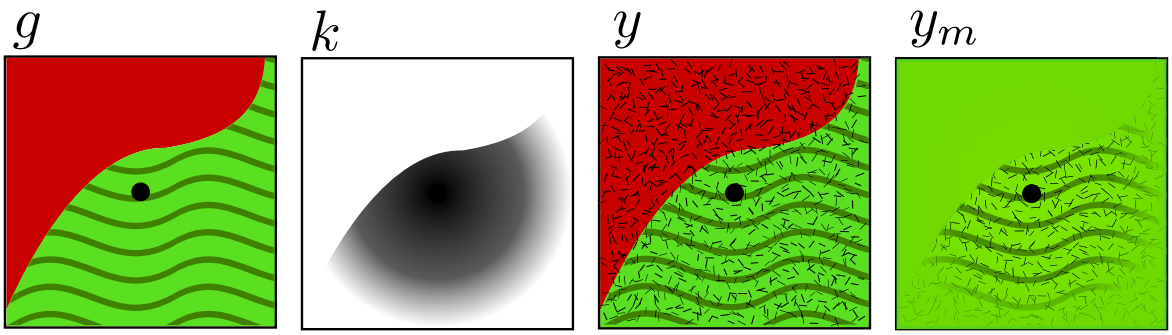


FIGURE 1 – Illustration du prétraitement d'un patch par DDID. Le noyau k est calculé en utilisant le guide g. Dans le patch modifié y_m, toutes les discontinuités d'objets ont été supprimées, laissant seulement l'information de texture correspondant à l'objet sélectionné par le noyau k. Les pixels supprimés sont remplacés par la moyenne de la partie significative du patch.

méthodes basées sur des patchs sont non locales puisqu'elles sont débruitées en faisant la moyenne de patchs similaires dans l'image. Le débruitage basé sur les patchs s'est développé en tentatives de modélisation de l'espace patch d'une image ou d'un ensemble d'images. Ces techniques modélisent le patch en tant que représentations éparses sur les dictionnaires cite Elad2006, Mairal2008, Mairal2009, Yu2010, Dong2013, en utilisant des modèles de Mélanges d'Echelle Gaussienne [45,46,67] ou avec des approches non paramétriques Base de données de patches [32,33,39,43].

Les méthodes actuelles de débruitage à l'état de l'art telles que BM3D [7] et NL-Bayes [27] tirent parti des approches spatiale et de transformation. Ils regroupent des patchs d'image semblables et les suppriment conjointement par filtrage collaboratif sur un domaine transformé. De plus, ils procèdent par l'application de deux étages de débruitage légèrement différents, le second utilisant la sortie du premier comme guide.

Certaines méthodes récemment proposées utilisent le résultat d'un algorithme différent comme guide pour une nouvelle étape de débruitage. Combinant par exemple des principes non locaux avec la décomposition spectrale [55], ou BM3D avec des réseaux de neurones [5]. Cela permet de mélanger différents principes de démontage, donnant ainsi des résultats de haute qualité. DDID [22] est un algorithme itératif qui utilise une image guide (à partir d'une itération précédente) pour déterminer des régions spatialement uniformes auxquelles le retrait de Fourier pourrait être appliqué sans introduire d'artefacts de sonnerie. Plusieurs méthodes [21,23,42] utilisent une seule étape de DDID avec une image de guide produite par un algorithme différent. Cela donne de bien meilleurs résultats que le DDID original. La raison de leur succès est l'utilisation de grandes (31×31) et la forme de patchs adaptatifs. En effet, le retrait de Fourier fonctionne mieux sur de grands blocs stationnaires.

Malheureusement, DDID a un coût de calcul prohibitif, car elle débruite paradoxalement un grand patch pour récupérer un pixel unique. De plus, contrairement à d'autres méthodes, l'agrégation de ces patchs n'améliore pas les résultats puisqu'elle introduit le flou. Nous montrons dans cet article que ces deux problèmes peuvent être résolus en introduisant une nouvelle sélection de patch, accompagnée d'une stratégie d'agrégation pondérée.

Notre premier chapitre présente le DA3D (Data Adaptive Dual Domain Denoising),

Method	$\sigma = 5$	$\sigma = 10$	$\sigma = 25$	$\sigma = 40$	$\sigma = 80$
BM3D	39.55	36.01	31.79	29.23	**26.83**
	39.59	**36.23**	**32.15**	**29.95**	**27.00**
	+0.05	+0.22	+0.36	+0.72	+0.16
DDID	39.20	35.91	31.88	29.75	26.41
	39.11	35.87	**31.99**	**29.98**	**26.85**
	-0.09	-0.04	+0.11	+0.23	+0.43
NLB	**39.77**	**36.09**	31.71	29.35	26.64
	39.67	**36.26**	**32.18**	**30.09**	26.74
	-0.10	+0.16	+0.47	+0.74	+0.11
NLDD	39.26	35.90	31.98	29.90	26.60
	39.17	35.87	**31.99**	**29.97**	26.63
	-0.09	-0.04	+0.02	+0.07	+0.03
PID	39.03	35.81	**32.05**	**30.10**	**27.00**
	38.99	35.80	**32.03**	**30.02**	**26.83**
	-0.04	-0.02	-0.02	-0.08	-0.17

TABLE 1 – Comparaison moyenne de PSNR entre les méthodes de pointe pour les images en couleur. La première ligne de chaque rangée montre la PSNR moyenne obtenue en débruitant les images de test. La seconde ligne représente le PSNR moyen de DA3D en utilisant l'algorithme de débruitage correspondant pour générer le guide. La troisième ligne montre l'amélioration moyenne due à DA3D. Le meilleur résultat pour chaque niveau de bruit est indiqué dans **bold**, et ceux dans une plage de 0.2 dB sont affichés en **gris**.

une nouvelle méthode de débruitage "last step" qui effectue le rétrécissement du domaine fréquentiel sur les patchs adaptatifs et adaptés aux données. DA3D améliore constamment les résultats des méthodes de pointe telles que BM3D ou NL-Bayes avec peu de temps de calcul supplémentaire.

Notre deuxième contribution dans DA3D améliore encore la qualité des résultats en adaptant le traitement aux données sous-jacentes. L'apparition de l'escalier est bien connue pour les méthodes non locales [3]. Pour minimiser l'influence de tels artefacts présents dans l'image guide, nous utilisons une régression linéaire non linéaire de premier ordre [53, 54] pour estimer, pour chaque patch, une approximation affine de l'image cohérente avec les données dans le pièce. Le débruitage est alors réalisé par rapport à cette approximation. Cette approche adaptée aux données est une autre innovation permise par l'utilisation de grands correctifs et améliore sensiblement la qualité des résultats sur des régions lisses de l'image.

La section 1.4 rappelle le post-process DDID et développe une explication de son efficacité, illustrée dans la Figure 1, où il devient évident que la méthode sépare chaque patch de texture, tout en évitant l'interférence des bords et d'autres textures voisines. Les sections 1.5 et 1.6 présentent l'algorithme DA3D, décrivant d'abord l'agrégation éparse proposée, puis les patchs adaptatifs de données. La performance de DA3D est largement validée dans les expériences de la section 1.7, où il devient évident qu'il améliore significativement le résultat de la plupart des algorithmes de débruitage, comme le montre le tableau 1.

0.2 Chapitre 2 : Un cadre de débruitage multi-échelle

Dans le chapitre 1, nous avons fourni une nouvelle vue sur le seuillage de la transformée de fréquence, montrant qu'il pourrait performer de manière excellente comme dernière étape de n'importe quel algorithme de débruitage basé sur un patch. Dans le Chapitre 2 nous adressons une autre question non traitée par la plupart des algorithmes de débruitage de l'état de l'art. Nous avons observé que la plupart d'entre eux limitent leur action à un voisinage pixel limité. Cela signifie clairement que le bruit à basse fréquence reste non traité, comme le montre clairement la figure 2. C'est un inconvénient qui produit des artefacts très visibles dans les zones plates, comme nous le verrons dans la partie expérimentale de ce chapitre, et dans notre étude expérimentale du chapitre 3. Ainsi, dans ce chapitre, nous proposons une nouvelle perspective d'un autre principe : la représentation à plusieurs échelles. Ce principe a déjà été exploré dans [4, 15]. Même si les résultats ne sont que partiellement satisfaisants, les idées présentées sont simples et prometteuses. Le problème principal de leur approche est que dans les échelles supérieures, le bruit devient corrélé, ce qui réduit la performance des algorithmes standard. Un autre travail qui tente d'utiliser un modèle multi-échelle est [47]. La différence avec notre travail est que [47] n'utilise pas d'algorithme classique de débruitage dans le processus et qu'il n'évite pas les artefacts dans la recomposition.

La représentation multi-échelle est aussi intrinsèquement présente dans les algorithmes de débruitage à base d'ondelettes [11,18,34,45]. Il est également présent dans [52], où l'algorithme KSVD est appliqué sur une décomposition en ondelettes de l'image. L'amélioration par rapport à une seule échelle KSVD est importante, en particulier pour les PSNR élevés, mais puisque la décomposition en ondelettes ne permet pas une recomposition conservatrice, les auteurs ont besoin de ce qu'ils appellent une stratégie de fusion, afin de réduire les artefacts.

Débruiter une pyramide multi-échelle est un moyen de traiter plus d'informations pour atteindre un meilleur résultat. D'autres efforts ont été faits pour fournir plus d'informations à l'algorithme de débruitage. En particulier, il convient de citer [5, 65], qui essaie d'améliorer les résultats d'un algorithme de débruitage en utilisant une base de données externe d'images similaires.

Dans le chapitre 2, nous présentons une nouvelle structure multi-échelle qui peut être appliquée à tout autre algorithme de débruitage existant, améliorant constamment ses résultats (voir Tableau 2). Le cadre utilise une simple pyramide DCT et n'est pas exigeant en termes de calcul.

Simplement en utilisant la pyramide DCT peut conduire à des artefacts de sonnerie. Nous résolvons ce problème en introduisant ce que nous appelons *recomposition conservatrice*, ce qui permet de conserver les avantages de la pyramide tout en évitant ses problèmes.

Le résultat est un moyen de transformer n'importe quel algorithme de débruitage en un multi-échelle, avec des améliorations tant en qualité visuelle et PSNR, et avec peu de coût supplémentaire.

0.3 Chapitre 3 : Critères de qualité pour le débruitage d'image

Habituellement, la performance des algorithmes de débruitage est mesurée en termes de PSNR. Un autre indice de qualité célèbre est le SSIM [58]. Malheureusement, aucune des

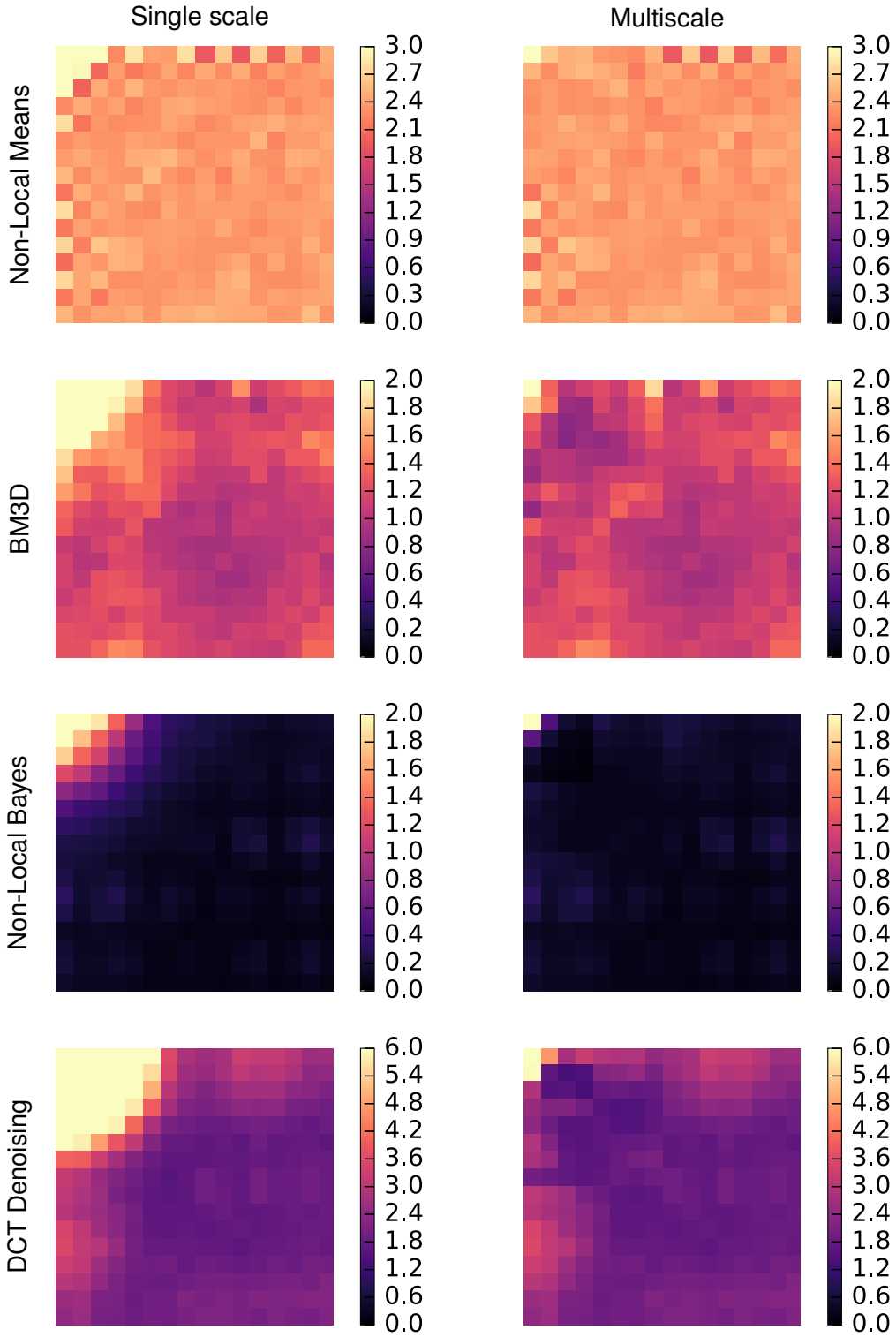


FIGURE 2 – DCT du résultat de divers algorithmes de débruitage appliqués à une image de bruit blanc pur, avec et sans le cadre multi-échelle. Notez qu'il reste du bruit dans le coin supérieur gauche de la version Single Scale, qui contient les basses fréquences de l'image. Dans la version multi-échelle, le bruit résiduel est beaucoup plus uniforme à travers les fréquences.

Noise	Best single-scale		Best multi-scale		Gain
$\sigma = 10$	38.59dB	BM3D	38.71dB	NLB	+0.12dB
$\sigma = 20$	34.97dB	BM3D	35.09dB	NLB	+0.12dB
$\sigma = 30$	32.87dB	BM3D	33.07dB	NLB	+0.20dB
$\sigma = 40$	31.37dB	BM3D	31.65dB	NLB	+0.28dB
$\sigma = 50$	30.30dB	NLB IPOL	30.61dB	NLB	+0.31dB
$\sigma = 60$	29.32dB	NLB IPOL	29.75dB	NLB	+0.43dB
$\sigma = 70$	28.65dB	BM3D	29.02dB	NLB	+0.37dB
$\sigma = 80$	27.97dB	BM3D	28.40dB	NLB	+0.43dB
$\sigma = 90$	27.30dB	BM3D	27.86dB	NLB	+0.56dB
$\sigma = 100$	26.75dB	NLB IPOL	27.46dB	NLB IPOL	+0.71dB

TABLE 2 – Résultats avec les meilleurs paramètres pour chaque algorithme. Les résultats sont une moyenne sur les images de la figure 2.10.

deux méthodes n'est étroitement corrélée avec la qualité visuelle, puisqu'elles ne sont pas suffisamment sensibles aux artefacts.

Différents algorithmes de débruitage se comportent différemment sur les textures et les zones planes d'images. La mesure du PSNR du résultat peut être trompeuse, car un algorithme peut fournir un bon résultat sur le premier et un mauvais sur celui-ci, ou vice versa. Par conséquent, nous proposons de découpler la mesure de qualité d'un résultat en deux valeurs : la PSNR plate (ou fPSNR) et la texture PSNR (ou tPSNR), en utilisant une carte de fluidité (Figure 3) pour discriminer des régions planes et texturées d'une image.

Nous avons appliqué les paramètres décrits dans ce chapitre aux résultats des algorithmes de débruitage les plus courants.

Le chapitre 1 et le chapitre 2 présentent une méthode pour améliorer le résultat d'autres algorithmes de débruitage. Dans cette section, nous présentons deux façons différentes de combiner les deux, afin d'obtenir un meilleur algorithme débruitage de performance.

Puisque DA3D et le framework Multiscale ont besoin d'un algorithme "baseline" pour fonctionner, nous avons choisi de nous focaliser sur Bayes non local [27] et BM3D [7]. Le premier algorithme est particulièrement adapté au DA3D et au framework Multiscale, car il a relativement peu d'artefacts et un résultat lisse, alors que nous avons décidé d'inclure le second pour offrir une comparaison équitable avec ce qui est encore considéré comme l'état de l'art.

Afin de combiner DA3D et le framework Multiscale, il y a fondamentalement deux possibilités : appliquer DA3D en premier ou appliquer le framework Multiscale en premier. Nous les présentons tous les deux. Figure 4 donne le diagramme de la combinaison gagnante.

0.4 Chapitre 4 : Débruitage externe

Un document séminal récente sur les limites absolues de la débruitage d'image [32] propose une méthode de dénoisage de patch qui réalise effectivement l'erreur quadratique moyenne minimale, étant donné tous les patchs d'image connus. Il est extrêmement important d'atteindre ces limites absolues, mais ils nécessitent le traitement d'une base de données illi-

FIGURE 3 – Exemple de carte de fluidité.

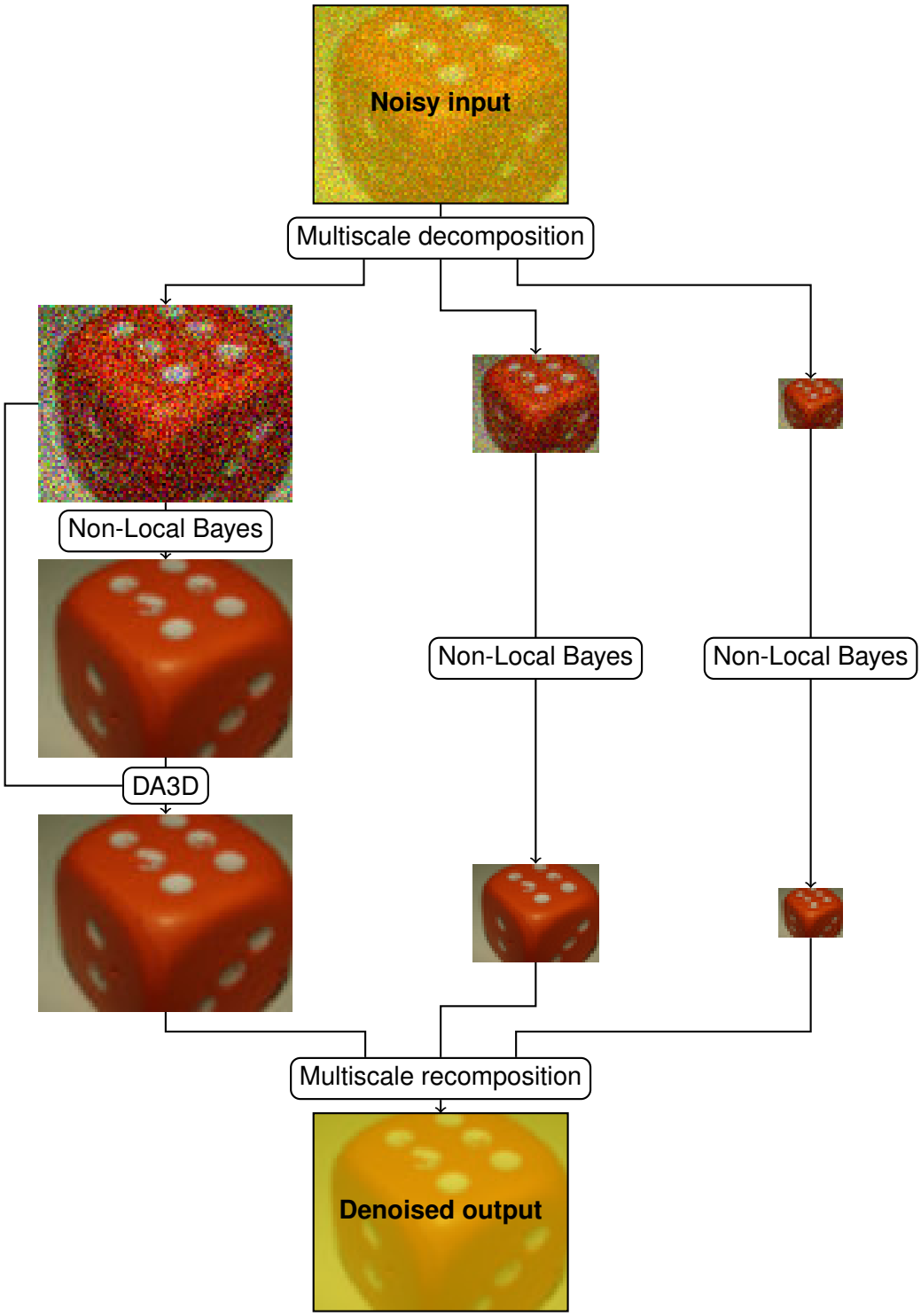


FIGURE 4 – Schéma de la méthode de débruitage *da3d_ms* qui combine non local Bayes, DA3D et le cadre multi-échelle. Les valeurs pour les recompositions multiscalaires sont celles utilisées pour les Bayes non locales dans le chapitre 2 (0.5 pour $\sigma = 10$ et $\sigma = 20$, 0.6 pour $\sigma = 40$ et 0.7 pour $\sigma = 70$). DA3D est appliqué en utilisant le résultat de Non-Local Bayes comme guide uniquement sur l'échelle plus fine. L'application de DA3D à une échelle plus grossière dégrade les résultats en raison de la présence de sonnerie dans la pyramide DCT.

DB Size	Shotgun method	NSD
10000	23.47 dB	26.79 dB
100000	25.92 dB	27.32 dB
4000000	27.66 dB	28.06 dB

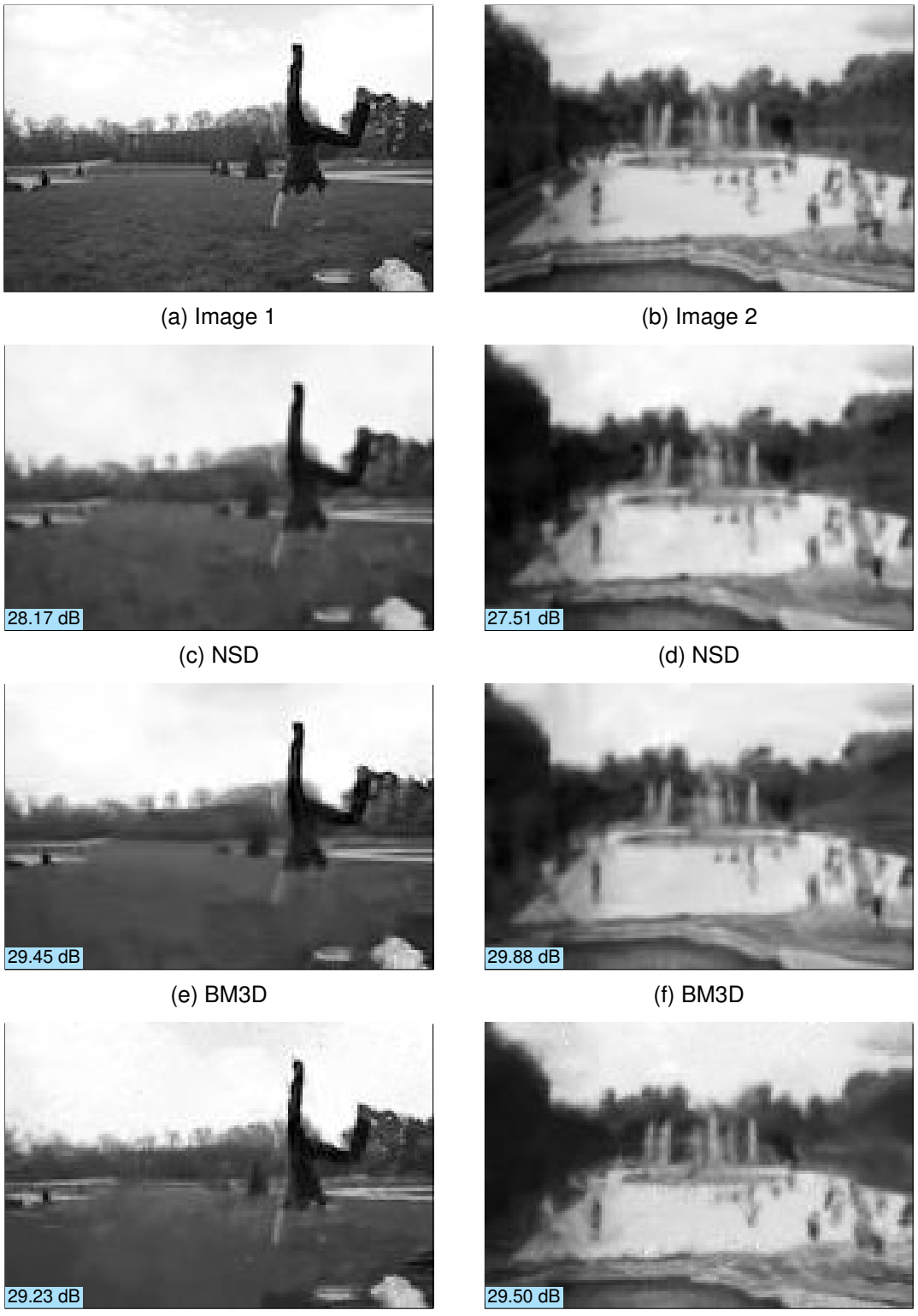
TABLE 3 – Moyenne PSNR obtenue après débruitage de 3500 patchs avec une taille de base de données fixe.

mitée. Dans le document susmentionné, cette base de données comportait 10 milliards de correctifs. Dans cet article, nous démontrons qu'en factorisant l'espace de patch, la méthode peut être accélérée par un facteur de plus de mille, tout en maintenant la prétention théorique que la méthode est optimale. L'utilisation de la méthode sur les images réelles démontre son potentiel pour battre l'état de l'art, car il fonctionne mieux sur les patchs difficiles.

Par "shotgun", on entend des méthodes qui ont l'intention de débruiter les patchs par un algorithme totalement non local, dans lequel le patch est comparé à un patch Modèle obtenu à partir d'un ensemble de patchs *très grand*, en supposant qu'ils représentent "tous les correctifs d'image naturels possibles".

Par exemple, Zoran et Weiss [67] ont introduit l'algorithme EPLL, dans lequel ils apprennent la distribution d'espace de patch à partir d'un mélange de gaussiens formés avec des patchs de 2×10^6. Ensuite, ils utilisent ce modèle comme un préalable pour débruiter les correctifs en essayant de maximiser la *Expected Patch Log Likelihood*, au lieu de débruiter chaque patch séparément. Pour ce faire, ils minimisent une fonction de coût en utilisant une méthode d'optimisation appelée "Half Quadratic Splitting" [48]. Zontak et Irani [66] ont analysé la performance de débruitage en utilisant des patchs extraits de la même image, ou ce qu'ils appellent des recherches de correctifs internes, contre l'extraction de correctifs "externes" à partir d'une base de données géante. Ils ont conclu que les patchs semblables ont tendance à se reproduire beaucoup plus souvent à l'intérieur de la même image que dans toute collection externe aléatoire d'images naturelles. En outre, la redondance de patch interne décroît rapidement avec la croissance de la distance spatiale à partir du patch, et sa teneur en gradient. Enfin, ils ont observé que la recherche d'un patch représentatif externe aussi bon pour chaque patch d'une image, nécessite une énorme base de données externe, ce qui rend le problème de débruitage computacionalement intraitable. Lee et al. [31] a utilisé le cadre bayésien pour dériver un algorithme de débruitage d'image non locale MMSE, qui utilise à la fois des correctifs d'image internes ainsi qu'un codebook externe avec des correctifs sans bruit. Ils effectuent le débruitage en normalisant l'espace-patch par le moyen, une idée qui sera encore exploitée ici. Plusieurs méthodes "shotgun" ont été récemment appliquées dans d'autres tâches de traitement d'image, telles que l'achèvement des scènes [19] (en utilisant 2,3 millions d'images) ou la reconnaissance de scènes [57] (avec 80 millions de patches 32×32).

Passons maintenant à la recherche de limites absolues pour le débruitage d'image. Les limites inférieures du type Cramer-Rao sur la performance RMSE atteignable donnée par Chaterjee et Milanfarci Chatterjee2010 sont optimistes : elles permettent la possibilité d'une augmentation significative de la performance de débruitage pour une large classe d'images à certains SNR, en particulier, Pour des images synthétiques par morceaux synthétiques. Pourtant, un article récent de A. Levin et B. Nadler [32] fournit une deuxième réponse à



(a) Image 1 (b) Image 2

(c) NSD (d) NSD

(e) BM3D (f) BM3D

(g) NL-Bayes (h) NL-Bayes

FIGURE 5 – Résultats et comparaison des méthodes de débruitage (valeurs PSNR entre parenthèses).

la question des limites absolues soulevées par [6]. Ce document séminal utilise une méthode de dérogation de fusil de chasse pour approcher les limites, qui est librement basé sur l'algorithme NL-moyens, avec des paramètres adéquats pour tenir compte d'une estimation minimale moyenne linéaire bayésienne carrée (LMMSE) du patch bruyant donné une base de données de 10^{10} patchs sans bruit. La seule et importante différence est que les correctifs similaires sont recherchés sur une base de données de correctifs, au lieu de sur l'image elle-même. De plus, grâce à un argument mathématique simple et à des simulations intensives sur l'espace du patch, les auteurs sont capables d'estimer *la meilleure erreur d'estimation moyenne atteignable par tout algorithme de débruitage par patch* (pour être plus précis, c'est la meilleure erreur d'estimation moyenne connaissant un seul patch et tous les observés ; mais la plupart des algorithmes de débruitage utilisent réellement la connaissance des patchs chevauchants dans l'image pour le débruiter). Après avoir effectué une expérimentation et les comparer avec des algorithmes de débruitage à la fine pointe de la technologie, ils concluent que les résultats pourraient être inférieurs à 0.1dB de la méthode de débruitage optimale possible, au moins en utilisant de petites tailles de patch ou sous de grandes quantités de bruit. Aucune autre marge d'amélioration. Ils proposent d'augmenter la taille du support pour obtenir de meilleures valeurs de PSNR. Pourtant, en raison de la malédiction de la dimensionnalité, les techniques non paramétriques ne seront pas capables de trouver de bons patchs similaires. Cela conduit les auteurs à renoncer à une méthode de fusil de chasse, et à revenir à des approches paramétriques. Cependant, une limitation de ce travail est que les contraintes de calcul limitaient l'expérimentation à de petits correctifs. Par conséquent, les limites réelles indépendamment de la taille du patch sont encore inconnues.

Dans [33], les mêmes auteurs étudient la dépendance de l'erreur de débruitage sur la taille du patch. De la même manière que Zontak et Irani [66], ils concluent que, lorsqu'on augmente la taille de la fenêtre, les patchs avec plus de détails nécessitent une base de données beaucoup plus grande et que le gain est très faible. Dans le même temps, le contraire se produit avec des correctifs plus lisses : l'augmentation de la taille de la fenêtre est beaucoup plus récompensé dans la performance finale débruitage. Ceci les amène à proposer une stratégie adaptative utilisant des tailles variables de fenêtre selon la complexité locale de correctif. Enfin, ils explorent à nouveau les limites fondamentales de débruitage, avec une taille de fenêtre infinie et en supposant une image naturelle parfaite avant. Pour ce faire, ils utilisent un modèle simplifié de formation d'images de feuilles mortes et la combinent avec une hypothèse d'invariance d'échelle pour des images naturelles, ce qui implique une convergence en loi de puissance. En utilisant cette loi de puissance pour modéliser le PSNR par rapport à la taille de fenêtre, ils s'adaptent au modèle et extrapolent son résultat pour obtenir des limites de débruitage optimales. Ils concluent qu'il ya encore place à amélioration (autour de 1dB).

Dans le chapitre 4, nous nous en tenons à l'approche du premier canon [32] et avons l'intention de le rendre possible en accélérant considérablement le calcul de leur intégrale de Monte Carlo. L'algorithme proposé obtient des résultats similaires en utilisant moins de ressources informatiques et en moins de temps. Pour ce faire, un espace de probabilité plus petit est utilisé en factorisant l'espace des patchs. Cela permet de Fonctionnent avec des classes d'équivalence entières de patchs au lieu de patchs simples, ce qui permet une diminution significative du nombre de calculs nécessaires et une réduction de la mémoire requise. Une comparaison est faite entre notre algorithme et deux méthodes à la fine pointe

de la technologie, l'algorithme BM3D [7, 26] et la méthode Non Local Bayes [27] montrant comment notre méthode surpasse à la fois pour l'algorithme Tâche de débruiter les patchs complexes.

Chapter 1

DA3D and the Dual Domain Methods

This chapter presents DA3D (Data Adaptive Dual Domain Denoising), a "last step denoising" method that takes as input a noisy image and as a guide the result of any state-of-the-art denoising algorithm. The method performs frequency domain shrinkage on shape and data-adaptive patches. Unlike other dual denoising methods, DA3D doesn't process all the image samples, which allows it to use large patches (64×64 pixels). The shape and data-adaptive patches are dynamically selected, effectively concentrating the computations on areas with more details, thus accelerating the process considerably. DA3D also reduces the staircasing artifacts sometimes present in smooth parts of the guide images. The effectiveness of DA3D is confirmed by extensive experimentation. DA3D improves the result of almost all state-of-the-art methods, and this improvement requires little additional computation time.

1.1 Introduction

Image denoising is one of the fundamental image restoration challenges [40]. It consists in estimating an unknown noiseless image $\mathbf{y}$ from a noisy observation $\mathbf{x}$. We consider the classic image degradation model

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{1.1}$$

where the observation $\mathbf{y}$ of the image $\mathbf{x}$ is contaminated by an additive white Gaussian noise $\mathbf{n}$ of variance σ^2.

All denoising methods assume some underlying image regularity. Depending on this assumption they can be divided, among others, into transform-domain and spatial-domain methods.

Transform domain methods work by shrinking (or thresholding) the coefficients of some transform domain [18, 34, 51]. The Wiener filter [60] is one of the first such methods operating on the Fourier transform. Donoho et al. [11] extended it to the wavelet domain.

Space-domain methods traditionally use a local notion of regularity with edge-preserving algorithms such as total variation [49], anisotropic diffusion [41], or the bilateral filter [56]. Nowadays however spatial-domain methods achieve remarkable results by exploiting the self-similarities of the image [2]. These patch-based methods are non-local as they denoise by averaging similar patches in the image. Patch-based denoising has developed into attempts to model the patch space of an image, or of a set of images. These techniques model the patch as sparse representations on dictionaries [10, 14, 35, 36, 64], using Gaussian Scale Mixtures models [45, 46, 67], or with non-parametric approaches by sampling from a huge database of patches [32, 33, 39, 43].

Current state-of-the-art denoising methods such as BM3D [7] and NL-Bayes [27] take advantage of both space- and transform-domain approaches. They group similar image patches and jointly denoise them by collaborative filtering on a transformed domain. In addition, they proceed by applying two slightly different denoising stages, the second stage using the output of the first one as its guide.

Some recently proposed methods use the result of a different algorithm as their guide for a new denoising step. Combining for instance, nonlocal principles with spectral decomposition [55], or BM3D with neural networks [5]. This allows one to mix different denoising principles, thus yielding high quality results. DDID [22] is an iterative algorithm that uses a guide image (from a previous iteration) to determine spatially uniform regions to which Fourier shrinkage could be applied without introducing ringing artifacts. Several methods [21, 23, 42] use a single step of DDID with a guide image produced by a different algorithm. This yields much better results than the original DDID. The reason for their success is the use of large (31×31) and shape-adaptive patches. Indeed, the Fourier shrinkage works better on large stationary blocks.

Unfortunately, DDID has a prohibitive computational cost, as it paradoxically denoises a large patch to recover a single pixel. Moreover, contrary to other methods, aggregation of these patches doesn't improve the results since it introduces blur. We show in this paper that these two problems can be solved by introducing a new patch selection, accompanied by a weighted aggregation strategy.

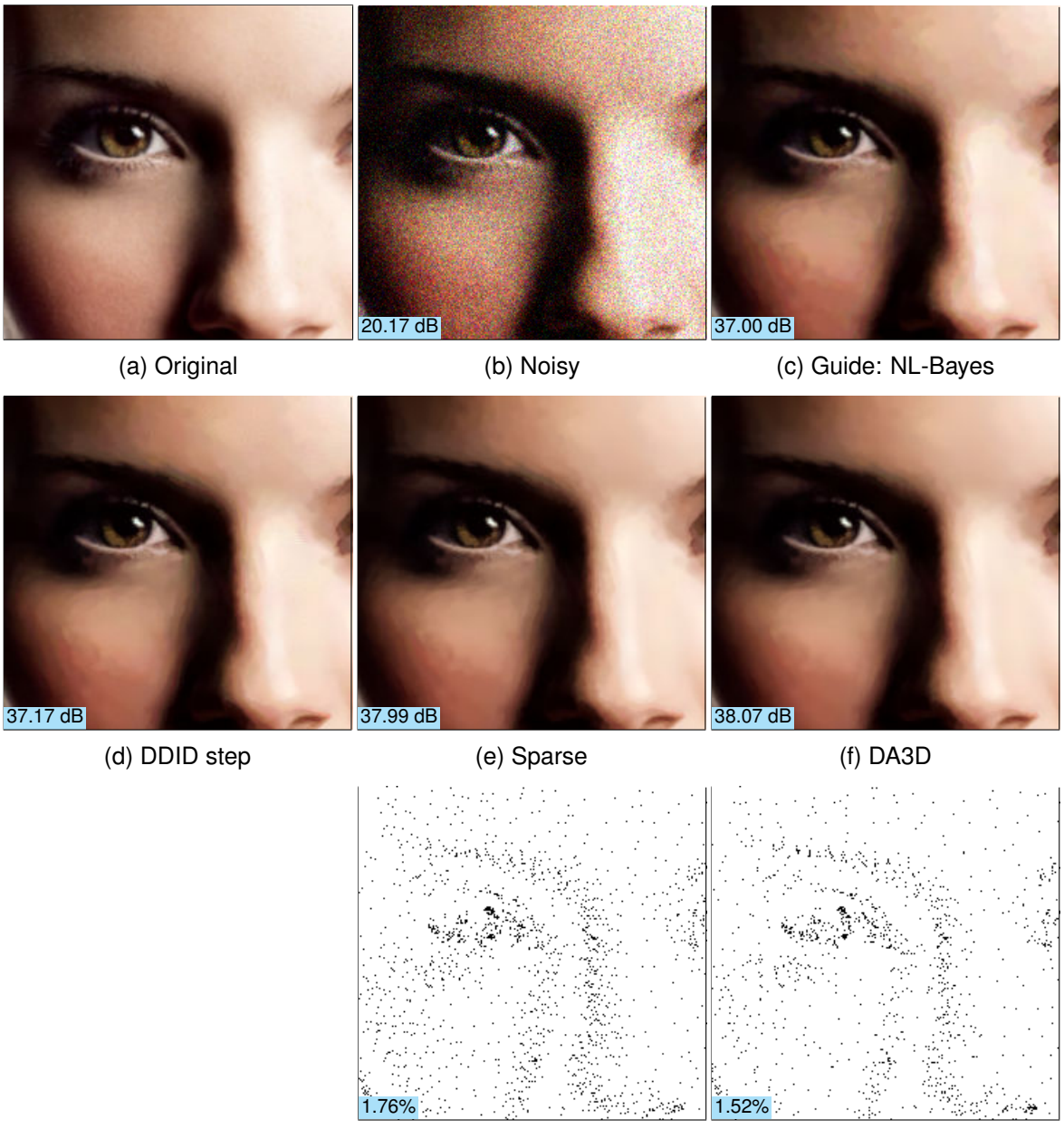
Contribution. This chapter presents DA3D (Data Adaptive Dual Domain Denoising), a new "last step" denoising method that performs frequency domain shrinkage on shape-adaptive and data-adaptive patches. DA3D consistently improves the results of state-of-the-art methods such as BM3D or NL-Bayes with little additional computation time.

Our second contribution in DA3D further improves the quality of the results by adapting the processing to the underlying data. The apparition of the staircasing is well known for non-local methods [3]. To mitigate the influence of such artifacts present in the guide image, we use a first order non-linear local kernel regression [53,54] to estimate, for each patch, an affine approximation of the image coherent with the data within the patch. The denoising is then performed with respect to this approximation. This data-adaptive approach is another innovation enabled by the use of large patches, and it noticeably improves the quality of the results on smooth regions of the image.

Section 1.4 recalls the DDID post-process. Sections 1.5 and 1.6 present the DA3D algorithm, first describing the proposed sparse aggregation, then the data-adaptive patches. The performance of DA3D is extensively validated in the experiments of section 1.7.

1.2 Related Work

The DDID algorithm [22] adapts the denoising patches (or blocks) to the shapes of the guide image and denoises them via Fourier shrinkage. The original algorithm iterated the process to obtain a guide image for the next step improving the results over consecutive iterations. However, the initial guide image is initialized with the noisy image, which generates considerable artifacts [42]. Several methods [21, 23, 42] use a single step of DDID as a "last step" denoising, yielding much better results. The reasons for the success of these methods are their large (31×31) and shape-adaptive patches. Indeed, the Fourier shrinkage works better on large stationary blocks. On this line, DA3D re-introduces aggregation showing

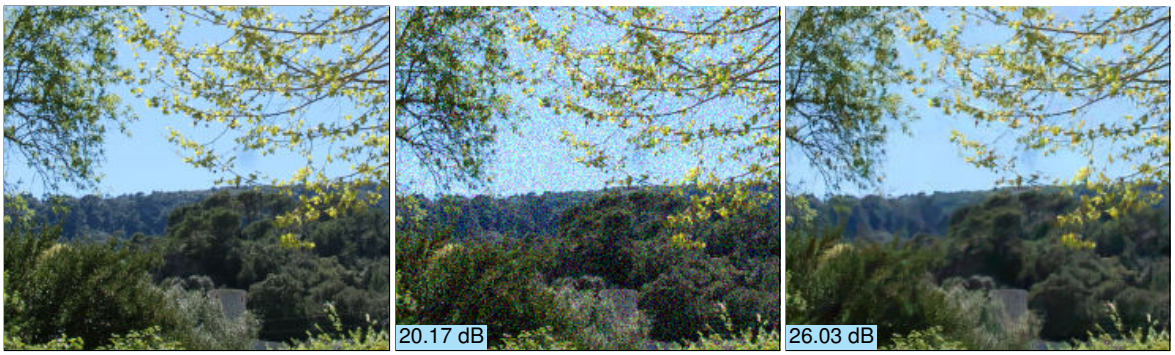


(a) Original (b) Noisy (c) Guide: NL-Bayes

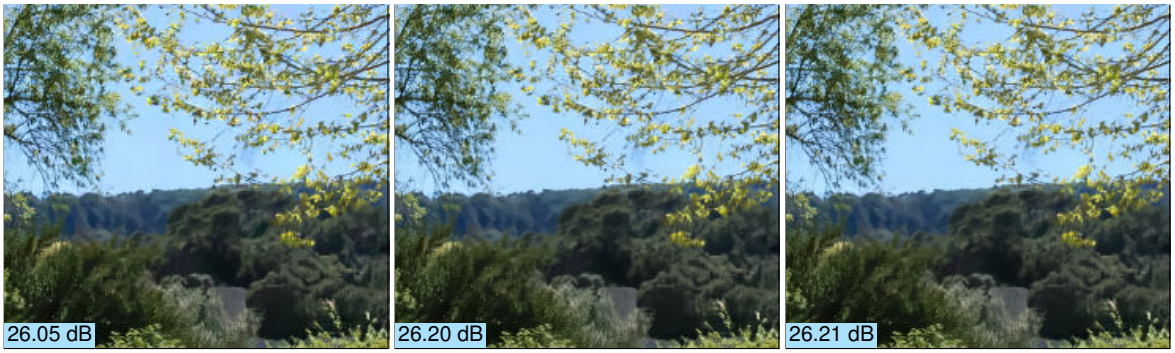
(d) DDID step (e) Sparse (f) DA3D

(g) Sparse samples (h) DA3D samples

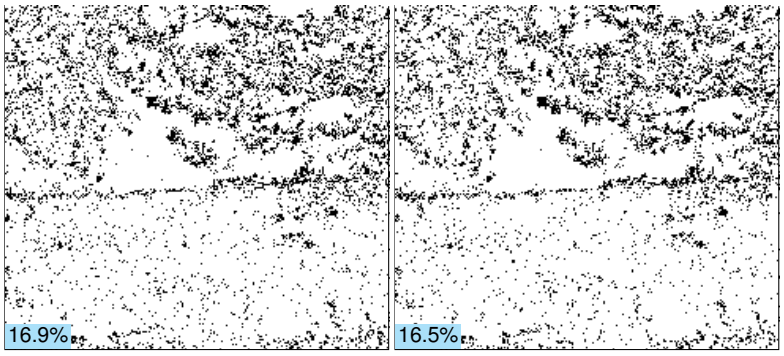
Figure 1.1 – Results of applying different post-processing steps on the same image (with noise $\sigma = 25$). The guide was produced with Non-Local Bayes. Figs. (g) and (h) show the centers of the blocks used for denoising (e) and (f) respectively (with $\tau = 2$). Image (d) was generated with 31×31 blocks, while for (e) and (f) 64×64 blocks were used. Notice in (e) and (f) the difference on the cheek and the forehead of the girl. Also, notice that in this case, with many smooth areas, the amount of patches used in the process is small (1.52% in the case of DA3D.



