Published in Image Processing On Line on 2013–06–17.
Submitted on 2012–05–22, accepted on 2013–02–14.
ISSN 2105–1232 © 2013 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2013.16

Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm

Marc Lebrun[1], Antoni Buades[2], Jean-Michel Morel[3]

[1] CMLA, ENS Cachan, France (marc.lebrun@cmla.ens-cachan.fr)
[2] Universitat de les Illes Balears, Spain (toni.buades@uib.es)
[3] CMLA, ENS Cachan, France (jean-michel.morel@cmla.ens-cachan.fr)

Communicated by Bartomeu Coll *Demo edited by* Miguel Colom

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:
A. Buades, M. Lebrun, and J.M. Morel. "A Non-local Bayesian Image Denoising Algorithm." *SIAM Journal on Imaging Sciences*, 2013. (to appear)

Abstract

This article presents a detailed implementation of the Non-Local Bayes (NL-Bayes) image denoising algorithm. In a nutshell, NL-Bayes is an improved variant of NL-means. In the NL-means algorithm, each patch is replaced by a weighted mean of the most similar patches present in a neighborhood. Images being mostly self-similar, such instances of similar patches are generally found, and averaging them increases the SNR. The NL-Bayes strategy improves on NL-means by evaluating for each group of similar patches a Gaussian vector model. To each patch is therefore associated a mean (which would be the result of NL-means), but also a covariance matrix estimating the variability of the patch group. This permits to compute an optimal (in the sense of Bayesian minimal mean square error) estimate of each noisy patch in the group, by a simple matrix inversion.

The implementation proceeds in two identical iterations, but the second iteration uses the denoised image of the first iteration to estimate better the mean and covariance of the patch Gaussian models. A discussion of the algorithm shows that it is close in spirit to several state of the art algorithms (TSID, BM3D, BM3D-SAPCA), and that its structure is actually close to BM3D. Thorough experimental comparison made in this paper also shows that the algorithm achieves the best state of the art on color images in terms of PSNR and image quality. On grey level images, it reaches a performance similar to the more complex BM3D-SAPCA (no color version is available for this last algorithm).

Source Code

The ANSI C implementation of NL-Bayes image denoising algorithm has been peer reviewed and accepted by IPOL. The source code, the code documentation, and the online demo are accessible at the IPOL web page of this article[1].

Keywords: denoising, patch, bayes

[1] https://doi.org/10.5201/ipol.2013.16

1 Introduction

Image denoising is the first step of any image processing chain. Early studies applied linear Wiener filters equivalent to a frequency reduction of the Fourier transform. These filters are more efficient when applied locally on the DCT (Yaroslavsky et al. [21, 20]). A recent and particularly efficient implementation of DCT denoising on color images is proposed by Yu et al. [22].

In the past two decades several studies have proposed non-linear variational methods like the total variation minimization (Rudin et al. [19]), for which a recent implementation is proposed by Getreuer [12]. Still more recently, several methods have expanded the Wiener filtering method to other linear transforms, in particular the wavelet transform (Donoho et al. [8]). All of the above mentioned methods rely on an underlying image regularity in a functional space. The latest denoising methods have attempted to take advantage of another regularity, the self-similar structures present in most images. The Nonlocal-means (NL-means) method (Buades et al. [1, 2]) seems to be one of the first methods of this kind. It proposed to look for similar patches of an image and average them. There is a faithful implementation of this method for color images [3]. Patch-based denoising methods have developed into attempts to model the patch space of an image, or of a set of images. Algorithms proposing parsimonious but redundant representations of patches on patch dictionaries are proposed by Elad et al. [10], Mairal et al. [15, 16], Yu et al. [23]. Lebrun et al [14] proposed a recent analysis and implementation of the KSVD method. This parsimonious decomposition method has become a paradigm for all image restoration tools, including also de-blurring or in-painting.

The BM3D method (Dabov et al. [5]) is probably the most efficient patch-based current method. It merges the local DCT thresholding-based method and the NL-means method based on patch comparison. Indeed, BM3D creates a 3D block with all patches similar to a reference patch, on which a threshold on the 3D transformed block is applied. We refer to its detailed analysis and implementation by Lebrun [13].

A more recent NL-means variant shares with BM3D the idea of applying a transform threshold to the 3D block. This method, due to Zhang et al. [24], replaces the DCT by an adaptive local linear transform, the principal component analysis (PCA). The method proceeds in two identical steps which can only be distinguished by the noise parameter that is used. Like BM3D the method creates an array of vectors with all patches similar to a reference patch. A linear minimum mean square error (LMMSE) method is applied on the obtained coefficients before applying the inverse transform. Unlike what is done in BM3D, only the estimate obtained for the reference pixel is kept. The second step attempts to remove the noise left by the first step. A similar enhancement for the BM3D method replacing the DCT by a local PCA on similar blocks (with adaptive shape) has also been considered by Dabov et al. [6]. Nevertheless, according to this paper, the performance gain with respect to BM3D is modest.

Last but not least, the Bayesian approaches for image denoising have been proposed as early as 1972 by Richardson [18]. Being first parametric and limited to rather restrictive Markov random field models [11], they have expanded recently to non-parametric methods. The seed for the recent non parametric estimation methods is a now famous algorithm to synthesize textures from examples [9]. The underlying Markovian assumption is that, in a textured image, the stochastic model for a given pixel $\mathbf{i}$ can predicted from a local image neighborhood P of $\mathbf{i}$, which we shall call "patch".

As we will see in this paper, the Bayesian approach can be merged with Fourier methods like BM3D, in a new method called NL-Bayes. A natural extension of this method, called NL-PCA in the following, can be seen as we described it as a fusion of BM3D and TSID (Two-Step Image Denoising), where NL-PCA begins and ends like BM3D, the only change being the use of the PCA instead of the DCT or Bi-orthogonal spline wavelet.

We shall take advantage of the similarity of the steps of the method with BM3D, and follow closely the description used for BM3D [13]. Like in BM3D, each step of the NL-Bayes algorithm is

realized in three parts: a) finding the image patches similar to a given image patch and grouping then in a 3D block; b) collaborative filtering; c) aggregation. The collaborative filtering is realized in two parts: a) applying Bayes' formula on the 3D block; and b) repositioning the 3D block. This 3D filtering is applied simultaneously on a group of 2D image blocks. Since these filtered patches overlap, many estimates are obtained, which need to be combined for each pixel. Aggregation is a particular averaging procedure used to take advantage of this redundancy.

The paper is organized as follows. We introduce the Bayesian method in Section 2. The developed image denoising algorithm is described in Section 3. The detailed study of its parameters and variation can be found in Section 5. Experimental results and comparison to several state-of-the-art denoising algorithms is given in Section 6. A detailed comparison with original PCA-based denoising methods is given in Section 7. A glossary at the end of this document gives a synopsis of the relevant notation.

A complete presentation of the theory underlying NL-Bayes, and a discussion of all related algorithms, is done in the companion paper [4], to appear in SIIMS.

2 Theory

This section presents a short derivation of the main formulas used in the algorithm. For a detailed analysis, see our SIIMS paper [4]. Given u the noiseless ideal image and $\tilde{u}$ the noisy image corrupted with additive white Gaussian noise of standard deviation σ so that

$$\tilde{u} = u + n, \tag{1}$$

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

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

where N is the total number of pixels in the image. Given a noiseless patch P of u with dimension $k \times k$ and $\tilde{P}$ an observed noisy version of P, the independence of noise realizations for the different pixels implies that

$$\mathbb{P}(\tilde{P} \mid P) = c.e^{-\frac{\|\tilde{P}-P\|^2}{2\sigma^2}}, \tag{3}$$

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

$$\mathbb{P}(P \mid \tilde{P}) = \frac{\mathbb{P}(\tilde{P} \mid P)\mathbb{P}(P)}{\mathbb{P}(\tilde{P})}. \tag{4}$$