(a) Original (b) Noisy (c) Guide: NL-Bayes



(d) DDID step (e) Sparse (f) DA3D



(g) Sparse samples (h) DA3D samples

Figure 1.2 – Results of applying different post-processing steps on the same image (with noise $\sigma = 25$). The guide was produced with Non-Local Bayes. Figs. (g) and (h) show the centers of the blocks used for denoising (e) and (f) respectively (with $\tau = 2$). Image (d) was generated with 31×31 blocks, while for (e) and (f) 64×64 blocks were used. Unlike Fig. 1.1, DA3D uses a bigger portion of the image patches to denoise the image. This is due to a prominence of textured areas.

that these large blocks are also valid in the vicinity of the current pixel. This accelerates the process considerably and allows to use even larger (64×64) patches, which in turn leads to improved results.

Other shape adaptive denoising methods [8,9] build nonlocal groups of similar patches to collectively denoise them. However the considered shape adaptive patches are usually small (contained in an 8×8 block for BM3D-SAPCA). Although DA3D doesn't consider groups of patches, the patches are much larger allowing to extract more information about the underlying image structure. The process of DA3D is somehow complementary to the collaborative denoising, since it is better adapted to recover textures but relies on the structures provided in the guide image.

Among the guided approaches, the Global Image Denoising [55] is interesting because it takes the filter associated to a nonlocal method such as BM3D or NLM and approximates its principal eigenvectors. The filter obtained by combining these eigenvectors can average pixels from the entire image. The authors show that this method can obtain almost perfect results when the ideal guide image (noiseless) is used. DA3D uses large blocks but processes them in a transformed domain. We will see in the experiments that this yields better results when the guide is not perfect.

Locally adaptive regression kernel (LARK) [53, 54] extends the bilateral filtering [56] to data-adaptive filtering. LARK estimates the dominant orientation of the data at each point, and then applies a local steerable kernel with the estimated orientation. DA3D's affine regression (see Section 1.6) can be seen as a data-adaptive cross bilateral filter kernel. However instead of filtering using this kernel we use it to model the local smoothness of the image.

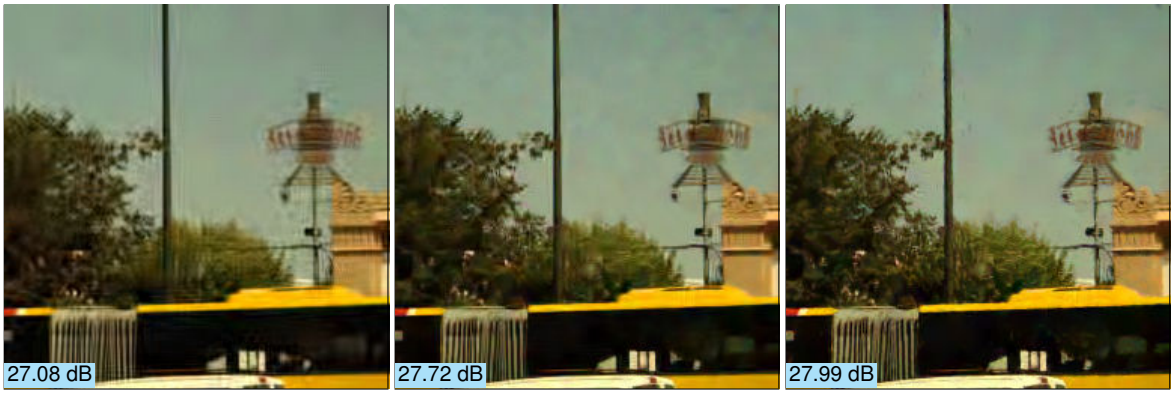
1.3 The Artifacts of Denoising Algorithms and their interpretation

The problem of image denoising is inherently ill-defined. The various algorithms designed to solve this problem must rely on some prior knowledge about the model of the image, in order to differentiate it from the noise [28]. The vast majority of modern denoising algorithms are patch-based, frequency-based or a combination of the two.

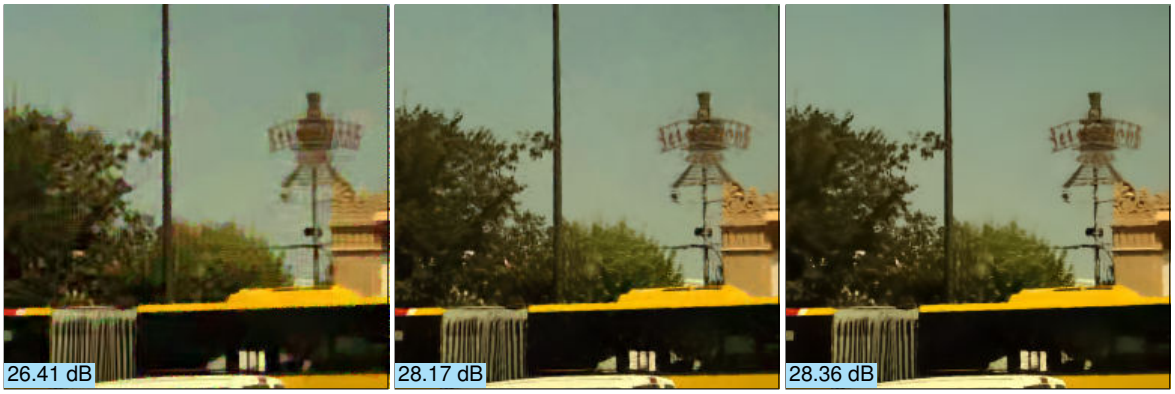
Patch-based algorithms exploit the self-similarity model, that takes into account the fact that natural images often present repeated elements. The idea behind patch-based algorithms is that, by grouping together similar patches, a model for those patches can be deduced. This model can be used to reduce the noise. The first, most famous denoising algorithms using self-similarity is Non-Local Means [2], where the similar patches are simply averaged in order to denoise. Relying on similar patches yields very good results, especially near edges and geometric structures; however, the results on textures can be blurry (see Fig. 1.3(c)), since the hypothesis of *perfect* self-similarity is often too strong. Recently, patch-based denoising has developed into attempts to model the patch space of an image, or of a set of images. These techniques model the patch as sparse representations on dictionaries [10, 14, 35, 36, 64], using Gaussian Scale Mixtures models [45, 46, 67], or with non-parametric approaches by sampling from a huge database of patches [32,33,39,43].

Other algorithms are *frequency based*. These algorithms suppose a certain degree of regularity on the image, and therefore they try to represent its patches in a basis that "sparsifies"

(a) Original (b) Noisy (c) Non-Local Means

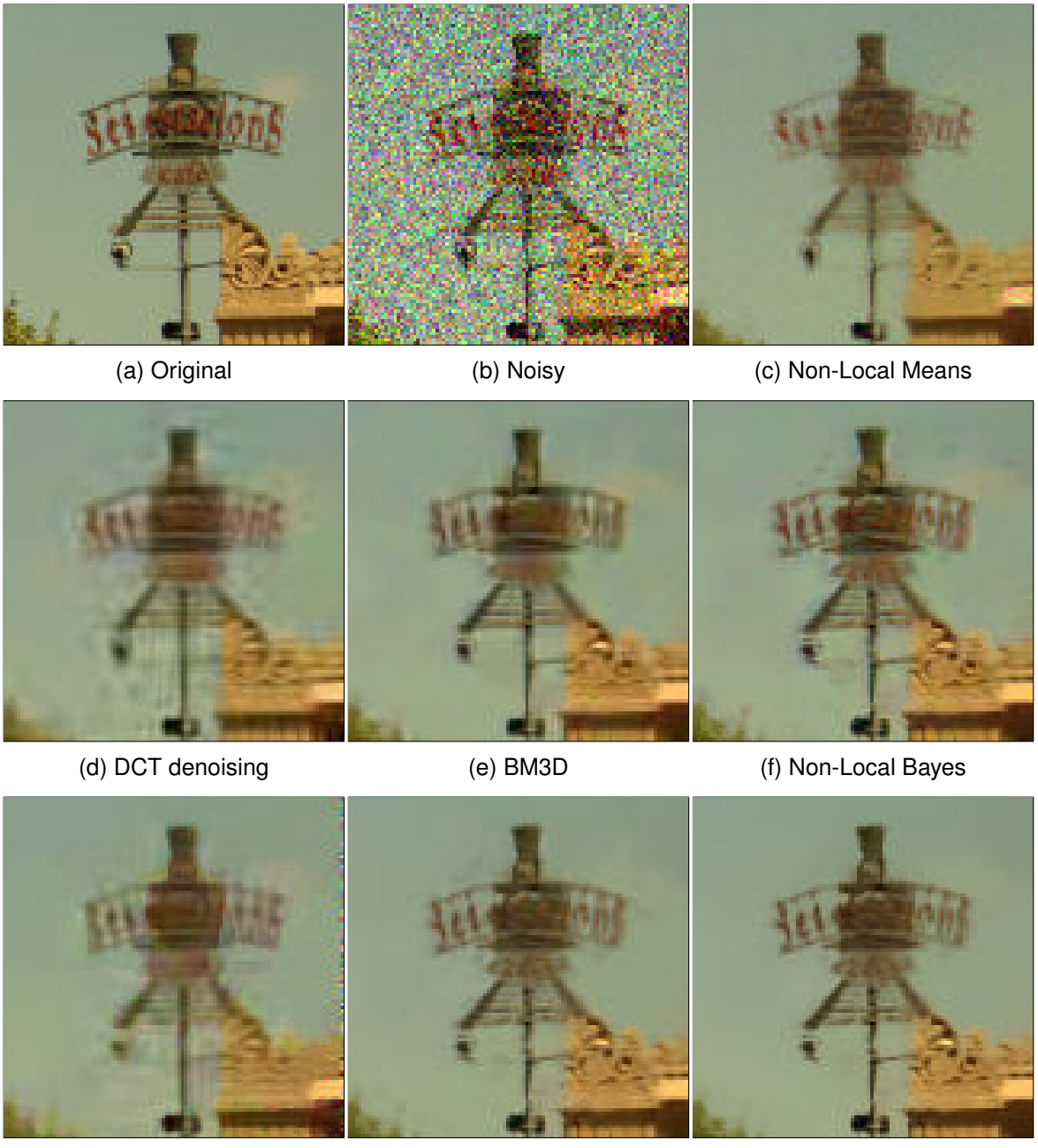


(d) DCT denoising (e) BM3D (f) Non-Local Bayes



(g) DCT on texture (h) Non-Local Bayes 2 (i) Non-Local Bayes 2 + DA3D

Figure 1.3 – Comparison of different denoising principles. Details of this image are shown in Fig. 1.4 and Fig. 1.5. In Non-Local Means, the results on textures can be blurry, since the hypothesis of *perfect* self-similarity is often too strong. Frequency-based algorithms like DCT Denoising fail near edges, by showing strong ringing artifacts. This is due to the fact that edges are non-smooth, and patches containing them do not have a sparse representation in the DCT basis.

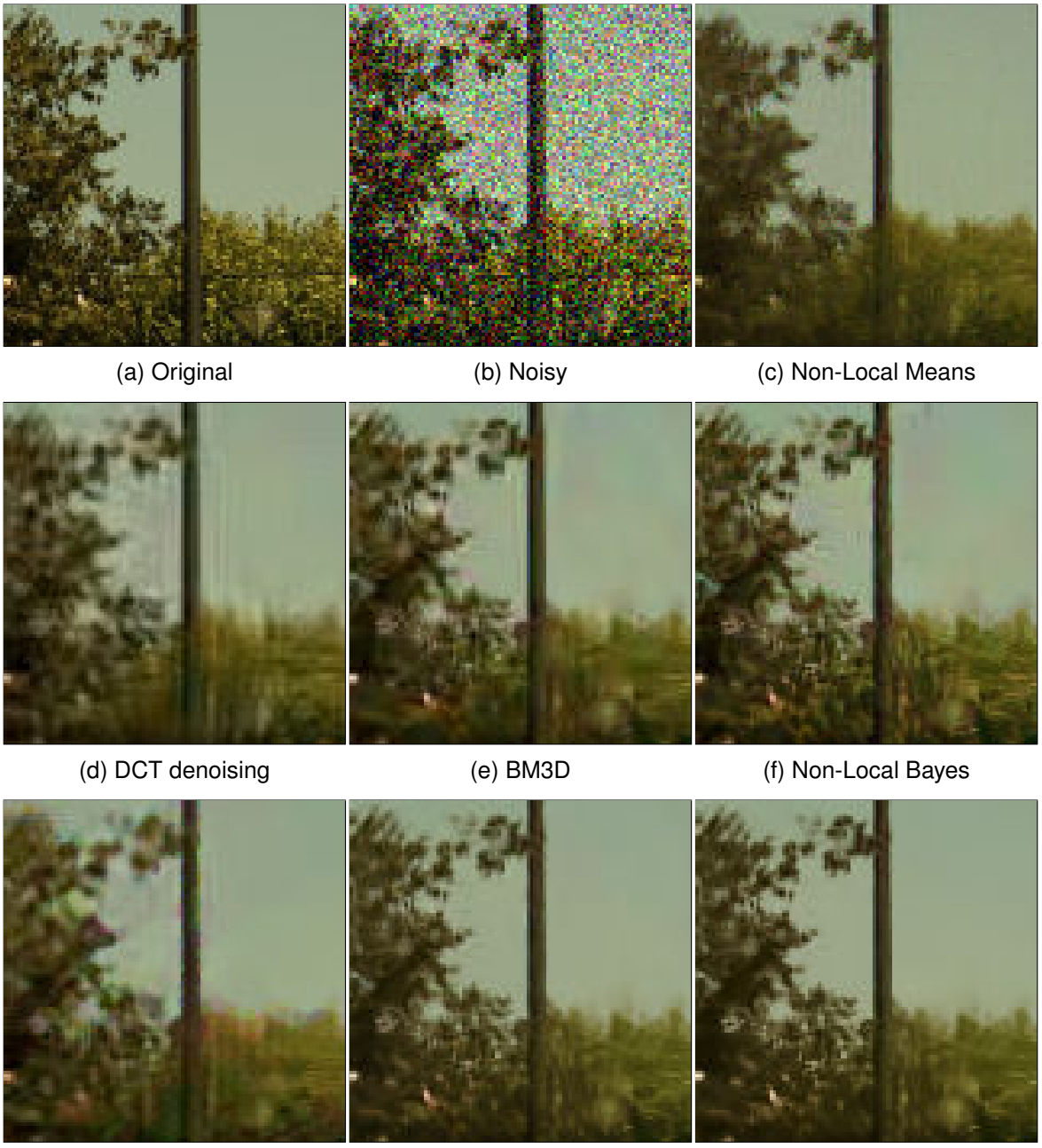


(a) Original (b) Noisy (c) Non-Local Means

(d) DCT denoising (e) BM3D (f) Non-Local Bayes

(g) DCT on texture (h) Non-Local Bayes 2 (i) Non-Local Bayes 2 + DA3D

Figure 1.4 – Comparison of different denoising principles and their best representative algorithms. Detail of borders and high-contrast textures. Notice that in DA3D removes all the artifacts from Non-Local Bayes, and at the same time it improve the contrast in textures.

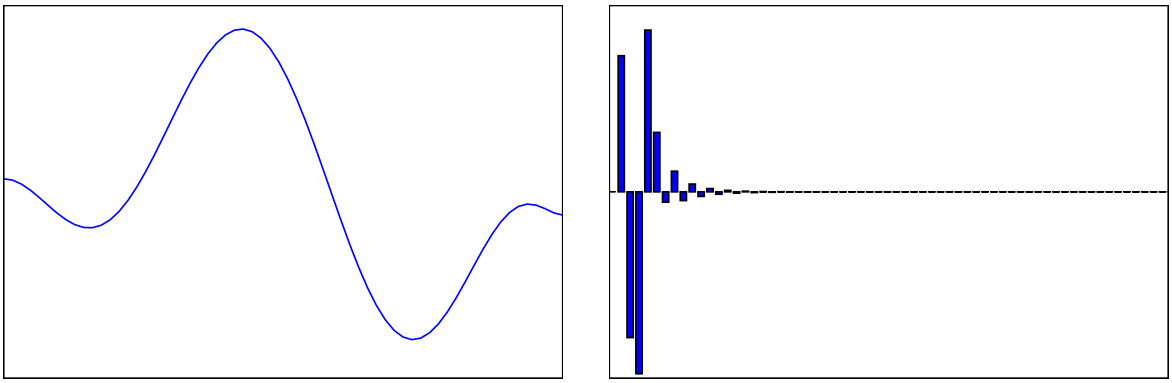


(a) Original (b) Noisy (c) Non-Local Means

(d) DCT denoising (e) BM3D (f) Non-Local Bayes

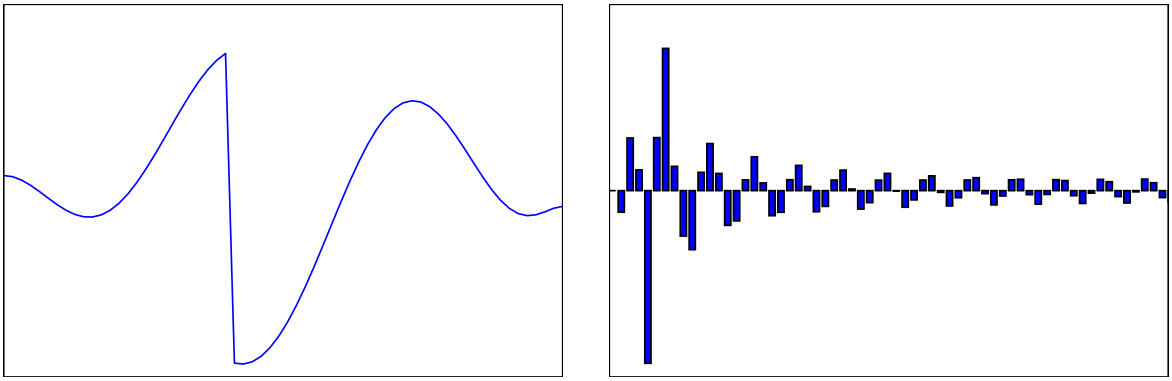
(g) DCT on texture (h) Non-Local Bayes 2 (i) Non-Local Bayes 2 + DA3D

Figure 1.5 – Comparison of different denoising principles and algorithms. Detail of smooth and textured areas.



(a) Smooth signal

(b) DCT of smooth signal



(c) Signal with discontinuity (edge)

(d) DCT of signal with discontinuity

Figure 1.6 – Behaviour of DCT coefficients. Signals containing discontinuities have their energy less concentrated in the DCT domain. This makes the DCT basis less effective for denoising purposes in presence of edges.

them [18, 34, 51]. Since white noise is unaffected by an orthogonal change of basis, in the transform domain the signal regarding the image is concentrated in few frequencies, while the noise itself remains evenly distributed, and therefore can be easily removed (for example, by a simple thresholding or with a Wiener filter [60]). A simple algorithm that uses this model is DCT denoising [63], that uses the DCT basis and a Wiener filter on the coefficients. The DCT basis has the property that smooth signals have most of the energy concentrated on the low frequencies. The problem with frequency-based denoising is that, sometimes, the patches do not follow the model that works well with the basis. For DCT denoising, in Fig. 1.3(d), the failures of the algorithm are evident near the edges. This follows from the fact that edges are non-smooth parts of the image, and patches containing them have their energy less concentrated in the DCT domain (as shown in Fig. 1.6). Denoising by DCT thresholding therefore reverberates around the edges, creating oscillations known as *Gibbs effect*. The class of frequency based algorithms is not limited to the DFT or DCT basis. Similar Gibbs effects are observed with algorithms that perform filtering in other wavelet domains [11].

Modern algorithms try to use both self-similarity and frequency priors, to achieve better results. Among them, the most famous is BM3D [7], that denoises blocks of similar patches in the DCT domain. Since the patches are denoised together, this effectively allows to take into account their similarity. BM3D is nowadays considered as one of the best denoising algorithms, and it has very good results in terms of PSNR. Its main drawback is that, since it uses the DCT basis for the thresholding operation, it can still present Gibbs artefacts around the edges (as in Fig. 1.3(e)).

An evolution of BM3D that uses an adaptive Bayesian model instead of the fixed DCT basis used in BM3D is Non-Local Bayes. This algorithm presents fewer artifacts than BM3D, especially in smooth areas, and has a similar performance in terms of PSNR. Compared to BM3D, Non-Local Bayes loses some contrast in textures.

A simple attempt to "correct" the issues of frequency-based denoising (namely, the Gibbs effect), is to apply the denoising on a *regularized* version of the image, where the edges have been removed. Fig 1.3(g) shows this attempt, where DCT denoising has been applied on the Texture of the Cartoon+Texture decomposition presented in [25]. The results shows significantly fewer ringing artifacts, but the overall quality of the image is still *sub par*.

DA3D applied tho the result of a patch-based denoising algorithm such as Non-Local Bayes gives the best result in terms of both PSNR and visual quality. Notice in particular the results near borders and textures (Fig. 1.4 and 1.5).

1.4 Interpreting the performance of a Dual Domain Image Denoising step

A DDID step is a single iteration of the DDID algorithm [22], but it can also be used as a last denoising step for other methods [21, 42]. This section, along with the pseudocode in Table 1, summarizes it.

This interpretation differs from the one originally proposed by Knaus and Zwicker [22], but it leads to the same exact algorithm. The original interpretation of DDID splits the image into a low- and a high-contrast layer, which are treated respectively with a spatial

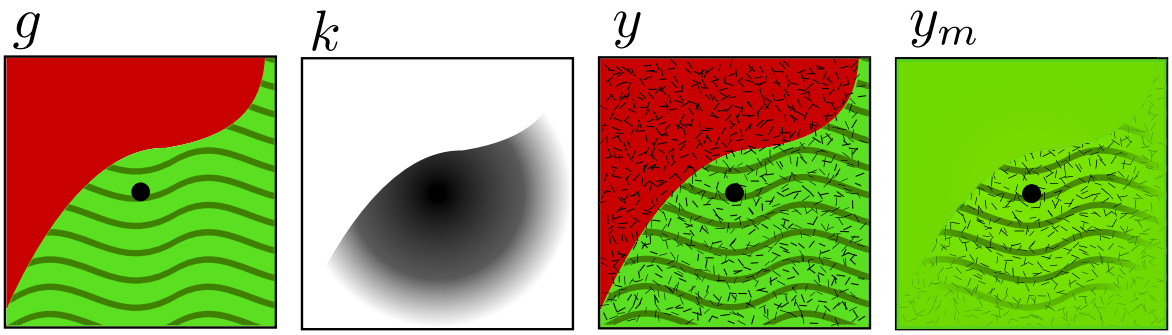


Figure 1.7 – Illustration of DDID's preprocessing of a patch. The kernel k is computed using the guide g. In the modified patch y_m all object discontinuities have been removed, leaving only the texture information corresponding to the object selected by the kernel k. The removed pixels are replaced by the average of the meaningful portion of the patch.

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.

To denoise a pixel p from the noisy image $\mathbf{y}$ the DDID step extracts a 31×31 pixel block around it (denoted y) and the corresponding block g from the guide image $\mathbf{g}$.

The blocks are processed to eliminate discontinuities that may cause artifacts in the subsequent frequency-domain denoising. To that end, the weight function k is derived from g. The weights identify the pixels of the block belonging to the same object as the center p. This weight function has the form of the bilateral filter [56, 61]

$$k(q) = \exp\left(-\frac{|g(q) - g(p)|^2}{\gamma_r \sigma^2}\right) \exp\left(-\frac{|q-p|^2}{2\sigma_s^2}\right). \tag{1.2}$$

The first term in this function is the *range kernel*, and it is used to identify the pixels belonging to the same structure as p, by selecting the ones with a similar color in the guide. The parameter γ_r is chosen empirically in [22] and it is equal to 0.7, and σ is the standard deviation of the noise. The way (1.2) is constructed means that, when there is less noise, a much more *uniform*, albeit smaller, area is taken into account.

The second term of (1.2) is the *spatial kernel*, and it removes the periodization discontinuities associated with the Fourier transform. This term does not depend on the noise, but it is roughly related to the size of the block and in [22] it is set to 7.0.

As said in Section 1.3, denoising by filtering Fourier coefficients presents problems in presence of edges (due to the Gibbs phenomenon). To avoid that, the blocks are made as *regular* as possible, by removing the parts that are not selected by k (and therefore not relevant to the denoising of the central pixel). First, the average of the "relevant" part of both the noisy and the guide blocks is computed:

$$\tilde{s} = \frac{\sum k(q)y(q)}{\sum k(q)}, \qquad \tilde{g} = \frac{\sum k(q)g(q)}{\sum k(q)}, \tag{1.3}$$

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

$$y_m(q) = k(q)y(q) + (1 - k(q))\tilde{s}. \tag{1.4}$$

As illustrated in Fig. 1.7 the block 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 block extracted from the guide

$$g_m(q) = k(q)g(q) + (1 - k(q))\tilde{g}. \tag{1.5}$$

In this way the "relevant" part of the blocks (similar to the central pixel) is retained by y_m and g_m, and their average value is assigned to the rest.

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

$$G(f) = \sum_{q \in \mathcal{N}_p} \exp\left(-\frac{2i\pi(q-p)f}{d}\right) g_m(q), \tag{1.6}$$

$$S(f) = \sum_{q \in \mathcal{N}_p} \exp\left(-\frac{2i\pi(q-p)f}{d}\right) y_m(q). \tag{1.7}$$

Assuming that y contains an additive white Gaussian noise of variance σ^2, the amount of noise present in y_m only 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

$$\sigma_f^2 = \sigma^2 \sum_{q \in \mathcal{N}_p} k(q)^2. \tag{1.8}$$

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

$$K(f) = \begin{cases} 1 & \text{if } f = 0, \\ \exp\left(-\frac{\gamma_f \sigma_f^2}{|G(f)|^2}\right) & \text{otherwise,} \end{cases} \tag{1.9}$$

where γ_f is a parameter of the algorithm, and it is determined in [22] to be 0.8. Section 1.9 deals with the problem of shrinkage in more detail. The denoised value of the central pixel is finally recovered by reversing the Fourier transform. Inverting equation (1.4) is unnecessary, since $k(p) = 1$.

Equations (1.6-1.10) are slightly different from the ones presented in [22]. 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 $K(0) = 1$. Since discontinuities have been removed from the blocks, filtering in the Fourier domain doesn't introduce ringing, which is a major advance made by DDID in transform thresholding methods.

Finally, the denoised value of the central pixel is recovered by reversing the Fourier transform. Since the inverse Fourier transform evaluated in the center of the patch is the average of the frequencies, the central pixel's value can be computed as

$$x(p) = \frac{1}{d^2} \sum_f S(f)K(f). \tag{1.10}$$

Algorithm 1 Pseudo-code for DDID step, Sparse DDID and DA3D. The lines used only in the DDID step are highlighted in blue. Bullets show the operations of each algorithm. Variables in **bold** denote whole images, while *italics* denote single blocks. Multiplication and division are pixel-wise.

DDID	Sparse	DA3D		
				Input: **y** (noisy image), **g** (guide) **Output:** denoised image
-	•	•	1	$\mathbf{w} \leftarrow 0$
•	•	•	2	$\mathbf{out} \leftarrow 0$
•	-	-	3	**for all** pixels $p \in \mathbf{y}$ **do**
-	•	•	4	**while** $\min(\mathbf{w}) < \tau$ **do**
-	•	•	5	$p \leftarrow \arg\min(\mathbf{w})$
•	•	•	6	$y \leftarrow \text{EXTRACTPATCH}(\mathbf{y}, p)$
•	•	•	7	$g \leftarrow \text{EXTRACTPATCH}(\mathbf{g}, p)$
				// regression weight, eq. 1.12
-	-	•	8	$K_{reg} \leftarrow \text{COMPUTEKREG}(g)$
				// regression plane, eq. 1.11
-	-	•	9	$P \leftarrow \arg\min_P \sum [y(q) - P(q)]^2 \cdot K_{reg}(q)$
-	-	•	10	$y \leftarrow y - P$ // subtract plane from the block
-	-	•	11	$g \leftarrow g - P$ // and from guide
•	•	•	12	$k \leftarrow \text{COMPUTEK}(g)$ // eq. 1.2
•	•	•	13	$y_m \leftarrow k \cdot y + (1-k) \cdot \left(\frac{\sum k(l)y(l)}{\sum k(l)}\right)$
•	•	•	14	$g_m \leftarrow k \cdot g + (1-k) \cdot \left(\frac{\sum k(l)g(l)}{\sum k(l)}\right)$
•	•	•	15	$Y \leftarrow \text{DFT}(y_m)$
•	•	•	16	$G \leftarrow \text{DFT}(g_m)$
•	•	•	17	$\sigma_f^2 \leftarrow \sigma^2 \sum k(q)^2$
•	•	•	18	$K \leftarrow \text{COMPUTESHRINKAGEFACTOR}\left(\frac{\lvert G(f)\rvert^2}{\sigma_f^2}\right)$ // shrinkage
•	•	•	19	$x_m \leftarrow \text{IDFT}(K \cdot Y)$
				// revert line 13
-	•	•	20	$x \leftarrow \left[x_m - (1-k)\left(\frac{\sum k(l)y(l)}{\sum k(l)}\right)\right] / k$
-	-	•	21	$x \leftarrow x + P$ // add plane back to the block
-	•	•	22	$aggw \leftarrow k \cdot k$ // aggregation weight
•	-	-	23	$\mathbf{out}(p) \leftarrow \text{EXTRACTCENTRALPIXEL}(x_m)$
				// accumulate patch in the correct position
-	•	•	24	$\mathbf{w} \leftarrow \text{ADDPATCHAT}(p, \mathbf{w}, aggw)$
-	•	•	25	$\mathbf{out} \leftarrow \text{ADDPATCHAT}(p, \mathbf{out}, aggw \cdot x)$
•	-	-	26	**return out**
-	•	•	27	**return out**/**w**

This process is repeated for every pixel of the image.

For color images, k is computed by using the Euclidean distance, while the shrinkage is done independently on each channel of the YUV color space. For more details about DDID refer to [22, 42]. An example of the result is shown in Fig. 1.2(d).

1.5 Sparse DDID step

The DDID step explained in section 1.4 is slow in practice because it has to process a block for every pixel. In fact, each pixel is denoised several times, but the result is discarded every time that it is not in the center of the current block. Since the denoising remains valid for all pixels in the "relevant" part of the block, we propose to aggregate the processed blocks to form the final result. As a result only a small number of blocks needs to be processed, thus accelerating the algorithm considerably. Note that since the processing is done with the modified block, line 13 of Table 1 must be reverted to obtain the denoised block (line 20).

In our selection-aggregation process, the image is treated by color-coherent blocks and the results are aggregated with weights deduced from the guide image. This weighted average can also be seen as the interpolation of the denoised image from a subset of processed blocks [1, 12, 16]. We found that the best aggregation weights are the squares of the weights (1.2).

We now describe the greedy approach used for selecting the image blocks to be processed. At each iteration a weight map w with the sum of the aggregation weights is updated. This weight map permits to identify the pixel in the image with the lowest aggregation weight, which will be selected as the center of the next block to process (line 5 of Table 1). This process iterates until the total weight for each pixel becomes larger than a threshold τ. The weight function k is always equal to 1 in the center, so the algorithm always terminates. The procedure is detailed in Table 1. This variant is faster to execute than a single DDID step, since only a small number of blocks are actually processed. This allows bigger patches to be used, that in turn gives better results in terms of denoising quality. Experimentally, good results are achieved with patches as large as 64×64, which is to be contrasted to patch based methods using mostly 8×8 patches. An example of the result is shown in Fig. 1.2(e). The total number of processed blocks depends on the image complexity. The centers of the effectively processed blocks are shown in Fig. 1.2(g). They concentrate on edges and details.

1.6 Data Adaptive Dual Domain Denoising

We now address a main drawback of the weight function (1.2), used for the bilateral filter and for many bilateral-inspired filters, including patch based methods. This weight function selects pixels of the block with a similar value. As a result, Sparse DDID works by processing parts of the image that are piecewise constant, considering the image as composed by many "flat" layers. This model is not well adapted for images that contain gradients or shadings, as the same smooth region may be split in many thin regions. The previous method can be extended to "normalize" each patch by subtracting an estimation of the gradient around the patch center. In practice, this means estimating an affine model

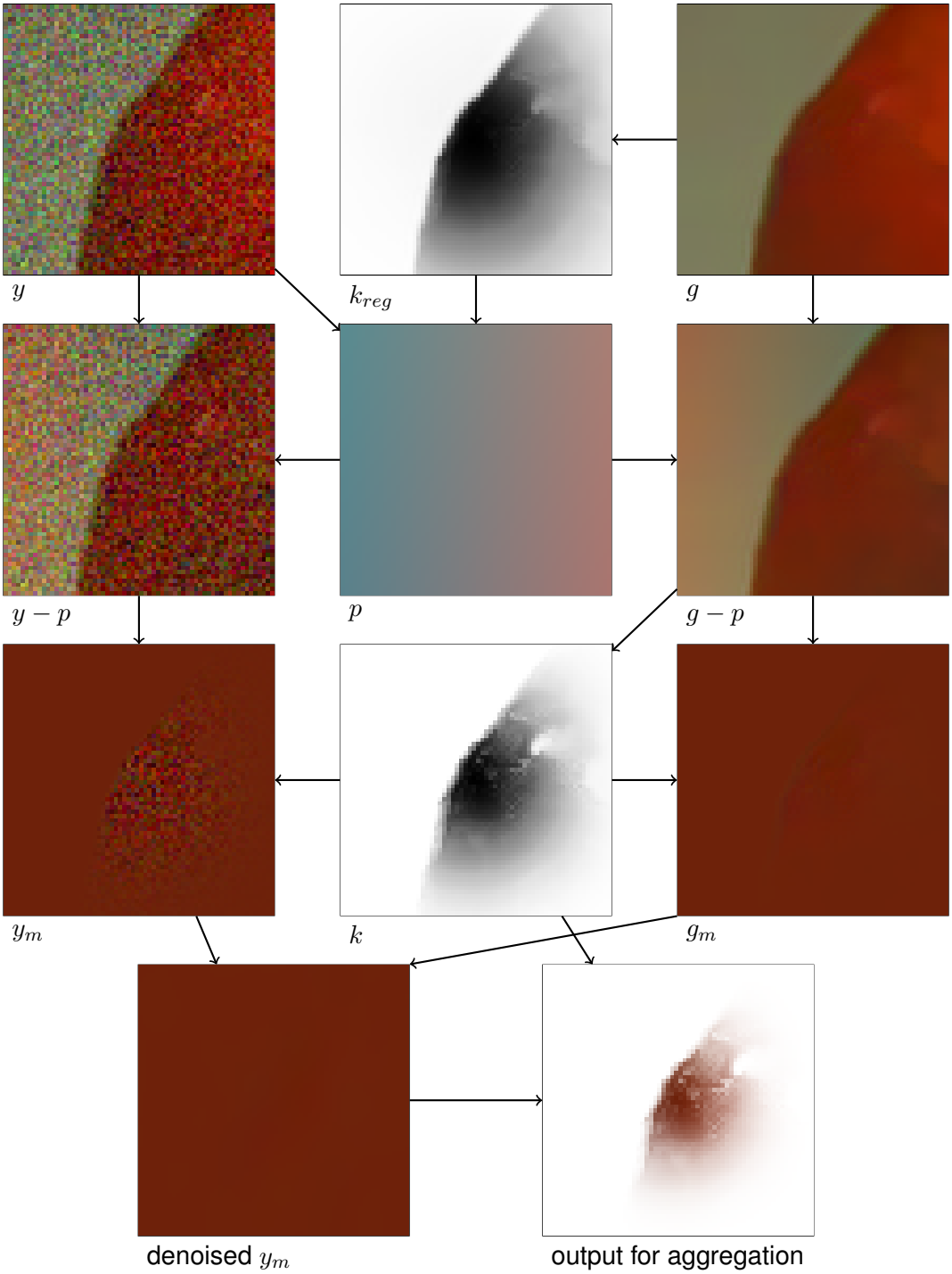


Figure 1.8 – Steps of the DA3D algorithm. This figure shows what happens to a noisy patch taken in a natural image, containing an edge. The arrows indicate the elements needed to compute every step of the algorithm. Notice that, thanks to the weight function, the useful part of the patch is kept, while the discontinuities are completely removed. See Fig. 1.9 for the steps on the white part of the patch and Fig. 1.10 for the steps on texture.

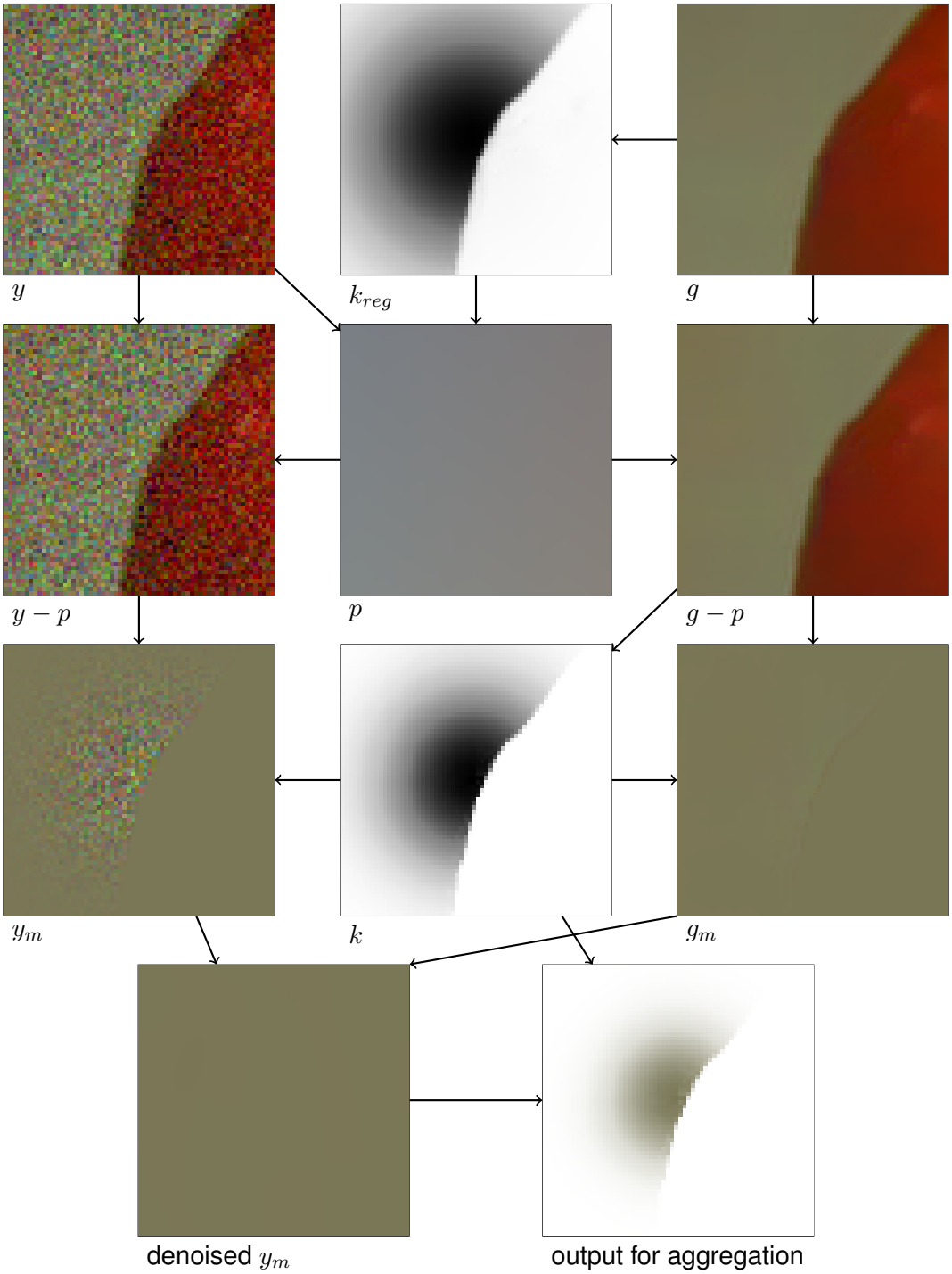


Figure 1.9 – Steps of the DA3D algorithm. This figure shows what happens to a noisy patch taken in a natural image, containing an edge. The arrows indicate the elements needed to compute every step of the algorithm. Notice that, thanks to the weight function, the useful part of the patch is kept, while the discontinuities are completely removed. See Fig. 1.8 for the steps on the white part of the patch and Fig. 1.10 for the steps on texture.