$\tilde{P}$ being observed, this formula could be used to deduce the patch P maximizing the right term, viewed as a function of P. This is unfortunately not possible, unless we have a probability model for P. We shall now discuss how to proceed when all observed patches are noisy. Assume that the patches Q similar to P follow a Gaussian model where $\mathbf{C}_P$ denotes the covariance matrix of the patches similar to P and $\overline{P}$ the expectation of the patches similar to P. This means that

$$\mathbb{P}(Q) = c.\ e^{-\frac{(Q-\overline{P})^t \mathbf{C}_P^{-1}(Q-\overline{P})}{2}} \tag{5}$$

From (3) and (4) we obtain for each observed $\tilde{P}$ the following equivalence of problems:

$$\begin{aligned}\underset{P}{\text{Arg max}}\ \mathbb{P}(P \mid \tilde{P}) &\Leftrightarrow \underset{P}{\text{Arg max}}\ \mathbb{P}(\tilde{P} \mid P)\mathbb{P}(P)\\ &\Leftrightarrow \underset{P}{\text{Arg max}}\, e^{-\frac{\|P-\tilde{P}\|^2}{2\sigma^2}} e^{-\frac{(P-\overline{P})^t \mathbf{C}_P^{-1}(P-\overline{P})}{2}}\\ &\Leftrightarrow \underset{P}{\text{Arg min}}\ \frac{\|P-\tilde{P}\|^2}{\sigma^2} + (P-\overline{P})^t \mathbf{C}_P^{-1}(P-\overline{P}).\end{aligned}$$

This expression is not satisfactory, because the noiseless patch P and the patches similar to P cannot be observed directly. Yet, since we are observing the noisy version $\tilde{P}$, we can at least compute the patches $\tilde{Q}$ similar to $\tilde{P}$. An empirical covariance matrix can therefore be obtained with enough such observable samples $\tilde{Q}$. P and n being independent, we deduce from (1), under the assumption that $\mathbf{C}_{\tilde{P}}$ is a Gaussian vector that

$$\mathbf{C}_{\tilde{P}} = \mathbf{C}_P + \sigma^2\mathbf{I}; \qquad E\tilde{Q} = \overline{P}. \tag{6}$$

These relations assume that patches similar to $\tilde{P}$ have been searched in a neighborhood large enough to include all patches similar to P, but not too large either, to avoid containing outliers. A safe strategy for that is to search for similar patches in a distance slightly larger than the plausible distance caused by noise. If the above estimates are correct, the MAP (maximum *a posteriori* estimation) problem boils down by (6) to the feasible minimization problem:

$$\underset{P}{\text{Arg max}}\ \mathbb{P}(P \mid \tilde{P}) \Leftrightarrow \underset{P}{\text{Arg min}}\ \frac{\|P-\tilde{P}\|^2}{\sigma^2} + (P-\overline{\tilde{P}})^t(\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I})^{-1}(P-\overline{\tilde{P}}).$$

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

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

and therefore

$$P = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I}\right]\mathbf{C}_{\tilde{P}}^{-1}(\tilde{P}-\overline{\tilde{P}}) \tag{7}$$

Thus a restored patch $P^{\mathbf{basic}}$ can be obtained from the observed patch $\tilde{P}$ by the one step estimation

$$P^{\mathbf{basic}} = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I}\right]\mathbf{C}_{\tilde{P}}^{-1}(\tilde{P}-\overline{\tilde{P}}). \tag{8}$$

In a second step, where all patches have been denoised at the first step, all the denoised patches can be used again to obtain a better unbiased estimation $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ for $\mathbf{C}_P$, the covariance of the cluster containing P, and $\overline{\tilde{P}}^{\mathbf{basic}}$ a new estimation of $\overline{\tilde{P}}$, the average of patches similar to $\tilde{P}$. Indeed, the patch similarity is better estimated with the denoised patches, then sets of similar patches are more accurate. Then it follows from (6) and (8) that we can obtain a second better denoised patch,

$$P^{\mathbf{final}} = \overline{\tilde{P}}^{\mathbf{basic}} + \mathbf{C}_{\tilde{P}}^{\mathbf{basic}}\left[\mathbf{C}_{\tilde{P}}^{\mathbf{basic}} + \sigma^2\mathbf{I}\right]^{-1}(\tilde{P} - \overline{\tilde{P}}^{\mathbf{basic}}) \tag{9}$$

The computation of $\mathbf{C}_{\tilde{P}}$ and $\overline{\tilde{P}}$ will be discussed in section 3.