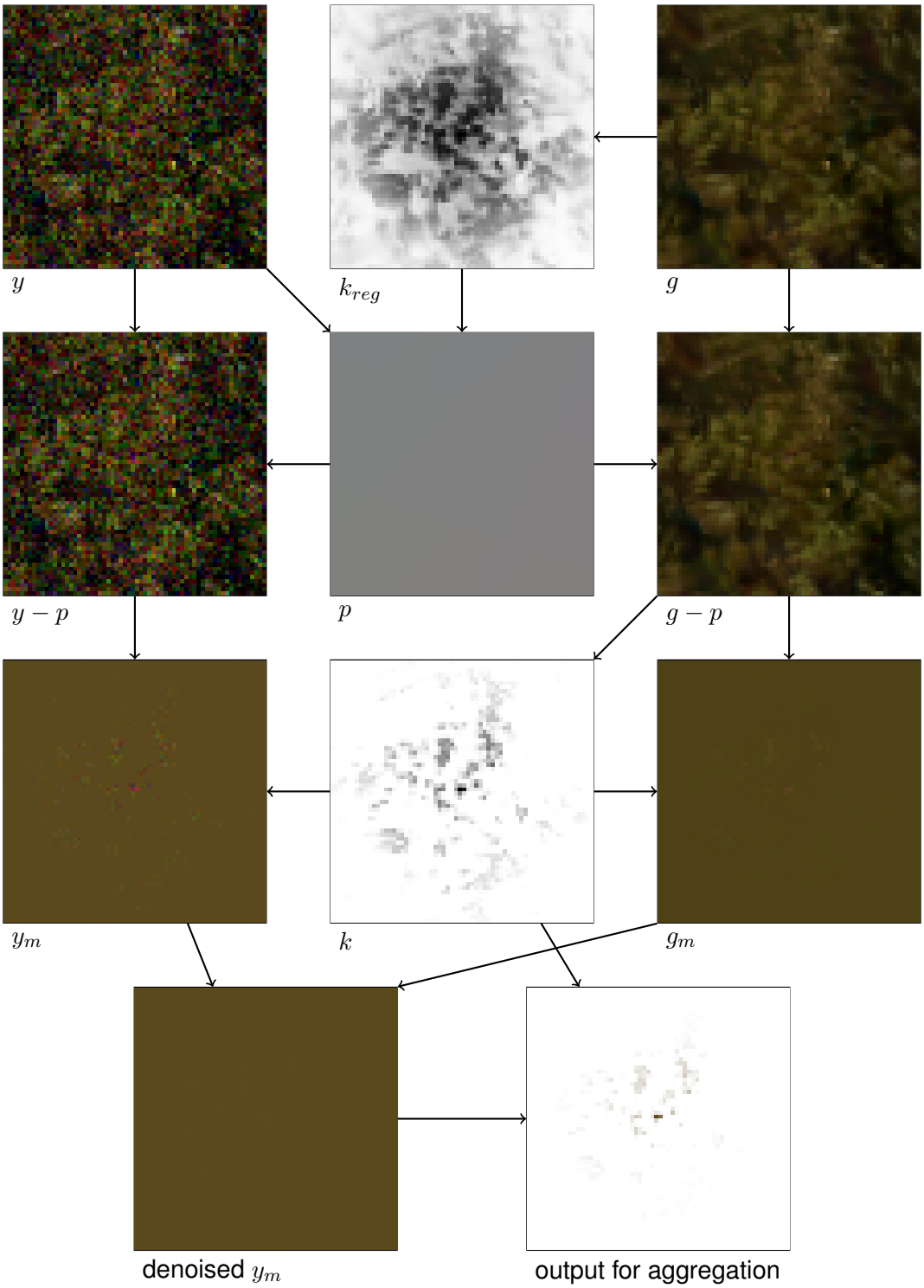


Figure 1.10 – Steps of the DA3D algorithm, for a patch containing only a texture. The arrows indicate the elements needed to compute every step of the algorithm. Notice that, thanks to the weight function, the useful part of the patch is kept, while the discontinuities are completely removed. See Fig. 1.8 and Fig. 1.9 for the steps near an edge.

of the block, as proposed in [54], which can be computed using a weighted least squares regression

$$\min_P \sum [y(q) - P(q)]^2 \cdot K_{reg}(q), \tag{1.11}$$

where the sum is computed over the domain of y and K_{reg} is a bilateral weight function

$$K_{reg}(q) = \exp\left(-\frac{|g(q) - g(p)|^2}{\gamma_{rr}\sigma^2} - \frac{|q-p|^2}{2\sigma_{sr}^2} \right), \tag{1.12}$$

which selects the parts of the block that gets approximated by P. To ensure that the central pixel gets denoised, the constraint $P(p) = g(p)$ is also added. Since the weights K_{reg} should capture the overall shape of the block, they are computed using a larger range parameter than the bilateral weight function in (1.2). It is worth noting that although K_{reg} uses the guide to select the parts of the block in which to perform the estimation, the regression is performed directly on the noisy data y, thus allowing to correct any staircasing effect already present in the guide. The parameters σ_{sr} and γ_{rr} are specific of the algorithm. The case of color images is identical, since (1.11) is separable and can be computed independently on every channel.

Once estimated, the local plane P is subtracted from the patch, effectively removing shades and gradients. Then the standard DDID step is used to denoise the block and at the end the plane is added back. The whole procedure is detailed in Table 1 (lines 8-11, 21). An real example of the algorithm run on a patch is shown in Fig. 1.8-1.9-1.10.

An example of the result is shown in Fig. 1.2(f). Observe that the result presents fewer staircasing effect and has a larger PSNR. In addition, fewer blocks are treated to denoise the gradients (Fig. 1.2(h)).

1.7 Experiments

Our implementation of the DA3D algorithm (available in the support website [44]) has been tested against the set of images shown in Fig. 1.11. For γ_r and γ_f, the parameters of DDID [22] were kept ($\gamma_r = 0.7$, $\gamma_f = 0.8$), but since DA3D does not need to process all patches, the size of the patches themselves was chosen as 64×64, with $\sigma_s = 14$. The parameters $\tau = 2$, $\sigma_{sr} = 20$ and $\gamma_{rr} = 7$ were chosen experimentally on images outside the test database.

The DA3D method was applied to the results of several state-of-the-art algorithms. Each method was tested with noises of $\sigma = 5, 10, 25, 40, 80$. The results are summarized in Table 1.1.

DA3D improves the PSNR of every algorithm except NLDD and PID (which is to be expected, since they are based on a similar shrinkage strategy). This improvement is more marked with higher noises, which makes sense since the parameters of DDID were optimized for medium to high noise. It is worth mentioning that DA3D is even able to improve over BM3D-SAPCA, which is considered the best denoising algorithm up to date for grayscale images. Similar results are obtained using SSIM [58] as metric.

In general BM3D+DA3D (DA3D using BM3D as guide) offers one of the best performances with a reasonable computational cost. As an example the results of the best two performing methods for the image "Montage" are shown in Fig. 1.12, along with the result

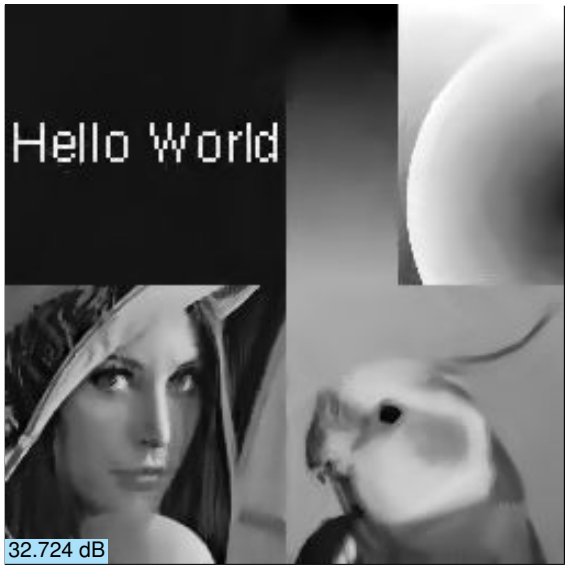
Figure 1.11 – Test images used in the experiments. No parameter learning or fitting was performed on this database. The image were chosen to include different features normally found in natural images, i.e. smooth areas, textures and geometric elements.

Method	$\sigma = 5$	$\sigma = 10$	$\sigma = 25$	$\sigma = 40$	$\sigma = 80$
BM3D	**38.43**	34.94	30.58	28.27	24.69
-	38.39	**34.95**	**30.67**	**28.41**	**25.09**
SAPCA [8]	-0.04	+0.01	+0.10	+0.14	+0.39
BM3D [7]	38.24	34.70	30.37	27.99	24.94
	38.24	34.78	30.54	28.23	25.03
	+0.00	+0.08	+0.16	+0.24	+0.09
DDID [22]	37.95	34.55	30.34	28.05	24.68
	37.90	34.52	30.39	28.16	24.84
	-0.04	-0.03	+0.05	+0.11	+0.16
EPLL [67]	37.91	34.29	29.90	27.64	24.46
	37.92	34.39	30.21	28.04	24.80
	+0.01	+0.10	+0.31	+0.40	+0.34
G-NLM [55]	35.07	33.54	29.19	26.87	23.45
	35.67	33.97	29.77	27.49	24.10
	+0.60	+0.43	+0.58	+0.63	+0.65
LSSC [35]	38.29	34.75	30.35	28.07	24.78
	38.34	34.88	30.61	28.32	24.97
	+0.05	+0.13	+0.27	+0.25	+0.19
MLP		34.63	30.44		
+		34.78	30.61		
BM3D [5]		+0.15	+0.17		
NLB [27]	38.19	34.62	30.13	27.86	24.45
	38.20	34.72	30.38	28.14	24.78
	+0.02	+0.10	+0.25	+0.28	+0.33
NLDD [42]	38.12	34.62	30.30	28.11	24.83
	38.09	34.60	30.29	28.09	24.77
	-0.04	-0.02	-0.01	-0.02	-0.05
NLM [2]	37.31	33.58	28.97	26.50	22.72
	37.49	33.98	29.66	27.45	24.17
	+0.18	+0.40	+0.69	+0.95	+1.44
PID [23]	37.97	34.56	30.38	28.18	24.99
	37.90	34.48	30.28	28.08	24.81
	-0.07	-0.08	-0.10	-0.11	-0.18
SAIST [10]	38.31	34.78	30.39	28.16	25.00
	38.31	34.82	30.53	28.27	25.07
	-0.01	+0.04	+0.14	+0.11	+0.07

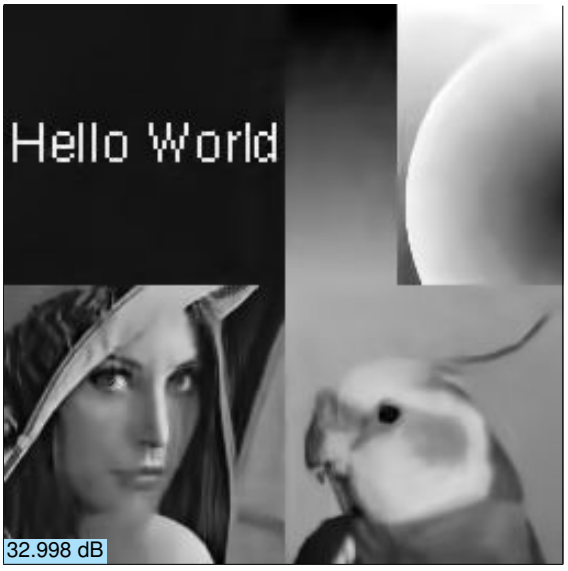
Table 1.1 – Average PSNR comparison between state-of-the-art methods on grayscale images. The first line of each row shows the average PSNR. The second line shows the average PSNR of DA3D using the corresponding algorithm to generate the guide. The third line shows the average improvement due to DA3D. The best result for each noise level is shown in **bold**, and the ones within a range of 0.2 dB are shown in **gray**. MLP+BM3D only works for some specific levels of noise, the other levels are left blank.

(a) Original

(b) SAPCA



(c) PID



(d) BM3D+DA3D

Figure 1.12 – Denoising results for *Montage*, $\sigma = 25$. Staircasing effects are present in (b) and (c), and they are significantly reduced in (d) by DA3D.

Method	$\sigma = 5$	$\sigma = 10$	$\sigma = 25$	$\sigma = 40$	$\sigma = 80$
BM3D [7]	39.55	36.01	31.79	29.23	**26.83**
	39.59	**36.23**	**32.15**	**29.95**	**27.00**
	+0.05	+0.22	+0.36	+0.72	+0.16
DDID [22]	39.20	35.91	31.88	29.75	26.41
	39.11	35.87	**31.99**	**29.98**	**26.85**
	-0.09	-0.04	+0.11	+0.23	+0.43
NLB [27]	**39.77**	**36.09**	31.71	29.35	26.64
	39.67	**36.26**	**32.18**	**30.09**	26.74
	-0.10	+0.16	+0.47	+0.74	+0.11
NLDD [42]	39.26	35.90	31.98	29.90	26.60
	39.17	35.87	**31.99**	**29.97**	26.63
	-0.09	-0.04	+0.02	+0.07	+0.03
PID [23]	39.03	35.81	**32.05**	**30.10**	**27.00**
	38.99	35.80	**32.03**	**30.02**	**26.83**
	-0.04	-0.02	-0.02	-0.08	-0.17

Table 1.2 – Average PSNR comparison between state-of-the-art methods for color images. The first line of each row shows the average PSNR obtained by denoising the test images. The second line shows the average PSNR of DA3D using the corresponding denoising algorithm to generate the guide. The third line shows the average improvement due to DA3D. The best result for each noise level is shown in **bold**, and the ones within a range of 0.2 dB are shown in **gray**.

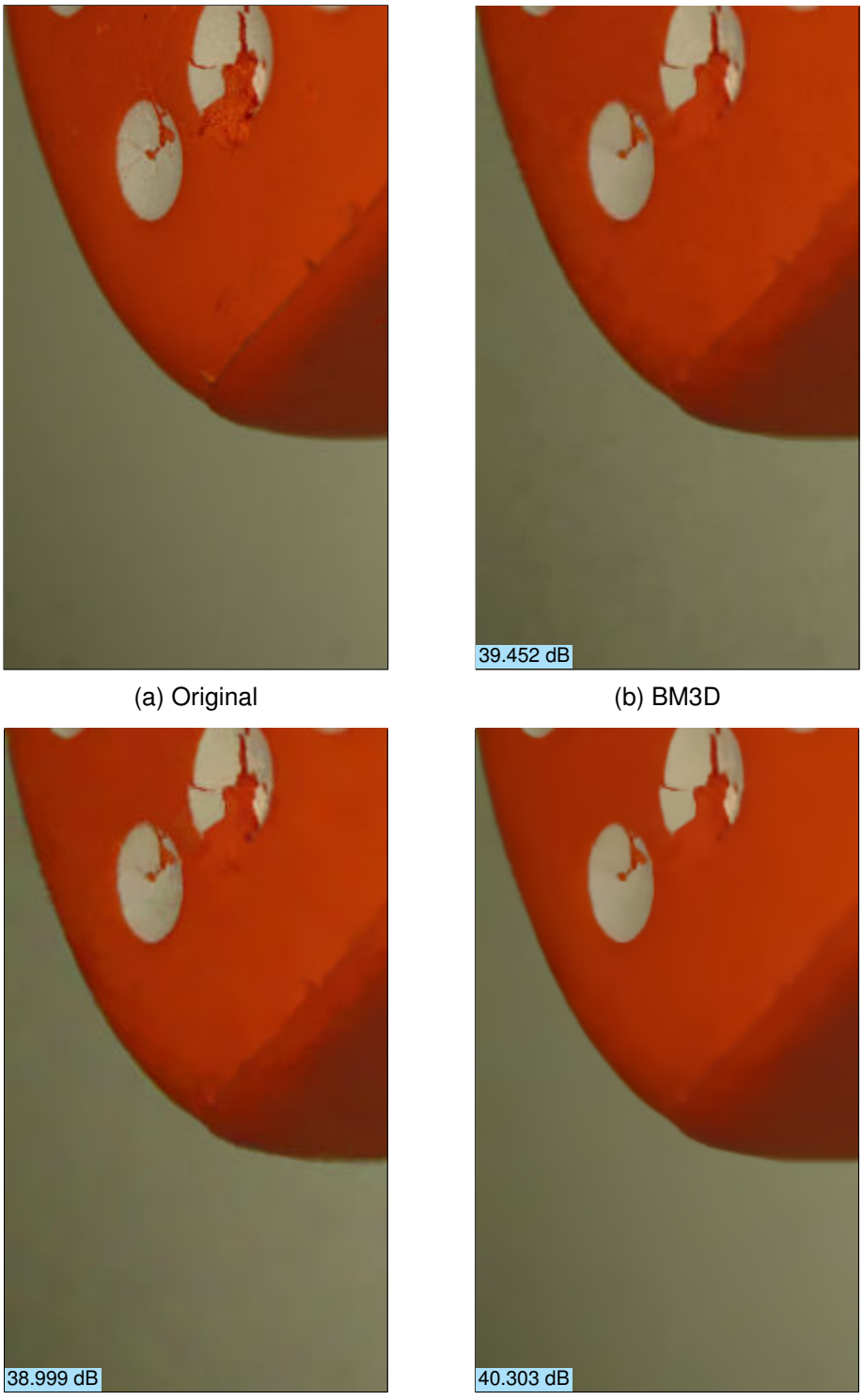
of BM3D+DA3D. The latter outperforms the other algorithms in terms of PSNR and image quality. Despite having a high PSNR value, the result of the other two algorithms present artifacts close to the edges and some staircasing (BM3D-SAPCA in particular).

Table 1.2 shows the results obtained for color images. In this case fewer algorithms have been tested since most methods only work for grayscale images. While for small noise levels DA3D does not improve the PSNR of the existing methods, for higher noise levels the gain is considerable. In the case of BM3D and NL-Bayes, this improvement is even larger than for grayscale images. As in the grayscale case, NLDD and PID are not improved (or just marginally improved) by DA3D. Results for SSIM are available in the supplementary materials [44].

Fig. 1.13 shows the two best denoising results for the image "Dice", along with the result of BM3D+DA3D. Most of the artifacts generated by BM3D disappear with the post-processing, and at the same time the edges becomes sharper and the gradients smoother.

1.7.1 Comparison with other last-step denoising methods

Figure 1.14 compares the result of three algorithms: DA3D, G-NLM [55] and the third iteration of DDID [22] (as done in NLDD [42]). The PSNR gain over Non-Local Means is shown for every algorithm and for every image in the test set. Non-Local Means has been chosen as a guide to make a fairer comparison, since it is the one used in G-NLM. It can be seen that DA3D achieves better results for all levels of noise. Furthermore, especially in case of low noise ($\sigma \leq 10$), DA3D consistently improves the result.



(a) Original (b) BM3D

(c) NL-Bayes (d) BM3D+DA3D

Figure 1.13 – Denoising results for *Dice*, $\sigma = 25$.

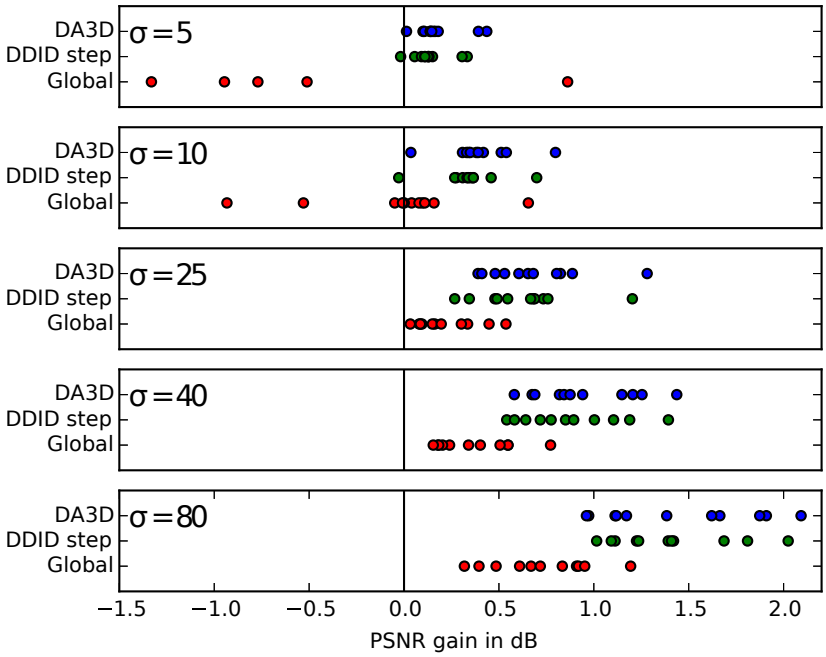


Figure 1.14 – PSNR gain for three different "last step" denoising methods. NLM was used to generate the guide. Each dot represents an image of the test set. Note that DA3D does not only improve the results on average, but it does it consistently.

1.8 Reducing the computational complexity

The algorithm, as it is described in the previous sections, can be slow if implemented naively. In particular, keeping track of the minimum weight (Algorithm 1, lines 4–5) can require scanning the whole image. In addition, the matter of parallel processing was not addressed. This section deals with those two problems.

1.8.1 Copying the guide

We observed that if the patch being processed by DA3D does not contain enough information, then the frequency processing does not improve the overall result. Therefore, when k (computed in Algorithm 1, line 12) has a total weight below a certain threshold η, the patch is not further processed and its guide value is used for the aggregation.

We found that a value of $\eta = 10$ allows a substantial speed-up, especially on heavily textured images, without significant changes to the result.

1.8.2 Tracking the minimum weight

In our first implementation, we selected the position with the lowest aggregation weight (as explained in Section 1.5) with a simple linear search. This approach shows its limits when the size of the image increases. Selecting a new block to process requires approximately $O(n)$ operations, with n the number of pixels of the image. Therefore, under the reasonable assumption that the number of processed blocks is a fraction of the total (from the original

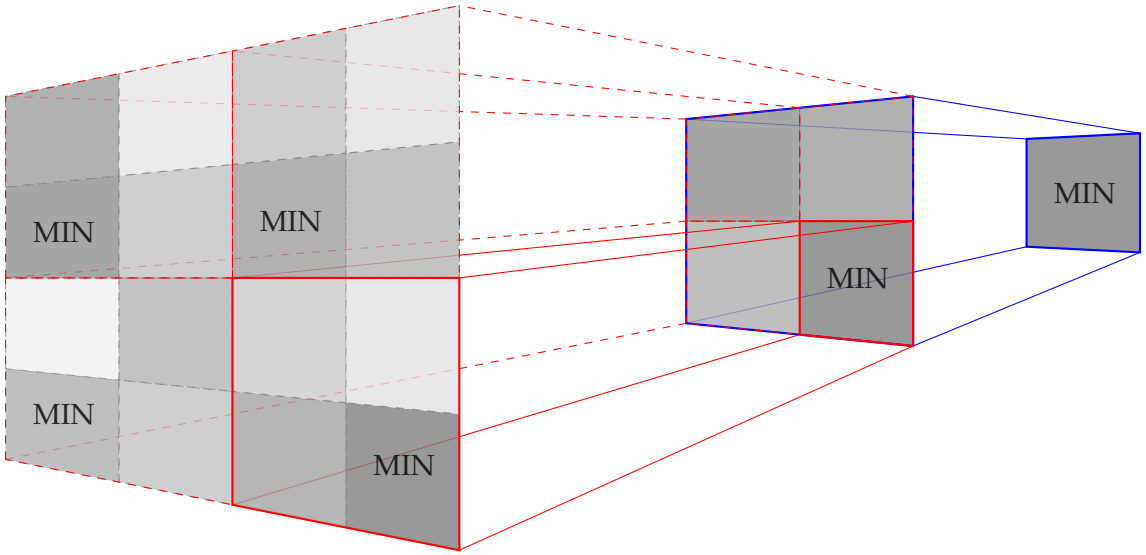


Figure 1.15 – Schema of the quad-tree used in DA3D to keep track of the minimum weight. Each node of the tree contains the minimum value of its four children. In order to retrieve the position of the minimum value, one has simply to traverse the tree from the root to the leaf, always choosing one of the children with the minimum value.

article, between 1% and 20%), and since the denoising of a block is performed in a bounded time, the complexity of the algorithm is $O(n^2)$.

We therefore propose to use a quad-tree to keep track of the minimum. The weight map $\mathbf{w}$ is in the leaves of the tree, and every node contains the minimum value of its four children. This can also be interpreted as a multi-scale version of $\mathbf{w}$, built using a *min* filter. The space complexity for this data structure is

$$\sum_{i=0}^{\log n} \frac{n}{4^i} \leq \frac{4}{3} n = O(n) \tag{1.13}$$

because every "layer" of the tree contains a fourth of the values of the previous one.

In order to retrieve the position of the minimum value, one has simply to traverse the tree from the root to the leaf, always choosing one of the children with the minimum value. This guarantees that the chosen pixel is a global minimum for $\mathbf{w}$, and has time complexity $O(\log n)$.

To update the tree, it suffices to update the appropriate leaves, and then recompute the minima in the upper nodes until the top. Since the aggregation is done one patch at a time, it is simple to calculate which nodes need to be updated, thus avoiding to recompute the values for areas in which $\mathbf{w}$ has not changed. The time complexity for this update is $O(k)$, where k is the number of pixels of the patch that is being aggregated. Since k is constant, the aggregation does not increase the complexity of the algorithm.

One could be tempted to update the values one by one. Although this could be simpler to implement, it is slower, having a time complexity of $O(k \log n)$. Using this data structure instead, the total complexity of the algorithm becomes $O(n \log n)$, which allows to denoise bigger images.

Size	Global	SAPCA	NLB	PID	BM3D	SAIST	MLP	EPLL	DDID	BM3D +DA3D
256×256	357 s	639 s	0.48 s	188 s	1.26 s	37.8 s	16.5 s	71.7 s	5.26 s	**1.52 s**
512×512	3359 s	2490 s	0.80 s	725 s	4.94 s	140 s	60.7 s	272 s	20.4 s	**5.59 s**

Table 1.3 – Average running time depending on image size between grayscale denoising methods. The experiments were performed on a 8-core 2.67GHz Xeon CPU. Every algorithm was tested using its official implementation. For DA3D the implementation available at [44] was used.

1.8.3 Parallel processing

Since DA3D selects the patches to denoise in a greedy fashion, it is impossible to know where the next patch will be prior to the aggregation step of the current one. This makes parallelization more complex than in other denoising algorithms.

In order to denoise a pixel p, the algorithm uses the other pixels inside a (64×64) window, all the pixels needed to denoise p are at a distance of at most 32. This makes the algorithm *local*, and allows to solve the problem of parallelism by just dividing the image in tiles. Each tile can be denoised separately, and then the results can be combined together.

It is clear that the patches chosen in this way will not correspond exactly to the patches chosen without parallelism. The main difference can be an over-sampling of the areas near the edges, since the weights from a neighboring tile are not taken into account. This could result in a slight overhead in the processing time. However, the experiments show that the overhead is negligible, and the results of the simple and parallel versions of the algorithm are identical from a practical standpoint. With bigger images, the overlap area becomes smaller, therefore the factor of acceleration becomes even closer to the number of processors.

1.8.4 Running time

The time needed to run the analyzed algorithms is summarized in Table 1.3. Using DA3D as a post-processing method demands little additional time, while the gain is substantial (in PSNR and in visual quality). For example, BM3D + DA3D turns out to take $1.52s$, while BM3D alone requires $1.26s$. Therefore, while BM3D+DA3D is comparable to BM3D-SAPCA in terms of performance, its computation is more than 200 time faster.

1.9 Learning the Shrinkage Curve

In DA3D the shrinkage is performed via a simple exponential dampening, the same that is used in DDID. Since DA3D is conceived to be applied on top of another algorithm, we can expect to have a better result if the shrinkage function is adapted to the guide. In order to compute a better shrinkage curve, we trained the algorithm on the dataset of Figure 1.16.

For each algorithm, and for each noise level (since most algorithms behave differently for different noise levels), the perfect curve to maximise the global PSNR was computed as a look-up table. Similarly to the equation in line 18, we searched a function dependent on $\frac{|G(f)|^2}{\sigma_f^2}$.

Figure 1.16 – Images used to train the shrinkage curve.

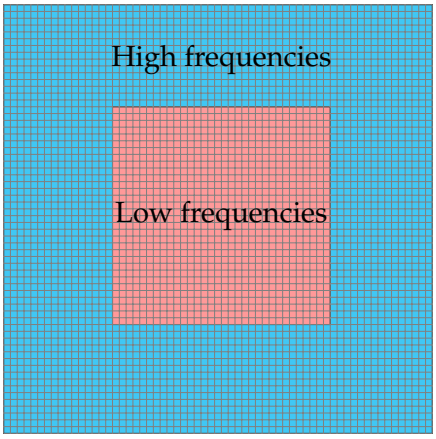


Figure 1.17 – The frequencies of the patch are divided in two sets.

Since most denoising algorithms have different performances depending on the frequency, another way to improve the result is to train different functions for high and low frequencies. Therefore, we decided to divide the frequencies in two groups, and to train the two shrinkage functions as look-up tables. In order to find the best function, we optimized its value on the entries of the tables one by one, and we repeated the process until convergence.

The resulting shrinkage curves for six different algorithms are shown in Figures 1.18–1.23, and the comparison between the original version of DA3D and the version with adapted curves is shown in Table 1.4.

From the results, two main observations can be made. First, for most algorithms, the "optimal" shrinkage curve resembles to a hard thresholding in the case of high frequencies, while the shape of the low frequency curve varies from algorithm to algorithm.

Second, and more important, the optimization of the shrinkage curve appears less important than expected. From Table 1.4, it looks like the improvement due to the learning process is not consistent. Moreover, sometimes the optimal curves are decreasing in some part. This should not happen in theory. An explanation for those two facts can be given. These results indicate that the effectiveness of DA3D lies more on the bilateral weight and on the sparse aggregation than on the shrinkage step itself. While conducting the experi-

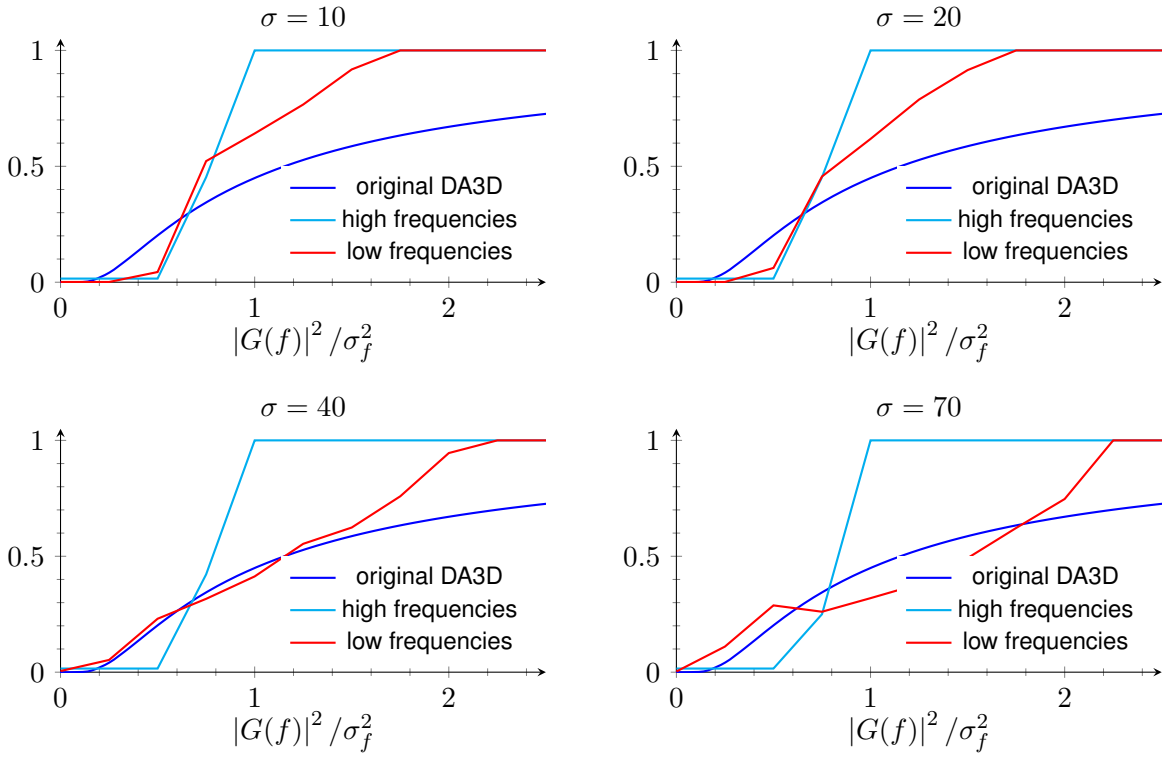


Figure 1.18 – Optimal shrinkage curves for BLS-GSM.

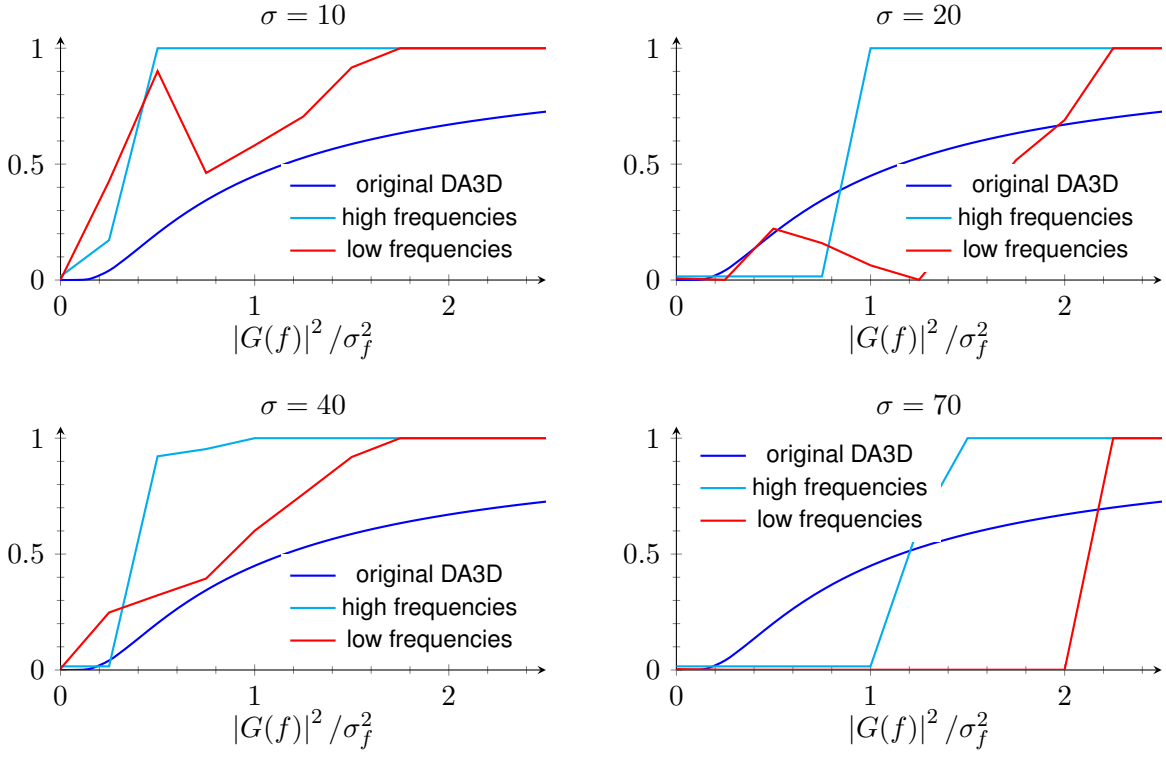


Figure 1.19 – Optimal shrinkage curves for DCT denoising.

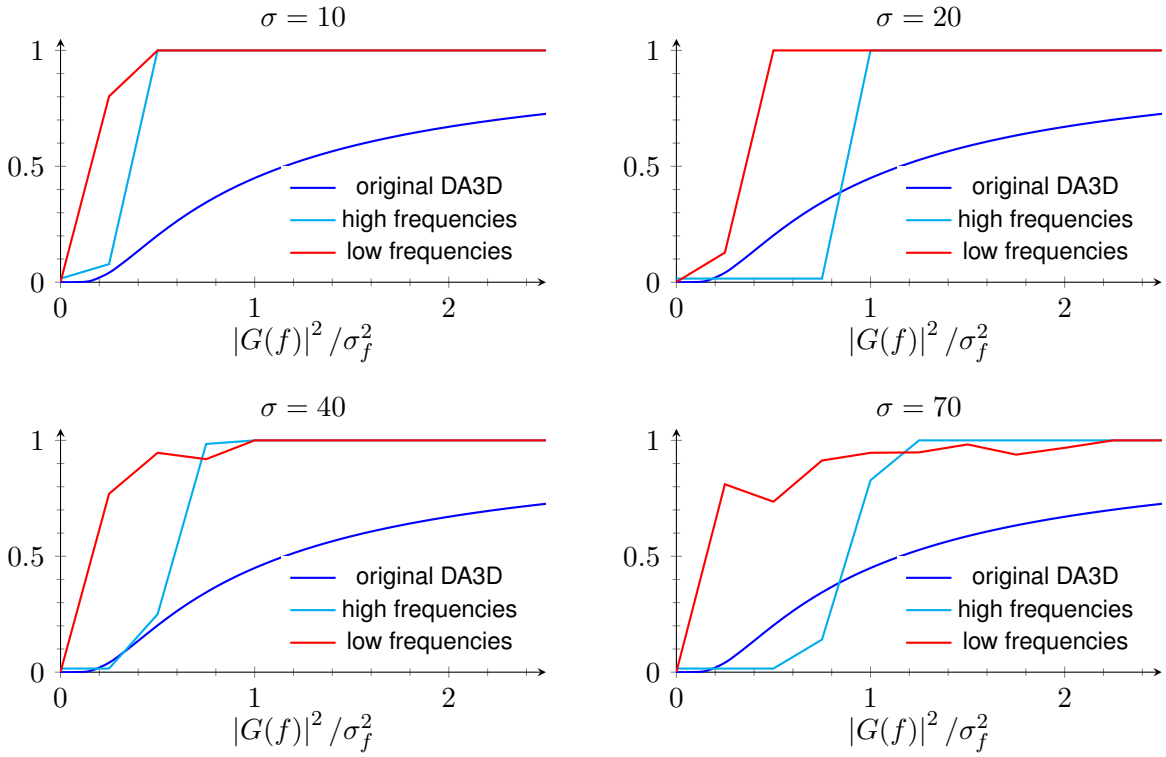


Figure 1.20 – Optimal shrinkage curves for Non-Local Means.

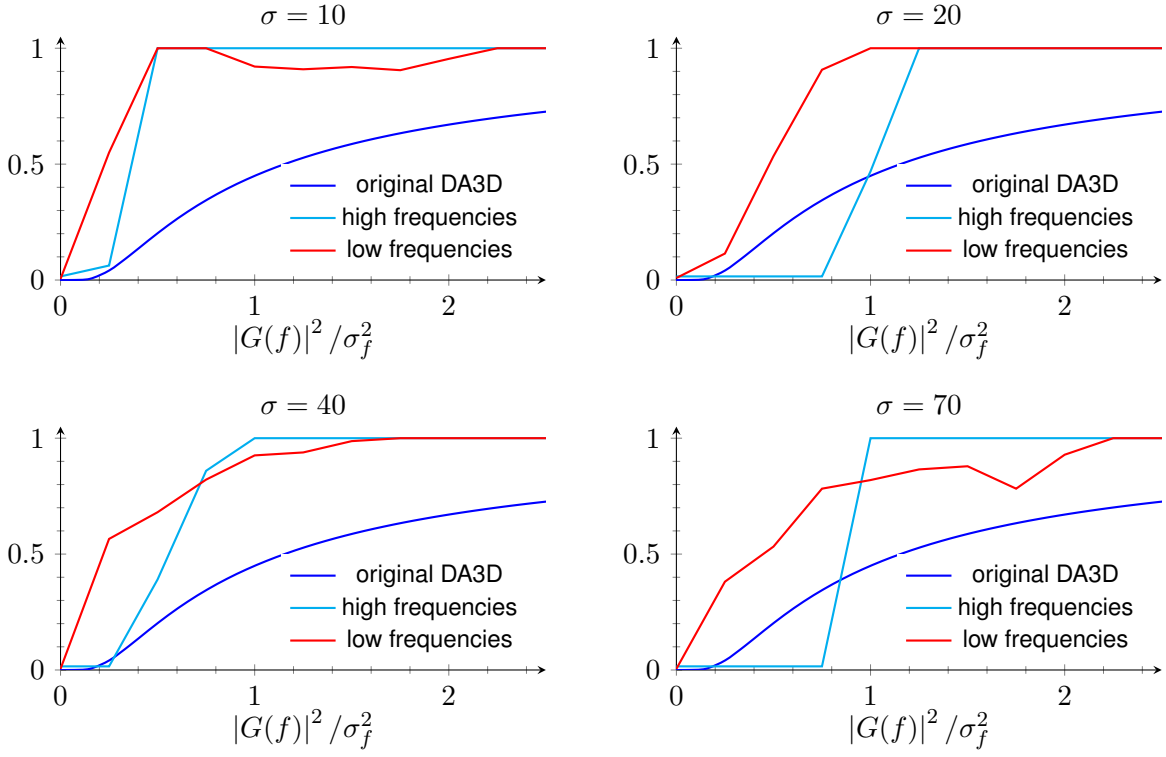


Figure 1.21 – Optimal shrinkage curves for K-SVD.

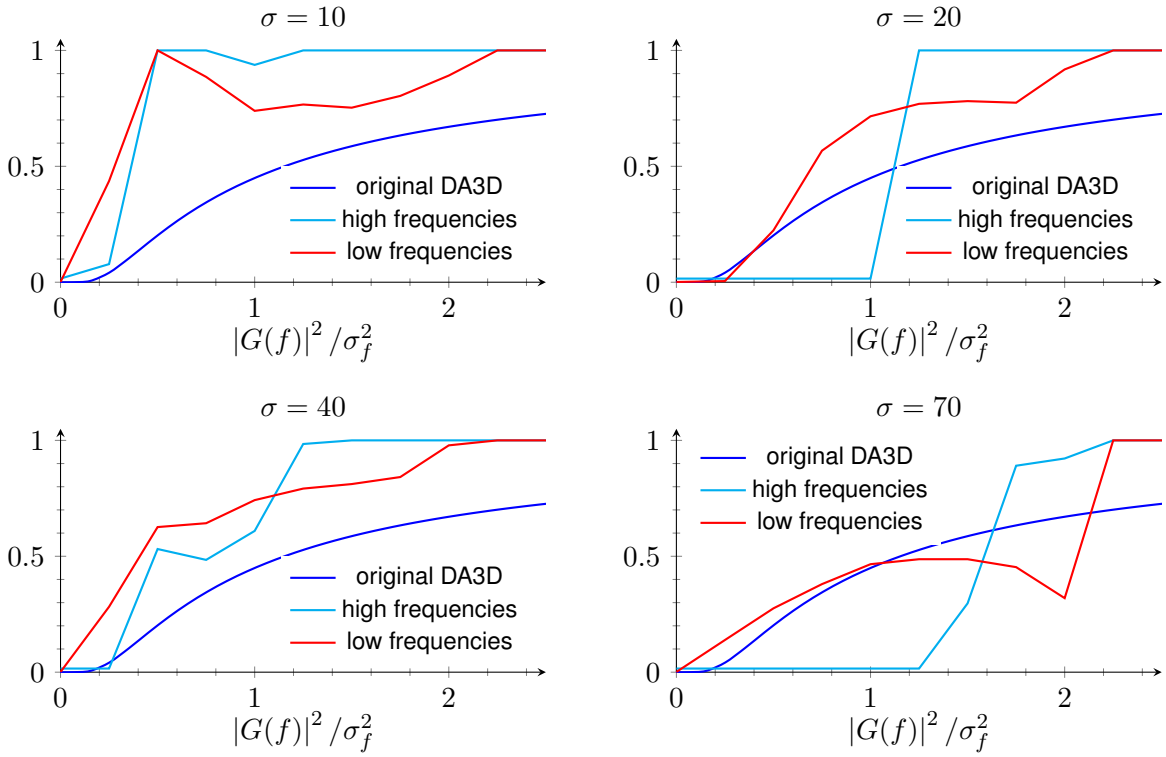


Figure 1.22 – Optimal shrinkage curves for BM3D.

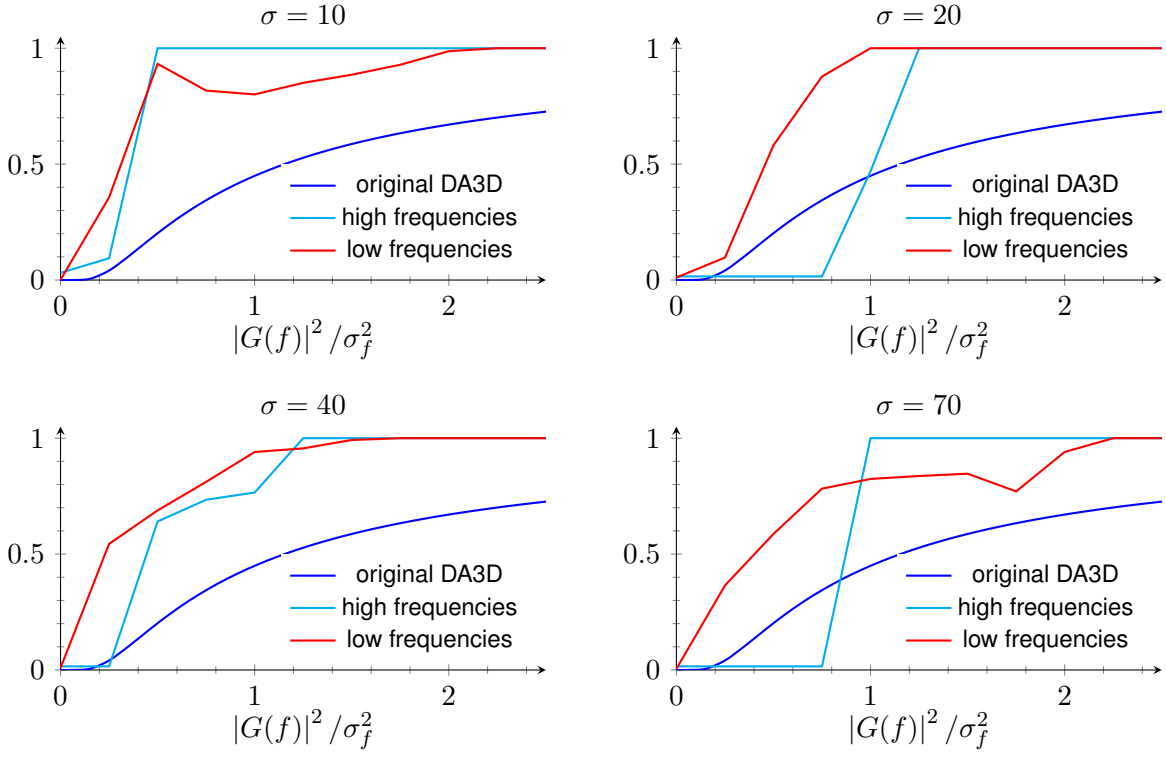


Figure 1.23 – Optimal shrinkage curves for Non-Local Bayes.

$\sigma = 10$	PNSR	With DA3D		Custom Curves	
BLS-GSM	35.64	36.70	+1.06	36.68	+1.04
DCT Denoising	36.31	36.94	+0.63	36.78	+0.48
Non-Local Means	35.79	36.64	+0.86	36.25	+0.46
K-SVD	36.35	37.06	+0.71	36.60	+0.25
BM3D	37.03	37.74	+0.72	**38.00**	+0.97
Non-Local Bayes	36.80	37.68	+0.88	**37.94**	+1.14

$\sigma = 20$	PNSR	With DA3D		Custom Curves	
BLS-GSM	31.99	33.49	+1.51	33.46	+1.47
DCT Denoising	32.87	33.78	+0.91	33.80	+0.94
Non-Local Means	32.35	33.70	+1.35	33.70	+1.35
K-SVD	32.59	33.74	+1.15	33.20	+0.61
BM3D	33.69	**34.25**	+0.55	**34.18**	+0.49
Non-Local Bayes	33.78	**34.24**	+0.46	**34.18**	+0.40

$\sigma = 40$	PNSR	With DA3D		Custom Curves	
BLS-GSM	28.67	30.50	+1.83	30.44	+1.77
DCT Denoising	29.55	30.60	+1.04	30.57	+1.02
Non-Local Means	28.83	30.55	+1.72	30.58	+1.74
K-SVD	29.45	30.62	+1.17	29.77	+0.32
BM3D	29.99	**31.33**	+1.34	**31.41**	+1.43
Non-Local Bayes	29.96	**31.40**	+1.45	**31.24**	+1.28

$\sigma = 70$	PNSR	With DA3D		Custom Curves	
BLS-GSM	26.37	28.13	+1.76	28.05	+1.69
DCT Denoising	27.16	28.02	+0.86	27.88	+0.72
Non-Local Means	26.06	27.53	+1.46	28.03	+1.97
K-SVD	26.97	27.99	+1.01	27.56	+0.58
BM3D	28.18	**28.58**	+0.40	**28.55**	+0.38
Non-Local Bayes	27.69	28.36	+0.67	**28.42**	+0.74

Table 1.4 – Difference in performance between standard DA3D and the version with tuned shrinkage curves. The results are contrasting, and show that the performance varies with the database. This suggests that the effectivness of DA3D lies more on the bilateral weight and on the sparse aggregation than on the shrinkage step itself.

ments to compute the optimal curves, we observed that even big changes in the parameters were reflected in small improvements on the overall PSNR.

Moreover, the patches that are more affected by an optimal shrinkage curve are the ones with a big support. At the same time, if a patch has a big support (or, in other terms, a large weight k), it means that the patch is mostly flat. A *mostly flat* patch will have almost all its coefficients already close to zero, so changing the shrinkage curve will not yield a big difference.

1.10 Conclusions

This chapter presented DA3D, a fast Data Adaptive Dual Domain Denoising algorithm for "last step" processing. It performs frequency domain shrinkage on shape and data-adaptive patches. The key innovations of this method are a sparse processing that allows bigger blocks to be used and a plane regression that greatly improves the results on gradients and smooth parts. The experiments show that DA3D improves the results of most denoising algorithms with reasonable computational cost, achieving a performance superior to the state-of-the-art.

Chapter 2

A Multi-Scale Denoising Framework

We reconsider in this chapter the class of patch based denoising algorithms and prove that they underperform at coarse scale. We solve this problem by applying them at several scales in a multiscale structure. The main technical issue is to maintain a white noise at the low scales, which implies applying a hard frequency cut-off, yet avoiding the Gibbs reconstruction artefacts caused by this cut-off. We solve this dilemma by applying after denoising at all scales a "soft fusion" of the scales, that only retains the lower frequencies of each scale with the exception, of course, of the finest scale. This eliminates the frequency involved in ringing artefacts. This method is demonstrated on several denoising algorithms. It gives a significant PSNR improvement for all of them. The visual aspect improves also significantly by removing staircasing artifacts and low frequency bumps.

As we shall see in Chapter 3, this multiscale improvement is complementary to DA3D, presented in Chapter 1. In Chapter 3 we shall explore systematically how both image denoising improvements can be combined to yield the best PSNR with minimal artefacts. In the present chapter, for a sake of clarity, we limit ourselves to demonstrate how our new multiscale framework is effective on classic algorithms.

2.1 Introduction

In the preceding chapter, we provided a new view on frequency transform thresholding, showing that it could perform excellently as the last step of any patch based denoising algorithm. In this chapter we address another question left untreated by most state of the art denoising algorithms. We observed that most of them limit their action to a limited pixel neighborhood. This clearly means that low frequency noise remains untreated. This is a drawback that yields very visible artifacts in flat areas, as we shall check in the experimental part of this chapter, and in our experimental study of Chapter 3. Thus, in this chapter we provide a new perspective of another principle: the multi-scale representation. This principle has already been explored in [4, 15]. Even though the results were only partially satisfying, the ideas presented are simple and promising. The main problem with their approach is that in higher scales, the noise becomes correlated, thus reducing the performance of standard algorithms. Another work that tries to use a multi-scale model is [47]. The difference with our work is that [47] does not use classical denoising algorithm in the process and that it does not avoid artifacts in the recomposition.

The multi-scale representation is also intrinsically present in wavelet-based denoising algorithms [11, 18, 34, 45]. It is also present in [52], where the KSVD algorithm is applied on a wavelet decomposition of the image. The improvement over a single scale KSVD is

important, especially for high PSNR, but since the wavelet decomposition does not allow for a conservative recomposition, the authors need what they call a fusion strategy, in order to reduce the artifacts. In [13] the authors propose to reduce the artifact in wavelet-based denoising by using a constrained total variation minimization.

Denoising a multi-scale pyramid is a way to process more information to obtain a better result. Other efforts have been made to provide more input to denoising algorithm. In particular, it is worth citing [5, 65], that try to improve the results of a denoising algorithm by using an external database of similar images.

2.1.1 Our contribution

In this chapter we present a new multi-scale framework that can be applied to any other existing denoising algorithm, consistently improving its results. The framework uses a simple DCT pyramid, and is not computationally demanding.

Simply using the DCT pyramid may lead to ringing artifacts. We solve this issue by introducing what we call *conservative recomposition*, which allows to keep the advantages of the pyramid while avoiding its problems.

The result is a way of transforming any denoising algorithm into a multi-scale one, with improvements both in visual quality and PSNR, and with little additional cost.

2.2 A Multi-scale Framework

We take the classic assumption [20] that the statistics of natural images are invariant to a change of scale. A possible justification for this is that scenes are equally likely to be viewed from different distances. A classic theoretical model for scale invariance can be found in [30]. The scale invariance assumption is used for several multi-scale algorithms, such as [4, 45].

Local and non-local denoising methods, because of the size of patches and search windows, are usually better at removing high-frequency noise, while they underperform at low frequencies. In order to corroborate this claim, we show in Figure 2.1 the frequency distribution of the result of some popular denoising algorithms on an image composed only of white noise, together with the results for our multi-scale version of those algorithms proposed in this paper.

Historically, wavelet thresholding was a common way of denoising every scale of an image, but it has proven difficult to extend, and currently it is surpassed by modern patch-based method. Nevertheless, the study of wavelet methods for thresholding is a first successful attempt at multiscale image denoising, and it has great value as a source of inspiration.

Given any sort of multiscale representation of an image, a possible way of exploiting it in order to improve the performance of the denoising is to apply the denoising algorithm at each scale, and then to recompose the image while keeping duplicate information from the lowest scale when available.

There are two restrictions to this: first, we need every layer of the multiscale representation to have white Gaussian additive noise; second, we need a practical and effective way of joining the different scales together to obtain the final result.

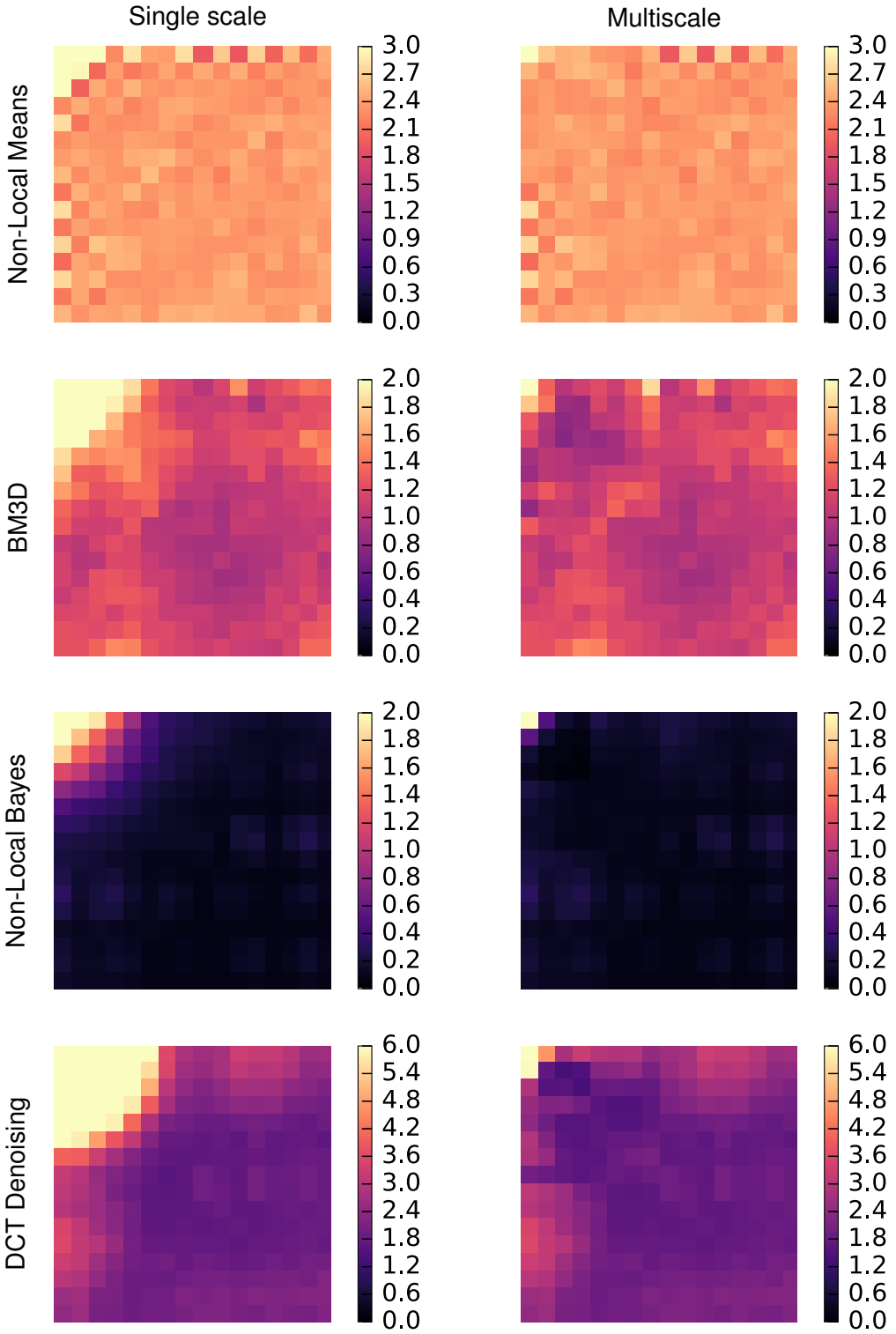


Figure 2.1 – DCT transform of the result of various denoising algorithms applied to an image of pure white noise, with and without the Multi-Scale Framework. Notice that there is still noise remaining in the upper left corner of the Single Scale version, that contains the low frequencies of the image. In the Multi-Scale version, the residual noise is much more uniform across frequencies.

2.2.1 DCT pyramid

A way of satisfying both of those conditions is to use a DCT Pyramid. The Discrete Cosine Transform, or DCT given in (2.1) is a real separable orthogonal transform. For 2-D signals, the DCT can be computed by applying (2.1) to the rows and the columns. Its inverse is the IDCT (2.2).

$$Y_k = \frac{1}{N} \sum_{j=0}^{N-1} X_j \cos \left[\pi \left(j + \frac{1}{2} \right) \frac{k}{N} \right], \tag{2.1}$$

$$X_k = Y_0 + 2 \sum_{j=1}^{N-1} Y_j \cos \left[\pi \left(k + \frac{1}{2} \right) \frac{j}{N} \right]. \tag{2.2}$$

The DCT of an image (as in Figure 2.1) has the coefficients relative to the low frequencies in the upper-left corner. Moreover, it transforms additive Gaussian white noise in additive Gaussian white noise.

Therefore, the DCT transform can be used to form a multiscale representation of an image. The downsampling of the image is simply done by extracting the low frequencies from the DCT transform of the image, and by computing the IDCT on just those frequencies. Each layer of the pyramid has half the width and half the length of the previous one.

Using (2.1) and (2.2) for this procedure keeps the values of the image on the same range. On the other hand, the standard deviation of the noise gets halved at each successive scale. This makes sense in retrospect, since zooming out an image is a way to reduce its noise.

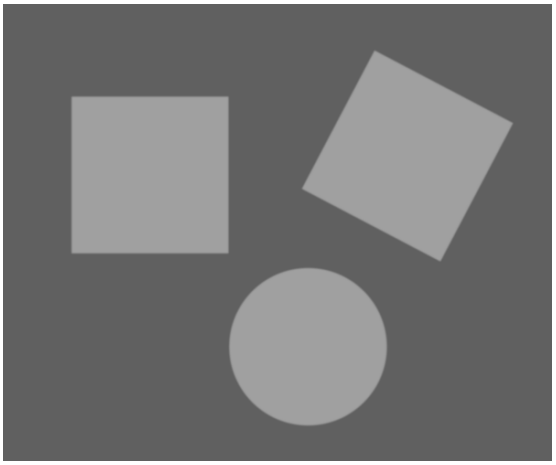
This representation has the advantage that, since an additive white Gaussian noise remains so under the DCT transform, the model of the noise remains the same in every layer of the pyramid. Thus, no particular adaptation of the denoising algorithm is needed to denoise the coarse layers. This is a very important property, since it allows a straight-forward extension of any denoising algorithm. Recomposing the pyramid is trivial, since it can be reduced to substituting the low frequencies of a layer with the frequencies of the coarser layer.

The drawback of this model is that, since each layer is essentially the result of the convolution of the previous one with a sinc-like function, ringing artifacts due to the Gibbs effect can appear in the result. If the denoising algorithms alters the high frequencies of the upper layer, the recomposition can present some unpleasant artifacts.

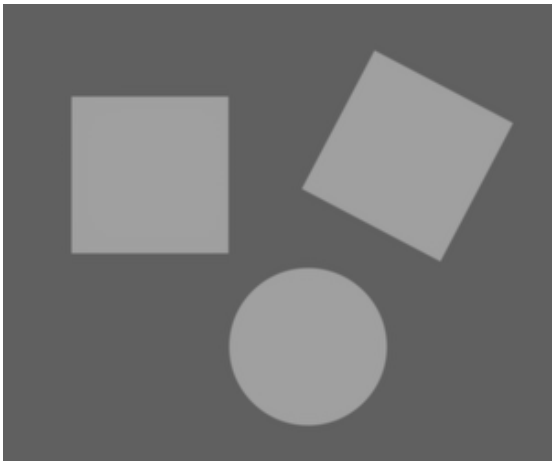
2.2.2 Conservative recomposition

We have to consider that the ringing artifacts are formed when two consecutive layers of the pyramid do not merge properly. To solve this issue, we found an easy and tentatively new solution. In the recomposition, we just consider the *lower frequencies* of the upper layers. Those frequencies are selected by a parameter, that is specific of the denoising algorithm used with the framework.

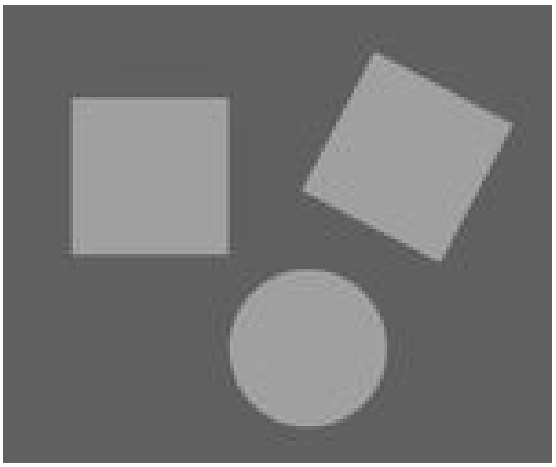
This conservative recomposition is the main reason why we cannot use a pyramid based on a wavelet decomposition. Since in the wavelet base the frequencies are in discrete group, there is no way to discard a part of them to better improve the fusion of the layers.



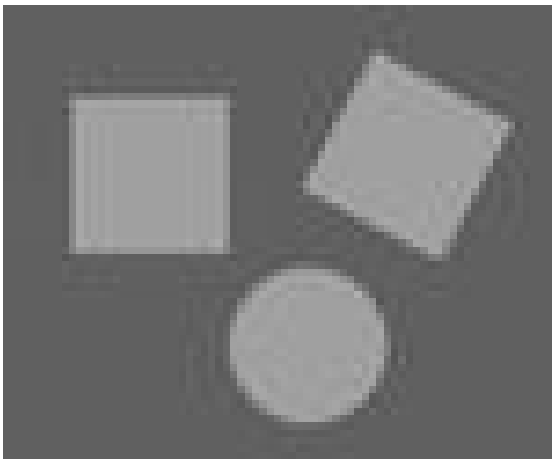
(a) Layer 0



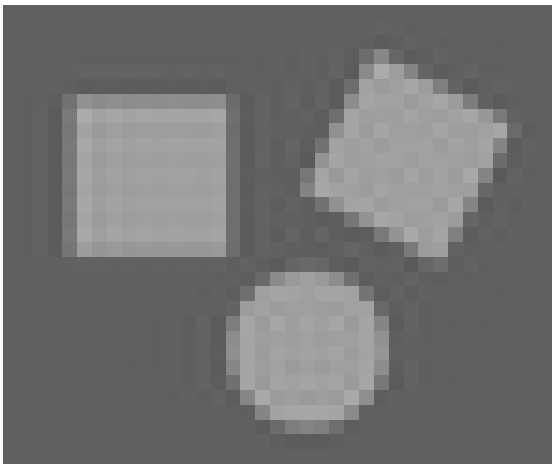
(b) Layer 1



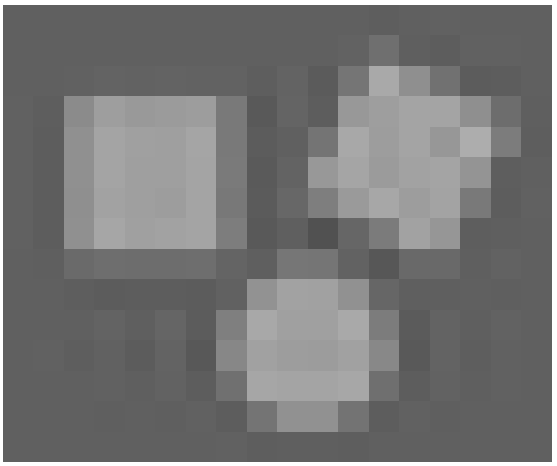
(c) Layer 2



(d) Layer 3



(e) Layer 4



(f) Layer 5

Figure 2.2 – Ringing artifacts in the upper layer of the DCT pyramid. The layers are resized for easier comparison. Since keeping all the coefficients in the low frequency of the DCT is comparable to a convolution with a sinc function, the ripples are visible in the upper layer.

Algorithm 2 Pseudo-code for the Multi-Scale Framework.

function MULTISCALE($input$, σ_{noise}, n_{scales}, f_{rec})
  $result \leftarrow$ NULL
  for $scale \leftarrow n_{scales} - 1, \ldots, 0$ **do**
    $layer \leftarrow$ EXTRACTSCALE($input$, $scale$)
    $tmp \leftarrow$ DENOISE($layer$, $\sigma_{noise}/2^{scale}$)
    $result \leftarrow$ MERGECOARSE(tmp, $result$, f_{rec})
  end for
  return $result$
end function

function EXTRACTSCALE($image$, $scale$)
  $w, h \leftarrow$ SIZE($image$)
  $w_{out} \leftarrow \lfloor w/2^{scale} \rfloor$
  $h_{out} \leftarrow \lfloor h/2^{scale} \rfloor$
  $freq \leftarrow$ DCT($image$)
  $tmp \leftarrow$ ZEROS(w_{out}, h_{out})
  for $i \leftarrow 0, \ldots, h_{out} - 1$, $j \leftarrow 0, \ldots, w_{out} - 1$ **do**
    $tmp[i, j] \leftarrow freq[i, j]$
  end for
  return IDCT(tmp)
end function

function MERGECOARSE($image$, $coarse$, f_{rec})
  if $coarse =$ NULL **then**
    return $image$
  else
    $freq \leftarrow$ DCT($image$)
    $tmp \leftarrow$ DCT($coarse$)
    $w, h \leftarrow$ SIZE($coarse$)
    $w_{rec} \leftarrow w \cdot f_{rec}$
    $h_{rec} \leftarrow h \cdot f_{rec}$
    for $i \leftarrow 0, \ldots, h_{rec} - 1$, $j \leftarrow 0, \ldots, w_{rec} - 1$ **do**
      $freq[i, j] \leftarrow tmp[i, j]$
    end for
    return IDCT($freq$)
  end if
end function

2.2.3 Algorithm

The pseudocode for the Multiscale Framework is shown in Algorithm 2, and a scheme showing the procedure for a sample image is shown in Figure 2.3.

In Algorithm 2, the support function EXTRACTSCALE($image$, $scale$) is used to extract a specific level from the DCT pyramid. The level 0 is the input image itself, and every other level is half the size of the previous one. Conversely, the support function MERGECOARSE($image$, $coarse$, f_{rec}) is used to join together two different levels. The low frequency of $image$ get replaced by the ones from $coarse$, in a ratio proportional to f_{rec}. Finally, MULTISCALE($input$, σ_{noise}, n_{scales}, f_{rec}) performs the whole denoising process, using the previous two functions. Here DENOISE($image$, σ) is the denoising algorithm that is used with the framework.

Since each layer of the pyramid contains a quarter of the pixels of the previous one, the time needed to denoise the pyramid is similar to the time needed for a single-scale denoising. In fact, we observed in our experiments that the overhead is mainly due to the DCT transform, which is fast to compute [59].

2.3 Parameters of Multi-Scale Algorithms

In order to test the multi-scale framework, we first computed the best parameters for five classic denoising algorithms. The only parameters of the Multiscale Framework are the number of scales and the recomposition factor. To find the best values for different values of noise, we tried different combinations over the set of images in Figure 2.4. The internal parameters of the single algorithms were not modified.

2.3.1 Non-Local Means

We tried to use the Non-Local Means Denoising Algorithm [2] within our multi-scale framework. We tested it for various amounts of noise, and for various parameters of the framework. The results are in Figure 2.5. It is interesting to notice that different amounts of noise call for different parameters. This is to be expected, since the denoising algorithm utilizes different internal parameters depending on the kind of noise.

2.3.2 K-SVD

K-SVD [14, 29] Denoising uses sparse representations of the image patches in terms of a learned dictionary. It is an effective method, with good results. We tested it to find the best parameters within the multiscale framework. The results are in Figure 2.6. One can see that the K-SVD algorithm, when the noise is over $\sigma = 20$, benefits from the application of the framework.

2.3.3 DCT denoising

DCT Denoising [63] is a simple two-step denoising algorithm. It is very fast, but in general it is not regarded as the one with the best results. We included it in our testing because we wanted to see how much we could improve the result of a really simple algorithm. In order to further speed it up, we reduced the size of the patches to 4×4, instead of the

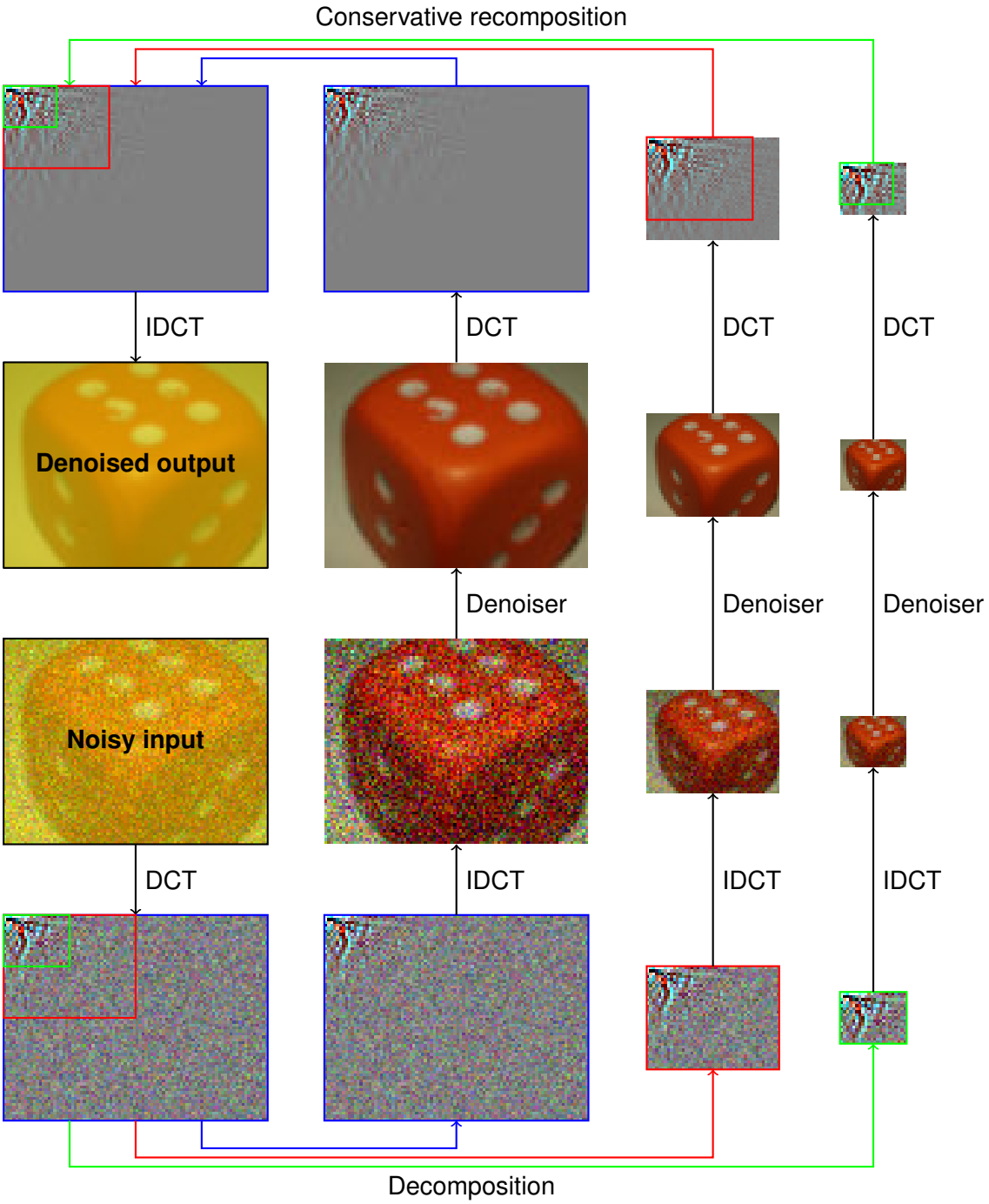


Figure 2.3 – Scheme of the Multi-Scale Framework with three levels. Notice that not all the frequencies of the upper layers are used for the recomposition. This is done in order to avoid ringing artifacts. The single denoising steps can be performed by any existing denoising algorithm.

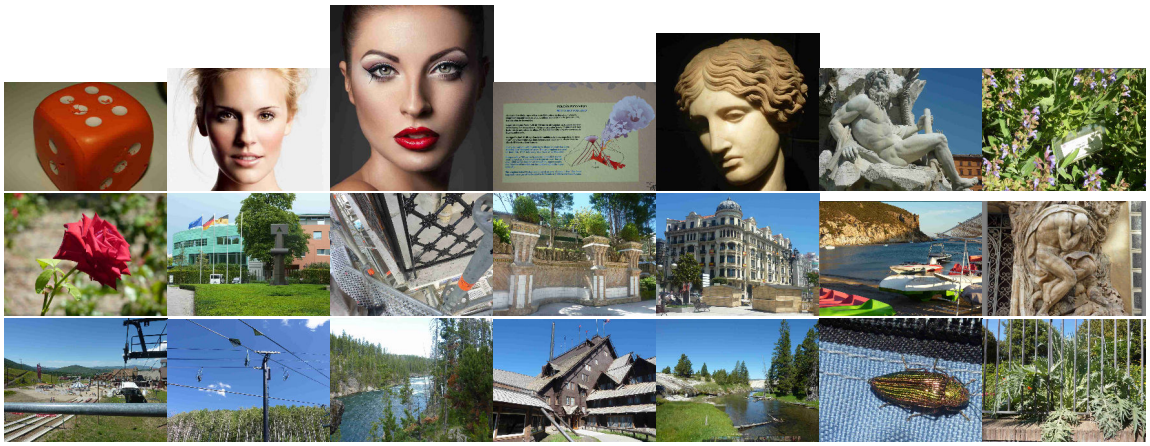


Figure 2.4 – Images used to find the best parameters of each multi-scale algorithm.

suggested 16×16. The results with different parameters for the Multi-Scale Framework are in Figure 2.7. Notice that the best results are obtained with a large number of scales. This is also due to the choice of using the algorithm with small patches, that allows the algorithm to only "see" the high frequency noise.

2.3.4 BM3D

BM3D [7] is considered the reference for denoising algorithms. Even though it can provide results that contain artifacts, especially with high levels of noise, it provides high PSNR values and overall a good image quality. The results with different parameters are provided in Figure 2.8. It may be worth noticing that for low values of noise ($\sigma = 10$) the best results are obtained with only one scale, i.e. with the original algorithm. This may be due to the fact that BM3D is highly optimized, especially for low levels of noise, and because the Multi-Scale Framework is only marginally useful with those levels of noise, since the upper layers are almost noise-free.

2.3.5 Non-Local Bayes

Non-Local Bayes [27] is a two-step denoising algorithm. It is currently considered a state-of-the-art algorithm, it is fast and it provides good results, both visually and in terms of PSNR. The results of the testing are shown in Figure 2.9.

2.4 Results

In Section 2.3, we have tried different parameters for a few denoising algorithms, in order to select the best ones. To test the Multi-Scale Framework, we used a different dataset (shown in Figure 2.10). This was necessary in order to understand if the parameters found were really consistent.

We tried the algorithm and the noise levels of Section 2.3. For each algorithm and noise we selected the parameters which gave the best results in the dataset of Figure 2.4, and we applied them. The results are shown in Table 2.1.

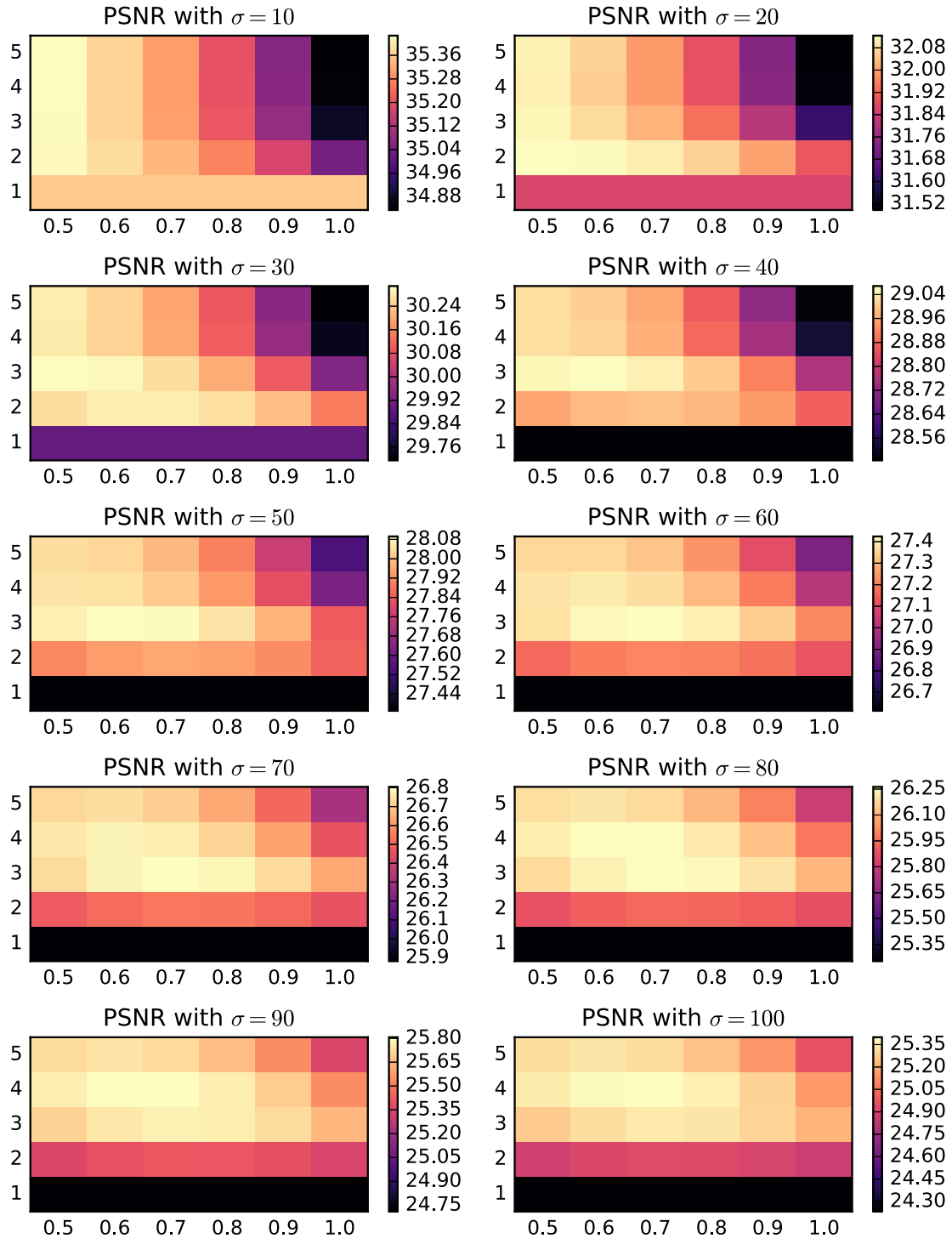


Figure 2.5 – Average PSNR with different parameters of the Multi-Scale Framework applied to the Non-Local Means denoising algorithm. The integers on the left of each figure (1, 2, . . . , 5) represent the number of scales n_{scales} used in Algorithm 2. The value at the bottom is the fraction f_{rec} of low frequencies at each scale being used in the recomposition.

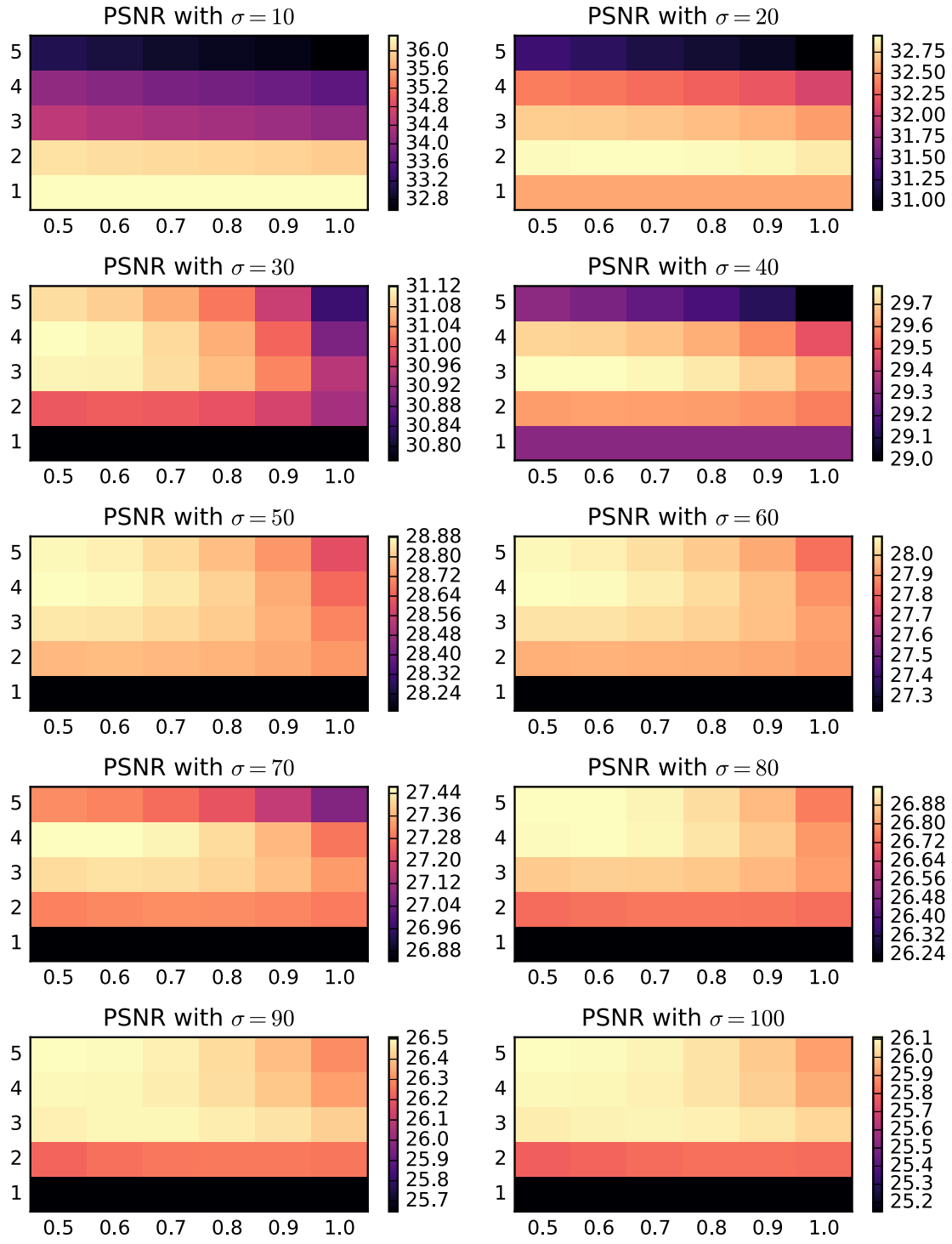


Figure 2.6 – Average PSNR with different parameters of the Multi-Scale Framework applied to the K-SVD denoising algorithm. The integers on the left of each figure (1, 2, . . . , 5) represent the number of scales n_{scales} used in Algorithm 2. The value at the bottom is the fraction f_{rec} of low frequencies at each scale being used in the recomposition.

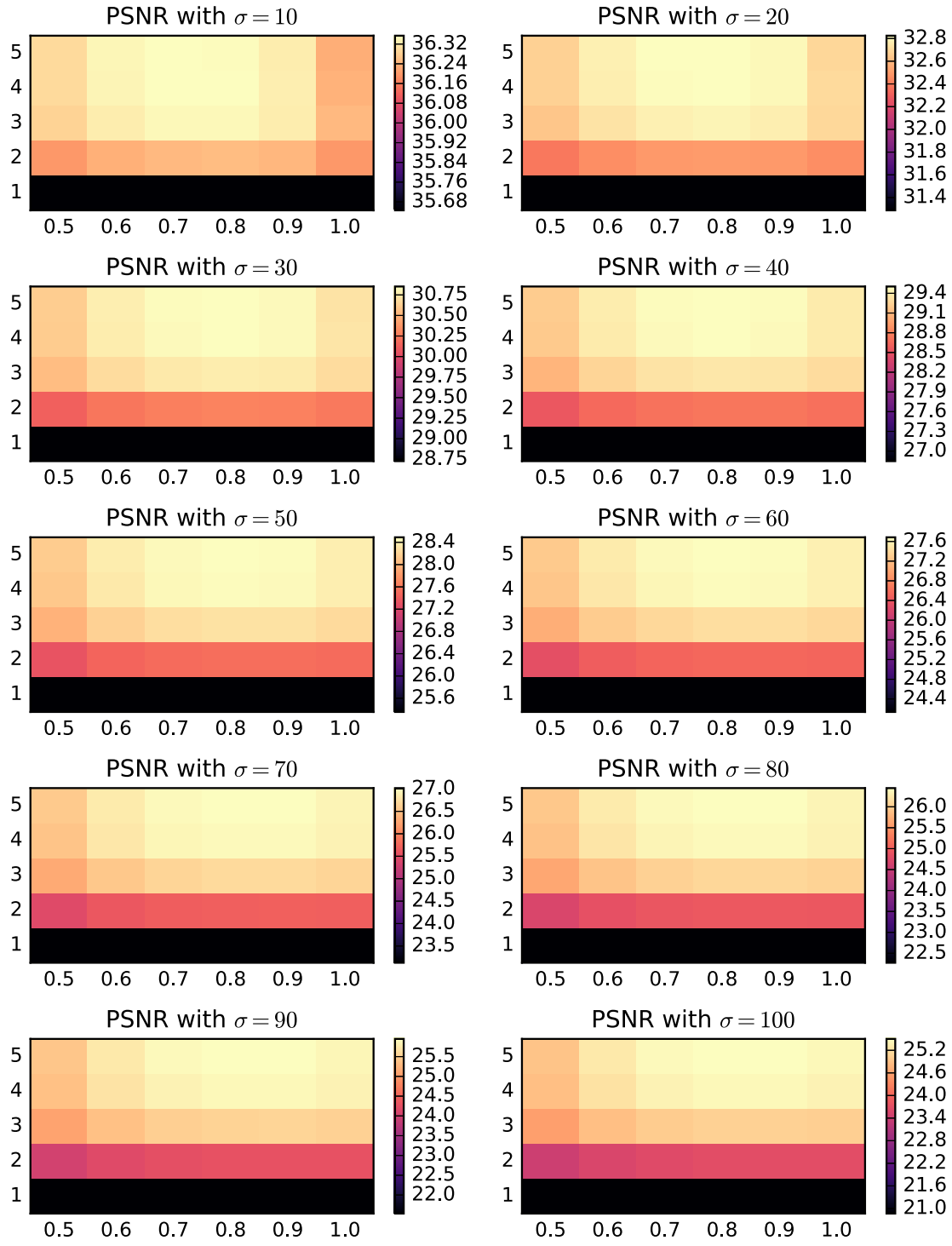


Figure 2.7 – Average PSNR with different parameters of the Multi-Scale Framework applied to the DCT denoising algorithm. The integers on the left of each figure (1, 2, . . . , 5) represent the number of scales n_{scales} used in Algorithm 2. The value at the bottom is the fraction f_{rec} of low frequencies at each scale being used in the recomposition.