3 Implementation

3.1 BM3D

The algorithm developed in this paper is very similar to BM3D and also has two successive steps. In order to use this similarity as much as possible our notation and exposition order will be as close as possible to those used in our previous article on BM3D [13]. Here is a brief overview of BM3D:

Step 1 : First denoising loop on the image. We denote by $\tilde{P}$ the reference current noisy patch.

Grouping : Stacking up similar patches to the reference one, using a similarity threshold applied to the distance between patches in order to build the 3D block $\mathcal{P}(\tilde{P})$;

Collaborative Filtering : A 3D linear transform is applied to the 3D block, then a hard thresholding is applied to the coefficients, and finally the inverse 3D transform is applied. A weight is associated with the whole 3D block, depending on its sparsity;

Aggregation : A first basic estimate of the denoised image is obtained by doing a weighted aggregation of every estimate obtained in the preceding step for each pixel. This basic estimate will be denoted by $u^{\mathbf{basic}}$.

Step 2 : Second denoising step using the result of the first as "oracle".

Grouping : The distance between patches is computed on the basic estimate. Two 3D blocks are formed:

- $\mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}})$ by stacking up patches from the basic estimation $u^{\mathbf{basic}}$ and,
- $\mathcal{P}^{\mathbf{basic}}(\tilde{P})$ by stacking up patches in the same order from the original noisy image $\tilde{u}$.

Collaborative Filtering : A 3D transform is applied on both 3D blocks, followed by a Wiener filtering of the group $\mathcal{P}^{\mathbf{basic}}(\tilde{P})$ using the empirical oracular coefficients obtained from the group $\mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}})$, and finally by the inverse 3D transform. A weight is computed for the whole 3D block. It depends on the norm of the empirical Wiener coefficients;

Aggregation : A final estimate of the denoised image is obtained by using a weighted aggregation of every estimate obtained for each pixel. This final estimate will be denoted by $u^{\mathbf{final}}$.

3.2 Comparison of the Structure of NL-Bayes with BM3D

Both algorithms will be described for color images. They can also be applied on grey level images; the changes in that case will be indicated. Like BM3D, NL-Bayes is applied in two successive steps, the result of the first one serving as oracle for the second one:

1. the first step provides a basic estimate $u^{\mathbf{basic}}$ by using (8) during the *collaborative filtering*. Parameters in this step are denoted by the exponent **1**;

2. the second step is based both on the original noisy image $\tilde{u}$ and on the basic estimate obtained during the first step $u^{\mathbf{basic}}$ in order to apply (9) during the *collaborative filtering*. Parameters in this step are denoted by the exponent **2**.

Table 1 permits to compare steps between BM3D and NL-Bayes for color images.

3.3 The First Step of NL-Bayes

Only for the first step, the noisy image $\tilde{u}$ in the usual RGB color space is converted in a different color space where an independent denoising of each channel will not create noticeable color artifacts. Most algorithms use the YUV system which separates the luminance and chromatic parts of the image. BM3D, uses a linear transform multiplying the RGB vector by the matrix