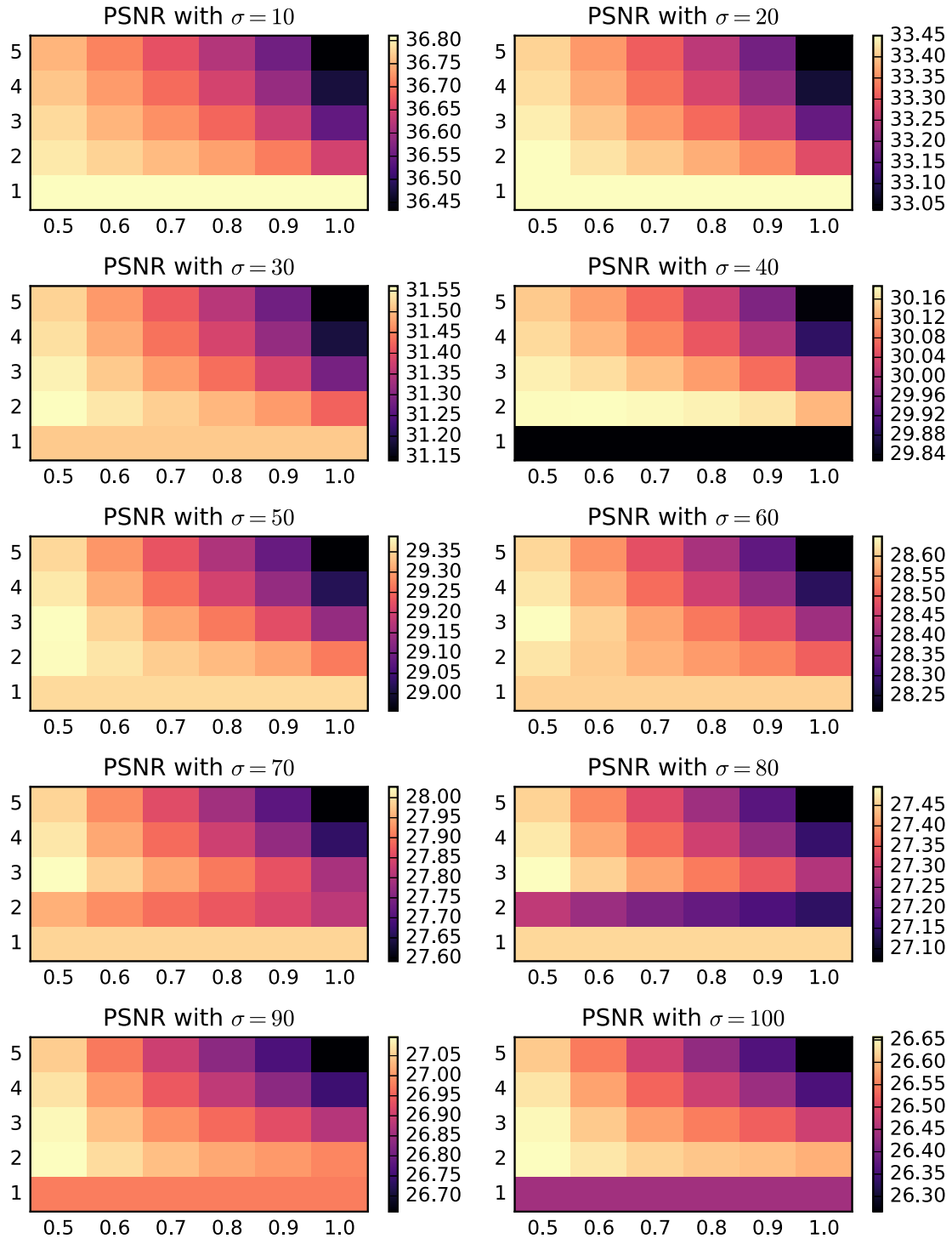


Figure 2.8 – Average PSNR with different parameters of the Multi-Scale Framework applied to the BM3D denoising algorithm. The integers on the left of each figure (1, 2, . . . , 5) represent the number of scales n_{scales} used in Algorithm 2. The value at the bottom is the fraction f_{rec} of low frequencies at each scale being used in the recomposition.

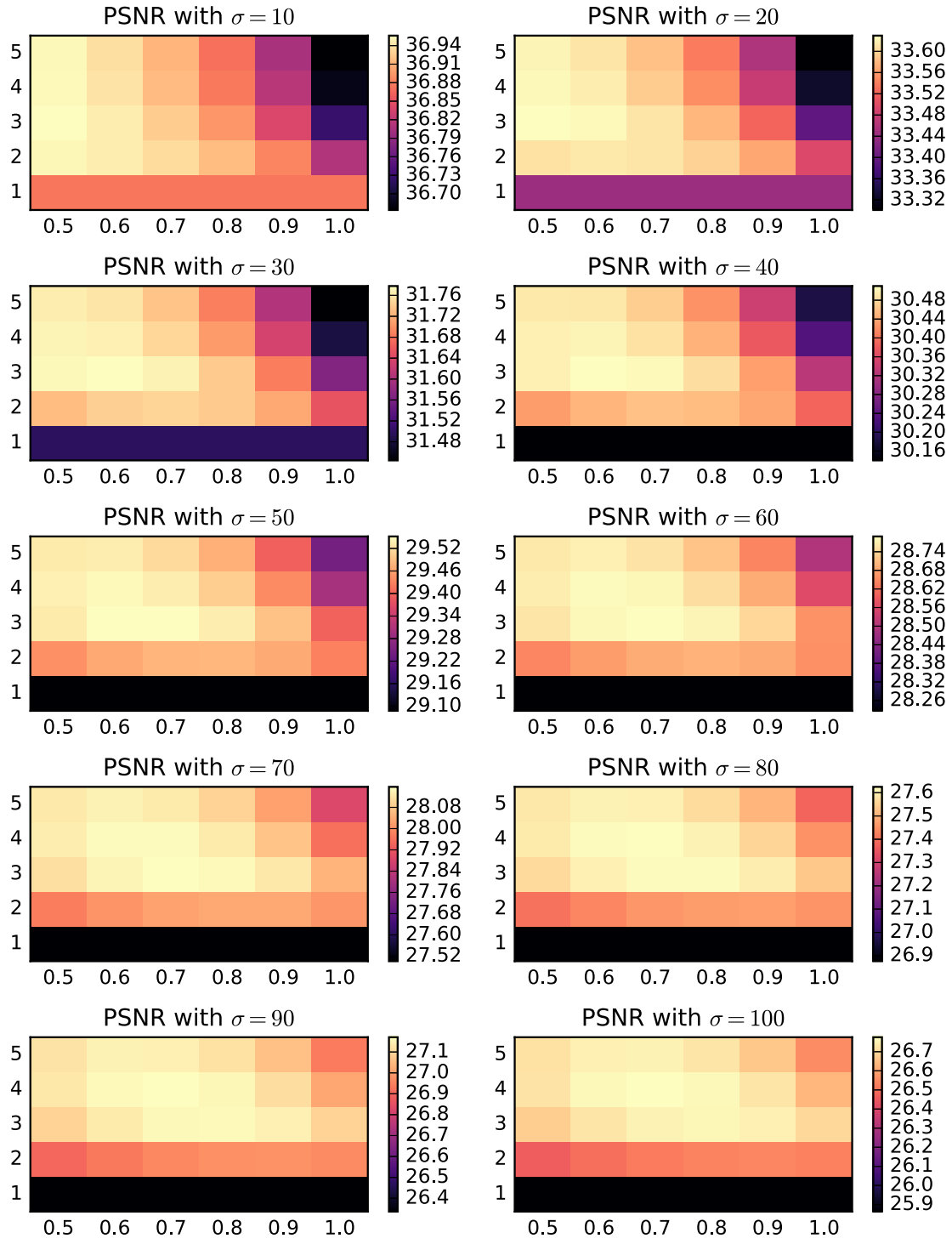


Figure 2.9 – Average PSNR with different parameters of the Multi-Scale Framework applied to the Non-Local Bayes denoising algorithm. The integers on the left of each figure (1, 2, . . . , 5) represent the number of scales n_{scales} used in Algorithm 2. The value at the bottom is the fraction f_{rec} of low frequencies at each scale being used in the recomposition.

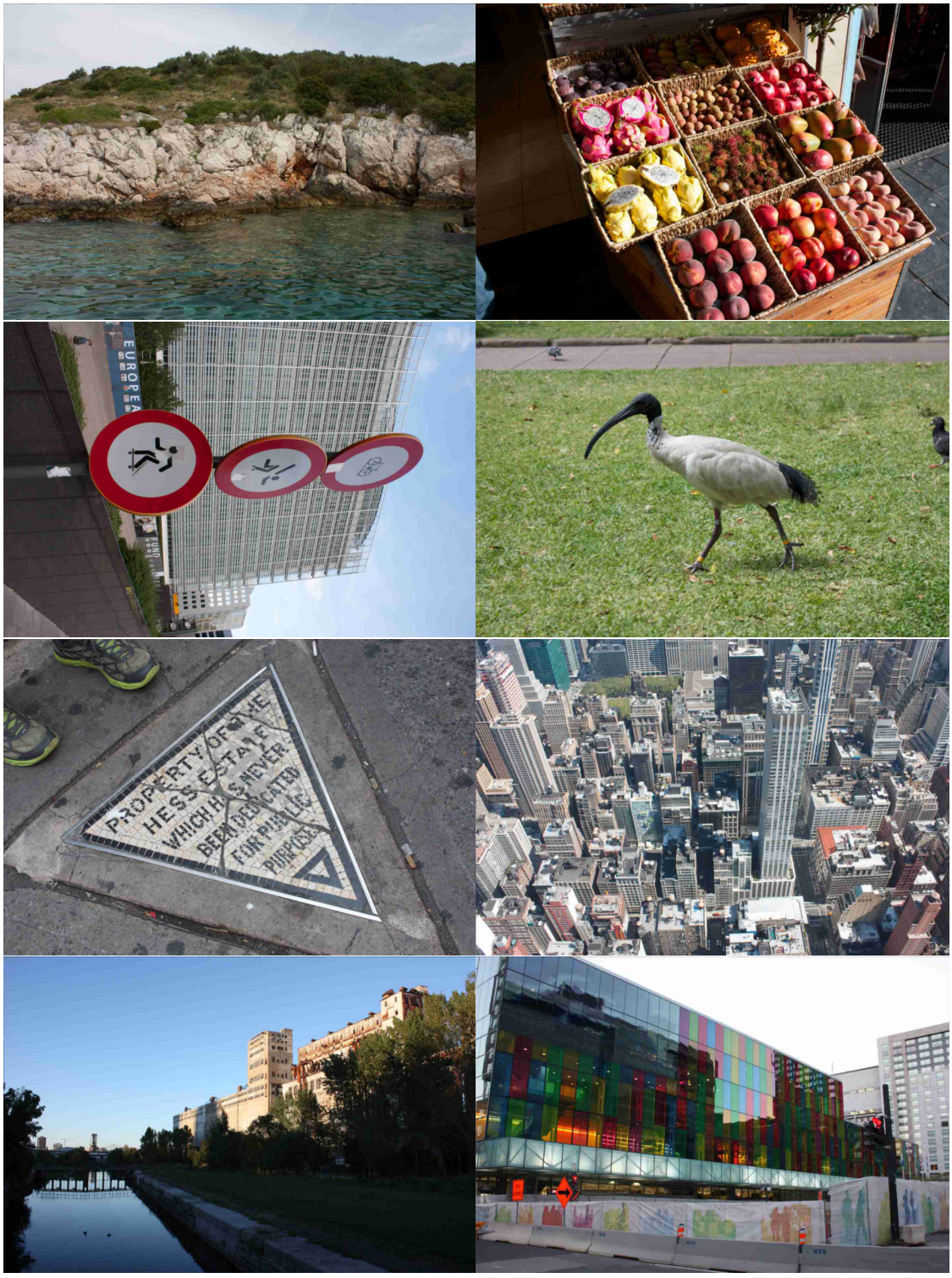


Figure 2.10 – Images used to test each multi-scale algorithm.

Noise	Non-Local Means			K-SVD Denoising		
	single	multi	gain	single	multi	gain
$\sigma = 10$	36.59dB	36.74dB	+0.15dB	38.18dB	38.18dB	+0.00dB
$\sigma = 20$	32.59dB	33.20dB	+0.61dB	34.05dB	34.49dB	+0.43dB
$\sigma = 30$	30.50dB	31.33dB	+0.83dB	32.06dB	32.07dB	+0.01dB
$\sigma = 40$	28.84dB	29.98dB	+1.14dB	30.42dB	30.97dB	+0.55dB
$\sigma = 50$	27.57dB	28.87dB	+1.31dB	29.19dB	29.81dB	+0.61dB
$\sigma = 60$	26.75dB	28.19dB	+1.44dB	28.17dB	28.95dB	+0.78dB
$\sigma = 70$	25.93dB	27.48dB	+1.55dB	27.65dB	28.32dB	+0.67dB
$\sigma = 80$	25.24dB	26.85dB	+1.61dB	26.97dB	27.72dB	+0.75dB
$\sigma = 90$	24.66dB	26.31dB	+1.65dB	26.36dB	27.14dB	+0.78dB
$\sigma = 100$	24.15dB	25.83dB	+1.68dB	25.82dB	26.69dB	+0.87dB
Noise	DCT Denoising			BM3D		
	single	multi	gain	single	multi	gain
$\sigma = 10$	37.43dB	37.86dB	+0.43dB	38.59dB	38.59dB	+0.00dB
$\sigma = 20$	33.04dB	34.05dB	+1.00dB	34.97dB	34.96dB	-0.01dB
$\sigma = 30$	30.38dB	31.89dB	+1.52dB	32.87dB	32.91dB	+0.04dB
$\sigma = 40$	28.42dB	30.40dB	+1.98dB	31.37dB	31.46dB	+0.09dB
$\sigma = 50$	26.85dB	29.25dB	+2.40dB	30.22dB	30.36dB	+0.14dB
$\sigma = 60$	25.55dB	28.32dB	+2.78dB	29.12dB	29.31dB	+0.19dB
$\sigma = 70$	24.42dB	27.56dB	+3.13dB	28.65dB	28.84dB	+0.19dB
$\sigma = 80$	23.44dB	26.88dB	+3.44dB	27.97dB	28.20dB	+0.23dB
$\sigma = 90$	22.57dB	26.29dB	+3.73dB	27.30dB	27.59dB	+0.29dB
$\sigma = 100$	21.78dB	25.76dB	+3.98dB	26.32dB	26.74dB	+0.42dB
Noise	Non-Local Bayes			Non-Local Bayes (IPOL)		
	single	multi	gain	single	multi	gain
$\sigma = 10$	38.58dB	38.71dB	+0.13dB	37.97dB	38.17dB	+0.20dB
$\sigma = 20$	34.75dB	35.09dB	+0.35dB	34.49dB	34.75dB	+0.25dB
$\sigma = 30$	32.49dB	33.07dB	+0.58dB	32.23dB	32.62dB	+0.39dB
$\sigma = 40$	30.88dB	31.65dB	+0.77dB	30.61dB	31.23dB	+0.62dB
$\sigma = 50$	29.65dB	30.61dB	+0.97dB	30.30dB	30.53dB	+0.22dB
$\sigma = 60$	28.65dB	29.75dB	+1.09dB	29.32dB	29.62dB	+0.30dB
$\sigma = 70$	27.83dB	29.02dB	+1.19dB	28.57dB	28.93dB	+0.36dB
$\sigma = 80$	27.14dB	28.40dB	+1.26dB	27.87dB	28.33dB	+0.46dB
$\sigma = 90$	26.56dB	27.86dB	+1.30dB	27.27dB	27.78dB	+0.51dB
$\sigma = 100$	26.05dB	27.39dB	+1.34dB	26.75dB	27.46dB	+0.72dB

Table 2.1 – Results with the best settings for every algorithm. The results are an average over the images of Figure 2.10.

Noise	Best single-scale		Best multi-scale		Gain
$\sigma = 10$	38.59dB	BM3D	38.71dB	NLB	+0.12dB
$\sigma = 20$	34.97dB	BM3D	35.09dB	NLB	+0.12dB
$\sigma = 30$	32.87dB	BM3D	33.07dB	NLB	+0.20dB
$\sigma = 40$	31.37dB	BM3D	31.65dB	NLB	+0.28dB
$\sigma = 50$	30.30dB	NLB IPOL	30.61dB	NLB	+0.31dB
$\sigma = 60$	29.32dB	NLB IPOL	29.75dB	NLB	+0.43dB
$\sigma = 70$	28.65dB	BM3D	29.02dB	NLB	+0.37dB
$\sigma = 80$	27.97dB	BM3D	28.40dB	NLB	+0.43dB
$\sigma = 90$	27.30dB	BM3D	27.86dB	NLB	+0.56dB
$\sigma = 100$	26.75dB	NLB IPOL	27.46dB	NLB IPOL	+0.71dB

Table 2.2 – Results with the best settings for every algorithm. The results are an average over the images of Figure 2.10.

We can see from the table that the Multi-Scale Framework consistently improves the results of the single-scale version of every algorithm. The only exception is BM3D, that save for $\sigma = 40$ has almost the same results in the single- and multi-scale version. This could be explained by the fact that, although BM3D has very good PSNR results, it produces artifacts. These artifacts can be amplified in the multi-scale version, so the advantages of the multi-scale are lost.

To judge the visual quality, some of the results for a noise of $\sigma = 40$ are available in Figures 2.11-2.22. It can be seen that the multi-scale counterpart of each algorithm generally increases the contrast and enhances lower-scale details. This is due to the fact that within this framework those details are better denoised.

In particular, observe how in the more complex algorithms (K-SVD, BM3D and Non-Local Bayes), the multi-scale version does indeed remove the low-frequency noise. This is particularly evident in smooth areas (Figures 2.12, 2.16, 2.18), but it is also visible within geometric patterns (Figure 2.24). Also, for geometric structures, the multi-scale better recovers the edges. A special mention should be done for Figure 2.20. The multi-scale version of the algorithm recovers some lines inside the windows of the building. At a first glance, this may look like the presence of ringing artifacts. In reality, looking at the original image, one can see that those structures are present in the original image too. No single-scale algorithm was able to retrieve them.

2.5 Conclusion

In this chapter we have reconsidered patch based denoising methods by observing that they lacked a natural multiscale structure. We proved that they underperform at lower scales. Extending them to make multiscale implied a multiscale representation that keeps white noise at all scales. We found that a DCT pyramid was the most evident way to proceeed, but had never been used, probably because of Gibbs effects caused by hard frequency thresholding. Our solution is to have a soft multiscale fusion that discards the high frequencies of the lower scales. We observed significant PSNR and visual quality improvements.

Nevertheless, this is not the end of the story. Indeed we used algorithms which have their parameters optimized for a single scale usage. Clearly immersing them in multiscale structure will lead to different parameter choices and probably to further improvements. We also thought of using wavelet bases, but found that they were not compatible with our soft multiscale fusion.

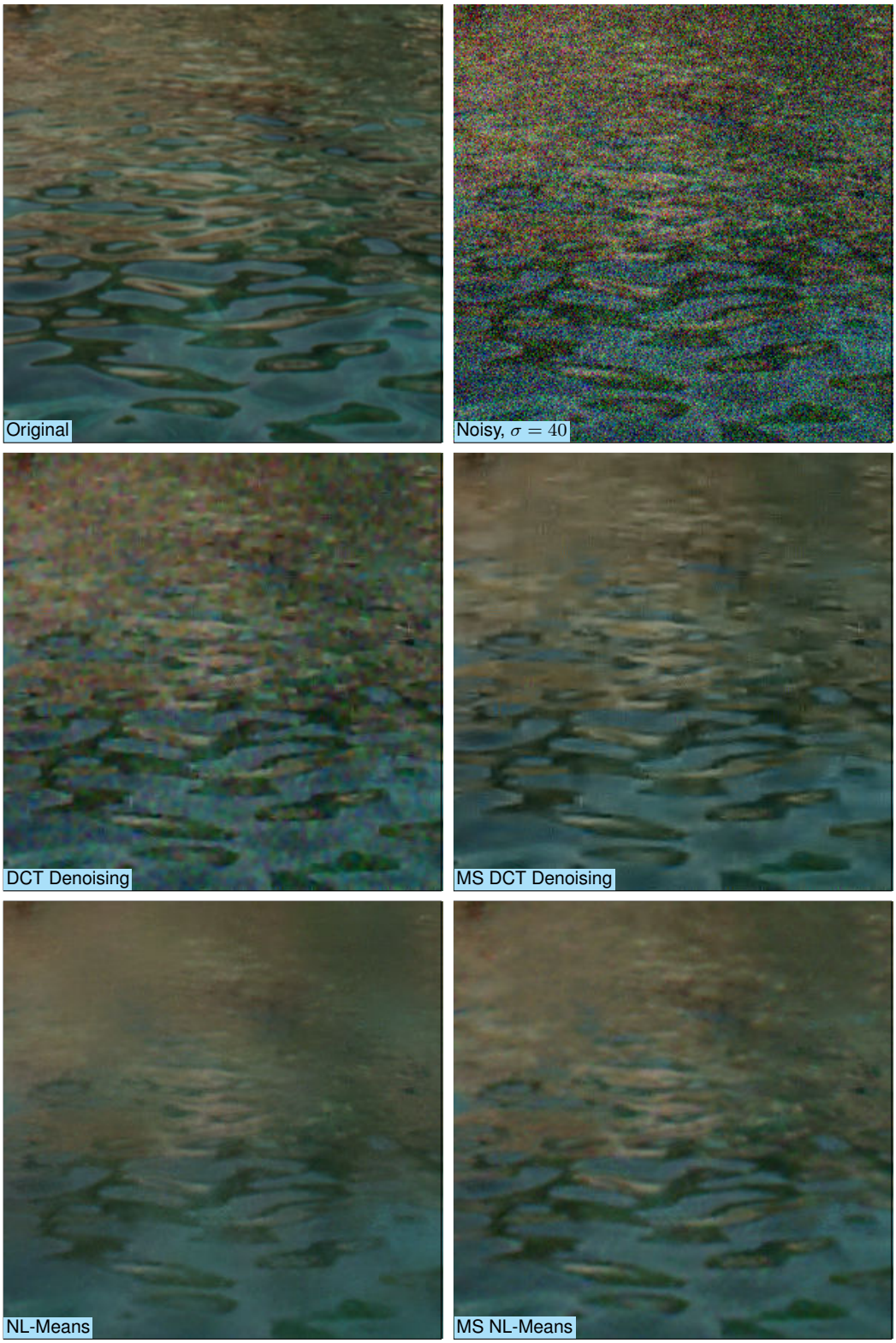


Figure 2.11 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

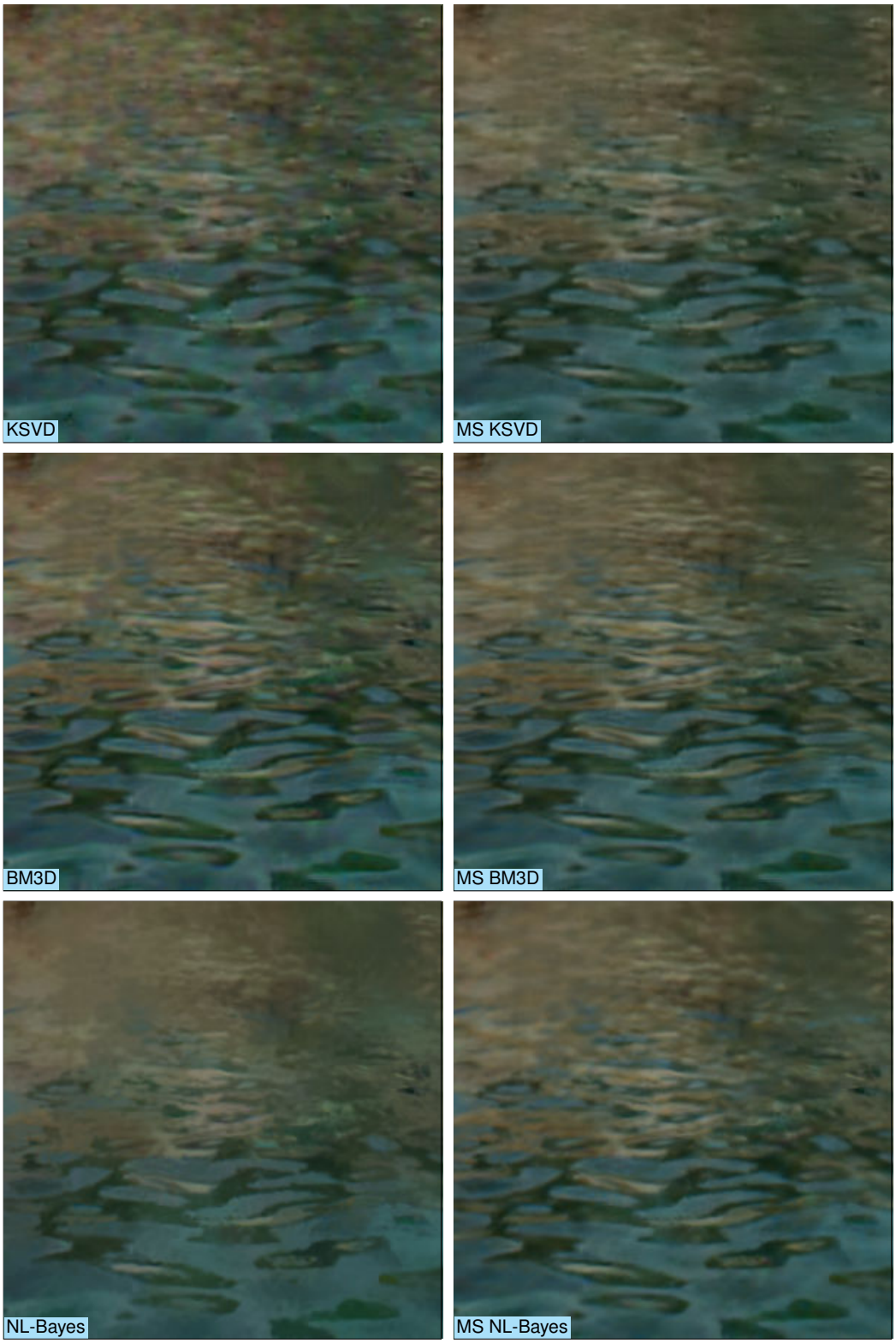


Figure 2.12 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

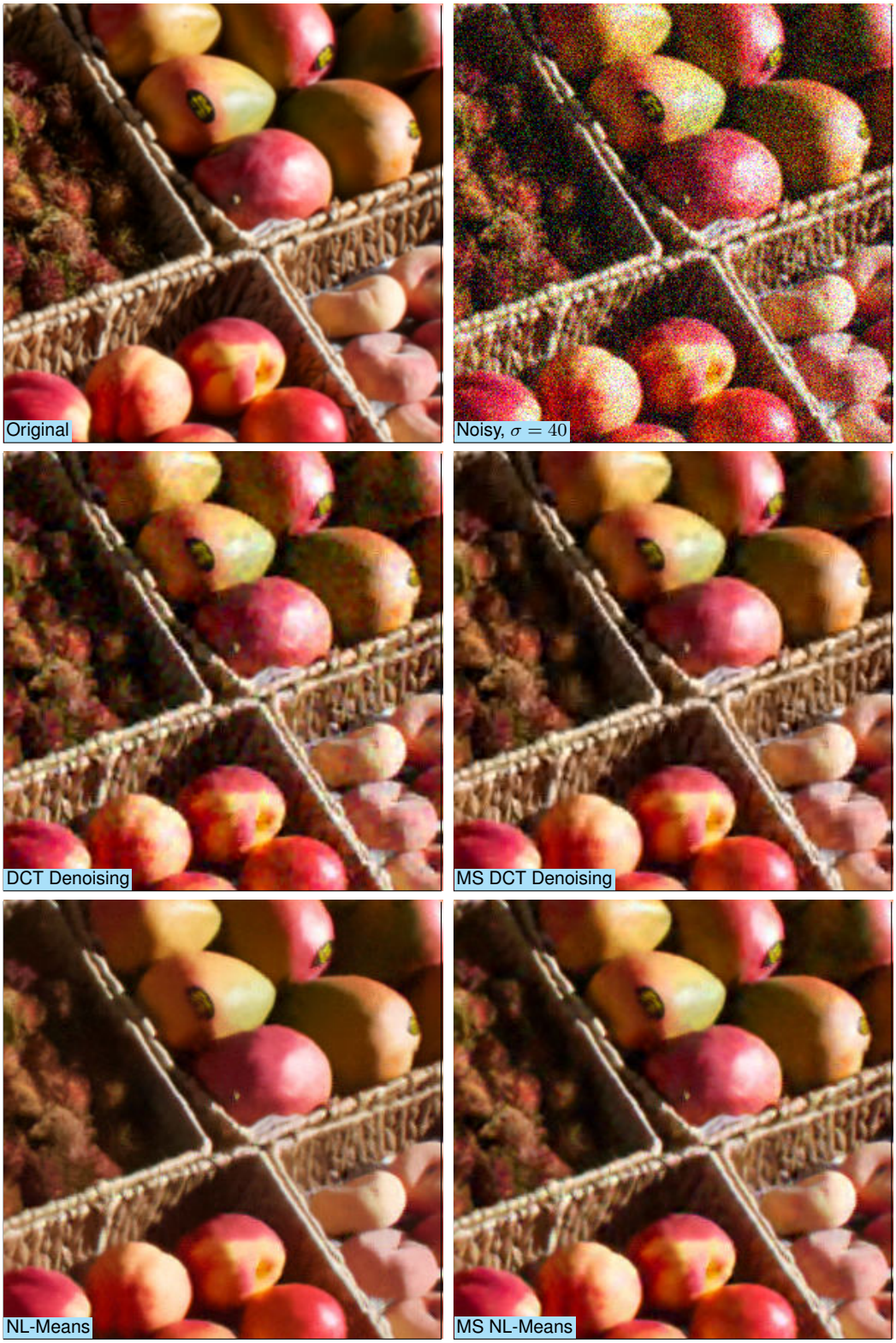


Figure 2.13 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

Figure 2.14 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

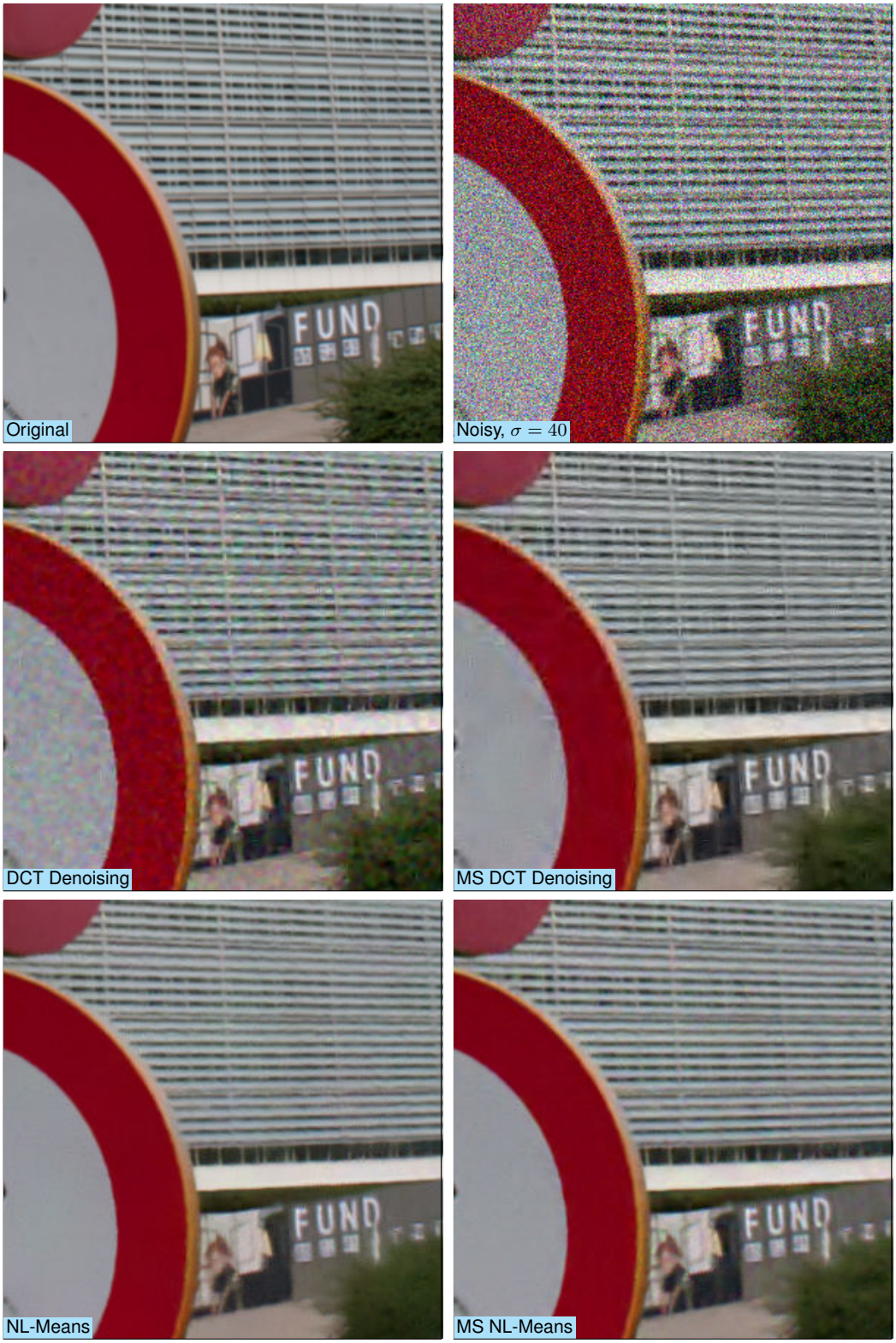


Figure 2.15 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

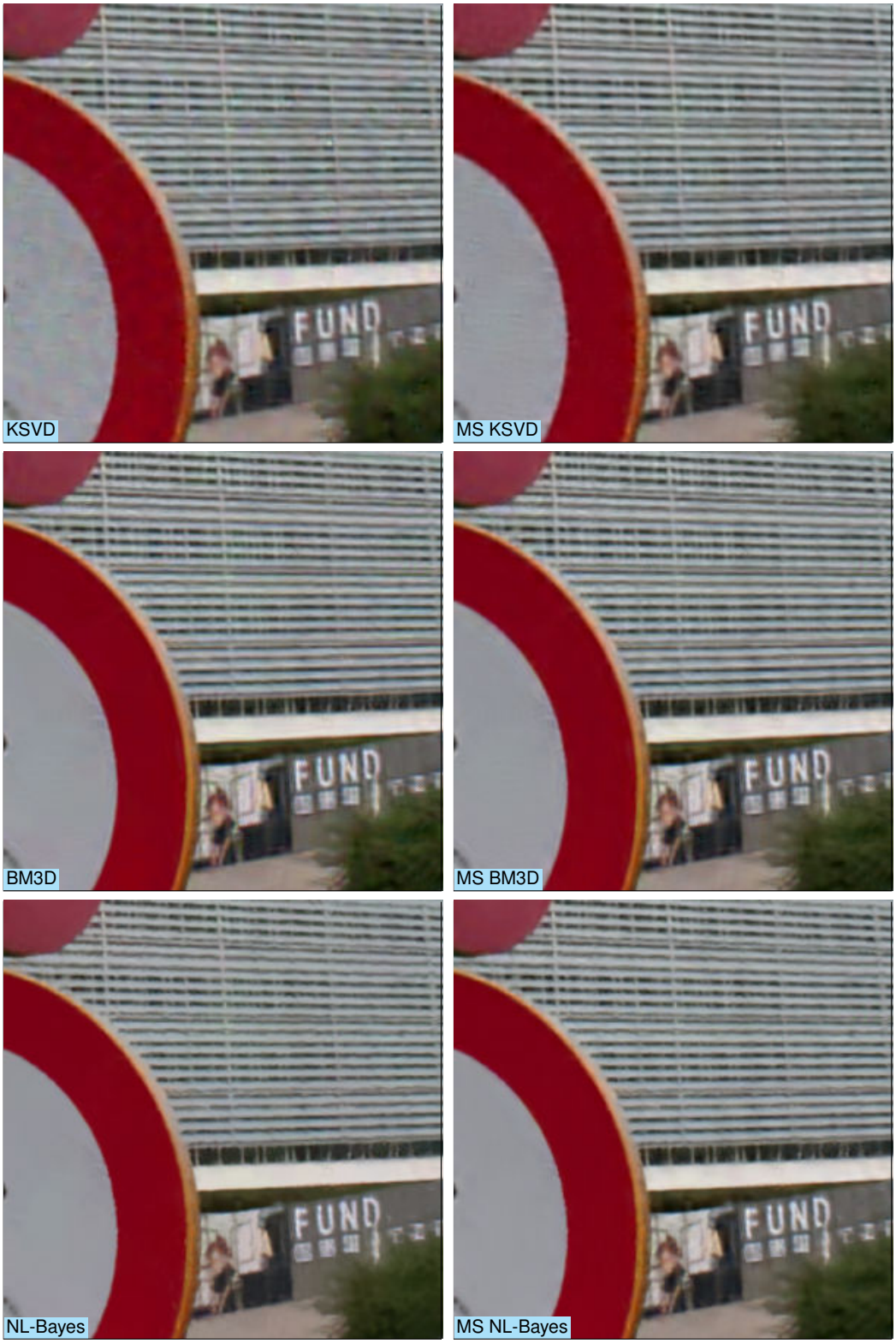


Figure 2.16 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

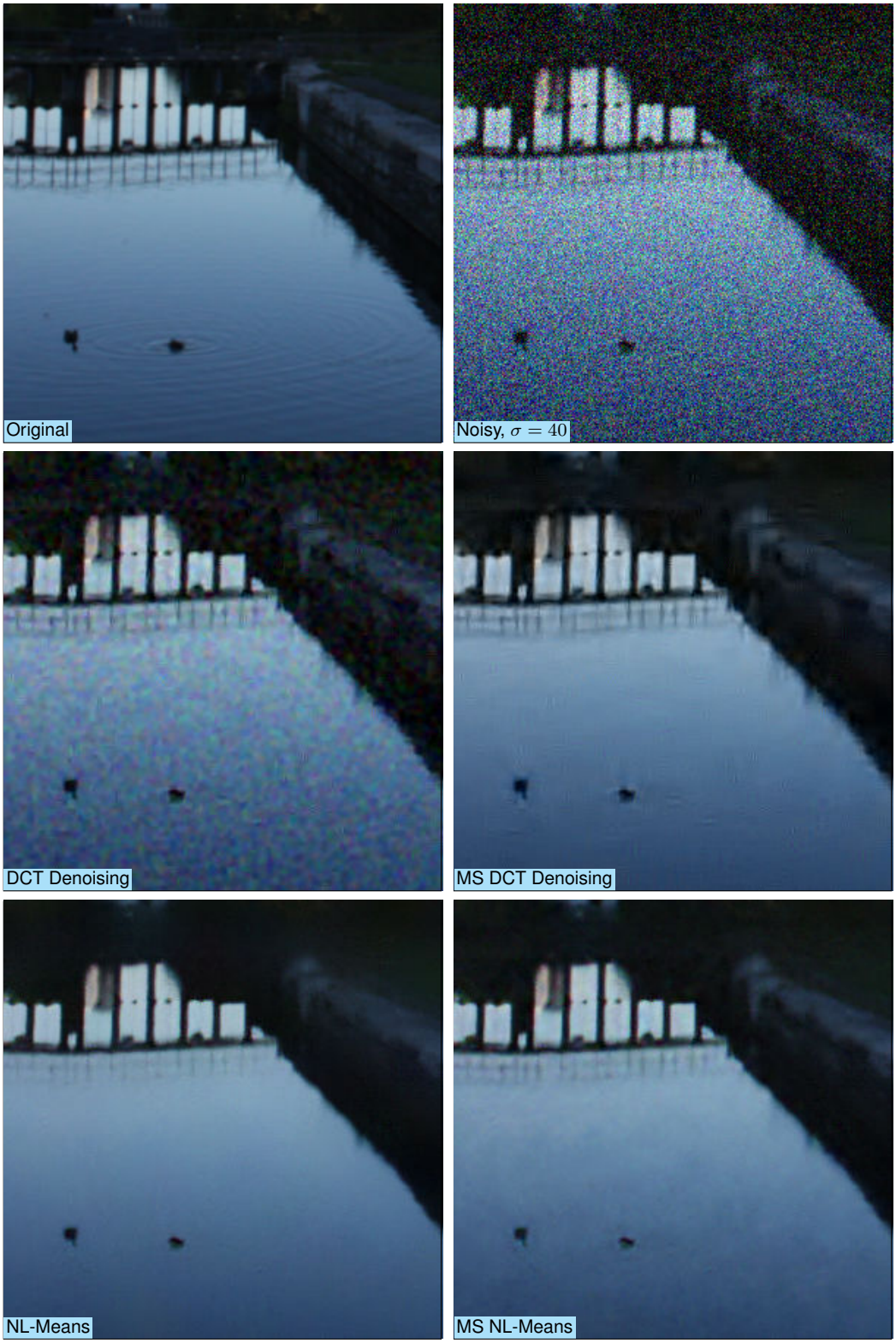


Figure 2.17 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

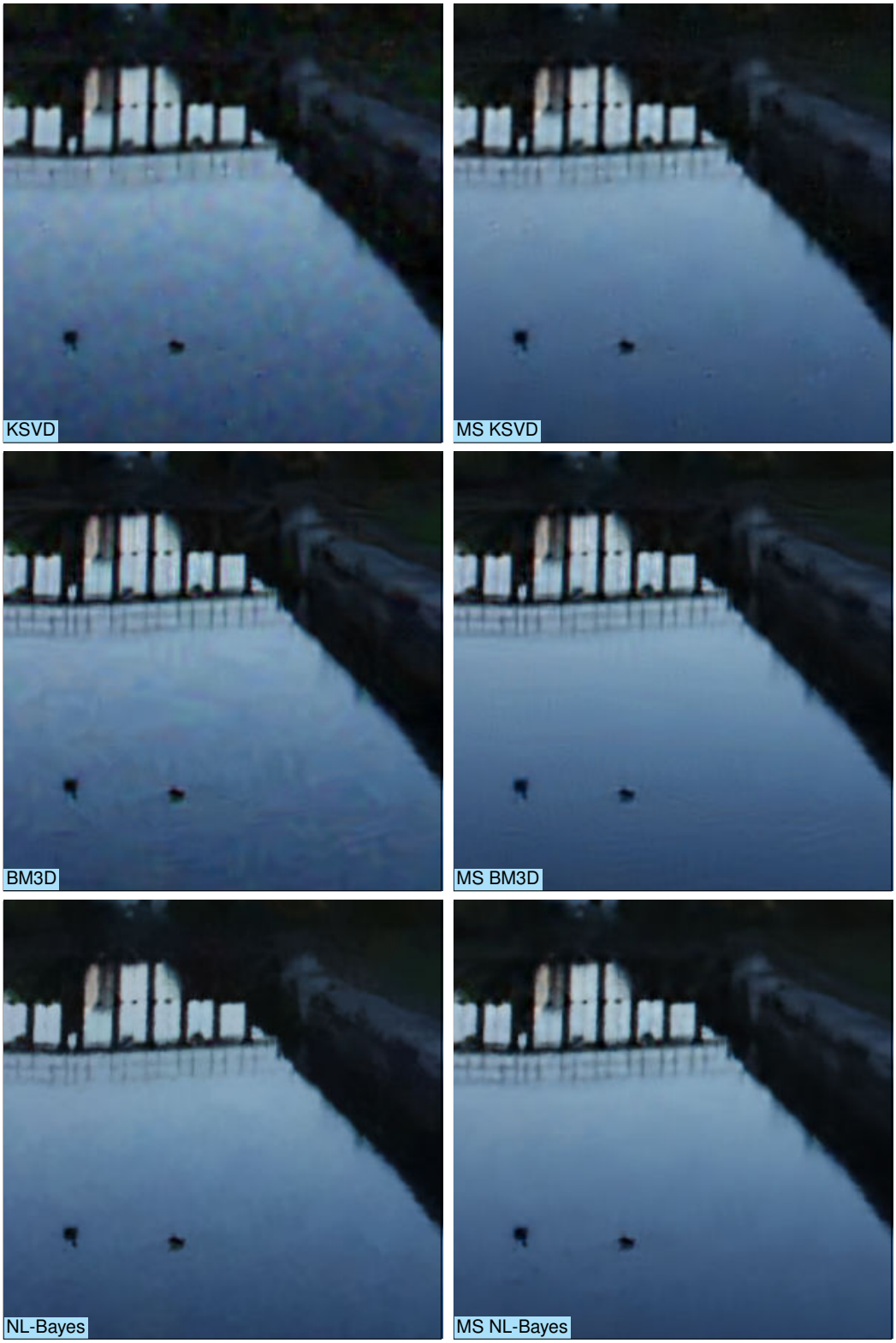


Figure 2.18 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

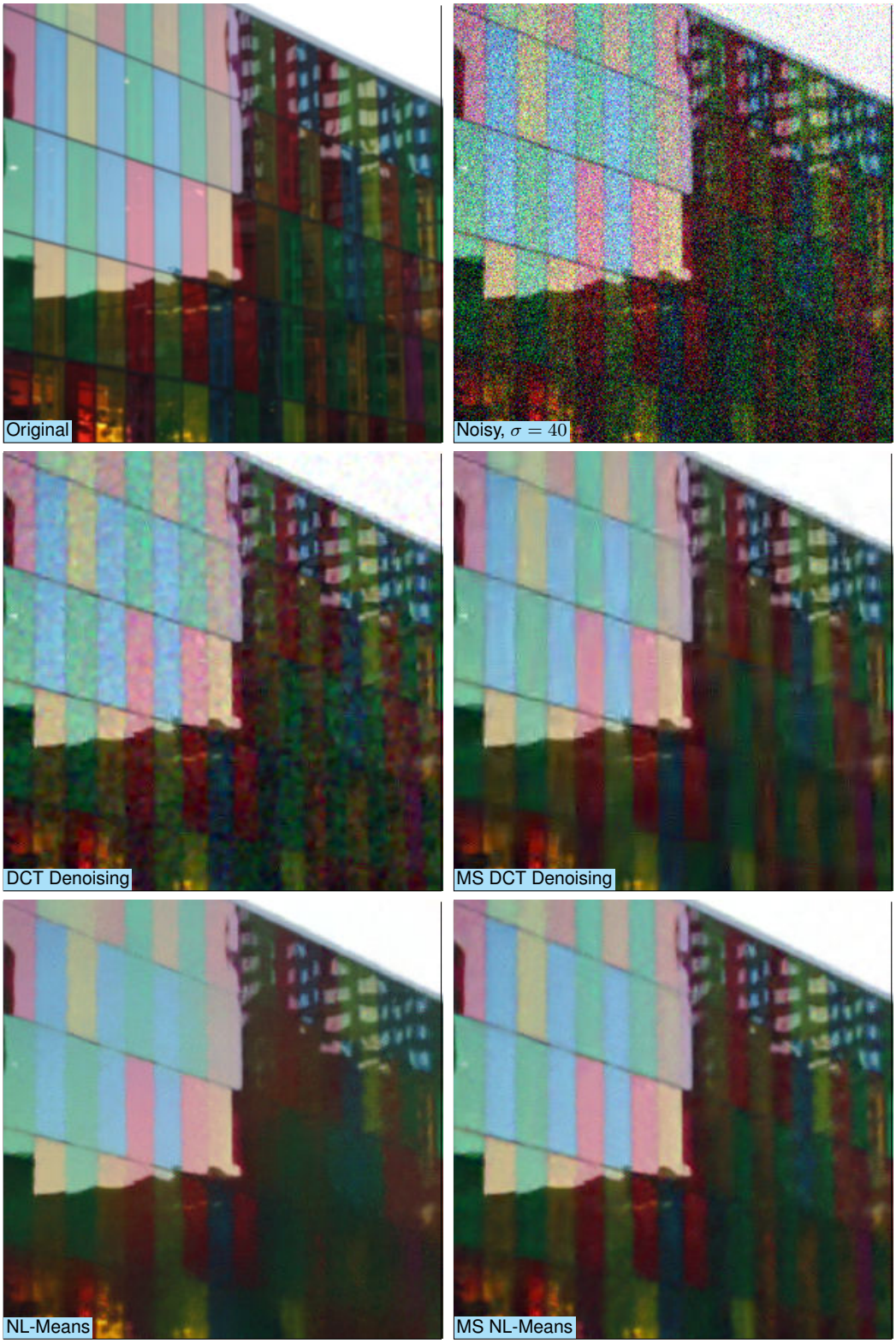


Figure 2.19 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

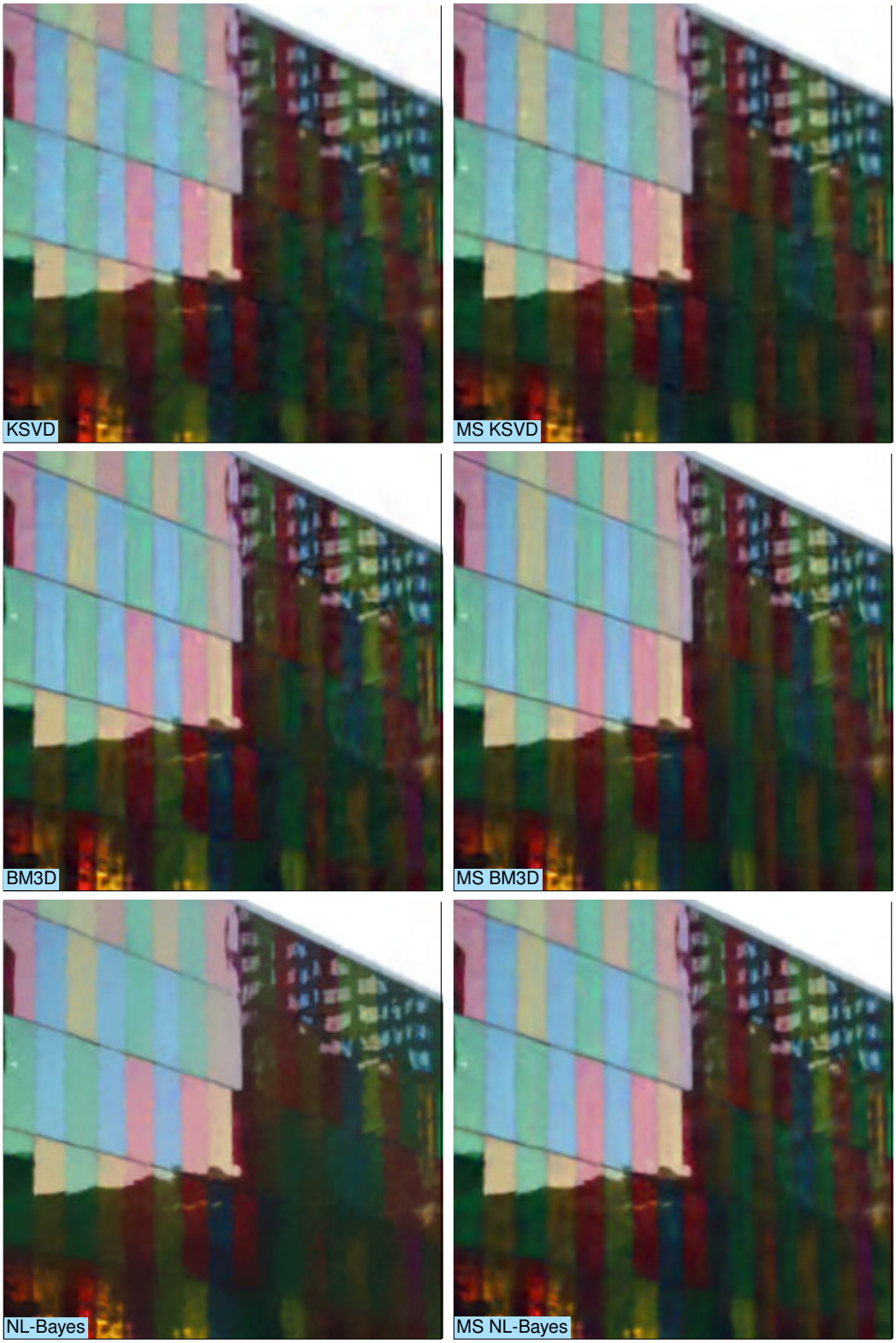


Figure 2.20 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

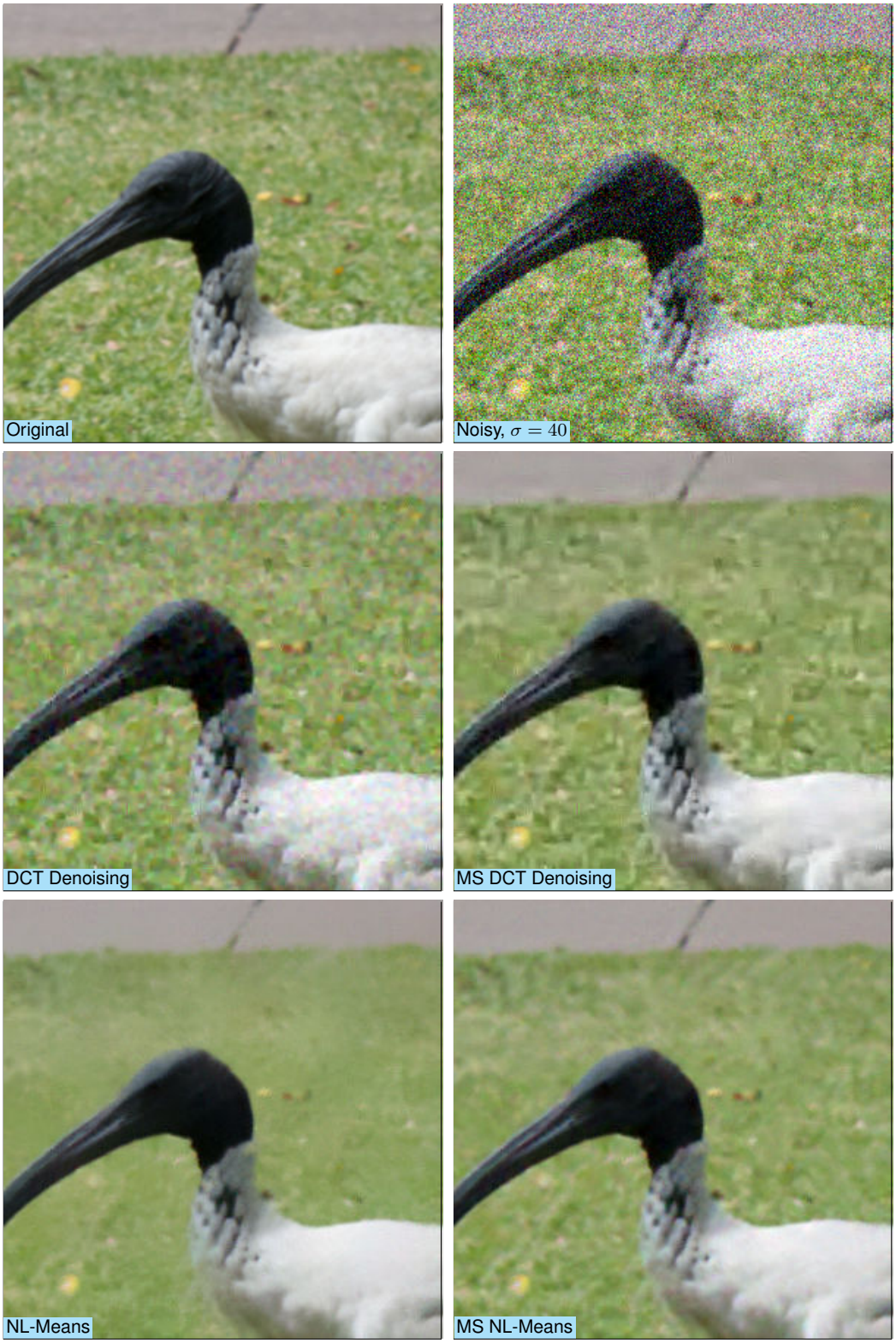


Figure 2.21 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

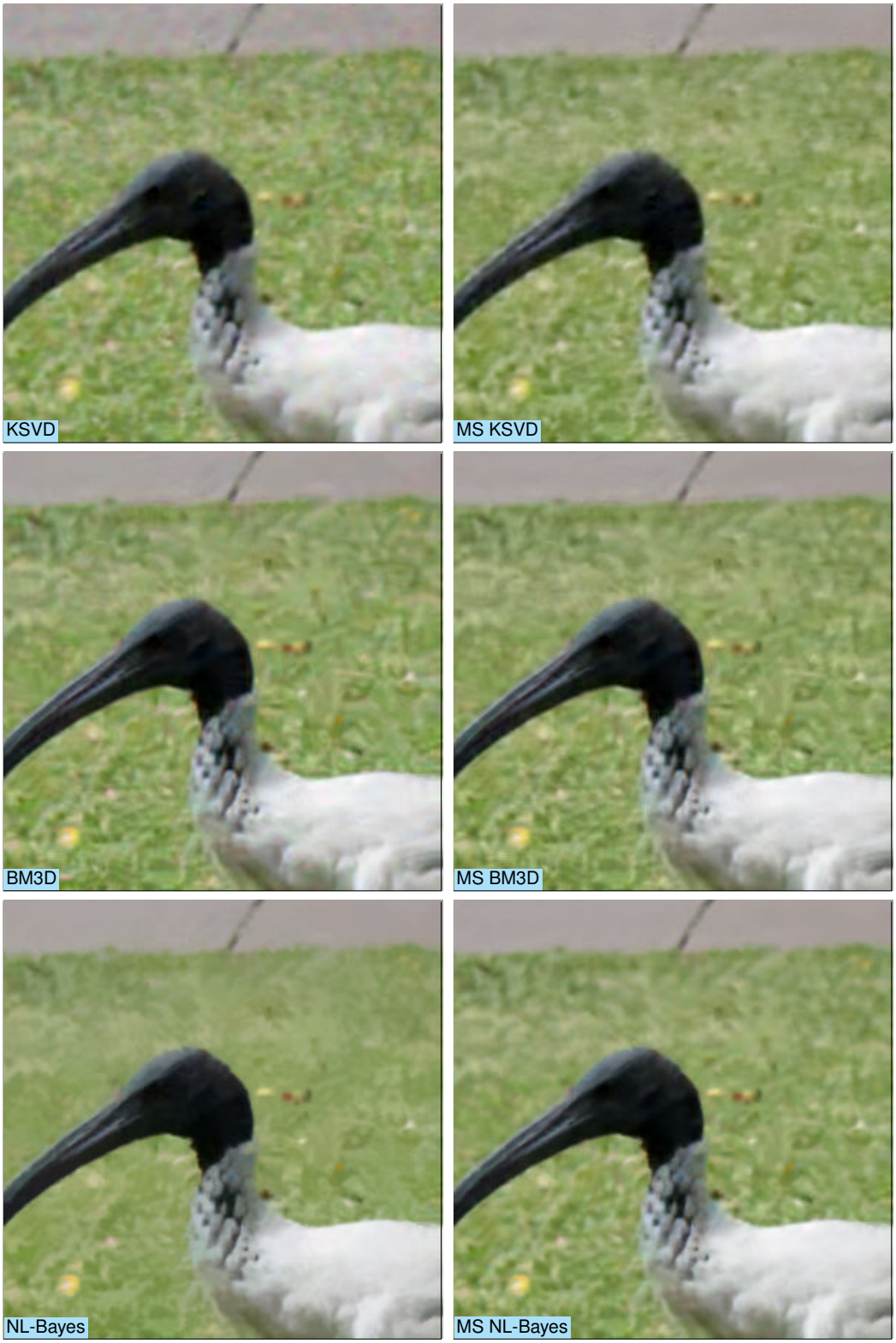


Figure 2.22 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

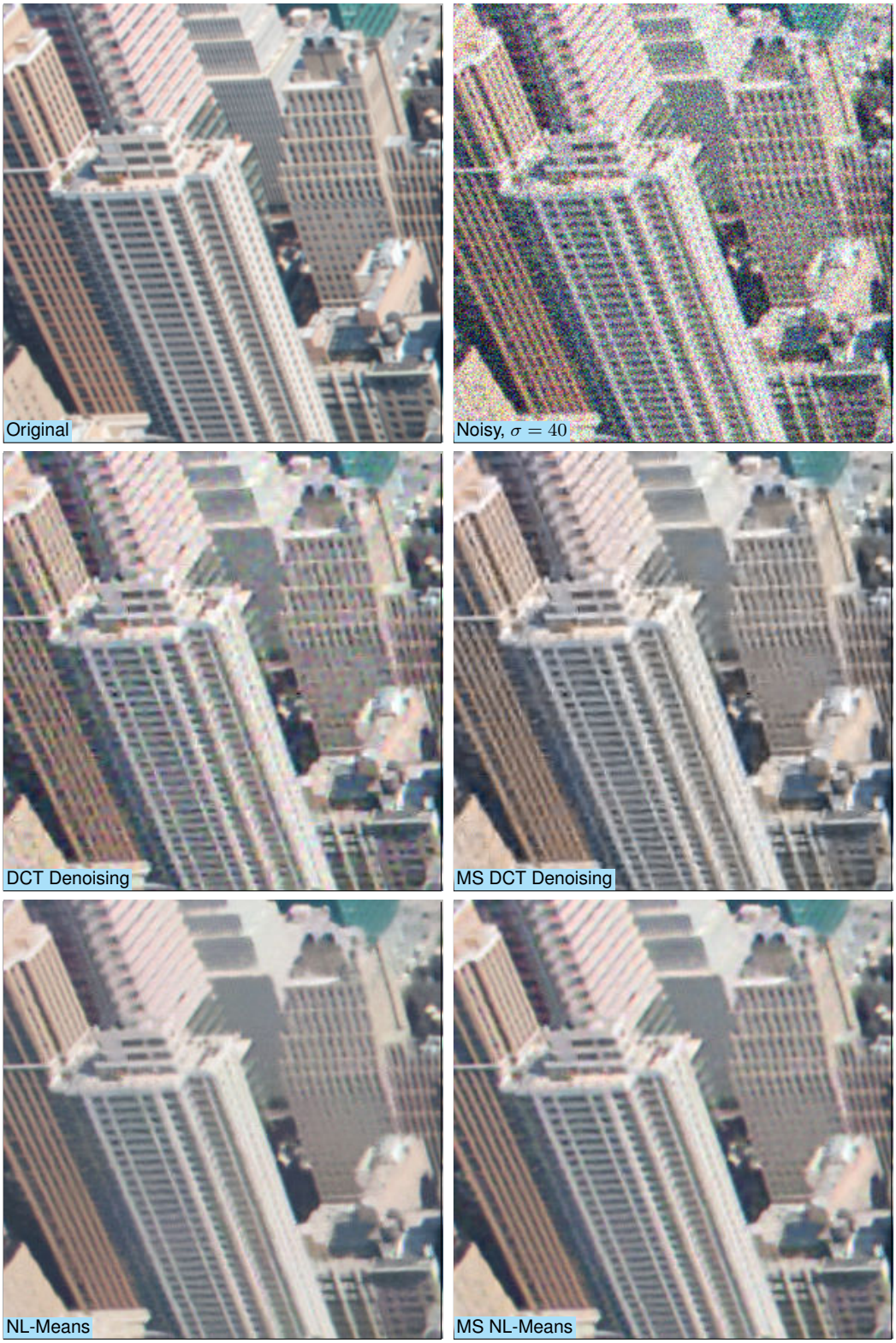


Figure 2.23 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

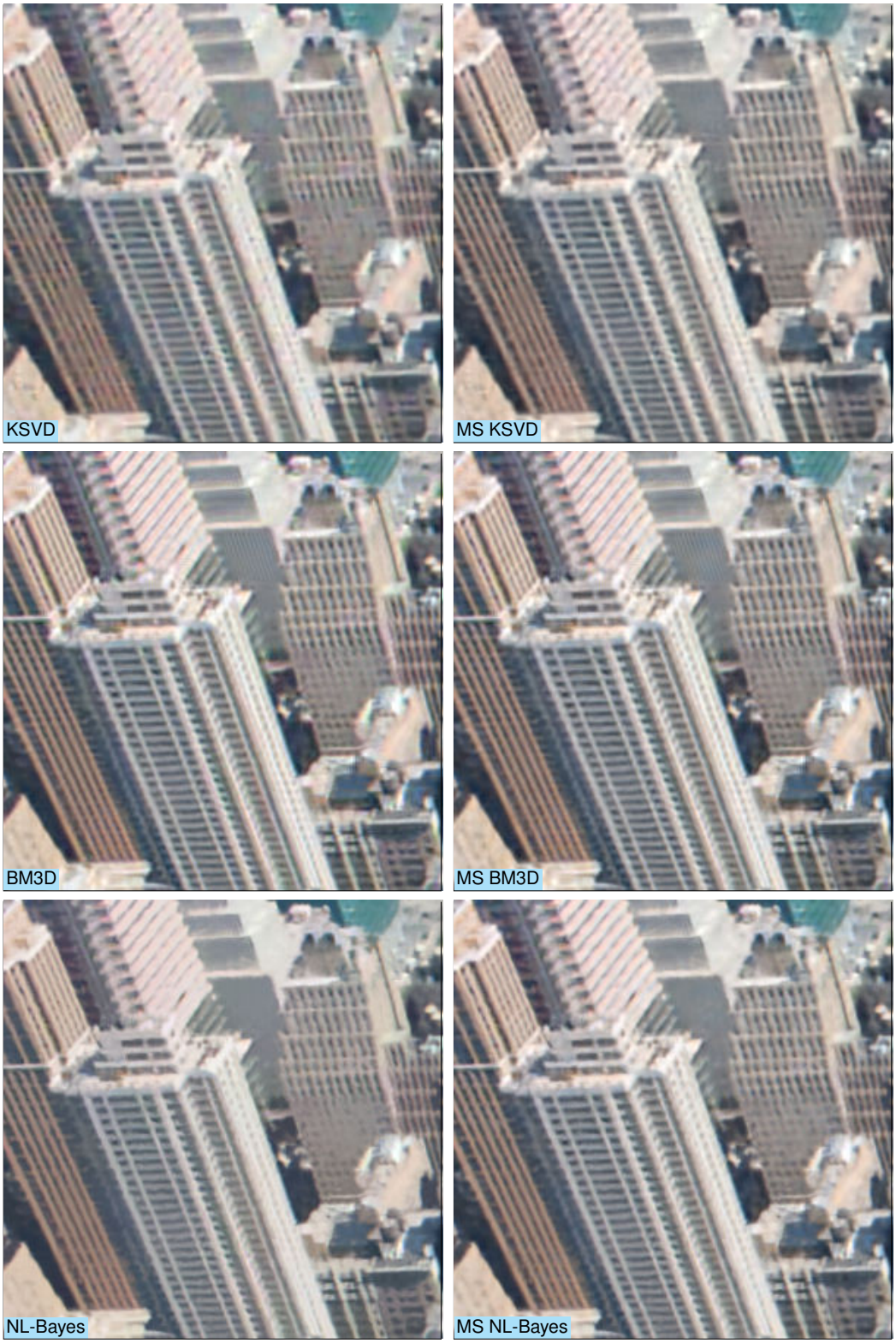


Figure 2.24 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

Figure 2.25 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

Figure 2.26 – Results of Single- and Multi-Scale algorithms applied to different images. The details are taken from the set of test images in Figure 2.10.

Chapter 3

Quality Criteria for Image Denoising

PSNR is the most used metric in the evaluation of denoising algorithm. In this chapter we try to find a metric that better discriminates the weakness and strength of each denoising algorithm, to allow a more fair comparison. Using this new metric, we find ways to combine the algorithms of Chapter 1 and Chapter 2 that uses the strength of the two. These methods are compared in terms of both the PSNR and the new metric.

Usually the performance of denoising algorithms is measured in terms of PSNR. Another famous quality index is the SSIM [58]. Unfortunately, neither of the two methods is closely correlated with the visual quality, since they are not sensitive enough to artifacts.

Different denoising algorithms behave differently on textures and flat areas of images. Measuring the PSNR of the result may be misleading, since an algorithm can provide a good result on the former and a bad one on the latter, or vice versa. Therefore, we propose to decouple the quality measure of a result in two values: the flat PSNR (or fPSNR) and the texture PSNR (or tPSNR).

In order to do that, it is necessary to define a method to detect the smooth areas of an image.

3.1 Detection of smooth areas

By *smooth area* we mean a part of the image that is regular and without texture. A simple way to define it is:

Definition 1. *A* smooth area *is an area of the image which is close to its affine regression surface.*

From this definition, we can find a *smoothness* of a noiseless image, simply constructing a local smooth approximation and computing its distance, as in Algorithm 3. The smoothness is computed for each pixel of each 16×16 patch as the distance between the pixels themselves and the regression plane of the patch. Then the global smoothness is computed by aggregating each patch by keeping, for each pixel, this minimum value. This is done in order to avoid false detections when, for example, a patch is composed of two different smooth parts. This aggregation allows each pixel to be declared smooth, since there exists a patch that includes the pixel itself without containing the edge.

This algorithm generates a result as the ones in Figures 3.1-3.2. Its pseudocode is listed in Algorithm 3.

Figure 3.1 – Example of smoothness map.

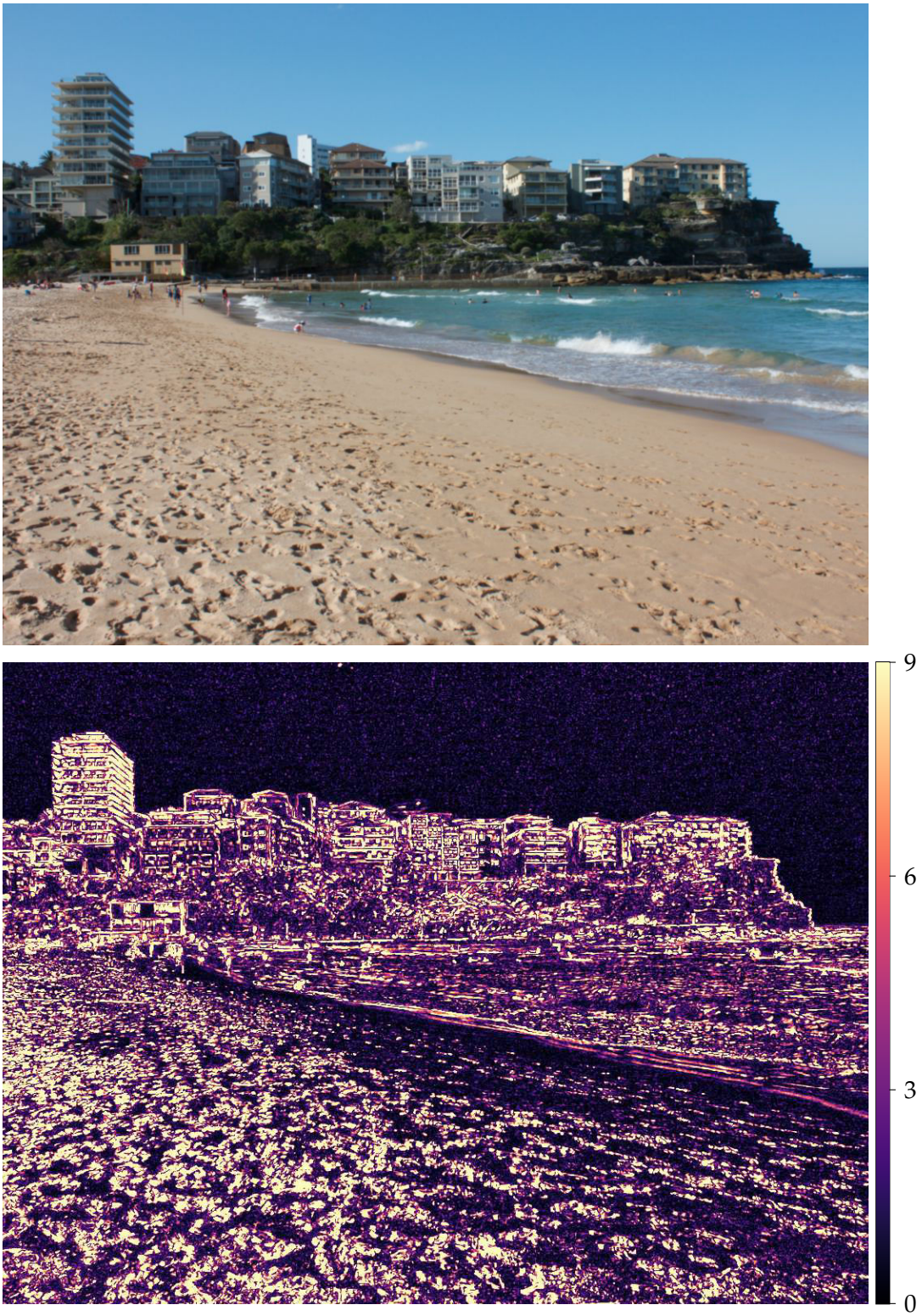


Figure 3.2 – Example of smoothness map.

Algorithm 3 Pseudo-code for the detection of Smooth Areas.

```
function SmoothDetector(image, level)
    result ← 2-D, one-channel image of +∞ with the same size as image
    for all pixels p ∈ image do
        y ← ExtractPatch(image, p)                          ▷ 16 × 16 patch
        P ← LeastSquaresRegression(y, level)
        d ← 2-D, one-channel patch with the same size as y
        for all pixels q ∈ y do
            d(q) ← ‖y(q) − P(q)‖_2                          ▷ Norm computed along channels
        end for
        result ← AggregateWithMin(p, result, d)
    end for
    return result
end function
```

3.2 fPSNR and tPSNR

With the mask of the smooth areas, one can easily define a quality measure for both the smooth and textured zones. Since the result of Algorithm 3 is a positive number, we must define a threshold in order to decide if an area of the image is smooth. To allow more flexibility, we define two thresholds, τ_1 and τ_2, with $\tau_1 < \tau_2$ and such that

- If the *smoothness* is $< \tau_1$ the area is considered smooth.
- If the *smoothness* is $> \tau_2$ the area is considered texture.
- If the *smoothness* is between τ_1 and τ_2, the area is considered "mixed".

In other words, we can define a *smoothness coefficient* as

$$sc(x) = \begin{cases} 1 & \text{if } x < \tau_1 \\ \dfrac{\tau_2 - x}{\tau_2 - \tau_1} & \text{if } \tau_1 < x < \tau_2 \\ 0 & \text{if } \tau_2 < x \end{cases} \tag{3.1}$$

Having identified the different areas of the image, we can easily (re-)define the quality measures of a reconstructed image $\tilde{\mathbf{y}}$ as:

$$fMSE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{\sum sc(x) \, \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2}{\sum sc(x)} \qquad fPSNR = -10 \log_{10} \frac{fMSE}{255} \tag{3.2}$$

$$tMSE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{\sum (1 - sc(x)) \, \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2}{\sum 1 - sc(x)} \qquad tPSNR = -10 \log_{10} \frac{tMSE}{255} \tag{3.3}$$

$$MSE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{\sum \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2}{\sum 1} \qquad PSNR = -10 \log_{10} \frac{MSE}{255} \tag{3.4}$$

The pseudo-code used to compute this measure is in Algorithm 4. It is worth noticing that, differently from the "standard" PSNR, both the fPSNR and the tPSNR are not symmetric functions, since the smoothness map needs to be computed on the original, clean image.

Algorithm 4 Pseudo-code for the computation of the the flat PSNR, the texture PNSR, and the PSNR.

function QUALITYMEASURES($\mathbf{y}$, $\tilde{\mathbf{y}}$)
  $fSum \leftarrow 0$
  $fW \leftarrow 0$
  $tSum \leftarrow 0$
  $tW \leftarrow 0$
  $\mathbf{sm} \leftarrow$ SMOOTHDETECTOR($\mathbf{y}$, linear)

  for all pixels $p \in$ **image do**
    $d \leftarrow \|\mathbf{y}(p) - \tilde{\mathbf{y}}(p)\|^2$ ▷ Average of square distances over channels
    $sc \leftarrow \begin{cases} 1 & \text{if } \mathbf{sm}(p) < \tau_1 \\ \dfrac{\tau_2 - \mathbf{sm}(p)}{\tau_2 - \tau_1} & \text{if } \tau_1 < x < \tau_2 \\ 0 & \text{if } \tau_2 < \mathbf{sm}(p) \end{cases}$
    $fSum \leftarrow fSum + sc \cdot d$
    $fW \leftarrow fW + sc$
    $tSum \leftarrow tSum + (1 - sc) \cdot d$
    $tW \leftarrow tW + (1 - sc)$
  end for
  $fMSE \leftarrow fSum/fW$
  $tMSE \leftarrow tSum/tW$
  $MSE \leftarrow (fSum + tSum)/(fW + tW)$

  $fPSNR \leftarrow -10 \log_{10}(fMSE/255^2)$
  $tPSNR \leftarrow -10 \log_{10}(tMSE/255^2)$
  $PSNR \leftarrow -10 \log_{10}(MSE/255^2)$
  return $(fPSNR, tPSNR, PSNR)$
end function

In addition, since the MSE can be seen as a convex combination of the $fMSE$ and the $tMSE$, we can conclude that the $PSNR$ always lies between the $fPSNR$ and the $tPSNR$.

3.3 Evaluation of existing denoising algorithms

We applied the metrics discussed in this chapter to the results of the most common denoising algorithms. The results are shown in Tables 3.1-3.2. The tables show the comparison between the "standard" algorithms and their multi-scale version, using the framework of Chapter 2. The comparison were made on the results of the algorithms in the dataset of Figure 2.10.

It is interesting to notice that the multiscale framework improves both flat areas and textures. The algorithms for which the framework is most effective are DCT Denoising and Non-Local Bayes. This is due to different reasons. DCT Denoising shows a big improvement since the patches used in the algorithm are 4×4. This means that the low frequency noise is left untouched. Non-Local Bayes is improved by the multiscale framework because

of its smooth results, that do not present artifacts. Even if the performance of Non-Local Bayes is comparable with the one of BM3D for their single-scale versions, BM3D presents artifacts along edges and on smooth areas, resulting in a more problematic recomposition, and in a relatively smaller improvement.

3.4 Combining different denoising techniques

Both Chapter 1 and Chapter 2 present a method to improve the result of other denoising algorithms. In this section, we present two different ways to combine both of them, in order to get a better performing denoising algorithm.

Since both DA3D and the Multiscale framework need a "baseline" algorithm to work, we chose to focus on Non-Local Bayes [27] and BM3D [7]. The first algorithm is particularly suited for both DA3D and the Multiscale framework, since it has relatively few artifacts and a smooth result, while we decided to included the second to offer a fair comparison with what is still considered the state-of-the-art.

In order to combine DA3D and the Multiscale framework, there are fundamentally two possibilities: apply DA3D first, or apply the Multiscale framework first. We present them both.

3.4.1 DA3D first, then Multiscale

The first method consists in applying the Multiscale framework on the noisy image, but applying DA3D to the finest level of the pyramid. The scheme for this method is shown in Figure 3.3. One might be tempted to apply DA3D to every layer of the pyramid. This does not work, since DA3D only works when the guide is smooth and does not have ringing. As explained in Chapter 2, the upper layers of the DCT pyramid have ringing around edges, therefore applying DA3D to those layers would actually worsen the result. We shall indicate this method as *da3d_ms* (da3d followed by a multiscale recomposition).

3.4.2 Multiscale as guide for DA3D

A second method consists in using the result of the multiscale algorithm as a guide for DA3D. This works well, since the guide provided to DA3D is better than the simple algorithm. An issue with this method is that, even if DA3D uses big 64×64 patches, some low-frequency noise remains in the result. To get rid of it, a second pass of the multiscale framework is needed. Since just the very low frequency noise is reintroduced, in this second step only the coarsest level of the pyramid is used for the recomposition. The scheme for this method is shown in Figure 3.4. We shall indicate this method as *ms_da3d* (multiscale, followed by da3d, followed by a second step of multiscale).

3.4.3 Results

Those two techniques are compared for the test set of Figure 3.5. These images are relatively noise-free, since they were all taken in conditions of good light and low ISO values with a good camera, and they were afterwards resized to eliminate the residual noise. They were

$\sigma = 10$	fPSNR	tPSNR	PSNR
Non-Local Means	40.03 dB	34.98 dB	36.59 dB
Multi-Scale	39.96 dB	35.19 dB	36.74 dB
Difference	-0.07 dB	+0.21 dB	+0.15 dB
DCT Denoising	39.48 dB	36.28 dB	37.43 dB
Multi-Scale	40.78 dB	36.39 dB	37.86 dB
Difference	+1.30 dB	+0.11 dB	+0.43 dB
KSVD	40.84 dB	36.81 dB	38.18 dB
Multi-Scale	40.84 dB	36.81 dB	38.18 dB
Difference	+0.00 dB	+0.00 dB	+0.00 dB
NL-Bayes (IPOL)	40.98 dB	36.46 dB	37.97 dB
Multi-Scale	41.01 dB	36.72 dB	38.17 dB
Difference	+0.03 dB	+0.25 dB	+0.20 dB
Non-Local Bayes	**41.72 dB**	37.05 dB	38.58 dB
Multi-Scale	41.70 dB	**37.22 dB**	**38.71 dB**
Difference	-0.02 dB	+0.17 dB	+0.13 dB
BM3D	41.48 dB	37.13 dB	38.59 dB
Multi-Scale	41.48 dB	37.13 dB	38.59 dB
Difference	+0.00 dB	+0.00 dB	+0.00 dB

$\sigma = 20$	fPSNR	tPSNR	PSNR
Non-Local Means	36.46 dB	30.90 dB	32.59 dB
Multi-Scale	36.67 dB	31.59 dB	33.20 dB
Difference	+0.21 dB	+0.69 dB	+0.61 dB
DCT Denoising	34.79 dB	32.03 dB	33.04 dB
Multi-Scale	37.26 dB	32.50 dB	34.05 dB
Difference	+2.47 dB	+0.47 dB	+1.00 dB
KSVD	36.56 dB	32.75 dB	34.05 dB
Multi-Scale	37.79 dB	32.93 dB	34.49 dB
Difference	+1.23 dB	+0.18 dB	+0.43 dB
NL-Bayes (IPOL)	37.79 dB	32.92 dB	34.49 dB
Multi-Scale	38.09 dB	33.16 dB	34.75 dB
Difference	+0.30 dB	+0.24 dB	+0.25 dB
Non-Local Bayes	38.58 dB	33.05 dB	34.75 dB
Multi-Scale	**38.65 dB**	**33.45 dB**	**35.09 dB**
Difference	+0.07 dB	+0.41 dB	+0.35 dB
BM3D	38.19 dB	33.41 dB	34.97 dB
Multi-Scale	38.31 dB	33.38 dB	34.96 dB
Difference	+0.12 dB	-0.04 dB	-0.01 dB

Table 3.1 – Showing fPSNR, tPSNR and PSNR for various algorithms. The normal version is compared with the multiscale version using the framework of Chapter 2.

$\sigma = 40$	fPSNR	tPSNR	PSNR
Non-Local Means	33.22 dB	27.05 dB	28.84 dB
Multi-Scale	33.93 dB	28.27 dB	29.98 dB
Difference	+0.70 dB	+1.22 dB	+1.14 dB
DCT Denoising	29.67 dB	27.64 dB	28.42 dB
Multi-Scale	33.76 dB	28.83 dB	30.40 dB
Difference	+4.08 dB	+1.19 dB	+1.98 dB
KSVD	33.68 dB	28.90 dB	30.42 dB
Multi-Scale	34.91 dB	29.26 dB	30.97 dB
Difference	+1.23 dB	+0.36 dB	+0.55 dB
NL-Bayes (IPOL)	34.04 dB	29.01 dB	30.61 dB
Multi-Scale	34.94 dB	29.56 dB	31.23 dB
Difference	+0.89 dB	+0.55 dB	+0.62 dB
Non-Local Bayes	35.38 dB	29.06 dB	30.88 dB
Multi-Scale	**35.68 dB**	**29.91 dB**	**31.65 dB**
Difference	+0.30 dB	+0.85 dB	+0.77 dB
BM3D	34.71 dB	**29.79 dB**	31.37 dB
Multi-Scale	35.11 dB	**29.80 dB**	**31.46 dB**
Difference	+0.40 dB	+0.01 dB	+0.09 dB

$\sigma = 70$	fPSNR	tPSNR	PSNR
Non-Local Means	30.63 dB	24.08 dB	25.93 dB
Multi-Scale	31.58 dB	25.74 dB	27.48 dB
Difference	+0.95 dB	+1.66 dB	+1.55 dB
DCT Denoising	25.28 dB	23.85 dB	24.42 dB
Multi-Scale	30.75 dB	26.04 dB	27.56 dB
Difference	+5.47 dB	+2.19 dB	+3.13 dB
KSVD	31.26 dB	26.04 dB	27.65 dB
Multi-Scale	32.71 dB	26.52 dB	28.32 dB
Difference	+1.45 dB	+0.47 dB	+0.67 dB
NL-Bayes (IPOL)	33.06 dB	26.75 dB	28.57 dB
Multi-Scale	33.12 dB	**27.16 dB**	**28.93 dB**
Difference	+0.06 dB	+0.41 dB	+0.36 dB
Non-Local Bayes	32.50 dB	26.00 dB	27.83 dB
Multi-Scale	**33.39 dB**	**27.22 dB**	**29.02 dB**
Difference	+0.89 dB	+1.22 dB	+1.19 dB
BM3D	32.02 dB	**27.07 dB**	28.65 dB
Multi-Scale	32.71 dB	**27.15 dB**	**28.84 dB**
Difference	+0.70 dB	+0.07 dB	+0.19 dB

Table 3.2 – Showing fPSNR, tPSNR and PSNR for various algorithms. The normal version is compared with the multiscale version using the framework of Chapter 2.

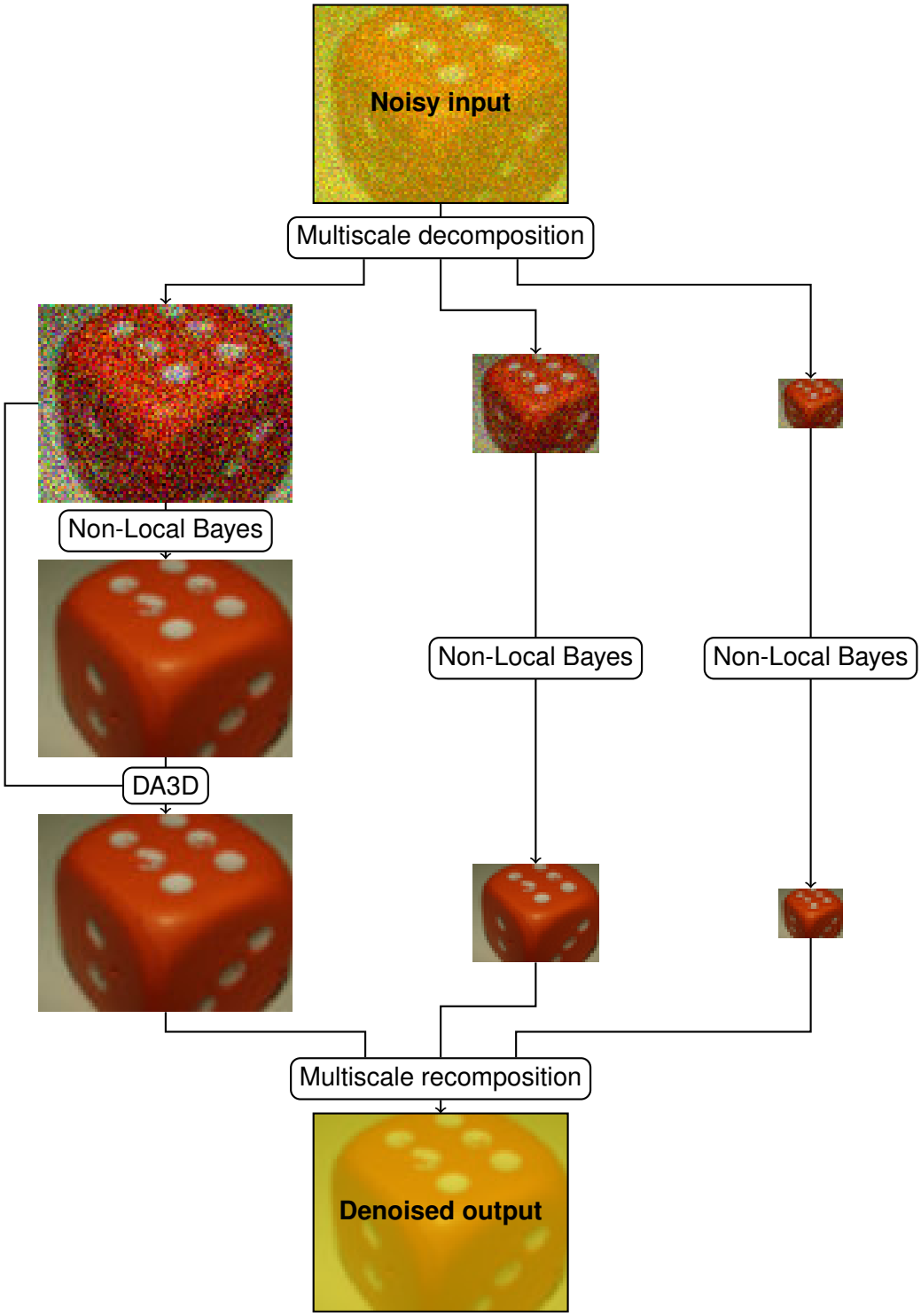


Figure 3.3 – Scheme of the *da3d_ms* denoising method that combines Non-Local Bayes, DA3D and the Multi-Scale Framework. The values for the multiscale recompositions are the one used for Non-Local Bayes in Chapter 2 (0.5 for $\sigma = 10$ and $\sigma = 20$, 0.6 for $\sigma = 40$ and 0.7 for $\sigma = 70$). DA3D is applied using the result of Non-Local Bayes as guide only on the finer scale. Applying DA3D to coarser scale degrades the results due to the presence of ringing in the DCT pyramid.