$$Y_oU_oV_o = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

Step 1	BM3D	NL-Bayes
Preprocessing	Transformation to the $Y_0U_0V_0$ color space	Transformation to the $Y_0U_0V_0$ color space
Scanning step between two patches	$p_{\mathbf{1}} = 3$	$p_{\mathbf{1}} = 1$
Processing already used patches	Yes	No
Grouping		
Distance between patches	Channel Y_0 Normalized quadratic distance	Channel Y_0 Normalized quadratic distance
Similarity threshold	Fixed, tabulated according to σ	-
Patches kept	$N_{\mathbf{1}}$ best	$N_{\mathbf{1}}$ best
Patches ordered	No	No
3D groups formed	One for each channel Y_0, U_0 and V_0	one for each channel Y_0, U_0 et V_0
Collaborative Filtering		
3D transform	2D Bior1.5 on each patch followed by 1D Hadamard transform along the third dimension	-
Filter	Hard thresholding on the coefficients of the DCT	Bayesian, based on (12)
Weighting	Depending on the number of non-zero coefficients after Hard thresholding	-
Aggregation	Identical part	
Post processing	-	Transform to the RGB color space
Step 2	BM3D	NL-Bayes
Step between two patches	$p_{\mathbf{2}} = 3$	$p_{\mathbf{2}} = 1$
Process of an already used patch	Yes	No
Grouping		
Distance	Channel Y_0 of $u^{\mathbf{basic}}$ Normalized quadratic distance	All channels of $u^{\mathbf{basic}}$ Normalized quadratic distance
Similarity threshold	Fixed, tabulated according to the σ	Adaptive according to the distance of the $N_{\mathbf{2}}$-th best patch
Patches kept	$N_{\mathbf{1}}$ best	All
Patches ordered	No	No
3D groups formed	Two for each channel	Two
Collaborative Filtering		
3D transform	2D DCT then 1D Hadamard transform on both groups	-
Filter	Wiener filter using $u^{\mathbf{basic}}$ as oracle	Bayesian, based on (15)
Weighting	Depending on the norm of the empirical Wiener coefficients	-
Aggregation	Identical part	
Post processing	Transformation to the RGB color space	-

Table 1: Comparison of NL-Bayes with BM3D

We wrote the matrix above without normalization for readability, but the matrix is normalized to become orthonormal. In that way, σ still is the value of the standard deviation of the noise on each channel Y_0, U_0 and V_0. The transform increases the SNR of the geometric component, the Y_0 component being an average of the three colors. The geometric component is perceptually more important than the chromatic ones, and the presence of less noise permits a better performance of the algorithm in this component. The components U_0 and V_0 are differences of channels, which cancel or attenuate the signal. Thus a higher noise reduction on the chromatic components U_0 and V_0 is possible, due to their observable regularity.

We denote by $\tilde{P}$ the current reference patch with size $k_{\mathbf{1}} \times k_{\mathbf{1}}$ (seen as a column vector) of the noisy image $\tilde{u}$.

Grouping : The original noisy image $\tilde{u}$ is explored in a $\tilde{P}$-centered $n_{\mathbf{1}} \times n_{\mathbf{1}}$ neighborhood for patches $\tilde{Q}$ similar to the reference patch $\tilde{P}$. The normalized quadratic distance between each patch $\tilde{Q}$ of the neighborhood and the reference patch $\tilde{P}$ is computed as

$$d^2\left(\tilde{P}, \tilde{Q}\right) = \frac{\|\tilde{P} - \tilde{Q}\|_2^2}{(k_{\mathbf{1}})^2}.$$

This distance is computed on the luminance channel Y_0 only. All patches $\tilde{Q}$ of the neighborhood are sorted according to their distance to the reference patch $\tilde{P}$, and the $N_{\mathbf{1}}$ closest patches to $\tilde{P}$ are kept. Then three sets of similar patches – one for each channel – are built: $\mathcal{P}_{Y_0}(\tilde{P})$, $\mathcal{P}_{U_0}(\tilde{P})$ and $\mathcal{P}_{V_0}(\tilde{P})$. But the sets of similar patches for the chromatic channels are built with patches whose index are the same as for $\tilde{Q} \in \mathcal{P}_{Y_0}(\tilde{P})$, and in the same order. After this step, the same procedure is applied on each channel, but separately. For a sake of simplicity, we will describe the procedure for a generic channel.