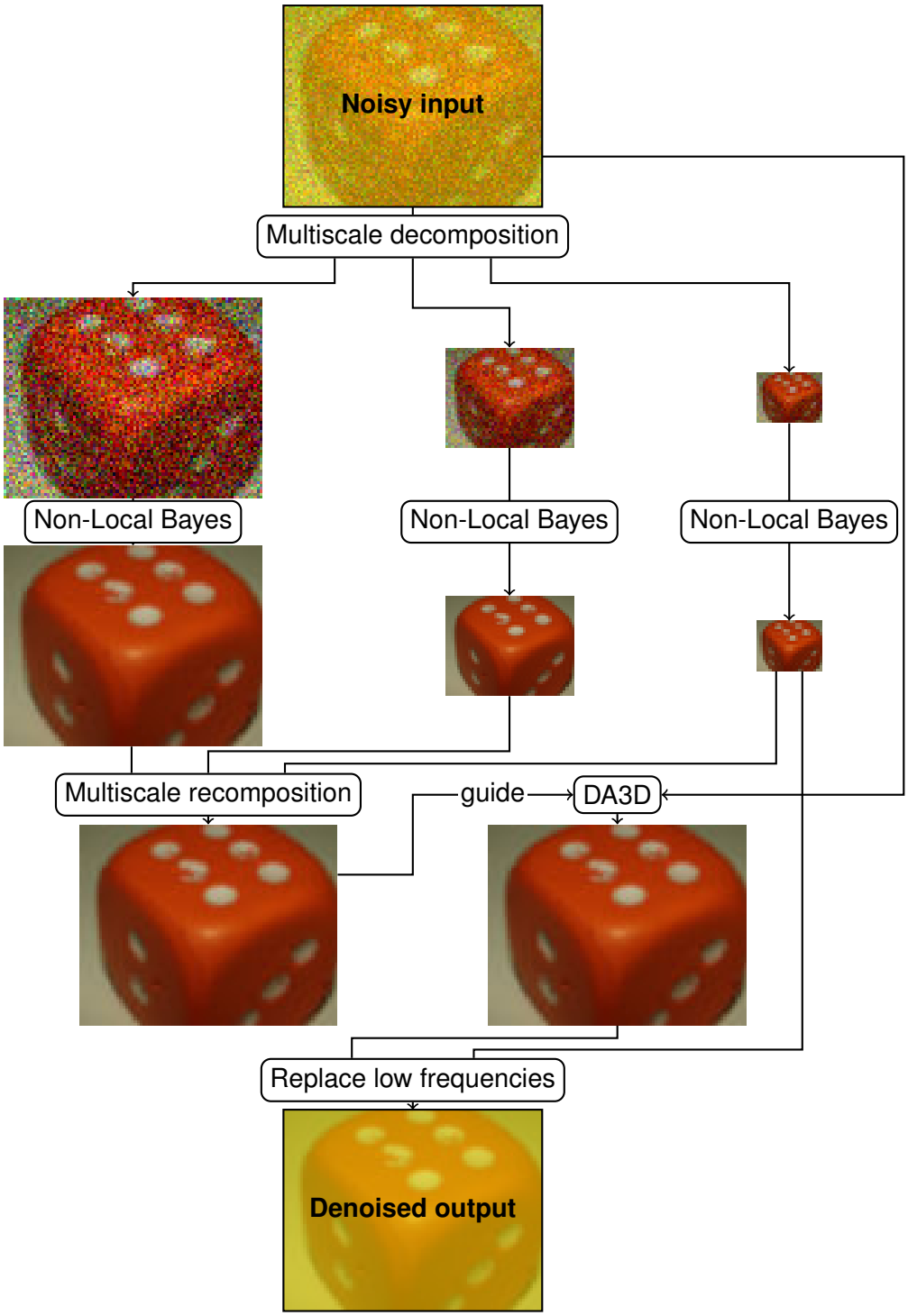


Figure 3.4 – Scheme of the *ms_da3d* denoising method that combines Non-Local Bayes, DA3D and the Multi-Scale Framework. The values for the first multiscale recompositions are the one used for Non-Local Bayes in Chapter 2 (0.5 for $\sigma = 10$ and $\sigma = 20$, 0.6 for $\sigma = 40$ and 0.7 for $\sigma = 70$). The second recomposition uses the same parameters, but it only merge the coarser level. DA3D is applied using the result of Multiscale Non-Local Bayes as guide, and the lower frequencies of the result of the first part are reinjected in the final result.

Figure 3.5 – Images used in the experiments.

taken from RAW sources, therefore they do not present compression artifacts. All images have about 1.4 MPixels after the resizing.

The results for different levels of noise using Non-Local Bayes as the base algorithm are shown in Tables 3.3-3.6. These results are worth commenting: it is clear that DA3D is the best method to improve the results on the flat areas, while only marginally improving the results on textures. Conversely, using the multiscale framework boosts the result quality on textures, with a small improvement on flat areas. This fact is not obvious from the PSNR values alone, and it shows that a finer metric allows to appreciate different aspects of denoising results. The two combined methods, *da3d_ms* and *ms_da3d*, balance the strength of the two algorithms. Even if they do not excel in neither flat areas or textures, they present good results in all areas, and in general a more pleasant visual result. In particular, we observed that when the noise has standard deviation of 10 or 20, the best compromise is achieved by *da3d_ms*, while for noise of 40 and 70 the best results come from *ms_da3d*. This also confirms the fact that DA3D was conceived to treat medium-to-high noise levels, so applying it at the end gives the best results in this case.

The same results using BM3D as the base algorithm are presented in Tables 3.7-3.10. In this case one can see that, since the output of BM3D contains artifacts, the multiscale framework fails to improve the results significantly. Therefore, the best result (except maybe for small values of noise, like $\sigma = 10$) are attained by DA3D.

Introducing new metrics, as FPSNR and TPSNR, is needed in order to evaluate the results of denoising algorithms more objectively. Images 3.6-3.7 show some examples of results in details that are commonly "hard" to denoise, to show how the algorithms behave in these cases.

All the examples are available at `http://dev.ipol.im/~pierazzo/CombinedDenoising/`.

FPSNR					
$\sigma = 10$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	46.79 dB	47.96 dB	47.89 dB	**48.08 dB**	**48.24 dB**
IMG_7113	**41.58 dB**	**41.56 dB**	**41.65 dB**	**41.55 dB**	**41.65 dB**
IMG_7626	45.02 dB	45.03 dB	**45.29 dB**	45.04 dB	**45.27 dB**
IMG_7627	40.91 dB	40.79 dB	**41.14 dB**	40.84 dB	**40.95 dB**
IMG_7673	42.49 dB	**42.50 dB**	**42.68 dB**	**42.51 dB**	**42.69 dB**
IMG_7739	**40.48 dB**	**40.36 dB**	**40.45 dB**	**40.33 dB**	**40.42 dB**
IMG_8275	**39.42 dB**	39.26 dB	**39.51 dB**	39.28 dB	**39.49 dB**
IMG_8336	43.77 dB	**43.91 dB**	**44.09 dB**	**43.90 dB**	**44.02 dB**
IMG_8339	42.90 dB	42.91 dB	**43.16 dB**	42.91 dB	41.20 dB
IMG_8586	**40.62 dB**	**40.62 dB**	**40.67 dB**	**40.59 dB**	**40.70 dB**
Average	42.40 dB	**42.49 dB**	**42.65 dB**	**42.50 dB**	**42.46 dB**

TPSNR					
$\sigma = 10$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	39.62 dB	**40.08 dB**	39.60 dB	**39.98 dB**	**39.89 dB**
IMG_7113	**35.98 dB**	**36.09 dB**	35.81 dB	**35.99 dB**	**35.92 dB**
IMG_7626	**38.53 dB**	**38.68 dB**	38.32 dB	**38.56 dB**	**38.49 dB**
IMG_7627	**36.39 dB**	**36.44 dB**	36.20 dB	**36.36 dB**	**36.28 dB**
IMG_7673	**36.33 dB**	**36.50 dB**	36.14 dB	**36.40 dB**	**36.30 dB**
IMG_7739	36.54 dB	**36.79 dB**	36.45 dB	**36.71 dB**	**36.61 dB**
IMG_8275	**36.43 dB**	**36.51 dB**	36.29 dB	**36.44 dB**	**36.41 dB**
IMG_8336	**37.46 dB**	**37.56 dB**	37.29 dB	**37.45 dB**	**37.40 dB**
IMG_8339	39.71 dB	**39.96 dB**	39.74 dB	**39.97 dB**	39.16 dB
IMG_8586	36.22 dB	**36.42 dB**	36.00 dB	**36.30 dB**	36.19 dB
Average	**37.32 dB**	**37.50 dB**	37.18 dB	**37.42 dB**	37.26 dB

PSNR					
$\sigma = 10$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	45.34 dB	**46.25 dB**	46.02 dB	**46.28 dB**	**46.34 dB**
IMG_7113	**38.07 dB**	**38.15 dB**	**37.95 dB**	**38.07 dB**	**38.04 dB**
IMG_7626	**41.33 dB**	**41.44 dB**	41.24 dB	**41.35 dB**	**41.36 dB**
IMG_7627	**37.43 dB**	**37.46 dB**	**37.30 dB**	**37.40 dB**	**37.34 dB**
IMG_7673	**38.44 dB**	**38.57 dB**	38.32 dB	**38.50 dB**	**38.46 dB**
IMG_7739	**37.57 dB**	**37.76 dB**	37.50 dB	**37.69 dB**	**37.62 dB**
IMG_8275	**37.09 dB**	**37.13 dB**	**36.99 dB**	**37.07 dB**	**37.08 dB**
IMG_8336	**40.22 dB**	**40.34 dB**	**40.18 dB**	**40.26 dB**	**40.25 dB**
IMG_8339	**41.11 dB**	**41.27 dB**	**41.21 dB**	**41.28 dB**	40.12 dB
IMG_8586	**37.70 dB**	**37.86 dB**	37.55 dB	**37.76 dB**	**37.70 dB**
Average	**39.43 dB**	**39.62 dB**	**39.43 dB**	**39.57 dB**	**39.43 dB**

Table 3.3 – Results using Non-Local Bayes as base algorithm, noise with $\sigma = 10$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**. From the results is clear that DA3D is the best method to improve the results on the flat areas, while applying the multiscale frameworks leads to an improvement in the textures of the images. For this level of noise, the combination *da3d_ms* offers a good combination of texture quality and flat areas quality.

FPSNR					
$\sigma = 20$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	43.17 dB	44.73 dB	44.67 dB	44.94 dB	**45.22 dB**
IMG_7113	38.59 dB	38.57 dB	**38.79 dB**	**38.60 dB**	**38.75 dB**
IMG_7626	42.22 dB	42.35 dB	**42.78 dB**	42.36 dB	**42.73 dB**
IMG_7627	37.82 dB	37.87 dB	**38.37 dB**	37.95 dB	**38.21 dB**
IMG_7673	39.70 dB	39.72 dB	**40.03 dB**	39.77 dB	**40.03 dB**
IMG_7739	37.85 dB	**37.87 dB**	**38.06 dB**	**37.89 dB**	**37.97 dB**
IMG_8275	36.23 dB	36.12 dB	**36.56 dB**	36.20 dB	**36.48 dB**
IMG_8336	40.72 dB	40.90 dB	**41.18 dB**	40.95 dB	**41.20 dB**
IMG_8339	39.28 dB	**39.93 dB**	**40.05 dB**	**40.00 dB**	39.06 dB
IMG_8586	37.63 dB	**37.83 dB**	**37.91 dB**	**37.82 dB**	**37.93 dB**
Average	39.32 dB	39.59 dB	**39.84 dB**	**39.65 dB**	**39.76 dB**

TPSNR					
$\sigma = 20$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	35.21 dB	**36.13 dB**	35.45 dB	**36.14 dB**	**36.04 dB**
IMG_7113	31.93 dB	**32.29 dB**	31.91 dB	**32.23 dB**	**32.12 dB**
IMG_7626	34.85 dB	**35.12 dB**	34.73 dB	**34.98 dB**	**34.99 dB**
IMG_7627	32.51 dB	**32.78 dB**	32.41 dB	**32.72 dB**	**32.63 dB**
IMG_7673	32.38 dB	**32.78 dB**	32.28 dB	**32.70 dB**	32.56 dB
IMG_7739	32.49 dB	**33.06 dB**	32.53 dB	**32.94 dB**	**32.88 dB**
IMG_8275	32.44 dB	**32.75 dB**	32.46 dB	**32.74 dB**	**32.71 dB**
IMG_8336	**33.60 dB**	**33.80 dB**	33.51 dB	**33.78 dB**	**33.71 dB**
IMG_8339	35.55 dB	**36.18 dB**	35.81 dB	**36.25 dB**	35.88 dB
IMG_8586	32.29 dB	**32.71 dB**	32.22 dB	**32.61 dB**	32.49 dB
Average	33.33 dB	**33.76 dB**	33.33 dB	**33.71 dB**	**33.60 dB**

PSNR					
$\sigma = 20$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	41.43 dB	42.73 dB	42.39 dB	**42.85 dB**	**42.96 dB**
IMG_7113	34.24 dB	**34.53 dB**	34.27 dB	**34.48 dB**	**34.43 dB**
IMG_7626	37.86 dB	**38.09 dB**	37.88 dB	**37.99 dB**	**38.08 dB**
IMG_7627	33.67 dB	**33.90 dB**	33.64 dB	**33.86 dB**	**33.81 dB**
IMG_7673	34.70 dB	**35.03 dB**	34.67 dB	**34.98 dB**	**34.90 dB**
IMG_7739	33.75 dB	**34.24 dB**	33.81 dB	**34.14 dB**	**34.11 dB**
IMG_8275	33.22 dB	**33.47 dB**	33.29 dB	**33.48 dB**	**33.49 dB**
IMG_8336	**36.58 dB**	**36.77 dB**	**36.61 dB**	**36.76 dB**	**36.76 dB**
IMG_8339	37.13 dB	**37.77 dB**	37.55 dB	**37.83 dB**	37.27 dB
IMG_8586	33.97 dB	**34.35 dB**	33.97 dB	**34.28 dB**	**34.19 dB**
Average	35.65 dB	**36.09 dB**	35.81 dB	**36.07 dB**	**36.00 dB**

Table 3.4 – Results using Non-Local Bayes as base algorithm, noise with $\sigma = 20$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**. From the results is clear that DA3D is the best method to improve the results on the flat areas, while applying the multiscale frameworks leads to an improvement in the textures of the images. For this level of noise, the combination *da3d_ms* offers a good combination of texture quality and flat areas quality.

FPSNR					
$\sigma = 40$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	39.42 dB	41.32 dB	41.14 dB	41.53 dB	**42.00 dB**
IMG_7113	35.64 dB	35.88 dB	**36.22 dB**	35.92 dB	**36.11 dB**
IMG_7626	38.77 dB	39.04 dB	**39.68 dB**	39.04 dB	**39.57 dB**
IMG_7627	34.62 dB	34.87 dB	**35.57 dB**	34.94 dB	35.29 dB
IMG_7673	36.77 dB	36.99 dB	**37.45 dB**	37.05 dB	**37.40 dB**
IMG_7739	35.19 dB	35.57 dB	**35.83 dB**	35.61 dB	**35.78 dB**
IMG_8275	33.13 dB	33.15 dB	**33.81 dB**	33.24 dB	33.58 dB
IMG_8336	37.45 dB	37.66 dB	**38.14 dB**	37.75 dB	**38.09 dB**
IMG_8339	35.47 dB	**36.83 dB**	**36.92 dB**	**36.94 dB**	36.47 dB
IMG_8586	34.87 dB	**35.24 dB**	**35.37 dB**	**35.27 dB**	**35.40 dB**
Average	36.13 dB	36.66 dB	**37.01 dB**	36.73 dB	**36.97 dB**

TPSNR					
$\sigma = 40$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	30.97 dB	**32.26 dB**	31.52 dB	**32.35 dB**	**32.32 dB**
IMG_7113	27.83 dB	**28.76 dB**	28.16 dB	**28.70 dB**	**28.65 dB**
IMG_7626	31.00 dB	**31.39 dB**	31.11 dB	**31.29 dB**	**31.41 dB**
IMG_7627	28.75 dB	**29.33 dB**	28.82 dB	**29.29 dB**	**29.22 dB**
IMG_7673	28.50 dB	**29.27 dB**	28.67 dB	**29.22 dB**	**29.09 dB**
IMG_7739	29.35 dB	**30.03 dB**	29.33 dB	**29.93 dB**	29.75 dB
IMG_8275	28.39 dB	**29.19 dB**	28.78 dB	**29.22 dB**	**29.22 dB**
IMG_8336	29.53 dB	**30.07 dB**	29.79 dB	**30.13 dB**	**30.11 dB**
IMG_8339	31.36 dB	**32.65 dB**	32.13 dB	**32.71 dB**	**32.59 dB**
IMG_8586	28.65 dB	**29.31 dB**	28.72 dB	**29.27 dB**	**29.13 dB**
Average	29.43 dB	**30.23 dB**	29.70 dB	**30.21 dB**	**30.15 dB**

PSNR					
$\sigma = 40$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	37.48 dB	39.12 dB	38.67 dB	39.27 dB	**39.50 dB**
IMG_7113	30.34 dB	**31.16 dB**	30.71 dB	**31.12 dB**	**31.11 dB**
IMG_7626	34.09 dB	**34.46 dB**	34.36 dB	34.38 dB	**34.59 dB**
IMG_7627	29.97 dB	**30.52 dB**	30.13 dB	**30.49 dB**	**30.46 dB**
IMG_7673	30.95 dB	**31.65 dB**	31.19 dB	**31.62 dB**	**31.55 dB**
IMG_7739	30.68 dB	**31.31 dB**	30.73 dB	**31.24 dB**	31.10 dB
IMG_8275	29.29 dB	**30.00 dB**	29.72 dB	**30.03 dB**	**30.08 dB**
IMG_8336	32.68 dB	**33.15 dB**	33.02 dB	**33.22 dB**	**33.27 dB**
IMG_8339	33.06 dB	**34.37 dB**	34.02 dB	**34.44 dB**	34.21 dB
IMG_8586	30.49 dB	**31.11 dB**	30.64 dB	**31.08 dB**	**30.98 dB**
Average	31.90 dB	**32.68 dB**	32.32 dB	**32.69 dB**	**32.69 dB**

Table 3.5 – Results using Non-Local Bayes as base algorithm, noise with $\sigma = 40$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**. From the results is clear that DA3D is the best method to improve the results on the flat areas, while applying the multiscale frameworks leads to an improvement in the textures of the images. For this level of noise, the combination *ms_da3d* offers a good combination of texture quality and flat areas quality.

FPSNR					
$\sigma = 70$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	35.73 dB	38.36 dB	38.07 dB	38.56 dB	**39.16 dB**
IMG_7113	32.91 dB	**33.98 dB**	**33.98 dB**	**34.02 dB**	**34.15 dB**
IMG_7626	34.99 dB	36.16 dB	36.53 dB	36.24 dB	**36.75 dB**
IMG_7627	31.74 dB	32.44 dB	**33.07 dB**	32.51 dB	32.78 dB
IMG_7673	33.94 dB	34.91 dB	**35.18 dB**	34.98 dB	**35.31 dB**
IMG_7739	32.77 dB	33.61 dB	**33.73 dB**	**33.67 dB**	**33.84 dB**
IMG_8275	30.44 dB	30.92 dB	**31.49 dB**	30.99 dB	31.25 dB
IMG_8336	34.17 dB	35.20 dB	**35.46 dB**	35.28 dB	**35.56 dB**
IMG_8339	32.04 dB	**34.17 dB**	**34.14 dB**	**34.26 dB**	**34.08 dB**
IMG_8586	32.56 dB	33.17 dB	**33.32 dB**	33.21 dB	**33.49 dB**
Average	33.13 dB	34.29 dB	**34.50 dB**	34.37 dB	**34.64 dB**

TPSNR					
$\sigma = 70$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	27.75 dB	**29.25 dB**	28.56 dB	**29.32 dB**	**29.35 dB**
IMG_7113	25.17 dB	**26.27 dB**	25.32 dB	**26.22 dB**	26.02 dB
IMG_7626	27.29 dB	**28.15 dB**	27.89 dB	**28.21 dB**	**28.30 dB**
IMG_7627	25.90 dB	**26.73 dB**	26.23 dB	**26.74 dB**	**26.65 dB**
IMG_7673	25.56 dB	**26.67 dB**	25.96 dB	**26.65 dB**	**26.54 dB**
IMG_7739	27.32 dB	**27.97 dB**	27.34 dB	**27.95 dB**	**27.81 dB**
IMG_8275	25.27 dB	**26.51 dB**	25.84 dB	**26.52 dB**	**26.48 dB**
IMG_8336	26.13 dB	**27.21 dB**	26.77 dB	**27.24 dB**	**27.28 dB**
IMG_8339	27.85 dB	**29.99 dB**	29.32 dB	**30.02 dB**	**30.05 dB**
IMG_8586	26.05 dB	**26.81 dB**	26.12 dB	**26.78 dB**	**26.62 dB**
Average	26.43 dB	**27.55 dB**	26.93 dB	**27.56 dB**	**27.51 dB**

PSNR					
$\sigma = 70$	nlb	ms	da3d	da3d_ms	ms_da3d
DSC_0767	33.98 dB	36.13 dB	35.66 dB	36.27 dB	**36.60 dB**
IMG_7113	27.68 dB	**28.77 dB**	27.96 dB	**28.73 dB**	**28.59 dB**
IMG_7626	30.37 dB	31.29 dB	31.16 dB	**31.36 dB**	**31.53 dB**
IMG_7627	27.12 dB	**27.93 dB**	27.55 dB	**27.95 dB**	**27.90 dB**
IMG_7673	28.03 dB	**29.12 dB**	28.53 dB	**29.11 dB**	**29.06 dB**
IMG_7739	28.59 dB	**29.27 dB**	28.73 dB	**29.26 dB**	**29.16 dB**
IMG_8275	26.22 dB	**27.37 dB**	26.84 dB	**27.39 dB**	**27.39 dB**
IMG_8336	29.30 dB	**30.37 dB**	30.07 dB	**30.41 dB**	**30.50 dB**
IMG_8339	29.57 dB	**31.71 dB**	31.22 dB	**31.75 dB**	**31.73 dB**
IMG_8586	27.95 dB	**28.68 dB**	28.11 dB	**28.66 dB**	**28.57 dB**
Average	28.88 dB	**30.06 dB**	29.58 dB	**30.09 dB**	**30.10 dB**

Table 3.6 – Results using Non-Local Bayes as base algorithm, noise with $\sigma = 70$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**. From the results is clear that DA3D is the best method to improve the results on the flat areas, while applying the multiscale frameworks leads to an improvement in the textures of the images. For this level of noise, the combination *ms_da3d* offers a good combination of texture quality and flat areas quality.

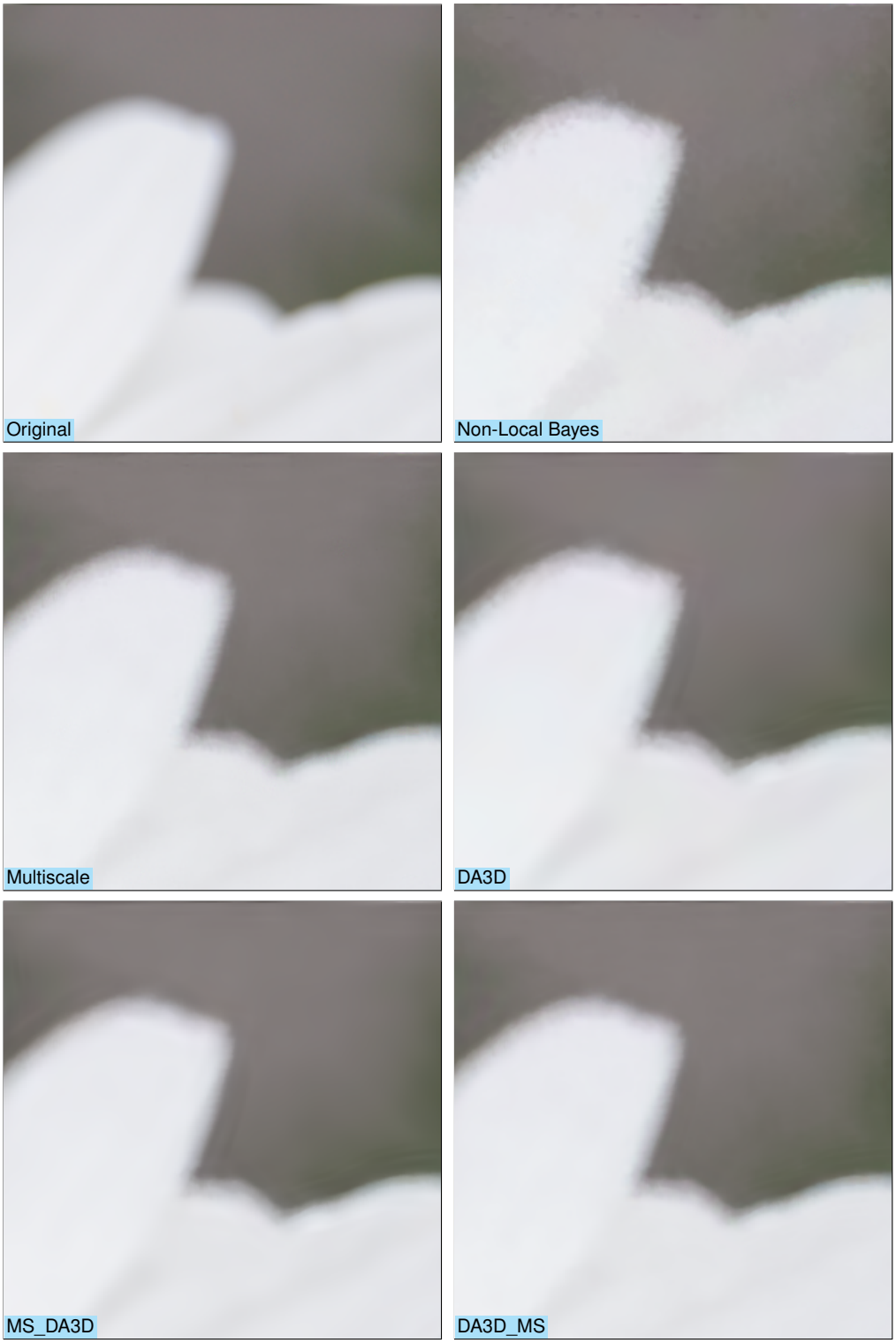


Figure 3.6 – Results of the different denoising methods treated in this chapters and in Tables 3.3-3.6, with a noise of $\sigma = 40$. The detail is taken from the set of test images in Figure 3.5. Notice the ringing artifacts produced by DA3D near smooth edges. The da3d_ms algorithm manages to remove them, and to provide the closest result to the original image.

Figure 3.7 – Results of the different denoising methods treated in this chapters and in Tables 3.3-3.6, with a noise of $\sigma = 40$. The detail is taken from the set of test images in Figure 3.5. Notice how Multiscale and da3d_ms are the best in recovering the texture of the bushes, while preserving the edges.

Figure 3.8 – Results of the different denoising methods treated in this chapters and in Tables 3.3-3.6, with a noise of $\sigma = 40$. The detail is taken from the set of test images in Figure 3.5. Notice how Multiscale and da3d_ms are the best in recovering the texture of the trees and the structure of the buildings.

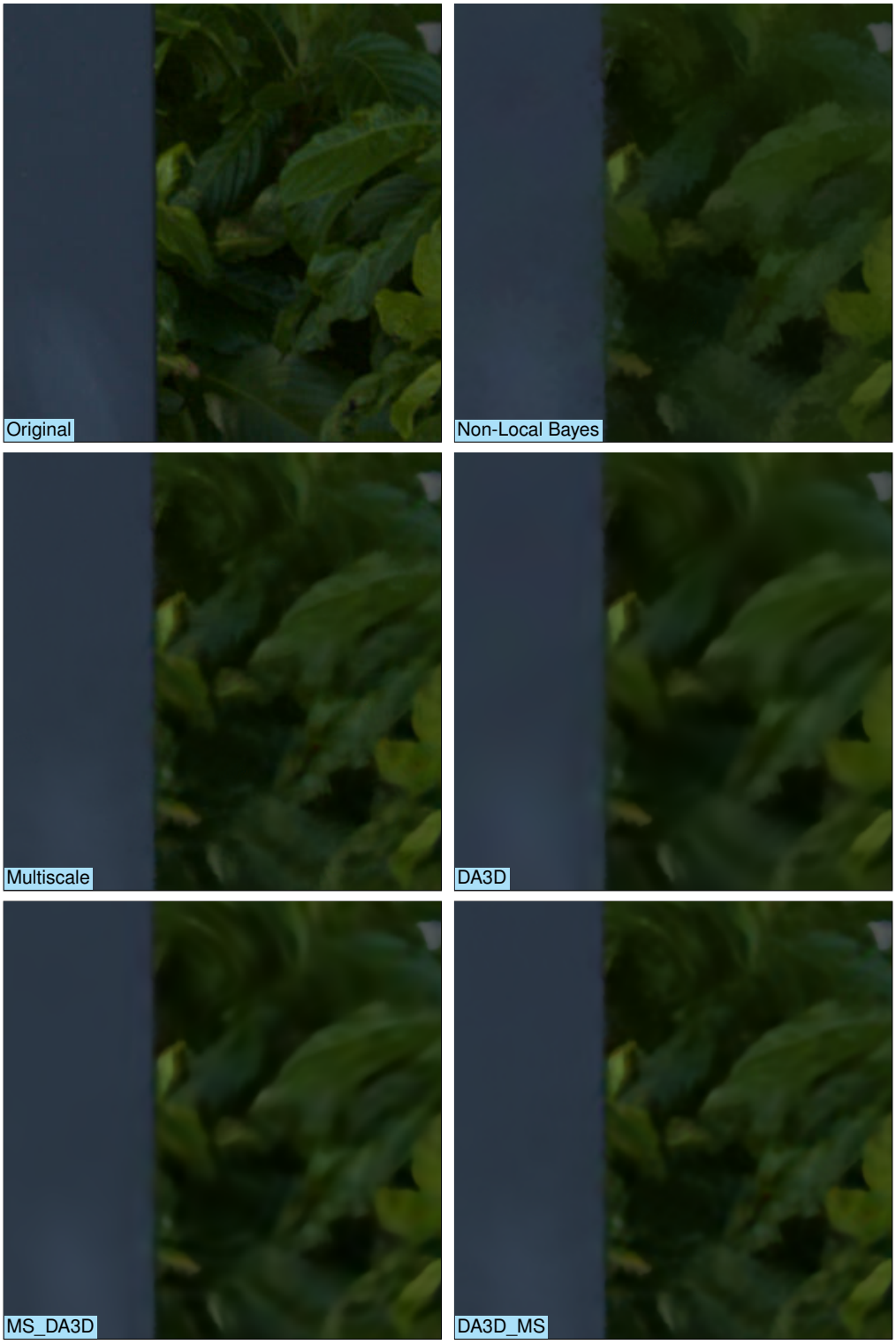


Figure 3.9 – Results of the different denoising methods treated in this chapters and in Tables 3.3-3.6, with a noise of $\sigma = 40$. The detail is taken from the set of test images in Figure 3.5. Multiscale and da3d_ms both perform well on the leaves, but da3d_ms has a slightly smoother result on the flat area.

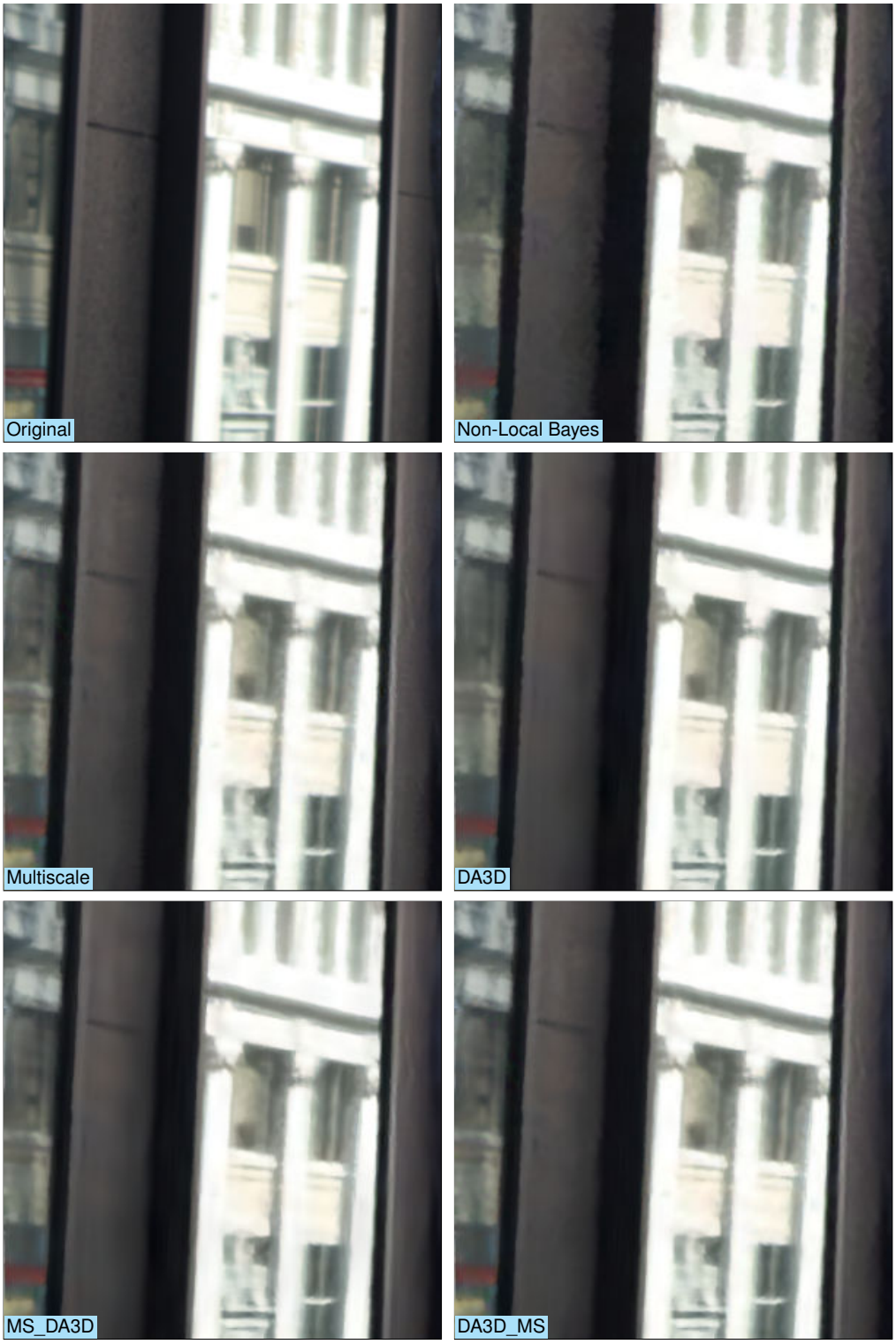


Figure 3.10 – Results of the different denoising methods treated in this chapters and in Tables 3.3-3.6, with a noise of $\sigma = 40$. The detail is taken from the set of test images in Figure 3.5. In the window reflection, DA3D performs better than Multiscale on the edges. This advantage is kept on both ms_da3d and da3d_ms.

FPSNR					
$\sigma = 10$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	46.89 dB	46.89 dB	**48.14 dB**	**48.14 dB**	47.81 dB
IMG_7113	41.24 dB	41.24 dB	**41.64 dB**	**41.64 dB**	41.41 dB
IMG_7626	44.61 dB	44.61 dB	**45.36 dB**	**45.36 dB**	44.86 dB
IMG_7627	40.75 dB	40.75 dB	**41.18 dB**	**41.18 dB**	40.78 dB
IMG_7673	42.12 dB	42.12 dB	**42.65 dB**	**42.65 dB**	42.35 dB
IMG_7739	40.23 dB	40.23 dB	**40.47 dB**	**40.47 dB**	**40.33 dB**
IMG_8275	39.27 dB	39.27 dB	**39.55 dB**	**39.55 dB**	39.29 dB
IMG_8336	43.44 dB	43.44 dB	**44.06 dB**	**44.06 dB**	43.73 dB
IMG_8339	42.90 dB	42.90 dB	**43.36 dB**	**43.36 dB**	43.09 dB
IMG_8586	40.38 dB	40.38 dB	**40.66 dB**	**40.66 dB**	40.46 dB
Average	42.18 dB	42.18 dB	**42.71 dB**	**42.71 dB**	42.41 dB

TPSNR					
$\sigma = 10$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	**40.20 dB**	**40.20 dB**	**40.07 dB**	**40.07 dB**	**40.13 dB**
IMG_7113	**35.93 dB**	**35.93 dB**	**35.84 dB**	**35.84 dB**	**35.87 dB**
IMG_7626	**38.29 dB**	**38.29 dB**	**38.45 dB**	**38.45 dB**	**38.37 dB**
IMG_7627	**36.25 dB**	**36.25 dB**	**36.22 dB**	**36.22 dB**	**36.21 dB**
IMG_7673	**36.24 dB**	**36.24 dB**	**36.18 dB**	**36.18 dB**	**36.21 dB**
IMG_7739	**36.77 dB**	**36.77 dB**	**36.62 dB**	**36.62 dB**	**36.69 dB**
IMG_8275	**36.44 dB**	**36.44 dB**	**36.38 dB**	**36.38 dB**	**36.39 dB**
IMG_8336	**37.51 dB**	**37.51 dB**	**37.46 dB**	**37.46 dB**	**37.44 dB**
IMG_8339	**40.23 dB**	**40.23 dB**	**40.16 dB**	**40.16 dB**	**40.17 dB**
IMG_8586	**36.18 dB**	**36.18 dB**	**36.05 dB**	**36.05 dB**	**36.10 dB**
Average	**37.40 dB**	**37.40 dB**	**37.34 dB**	**37.34 dB**	**37.36 dB**

PSNR					
$\sigma = 10$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	45.60 dB	45.60 dB	**46.36 dB**	**46.36 dB**	**46.18 dB**
IMG_7113	**37.95 dB**	**37.95 dB**	**37.98 dB**	**37.98 dB**	**37.95 dB**
IMG_7626	41.04 dB	41.04 dB	**41.35 dB**	**41.35 dB**	**41.16 dB**
IMG_7627	**37.29 dB**	**37.29 dB**	**37.32 dB**	**37.32 dB**	**37.26 dB**
IMG_7673	**38.29 dB**	**38.29 dB**	**38.35 dB**	**38.35 dB**	**38.31 dB**
IMG_7739	**37.71 dB**	**37.71 dB**	**37.64 dB**	**37.64 dB**	**37.67 dB**
IMG_8275	**37.07 dB**	**37.07 dB**	**37.07 dB**	**37.07 dB**	**37.03 dB**
IMG_8336	**40.17 dB**	**40.17 dB**	**40.30 dB**	**40.30 dB**	**40.20 dB**
IMG_8339	**41.44 dB**	**41.44 dB**	**41.56 dB**	**41.56 dB**	**41.47 dB**
IMG_8586	**37.62 dB**	**37.62 dB**	**37.58 dB**	**37.58 dB**	**37.58 dB**
Average	**39.42 dB**	**39.42 dB**	**39.55 dB**	**39.55 dB**	**39.48 dB**

Table 3.7 – Results using BM3D as base algorithm, noise with $\sigma = 10$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**.

FPSNR					
$\sigma = 20$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	42.83 dB	43.98 dB	**44.80 dB**	44.51 dB	**44.88 dB**
IMG_7113	38.06 dB	38.20 dB	**38.73 dB**	38.40 dB	**38.58 dB**
IMG_7626	41.32 dB	41.53 dB	**42.76 dB**	42.01 dB	42.20 dB
IMG_7627	37.65 dB	37.59 dB	**38.40 dB**	37.86 dB	37.98 dB
IMG_7673	39.08 dB	39.26 dB	**39.95 dB**	39.52 dB	**39.77 dB**
IMG_7739	37.42 dB	37.59 dB	**37.94 dB**	**37.81 dB**	**37.86 dB**
IMG_8275	36.01 dB	36.01 dB	**36.56 dB**	36.21 dB	36.34 dB
IMG_8336	40.18 dB	40.39 dB	**41.26 dB**	40.72 dB	41.03 dB
IMG_8339	39.31 dB	39.59 dB	**40.22 dB**	39.93 dB	39.89 dB
IMG_8586	37.38 dB	37.51 dB	**37.87 dB**	**37.67 dB**	**37.73 dB**
Average	38.92 dB	39.17 dB	**39.85 dB**	39.47 dB	39.62 dB

TPSNR					
$\sigma = 20$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	**36.11 dB**	**36.19 dB**	**36.22 dB**	**36.25 dB**	**36.28 dB**
IMG_7113	**32.21 dB**	**32.20 dB**	**32.16 dB**	**32.20 dB**	**32.16 dB**
IMG_7626	34.64 dB	34.54 dB	**35.06 dB**	34.83 dB	**34.94 dB**
IMG_7627	**32.64 dB**	**32.53 dB**	**32.67 dB**	**32.62 dB**	**32.62 dB**
IMG_7673	**32.51 dB**	**32.50 dB**	**32.52 dB**	**32.55 dB**	**32.54 dB**
IMG_7739	**33.26 dB**	**33.24 dB**	**33.18 dB**	**33.20 dB**	**33.16 dB**
IMG_8275	**32.69 dB**	**32.65 dB**	**32.75 dB**	**32.73 dB**	**32.73 dB**
IMG_8336	**33.75 dB**	**33.71 dB**	**33.84 dB**	**33.81 dB**	**33.80 dB**
IMG_8339	**36.29 dB**	**36.33 dB**	**36.49 dB**	**36.46 dB**	**36.44 dB**
IMG_8586	**32.51 dB**	**32.47 dB**	**32.45 dB**	**32.48 dB**	**32.43 dB**
Average	**33.66 dB**	**33.64 dB**	**33.73 dB**	**33.71 dB**	**33.71 dB**

PSNR					
$\sigma = 20$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	41.53 dB	42.30 dB	**42.80 dB**	42.65 dB	**42.88 dB**
IMG_7113	**34.36 dB**	**34.38 dB**	**34.46 dB**	**34.42 dB**	**34.43 dB**
IMG_7626	37.48 dB	37.46 dB	**38.14 dB**	37.80 dB	37.93 dB
IMG_7627	**33.75 dB**	33.65 dB	**33.87 dB**	**33.77 dB**	**33.78 dB**
IMG_7673	**34.69 dB**	**34.72 dB**	**34.85 dB**	**34.80 dB**	**34.84 dB**
IMG_7739	**34.34 dB**	**34.35 dB**	**34.36 dB**	**34.35 dB**	**34.32 dB**
IMG_8275	**33.40 dB**	**33.37 dB**	**33.53 dB**	**33.47 dB**	**33.49 dB**
IMG_8336	36.55 dB	36.57 dB	**36.88 dB**	**36.73 dB**	**36.80 dB**
IMG_8339	37.63 dB	37.75 dB	**38.06 dB**	**37.95 dB**	**37.93 dB**
IMG_8586	**34.10 dB**	**34.10 dB**	**34.15 dB**	**34.13 dB**	**34.11 dB**
Average	35.78 dB	35.87 dB	**36.11 dB**	**36.01 dB**	**36.05 dB**

Table 3.8 – Results using BM3D as base algorithm, noise with $\sigma = 20$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**.