Collaborative Filtering : Let $\mathcal{P}(\tilde{P})$ be the set of patches $\tilde{Q}$ similar to the reference patch $\tilde{P}$ obtained at the grouping step. We start by detecting if $\tilde{P}$ belongs to a homogeneous[2] area by processing the square of the standard deviation of $\mathcal{P}(\tilde{P})$:

$$\sigma_{\tilde{P}}^2 = \frac{M_{\mathbf{1}}}{M_{\mathbf{1}} - 1}\left(\frac{1}{M_{\mathbf{1}}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\sum_{x\in\tilde{Q}}\left(\tilde{Q}(x)\right)^2 - \left(\frac{1}{M_{\mathbf{1}}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\sum_{x\in\tilde{Q}}\tilde{Q}(x)\right)^2\right) \tag{10}$$

where $M_{\mathbf{1}} = N_{\mathbf{1}}(k_{\mathbf{1}})^2$. Since a huge number ($M_{\mathbf{1}}$) of realizations of the variable $u(i)$ is taken into account, in a homogeneous area this random variable should be very concentrated around its mean. Thus, fixing a threshold γ close to 1,

- if $\sigma_{\tilde{P}} \leq \gamma\sigma$, we can assume that with high probability $\tilde{P}$ belongs to a homogeneous area. In this case, the better result that can be obtained for the group is the average. Therefore, the estimate of all patches in the set of similar patches $\mathcal{P}(\tilde{P})$ is

$$\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), \forall x \in Q^{\mathbf{basic}}, Q^{\mathbf{basic}}(x) = \frac{1}{M_{\mathbf{1}}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\sum_{y\in\tilde{Q}}\tilde{Q}(y)$$

- else, $\tilde{P}$ is assumed to contain some signal, for which a Gaussian model is built. By the law of large numbers we have

$$\mathbf{C}_{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P}) - 1}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\left(\tilde{Q} - \overline{\tilde{P}}\right)\left(\tilde{Q} - \overline{\tilde{P}}\right)^t, \qquad \overline{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\tilde{Q}. \tag{11}$$

[2]By homogeneous we mean a flat area, where there is no geometric detail.

Once $\mathbf{C}_{\tilde{P}}$ and $\overline{\tilde{P}}$ have been computed, (8) yields an estimate for every patch in the set of similar patches,

$$\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), Q^{\mathbf{basic}} = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \beta_{\mathbf{1}}\sigma^2\mathbf{I}\right]\mathbf{C}_{\tilde{P}}^{-1}\left(\tilde{Q} - \overline{\tilde{P}}\right). \tag{12}$$

where $\beta_{\mathbf{1}}$ is a parameter of conservative attenuation close to 1.

Remark: As $\mathbf{C}_{\tilde{P}}$ is in principle real and symmetric positive definite, it should be invertible. Nevertheless it may sometimes (but seldom) be ill-conditioned. If the computation of the inverse does not end well, the algorithm simply sets $\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), Q^{\mathbf{basic}} = Q$.

Aggregation : When the collaborative filtering is achieved, an estimate is associated with every used patch. This yields a variable number of estimates for each pixel. To take advantage of these multiple estimates an aggregation must be done. Contrary to BM3D, this aggregation is not weighted. The final estimate after this first step is given by

$$u^{\mathbf{basic}}(x) = \frac{\displaystyle\sum_{\tilde{P}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})} \chi_{\tilde{Q}}(x) Q^{\mathbf{basic}}(x)}{\displaystyle\sum_{\tilde{P}}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})} \chi_{\tilde{Q}}(x)}$$

with $\chi_{\tilde{Q}}(x) = 1$ if and only if $x \in \tilde{Q}$, 0 otherwise.

Remark: For practical purposes, this computation is simplified by using two buffers $\boldsymbol{\nu}$ and $\boldsymbol{\delta}$, where respectively the numerator and the denominator are kept in memory

$$\forall \tilde{Q} \in \mathcal{P}(\tilde{P}), \forall x \in \tilde{Q}, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + Q^{\mathbf{basic}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + 1 \end{cases}$$

Thus the final estimate is simply obtained by dividing both buffers element-by-element.

Acceleration : To speed up the algorithm, each patch that has been used (and therefore denoised at least once) in a 3D group is no more considered as reference patch $\tilde{P}$. Nevertheless, it may be denoised several times, being potentially chosen in other groups.

Once $u_{Y_0}^{\mathbf{basic}}$, $u_{U_0}^{\mathbf{basic}}$ and $u_{V_0}^{\mathbf{basic}}$ have been obtained, inverting the color transform yields back $u^{\mathbf{basic}}$, the first estimate of the image in the RGB color space.