FPSNR					
$\sigma = 40$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	38.64 dB	40.33 dB	41.12 dB	41.27 dB	**41.65 dB**
IMG_7113	34.91 dB	35.39 dB	**35.96 dB**	35.75 dB	**36.01 dB**
IMG_7626	37.26 dB	37.77 dB	**39.42 dB**	38.54 dB	**39.24 dB**
IMG_7627	34.20 dB	34.30 dB	**35.42 dB**	34.74 dB	35.18 dB
IMG_7673	35.76 dB	36.39 dB	**37.14 dB**	36.88 dB	**37.26 dB**
IMG_7739	34.58 dB	35.04 dB	**35.61 dB**	35.45 dB	**35.67 dB**
IMG_8275	32.64 dB	32.84 dB	**33.60 dB**	33.20 dB	**33.56 dB**
IMG_8336	36.50 dB	37.03 dB	**38.04 dB**	37.62 dB	**38.09 dB**
IMG_8339	35.36 dB	35.98 dB	**36.86 dB**	36.58 dB	36.35 dB
IMG_8586	34.38 dB	34.80 dB	**35.23 dB**	**35.13 dB**	**35.26 dB**
Average	35.42 dB	35.99 dB	**36.84 dB**	36.52 dB	**36.83 dB**
TPSNR					
$\sigma = 40$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	32.10 dB	32.22 dB	**32.47 dB**	**32.40 dB**	**32.42 dB**
IMG_7113	**28.71 dB**	**28.76 dB**	**28.79 dB**	**28.79 dB**	**28.76 dB**
IMG_7626	30.76 dB	30.73 dB	**31.54 dB**	31.18 dB	**31.43 dB**
IMG_7627	29.18 dB	29.07 dB	**29.39 dB**	**29.21 dB**	**29.25 dB**
IMG_7673	**29.03 dB**	**29.05 dB**	**29.18 dB**	**29.15 dB**	**29.11 dB**
IMG_7739	**30.15 dB**	**30.15 dB**	**30.12 dB**	**30.13 dB**	**30.00 dB**
IMG_8275	29.09 dB	29.07 dB	**29.34 dB**	**29.22 dB**	**29.26 dB**
IMG_8336	29.92 dB	29.97 dB	**30.28 dB**	**30.21 dB**	**30.24 dB**
IMG_8339	32.26 dB	32.50 dB	**32.90 dB**	**32.83 dB**	**32.75 dB**
IMG_8586	**29.11 dB**	**29.10 dB**	**29.23 dB**	**29.17 dB**	**29.12 dB**
Average	30.03 dB	30.06 dB	**30.32 dB**	**30.23 dB**	**30.23 dB**
PSNR					
$\sigma = 40$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	37.40 dB	38.53 dB	39.10 dB	39.15 dB	**39.36 dB**
IMG_7113	30.93 dB	**31.07 dB**	**31.20 dB**	**31.16 dB**	**31.18 dB**
IMG_7626	33.56 dB	33.66 dB	**34.66 dB**	34.19 dB	**34.53 dB**
IMG_7627	30.30 dB	30.21 dB	**30.63 dB**	30.39 dB	**30.48 dB**
IMG_7673	31.25 dB	31.37 dB	**31.59 dB**	**31.53 dB**	**31.55 dB**
IMG_7739	**31.27 dB**	**31.34 dB**	**31.40 dB**	**31.39 dB**	**31.30 dB**
IMG_8275	29.84 dB	29.85 dB	**30.19 dB**	**30.03 dB**	**30.11 dB**
IMG_8336	32.76 dB	32.93 dB	**33.40 dB**	**33.25 dB**	**33.37 dB**
IMG_8339	33.63 dB	34.00 dB	**34.55 dB**	**34.42 dB**	34.28 dB
IMG_8586	30.79 dB	**30.86 dB**	**31.03 dB**	**30.97 dB**	**30.95 dB**
Average	32.17 dB	32.38 dB	**32.77 dB**	**32.65 dB**	**32.71 dB**

Table 3.9 – Results using BM3D as base algorithm, noise with $\sigma = 40$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**.

FPSNR					
$\sigma = 70$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	35.08 dB	37.09 dB	37.91 dB	38.30 dB	**38.82 dB**
IMG_7113	32.37 dB	33.31 dB	33.78 dB	33.82 dB	**34.08 dB**
IMG_7626	33.99 dB	35.01 dB	**36.31 dB**	35.91 dB	**36.45 dB**
IMG_7627	31.46 dB	31.86 dB	**32.89 dB**	32.39 dB	**32.74 dB**
IMG_7673	33.06 dB	34.12 dB	34.84 dB	34.78 dB	**35.27 dB**
IMG_7739	32.23 dB	33.02 dB	**33.62 dB**	33.57 dB	**33.78 dB**
IMG_8275	30.11 dB	30.58 dB	**31.27 dB**	31.01 dB	**31.33 dB**
IMG_8336	33.60 dB	34.42 dB	35.37 dB	35.13 dB	**35.64 dB**
IMG_8339	32.19 dB	33.21 dB	**34.00 dB**	**33.96 dB**	**33.96 dB**
IMG_8586	32.09 dB	32.81 dB	33.22 dB	**33.28 dB**	**33.42 dB**
Average	32.62 dB	33.54 dB	34.32 dB	34.22 dB	**34.55 dB**

TPSNR					
$\sigma = 70$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	28.78 dB	29.09 dB	**29.40 dB**	**29.36 dB**	**29.38 dB**
IMG_7113	**26.20 dB**	**26.26 dB**	**26.24 dB**	**26.24 dB**	**26.14 dB**
IMG_7626	27.74 dB	27.73 dB	**28.39 dB**	28.08 dB	**28.24 dB**
IMG_7627	26.61 dB	26.53 dB	**26.84 dB**	**26.66 dB**	**26.66 dB**
IMG_7673	**26.48 dB**	**26.52 dB**	**26.66 dB**	**26.60 dB**	**26.57 dB**
IMG_7739	**27.95 dB**	**28.06 dB**	**28.04 dB**	**28.10 dB**	**27.94 dB**
IMG_8275	26.40 dB	26.42 dB	**26.63 dB**	**26.54 dB**	**26.52 dB**
IMG_8336	27.13 dB	27.20 dB	**27.50 dB**	**27.41 dB**	**27.44 dB**
IMG_8339	29.16 dB	29.66 dB	**30.08 dB**	**30.09 dB**	**30.11 dB**
IMG_8586	**26.64 dB**	**26.66 dB**	**26.76 dB**	**26.71 dB**	**26.63 dB**
Average	27.31 dB	27.41 dB	**27.65 dB**	**27.58 dB**	**27.56 dB**

PSNR					
$\sigma = 70$	bm3d	ms	da3d	da3d_ms	ms_da3d
DSC_0767	33.92 dB	35.34 dB	35.95 dB	36.15 dB	**36.43 dB**
IMG_7113	28.42 dB	**28.65 dB**	**28.71 dB**	**28.72 dB**	**28.67 dB**
IMG_7626	30.48 dB	30.72 dB	**31.52 dB**	31.19 dB	**31.42 dB**
IMG_7627	27.70 dB	27.69 dB	**28.08 dB**	27.86 dB	**27.90 dB**
IMG_7673	28.67 dB	28.88 dB	**29.10 dB**	**29.04 dB**	**29.08 dB**
IMG_7739	29.04 dB	**29.26 dB**	**29.33 dB**	**29.38 dB**	**29.27 dB**
IMG_8275	27.17 dB	27.25 dB	**27.53 dB**	**27.41 dB**	**27.43 dB**
IMG_8336	29.94 dB	30.20 dB	**30.64 dB**	**30.51 dB**	**30.65 dB**
IMG_8339	30.50 dB	31.18 dB	**31.72 dB**	**31.72 dB**	**31.72 dB**
IMG_8586	28.35 dB	**28.50 dB**	**28.64 dB**	**28.61 dB**	**28.56 dB**
Average	29.42 dB	29.77 dB	**30.12 dB**	**30.06 dB**	**30.12 dB**

Table 3.10 – Results using BM3D as base algorithm, noise with $\sigma = 70$. The best result of each line is shown in **bold**, and the results inferior by less than 0.2 dB are shown in **gray**.

Chapter 4

External Denoising

A recent seminal paper on the absolute bounds of image denoising [32] proposes a patch denoising method effectively realizing the minimal mean square error, given all the known image patches. It is extremely important to reach these absolute limits, but they require processing a limitless database. In the above mentioned paper this database had 10 billion patches. In this paper we demonstrate that by factorizing the patch space the method can be sped up by a factor of more than a thousand, while maintaining the theoretical claim that the method is optimal. Using the method on real images demonstrates its potential to beat the state of the art, as it performs better on difficult patches.

4.1 Introduction

Recently the literature on image denoising has proposed extremely eficient methods [7, 26, 28, 67] for this ill-posed problem, and has even started exploring its absolute theoretical limits [6, 32]. There are two complementary early approaches to denoising, Fourier (or frequency-based) methods, and Bayesian estimation methods. Fourier methods have been extended in the past thirty years to other linear space-frequency transforms such as the windowed DCT [62] or the many wavelet transforms [37]. Being first parametric and limited to rather restrictive Markov random field models [17], Bayesian methods are becoming non-parametric. The idea for the recent non-parametric Markovian estimation methods is that, in a textured image, the stochastic model for a given pixel can be predicted from a local image neighborhood around it, which we shall call "patch". This principle yielded the image denoising algorithm called "non-local means" by Buades et al. [2]. The algorithm was called "non-local" because it uses patches that are far away from the processed pixel, and could even use *patches taken from other images*.

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 *very large* patch set, assuming they represent "all possible natural image patches".

For example, Zoran and Weiss [67] introduced the EPLL algorithm, in which they learn the patch space distribution from a mixture of Gaussians trained with 2×10^6 patches. Then, they use this model as a prior for denoising patches by trying to maximize the *Expected Patch Log Likelihood*, instead of denoising each patch separately. To do this, they minimize a cost function using an optimization method called "Half Quadratic Splitting" [48]. Zontak and Irani [66] analyzed the denoising performance using patches extracted from the same image, or what they call internal patch searches, against extracting "external" patches from a

huge database. They concluded that similar patches tend to recur much more frequently inside the same image than in any random external collection of natural images. Furthermore, internal patch redundancy rapidly decays with the growth of the spatial distance from the patch, and its gradient content. Finally, they observed that finding an equally good external representative patch for every patch of an image, requires a huge external database, which makes the denoising problem computationally intractable. Lee et al. [31] used the Bayesian framework to derive an MMSE non-local image denoising algorithm, that uses both internal image patches as well as an external codebook with noiseless patches. They perform denoising by normalizing the patch-space by the mean, an idea which will be further exploited here. Several "shotgun" methods have been recently applied in other image processing tasks, such as scene completion [19] (using 2.3 million images), or scene recognition [57] (using 80 million 32×32 patches).

Let us now turn on the search for absolute bounds for the image denoising. The Cramer-Rao type lower bounds on the attainable RMSE performance given by Chaterjee and Milanfar [6] are optimistic: they allow for the possibility of a significant increase in denoising performance for a wide class of images at certain SNRs, in particular, for synthetic piecewise constant images. Yet, a recent paper by A. Levin and B. Nadler [32] provides a second answer to the question of absolute limits raised by [6]. This seminal paper uses a shotgun denoising method to approach the bounds, which is loosely based on the NL-means algorithm, with adequate parameters to account for a Bayesian linear minimum mean square estimation (LMMSE) of the noisy patch given a database of 10^{10} noiseless patches. The only and important difference is that similar patches are searched on a database of 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 estimate *the best average estimation error attainable by any patch-based denoising algorithm* (to be more specific, it is the best average estimation error knowing a single patch and all the observed; but most denoising algorithms actually also use the knowledge of overlapping patches in the image to denoise it). After performing experimentation, and comparing them with state-of-the-art denoising algorithms, they conclude that results might be within 0.1dB of the optimal possible denoising method, at least when using small patch sizes or under big amounts of noise, therefore leaving no further room for improvement. They propose increasing the support size to obtain better PSNR values. Yet, due to the curse of dimensionality, non-parametric techniques won't be able to find good similar patches. This leads the authors to give up a shotgun method, and to switch again to parametric approaches. However, a limitation of this work is that computational constraints restricted the experimentation to small patches. Hence real bounds independent of patch size are still unknown.

In [33], the same authors study the dependence of denoising error on patch size. Similarly to Zontak and Irani [66], they conclude that when increasing window size, patches with more detail require a significantly larger database, and that the gain from doing so is very small. At the same time, the opposite happens with smoother patches: increasing the window size is much more rewarded in the final denoising performance. This leads them to propose an adaptive strategy using variable window sizes depending on local patch complexity. Finally, they again explore the fundamental limits of denoising, with an infinite window size and assuming a perfect natural image prior. To do this, they use a simplified dead leaves image formation model and combine it with a scale invariance assumption for natural images, which implies a power-law convergence. Using this power-law to model

the PSNR with respect to the window size, they fit the model and extrapolate its result to obtain optimal denoising bounds. They conclude that there is still some room for improvement (around 1dB).

In this chapter, we stick to the first shotgun approach [32] and intend to render it feasible by accelerating considerably the computation of their Monte Carlo integral. The proposed algorithm attains similar results using less computational resources and in shorter time. To achieve this, a smaller probability space is used by factorizing the space of patches. This allows to work with entire equivalence classes of patches instead of single patches, thus permitting a significant decrease of the number of needed computations and a reduction of the required memory. Following this, a comparison is made between our algorithm and two state-of-the-art methods, the BM3D algorithm [7,26] and the Non Local Bayes method [27] showing how our method outperforms both for the task of denoising complex patches.

This chapter is organized as follows. Section 4.2 describes how the patch-space is factorized. In Section 4.3, the Normalized Shotgun Denoising (NSD) algorithm is presented. Section 4.4 provides experimental results and further comparisons with state-of-the-art methods. To finalize the paper, in section 4.5 some directions for future research are given.

4.2 Space factorization

Following [24], the probability space of all $k \times k$ patches $(\Omega, \mathbb{P}(X))$ can be factorized into the product of three smaller subspaces, representing the shape, the brightness and the contrast of the patch. The terms *brightness* and *contrast* can be easily translated to our model by considering the *average value* of the patch and its standard deviation. Therefore, a patch $\mathbf{x}$ with mean $\mu_{\mathbf{x}}$ and standard deviation $\sigma_{\mathbf{x}}$ can be decomposed as

$$\mathbf{x} = \sigma_{\mathbf{x}}\tilde{\mathbf{x}} + \mu_{\mathbf{x}}\mathbf{1} \tag{4.1}$$

where $\mathbf{1} \in \Omega$ is the patch with all values equal to 1 and $\tilde{\mathbf{x}} \in \tilde{\Omega}$, the sample space of normalized patches. The patch $\tilde{\mathbf{x}}$ contains all information about the shape, with zero mean and standard deviation of 1. Then, we can factorize the patch space so that the underlying probability density, which we shall call $p(\mathbf{x})$, on a first approximation, becomes

$$p(\mathbf{x}) \approx \tilde{p}(\tilde{\mathbf{x}})p^{\sigma}(\sigma_{\mathbf{x}})p^{\mu}(\mu_{\mathbf{x}}) \tag{4.2}$$

where $\tilde{p}$ is the probability density of the normalized patch space, p^{μ} is a probability density over the space of possible mean values and p^{σ} is a probability density over the possible standard deviations.

The distribution of the average brightness of the patches p^{μ}, following [24], should be "almost" uniform over the interval, except near the endpoints (because the only patches with that mean are the flat ones). Also in [24], it is suggested that p^{σ} should have, at a first approximation, an exponential distribution. Computing the parameter λ from the exponential for finding the best distribution for natural images is easy, remembering that $\mathbb{E}[\sigma] = \frac{1}{\lambda}$, so one can just approximate it as $\lambda = \frac{1}{\mathbb{E}[\sigma]}$

4.3 Denoising with normalized patches

The method chosen to speed up the denoising process is based on the possibility of factorizing the space of patches. By factoring out the mean and the standard deviation, one

should be able to use an entire equivalence class of patches for each patch in the database, thus reducing both the amount of space needed and the time to compute the solution.

As stated by Levin and Nadler [32], if $\mathbf{y}$ is a noisy version of the original patch $\mathbf{x}$, $p(x)$ its probability in the natural image patch space and σ^2 the variance of the noise, the best MMSE estimator is given by

$$\mu(\mathbf{y}) = \frac{\int \mathbb{P}(\mathbf{y}|\mathbf{x})\mathbf{x}p(\mathbf{x})d\mathbf{x}}{\int \mathbb{P}(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}} = \frac{\int e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}\mathbf{x}p(\mathbf{x})d\mathbf{x}}{\int e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}p(\mathbf{x})d\mathbf{x}} \tag{4.3}$$

Theorem 1. ***Normalized shotgun denoising (NSD) algorithm*** *Let $\mathbf{y}$ be the noisy realization of a $k \times k$ patch $\mathbf{x}$ with mean $\mu_{\mathbf{y}}$. Assume a white Gaussian noise with standard deviation σ and let $\tilde{\mathbf{x}}$ be the patch $\mathbf{x}$ normalized by mean and standard deviation, such as $\tilde{\mathbf{x}} = \frac{\mathbf{x}-\mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}}$. Finally, let λ be the estimated parameter for the exponential distribution of p^{σ}. Then assuming that the (non noisy) patch density satisfies (4.1), the MMSE estimator (4.3) has a nonbiased discrete approximation as*

$$\hat{\mathbf{y}} = \frac{\sum_i (\beta + \alpha_i\gamma_i)\,\tilde{\mathbf{x}}_i}{\sum_i \gamma_i} + \mu_{\mathbf{y}}\mathbf{1} \tag{4.4}$$

where x_i are all available patches in a (large) image database and

$$\begin{aligned} \alpha_i &= \frac{\langle \tilde{\mathbf{x}}_i, \mathbf{y}\rangle - \sigma^2\lambda}{k^2}, \qquad & \beta &= \frac{\sqrt{2}}{k}\sigma, \\ \gamma_i &= \mathrm{e}^{\left(\frac{\alpha_i}{\beta}\right)^2}\sqrt{\pi}\,\mathrm{erfc}\left(-\frac{\alpha_i}{\beta}\right) \end{aligned} \tag{4.5}$$

and erfc *stands for the complementary error function.*

Sketch of Proof. Using (4.2), equation (4.3) approximates to

$$\hat{\mathbf{y}} = \frac{\int_{\tilde{\Omega}} \left(\iint f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}})(\sigma_{\mathbf{x}}\tilde{\mathbf{x}} + \mu_{\mathbf{x}}\mathbf{1})d\mu_{\mathbf{x}}d\sigma_{\mathbf{x}}\right) d\tilde{p}(\tilde{\mathbf{x}})}{\int_{\tilde{\Omega}} \left(\iint f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}})d\mu_{\mathbf{x}}d\sigma_{\mathbf{x}}\right) d\tilde{p}(\tilde{\mathbf{x}})} \tag{4.6}$$

with

$$f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}}) = \mathrm{e}^{-\frac{\|\sigma_{\mathbf{x}}\tilde{\mathbf{x}}_i + \mu_{\mathbf{x}}\mathbf{1} - \mathbf{y}\|^2}{2\sigma^2} - \lambda\sigma_{\mathbf{x}}}\, p^{\mu}(\mu_{\mathbf{x}}) \tag{4.7}$$

By using the law of large numbers as in [32], the integration over the normalized patch space can be approximated with the average over a huge normalized patch database. Then, by separating equation (4.6) into the sum of the following two terms

$$\frac{\sum_i \tilde{\mathbf{x}}_i \iint \sigma_{\mathbf{x}} f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}})d\mu_{\mathbf{x}}d\sigma_{\mathbf{x}}}{\sum_i \iint f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}})d\mu_{\mathbf{x}}d\sigma_{\mathbf{x}}} \tag{4.8}$$

and

$$\frac{\sum_i \iint \mu_{\mathbf{x}} f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}})d\mu_{\mathbf{x}}d\sigma_{\mathbf{x}}}{\sum_i \iint f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}})d\mu_{\mathbf{x}}d\sigma_{\mathbf{x}}}\mathbf{1} \tag{4.9}$$

and rewriting

$$f(\tilde{\mathbf{x}}_i, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}}) = \mathrm{e}^{\left(\frac{\alpha_i}{\beta}\right)^2}\mathrm{e}^{-\left(\frac{\sigma_{\mathbf{y}}}{\beta}\right)^2}\mathrm{e}^{-\left(\frac{\sigma_{\mathbf{x}}-\alpha_i}{\beta}\right)^2}\mathrm{e}^{-\left(\frac{\mu_{\mathbf{x}}-\mu_{\mathbf{y}}}{\beta}\right)^2} p^{\mu}(\mu_{\mathbf{x}}) \tag{4.10}$$

and using the fact that, following [24], $p^{\mu}(\mu_{\mathbf{x}})$ can be expressed as a constant, we arrive to the final formula of (4.4). □

The denoising process is performed patch-wise and the results are aggregated using a simple average of the estimates of each pixel.

4.4 Results

Using the algorithm devised in section 4.3, a test was conducted to see the performance of this denoiser. Following the experiments of [32], 5000000 sample patches (of size 8×8) were extracted from the LabelMe dataset [50]. To further reduce the noise and to remove JPEG artifacts the images were low-passed and down-sampled by a factor of 8. The low-pass filter used was Gaussian with a standard deviation of 6, to avoid aliasing artifacts. This choice is motivated by [38].

The test was performed on two images, not included in the database, of 150×100 pixels. A Gaussian white noise with standard deviation 25 was added to the images, which were then denoised using NSD and other state-of-the-art methods for comparison. The computation took less than one day on an average PC. The results are shown in Fig. 4.1.

Although the result of the denoising process depends on the size of the database, the convergence of the result is fast, as can be seen from the plot of the PSNR as a function of the number of patches in Fig. 4.2.

While the results of this algorithm, at a first glance, may seem discouraging, it must be remembered that the algorithm operates patch-wise. Therefore, the information about the rest of the image is not used. This puts NSD on a different category with respect to the state of the art. Thanks to its peculiarity, it can nevertheless perform better than traditional denoising algorithms in those areas for which "similar patches" are not present in the image. This is tested in Fig. 4.3. As can be seen from the images, the algorithm presented in this paper outperforms the state of the art algorithm BM3D in patches with small, non-repeated textures, and could therefore be used, along a traditional denoising algorithm, to improve the denoising in "difficult" areas of the image.

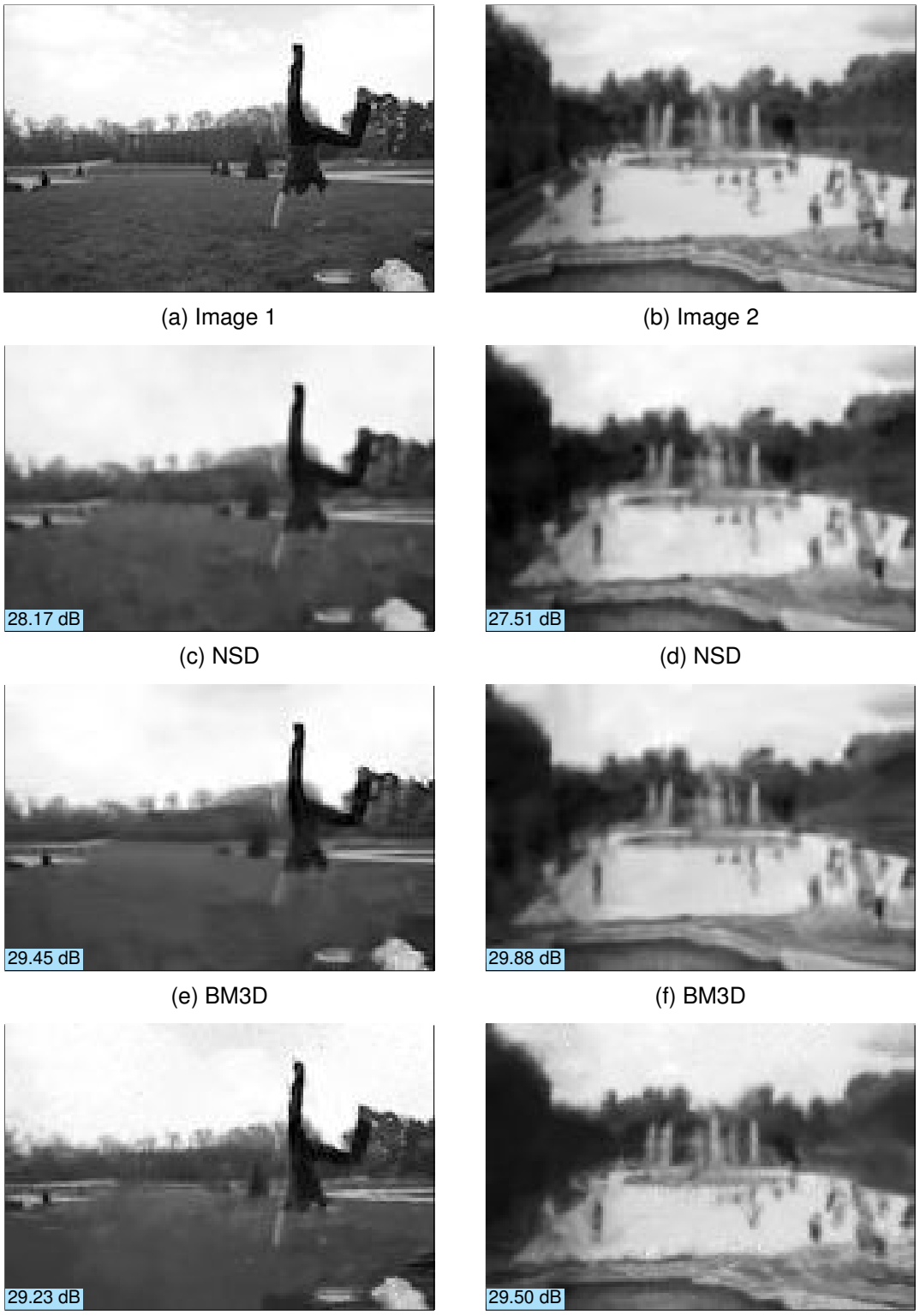
To further test this claim, we applied NSD only on patches that perform poorly with BM3D. This oracle gives an estimate of the best possible result attainable by combining the two algorithms. The oracle was a simple comparison with the original, non-noisy image. This, although clearly not applicable in the real case, demonstrates that a shotgun algorithm can effectively outperform the state of the art on difficult (non self-similar) image parts. The results of the "combined" denoising is shown in Table 4.1.

Method	Image 1	Image 2
NSD	28.17 dB	27.51 dB
BM3D	29.45 dB	29.88 dB
Combined	29.73 dB	30.30 dB

Table 4.1 – Results of combined denoising

As can be seen, the result obtained with this combination significantly outperforms BM3D.

Another simple test we performed was to fix the database size and to evaluate the original method [32] usign non-normalized patches with NSD. To do this, a set of 3500 random patches were denoised individually by each method. In table 4.2 it can be seen how our



(a) Image 1 (b) Image 2

(c) NSD (d) NSD

(e) BM3D (f) BM3D

(g) NL-Bayes (h) NL-Bayes

Figure 4.1 – Results and comparison of denoising methods (PSNR values in parenthesis).

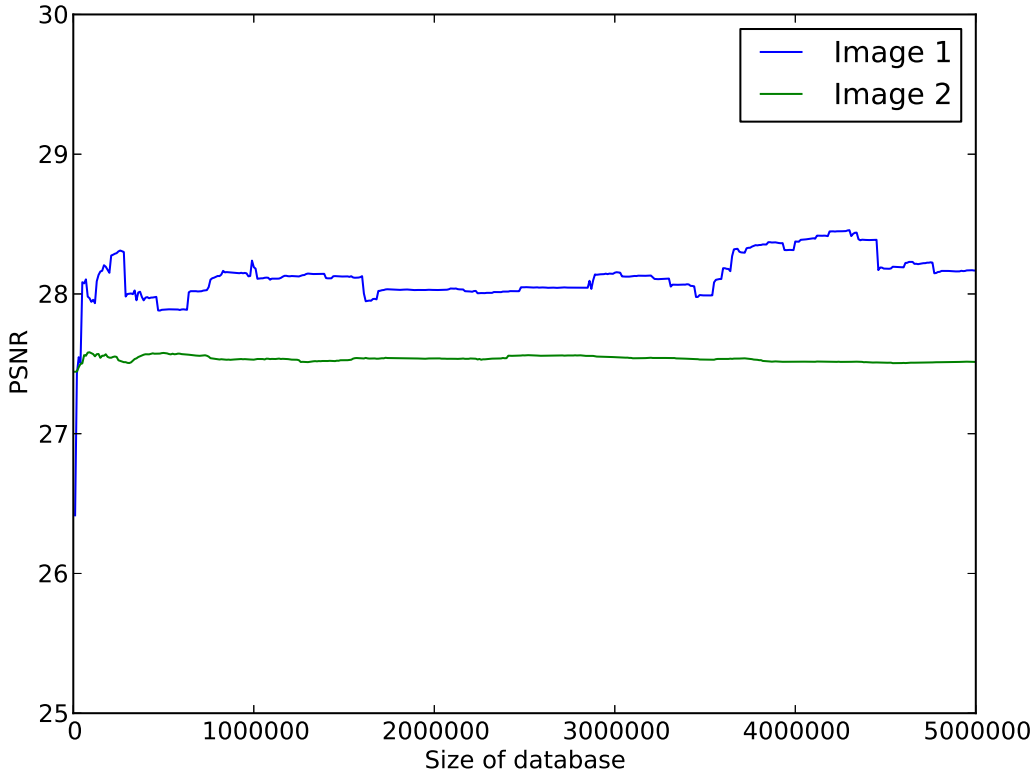


Figure 4.2 – Quality of the denoising, measured as PSNR, as a function of the database size.

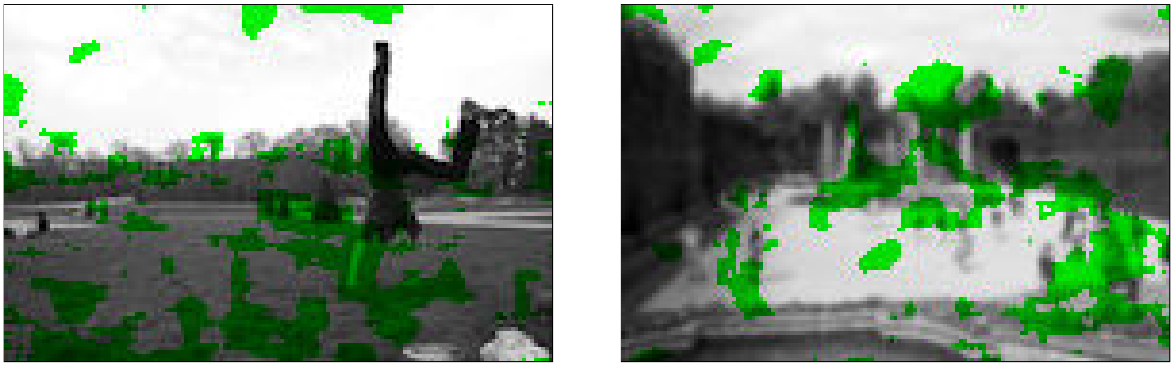
algorithm obtains improved results using the same amount of patches in the database. This improvement is logically diminished when bigger database sizes are used, however, for reasonably sized databases, our algorithm still manages to outperform Shotgun NLM.

DB Size	Shotgun method [32]	NSD
10000	23.47 dB	26.79 dB
100000	25.92 dB	27.32 dB
4000000	27.66 dB	28.06 dB

Table 4.2 – Average PSNR obtained after denoising 3500 random patches with a fixed Database size.

4.5 Conclusions

As can be seen from the results, the proposed algorithm as-is does not outperform BM3D, the current state-of-the-art. That said, since the method is local, it can be applied to single patches instead of entire images. The use of an external database for the denoising process can therefore be useful in particular applications. Other than that, the result of the combined



(a) Image 1 (b) Image 2

Figure 4.3 – Comparison between NSD and BM3D. The center of the patches in which a better result is achieved by the former are in green.

denoising is promising. It shows that there is still room for improving standard non-local denoising algorithms in patches with a low self-similarity, the "hard" patches for those algorithms.

Further developments should go in two different directions. First, a decision criterion must be elaborated to combine NSD with other algorithms without the aid of an oracle. Then, further efforts should be made to make this algorithm faster: while the factorization gives a huge boost in speed with respect to the original method of [32], NSD is still too slow compared to the other denoising algorithms.

Bibliography

[1] Andrew Adams. *High-dimensional gaussian filtering for computational photography*. Number July. Stanford University, 2011.

[2] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A Review of Image Denoising Algorithms, with a New One. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[3] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. The staircasing effect in neighborhood filters and its solution. *IEEE Transactions on Image Processing*, 15(6):1499–1505, 2006.

[4] Harold Christopher Burger and Stefan Harmeling. Improving denoising algorithms via a multi-scale meta-procedure. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6835 LNCS, pages 206–215, 2011.

[5] Harold Christopher Burger, Christian J. Schuler, and Stefan Harmeling. Learning how to combine internal and external denoising methods. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8142 LNCS of *Lecture Notes in Computer Science*, pages 121–130. Springer Berlin Heidelberg, 2013.

[6] Priyam Chatterjee and Peyman Milanfar. Is denoising dead? *IEEE Transactions on Image Processing*, 19(4):895–911, 2010.

[7] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, aug 2007.

[8] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image Denoising With Shape-Adaptive Principal Component Analysis. *SPARS*, 2008.

[9] Charles Alban Deledalle, Vincent Duval, and Joseph Salmon. Non-local methods with shape-adaptive patches (NLM-SAP). *Journal of Mathematical Imaging and Vision*, 43(2):103–120, 2012.

[10] Weisheng Dong, Guangming Shi, and Xin Li. Nonlocal image restoration with bilateral variance estimation: A low-rank approach. *IEEE Transactions on Image Processing*, 22(2):700–711, 2013.

[11] David L. Donoho and Jain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[12] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3):257–266, 2002.

[13] Sylvain Durand and Jacques Froment. Artifact free signal denoising with wavelets. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 6, pages 3685–3688. IEEE, 2001.

[14] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representation over learned dictionaries. *IEEE Transations on Image Processing*, 15(12):3736–3745, 2006.

[15] Francisco Estrada, David Fleet, and Allan Jepson. Stochastic Image Denoising. *Proceedings of the British Machine Vision Conference 2009*, pages 117.1—-117.11, 2009.

[16] Eduardo S. L. Gastal and Manuel M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics*, 31(4):1–13, 2012.

[17] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[18] D. Gnanadurai and V. Sadasivam. Image De-Noising Using Double Density Wavelet Transform Based Adaptive Thresholding Technique. *International Journal of Wavelets, Multiresolution and Information Processing*, 03(01):141–152, 2005.

[19] James Hays and Alexei a. Efros. Scene completion using millions of photographs. *Communications of the ACM*, 51(10):87, 2008.

[20] J. Huang and D. Mumford. Statistics of natural images and models. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 541–547, 1999.

[21] Claude Knaus. *Dual-Domain Image Denoising*. PhD thesis, Diss. Univ. Bern, 2013.

[22] Claude Knaus and Matthias Zwicker. Dual-domain image denoising. *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, (4):440–444, 2013.

[23] Claude Knaus and Matthias Zwicker. Progressive image denoising. *IEEE Transactions on Image Processing*, 23(7):3114–3125, 2014.

[24] Kostadin Koroutchev and José R Dorronsoro. Factorization and structure of natural 4x4 patch densities. *Optical Engineering*, 45(12):127003–127010, 2006.

[25] Vincent Le Guen. Cartoon + Texture Image Decomposition by the TV-L1 Model. *Image Processing On Line*, 4:204–219, 2014.

[26] Marc Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2012.

[27] Marc Lebrun, Antoni Buades, and Jean-michel Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. *Ipol*, 3:1–42, 2013.

[28] Marc Lebrun, Miguel Colom, Antoni Buades, and Jean-Michel Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21(April 2012):475–576, 2012.

[29] Marc Lebrun and Arthur Leclaire. An Implementation and Detailed Analysis of the K-SVD Image Denoising Algorithm. *Image Processing On Line*, 2012.

[30] Ann B. Lee, David Mumford, and Jinggang Huang. Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model. *International Journal of Computer Vision*, 41(1-2):35–59, 2001.

[31] Chul Lee, Chulwoo Lee, and Chang Su Kim. An MMSE approach to nonlocal image denoising: Theory and practical implementation. *Journal of Visual Communication and Image Representation*, 23(3):476–490, 2012.

[32] Anat Levin and Boaz Nadler. Natural image denoising: Optimality and inherent bounds. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2833–2840, 2011.

[33] Anat Levin, Boaz Nadler, Frédo Durand, and William T. Freeman. Patch complexity, finite pixel correlations and optimal denoising. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7576 LNCS(PART 5):73–86, 2012.

[34] Hong-qiao Li, Sheng-qian Wang, and Cheng-zhi Deng. New Image Denoising Method Based Wavelet and Curvelet Transform. *2009 WASE International Conference on Information Engineering*, 1, 2009.

[35] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2272–2279, 2009.

[36] Julien Mairal, Guillermo Sapiro, and Michael Elad. Learning Multiscale Sparse Representations for Image and Video Restoration. *Multiscale Modeling & Simulation*, 7(1):214–241, 2008.

[37] Y Meyer and R D Ryan. *Wavelets: Algorithms and Applications*. Society for Industrial and Applied Mathematics, 1993.

[38] Jean Michel Morel and Guoshen Yu. Is sift scale invariant? *Inverse Problems and Imaging*, 5(1):115–136, 2011.

[39] Inbar Mosseri, Maria Zontak, and Michal Irani. Combining the power of Internal and External denoising. *2013 IEEE International Conference on Computational Photography, ICCP 2013*, pages 1–9, 2013.

[40] Mukesh C Motwani, Mukesh C Gadiya, Rakhi C Motwani, and Frederick C Harris. Survey of Image Denoising Techniques. *GSPX*, pages 27–30, 2004.

[41] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[42] Nicola Pierazzo, Marc Lebrun, Martin Rais, Jean-Michel Morel, and Gabriele Facciolo. Non-Local Dual Image Denoising. *IEEE International Conference on Image Processing (ICIP)*, 2014.

[43] Nicola Pierazzo and Martin Rais. Boosting Shotgun Denoising By Patch Normalization. *IEEE International Conference on Image Processing (ICIP)*, 2013.

[44] Nicola Pierazzo, Martin Emilio Rais, Jean-Michel Morel, and Gabriele Facciolo. DA3D Support Website and Supplementary Material. http://dev.ipol.im/˜pierazzo/da3d, 2015.

[45] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003.

[46] Boshra Rajaei. An Analysis and Improvement of the BLS-GSM Denoising Method BLS-GSM Algorithm in Detail. *Image Processing On Line*, 4:44–70, 2014.

[47] Umesh Rajashekar and Eero P. Simoncelli. Multiscale Denoising of Photographic Images. In *The Essential Guide to Image Processing*, pages 241–261. 2009.

[48] Lars Lau Raket and Mads Nielsen. A splitting algorithm for directional regularization and sparsification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3094–3098. IEEE, 2012.

[49] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[50] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.

[51] Jean-Luc Starck, Emmanuel Jean Candes, and David L. Donoho. The Curvelet Transform for Image Denoising. *IEEE Transactions on Image Processing*, 11(6):670–684, 2002.

[52] Jeremias Sulam, Boaz Ophir, and Michael Elad. Image denoising through multi-scale learnt dictionaries. In *2014 IEEE International Conference on Image Processing, ICIP 2014*, pages 808–812, 2014.

[53] Hiroyuki Takeda, Sina Farsiu, and Peyman Milanfar. Higher Order Bilateral Filters and Their Properties. *Electronic Imaging*, pages 64980S–64980S–9, 2007.

[54] Hiroyuki Takeda, Sina Farsiu, and Peyman Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on Image Processing*, 16(2):349–366, 2007.

[55] Hossein Talebi and Peyman Milanfar. Global image denoising. *IEEE Transactions on Image Processing*, 23(2):755–768, 2014.

[56] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998.

[57] Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[58] Zhou Wang, A.C. Bovik, H.R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[59] Chen Wen-Hsiung, C Smith, and S Fralick. A Fast Computational Algorithm for the Discrete Cosine Transform. *Communications, IEEE Transactions on*, 25(9):1004–1009, 1977.

[60] Norbert Wiener. *The Extrapolation, Interpolation and Smoothing of Stationary Time Series*. The MIT Press, 1949.

[61] Leonid Yaroslavsky. Local adaptive image restoration and enhancement with the use of DFT and DCT in a running window. *... Applications in Signal and Image Processing IV, SPIE ...*, pages 1–13, 1996.

[62] Leonid P Yaroslavsky and Murray Eden. *Fundamentals of Digital Optics: Digital Signal Processing in Optics and Holography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996.

[63] Guoshen Yu and Guillermo Sapiro. DCT image denoising: a simple and effective image denoising algorithm. *Image Processing On Line*, 2011.

[64] Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Image modeling and enhancement via structured sparse model selection. *Proceedings - International Conference on Image Processing, ICIP*, pages 1641–1644, 2010.

[65] Huanjing Yue, Xiaoyan Sun, Jingyu Yang, and Feng Wu. CID: Combined Image Denoising in Spatial and Frequency Domains Using Web Images. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2933–2940, 2014.

[66] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 977–984, 2011.

[67] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. *Proceedings of the IEEE International Conference on Computer Vision*, pages 479–486, nov 2011.

université PARIS-SACLAY | École doctorale de mathématiques Hadamard (EDMH)

Titre : Quelque progrès en débruitage d'images

Mots clefs : Débruitage, Patches, Dual-Domain, Transformée de Fourier

Résumé : Cette thèse explore les dernières évolutions du débruitage d'images, et elle essaie de développer une vision synthétique des techniques utilisées jusqu'à présent. Elle aboutit à un nouvel algorithme de débruitage d'image évitant les artefacts et avec un meilleur PSNR que tous les algorithmes que nous avons pu évaluer. La première méthode que nous présentons est DA3D, un algorithme de débruitage fréquentiel avec guide, inspiré de DDID [Knaus-Zwicker 2013]. La surprise de cet algorithme, c'est que le débruitage fréquentiel peut battre l'état de l'art sans produire artefacts. Le deuxième résultat présenté est une méthode universelle de débruitage multi-échelle, applicable à tout algorithme. Le troisième problème sérieux que nous abordons est l'évaluation des algorithmes de débruitage. Il est bien connu que le PSNR n'est pas un indice suffisant de qualité. Un artefact sur une zone lisse de l'image est bien plus visible qu'une altération en zone texturée. Nous proposons une nouvelle métrique basée sur un *Smooth PSNR* et un *Texture PSNR*. Ces métriques sont finalement utilisées pour comparer les algorithmes de l'état de l'art avec notre algorithme final, qui combine les bénéfices du multi-échelle et du filtrage fréquentiel guidé. Les résultats étant très positifs, nous espérons que la thèse contribue à résoudre un vieux dilemme, pour lequel la méthode DDID avait apporté de précieuses indications : comment choisir entre le seuillage fréquentiel et les méthodes basées sur l'auto-similarité pour le débruitage d'images ? La réponse est qu'il ne faut pas choisir.

Title : Advances in Image Denoising

Keywords : Denoising, Patches, Dual-Domain, Fourier Transform

Abstract : This thesis explores the last evolutions on image denoising, and attempts to set a new and more coherent background regarding the different techniques involved. In consequence, it also presents a new image denoising algorithm with minimal artifacts and the best PSNR performance known so far. A first result that is presented is DA3D, a frequency-based guided denoising algorithm inspired form DDID [Knaus-Zwicker 2013]. This algorithm achieves good results not only in terms of PSNR, but also (and especially) with respect to visual quality. The second result presented is Multi-Scale Denoising, a multiscale framework applicable to any denoising algorithm. A third main issue addressed in this thesis is the evaluation of denoising algorithms. Experiences indicate that the PSNR is not always a good indicator of visual quality for denoising algorithms, since, for example, an artifact on a smooth area can be more noticeable than a subtle alteration in a texture. A new metric is proposed to improve on this matter. Instead of a single value, a "Smooth PNSR" and a "Texture PSNR" are presented. We found that the optimal solution for image denoising is the application of a frequency shrinkage, applied to regular regions only, while a multiscale patch based method serves as guide. This seems to resolve a long standing question for which DDID gave the first clue: what is the respective role of frequency shrinkage and self-similarity based methods for image denoising? We describe an image denoising algorithm that performs better in quality and PSNR than any other based on the right combination of both denoising principles.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France