3.4 Second Step of NL-Bayes

In this second step of the algorithm a basic estimate $u^{\mathbf{basic}}$ of the denoised image is available. The second step follows exactly the same scheme as the first, but performs a Wiener filter of the original noisy image $\tilde{u}$, using as oracle the basic estimate $u^{\mathbf{basic}}$.

Grouping : The patch matching is processed on the basic estimate only. But this time the distance involves all channels, which are assumed denoised by the first step:

$$\forall Q^{\mathbf{basic}}, d^2(P^{\mathbf{basic}}, Q^{\mathbf{basic}}) = \frac{1}{N_c}\sum_{c=1}^{N_c} \frac{\|P_c^{\mathbf{basic}} - Q_c^{\mathbf{basic}}\|_2^2}{(k_{\mathbf{2}})^2} \tag{13}$$

where N_c denotes the number of channels in the image. As a difference with the first step where only $N_{\mathbf{1}}$ patches were kept, here a threshold τ is used to obtain a set of similar patches

$$\mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}}) = \{Q^{\mathbf{basic}} : d^2(P^{\mathbf{basic}}, Q^{\mathbf{basic}}) \leq \tau\},$$

with

- $\tau = \max(\tau_0, d_{N_\mathbf{2}})$;
- τ_0 is a fixed parameter;
- $d_{N_\mathbf{2}}$ is the distance between the reference patch and its $N_\mathbf{2}$-th best similar patches, sorted by their distance to $P^{\mathbf{basic}}$.

Thus, using τ, many more similar patches can be picked in homogeneous areas. A second set of similar patches from the noisy image $\tilde{u}$ is built

$$\mathcal{P}^{\mathbf{basic}}(\tilde{P}) = \{\tilde{Q} : d^2(P^{\mathbf{basic}}, Q^{\mathbf{basic}}) \leq \tau\},$$

by stacking up patches together in the same order as $\mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}})$.

Collaborative Filtering : Once 3D-blocks are built the collaborative filtering is applied. Then by the law of large numbers,

$$\begin{aligned} \mathbf{C}_P^{\mathbf{basic}} &\simeq \frac{1}{\#\mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}}) - 1} \sum_{Q^{\mathbf{basic}} \in \mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}})} \left(Q^{\mathbf{basic}} - \overline{\tilde{P}}^{\mathbf{basic}}\right)\left(Q^{\mathbf{basic}} - \overline{\tilde{P}}^{\mathbf{basic}}\right)^t, \\ \overline{\tilde{P}}^{\mathbf{basic}} &\simeq \frac{1}{\#\mathcal{P}^{\mathbf{basic}}(\tilde{P})} \sum_{\tilde{Q} \in \mathcal{P}^{\mathbf{basic}}(\tilde{P})} \tilde{Q}. \end{aligned} \tag{14}$$

Once $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ and $\overline{\tilde{P}}^{\mathbf{basic}}$ are computed, (9) yields an estimate for every patch in the set of similar patches

$$\forall \tilde{Q} \in \mathcal{P}^{\mathbf{basic}}(\tilde{P}), Q^{\mathbf{final}} = \overline{\tilde{P}}^{\mathbf{basic}} + \mathbf{C}_{\tilde{P}}^{\mathbf{basic}} \left[\mathbf{C}_{\tilde{P}}^{\mathbf{basic}} + \beta_\mathbf{2}\sigma^2\mathbf{I}\right]^{-1} \left(\tilde{Q} - \overline{\tilde{P}}^{\mathbf{basic}}\right) \tag{15}$$

Aggregation : When the collaborative filtering is achieved, an estimate is associated with every used patch and therefore a variable number of estimates for every pixel. Once again, contrarily to BM3D, this aggregation is not weighted. The final estimate after this second step is given by

$$u^{\mathbf{final}}(x) = \frac{\displaystyle\sum_{\tilde{P}} \sum_{\tilde{Q} \in \mathcal{P}^{\mathbf{basic}}(\tilde{P})} \chi_{\tilde{Q}}(x) Q^{\mathbf{final}}(x)}{\displaystyle\sum_{\tilde{P}} \sum_{\tilde{Q} \in \mathcal{P}^{\mathbf{basic}}(\tilde{P})} \chi_{\tilde{Q}}(x)}$$

with $\chi_{\tilde{Q}}(x) = 1$ if and only if $x \in \tilde{Q}$, 0 otherwise.

Remark: For practical purposes, this computation is simplified by using two buffers $\boldsymbol{\nu}$ and $\boldsymbol{\delta}$, where respectively the numerator and the denominator are kept in memory

$$\forall \tilde{Q} \in \mathcal{P}^{\mathbf{basic}}(\tilde{P}), \forall x \in \tilde{Q}, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + Q^{\mathbf{final}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + 1 \end{cases}$$

Thus the final estimate is simply obtained by dividing both buffers element-by-element. Once again, in order to speed up the algorithm, each patch used once in a 3D group is no more processed as reference patch. It can be used anyway several times as secondary patch in other 3D blocks.

3.5 NL-PCA: a Particular Case of NL-Bayes

The Bayesian approach on which NL-Bayes is based can lead to formulate another algorithm, which we shall call NL-PCA. Its idea is to perform a Principal Component Analysis (PCA) on the 3D group $\mathcal{P}(\tilde{P})$. This idea was first proposed by Zhang et al. [24] and is also studied in detail by Deledalle et al. [7]. NL-PCA is obtained by replacing in BM3D the fixed linear transform (DCT, bi-orthogonal spline wavelet) by a learned basis for each patch, obtained by PCA on the patches of the 3D block. Indeed, applying a PCA to the 3D group amounts to diagonalize its covariance matrix, and the eigenvectors give the adaptive basis. In continuation, the collaborative filtering and the aggregation parts can be applied exactly like in BM3D. The algorithm developed by Deledalle et al. [7] is very close to the first step of the NL-PCA algorithm described hereafter. This article will be described and commented in more detail in section 7.

Diagonalizing the covariance matrix $\mathbf{C}_{\tilde{P}}$, denoting the associated isometry by $R_{\mathbf{1}}$ and denoting by $S_{\mathbf{1}}(j)$ the squares of the associated eigenvalues, the restoration formula (8) becomes on the eigenfunction basis:

$$\left(R_{\mathbf{1}}(P^{\mathbf{basic}} - \overline{\tilde{P}})\right)_j = \frac{S_{\mathbf{1}}(j) - \sigma^2}{S_{\mathbf{1}}(j)} \left(R_{\mathbf{1}}(\tilde{P} - \overline{\tilde{P}})\right)_j .$$

The only (classic) variation with respect to this estimate is that $\mathbf{C}_{\tilde{P}}$ should **be** positive semi-definite. Thus $S_{\mathbf{1}}(j) - \sigma^2$ is replaced in the above formula by $(S_{\mathbf{1}}(j) - \sigma^2)^+$ and, instead of $-\sigma^2$, a more conservative attenuation is applied, $-\beta^2\sigma^2$ where an empirical β slightly larger than 1 accounts for the error of model. A still more conservative estimate is applied for large noises (typically if $\sigma \leq 40$), where the estimate becomes

$$\left(R_{\mathbf{1}}(P^{\mathbf{basic}} - \overline{\tilde{P}})\right)_j = \begin{cases} \left(R_{\mathbf{1}}(\tilde{P} - \overline{\tilde{P}})\right)_j & \text{if } S_{\mathbf{1}}(j) \geq \beta^2\sigma^2 \\ 0 & \text{otherwise.} \end{cases}$$

In the same way, in the second step, diagonalizing $\mathbf{C}^{\mathbf{basic}}_{\tilde{P}}$ and denoting the associated isometry by $R_{\mathbf{2}}$ and the squares of the associated eigenvalues by $S_{\mathbf{2}}(j)$, the restoration formula (9) becomes, without any alteration to the model,

$$\left(R_{\mathbf{2}}(P^{\mathbf{final}} - \overline{\tilde{P}}^{\mathbf{basic}})\right)_j = \frac{S_{\mathbf{2}}(j)}{S_{\mathbf{2}}(j) + \sigma^2} \left(R_{\mathbf{2}}(\tilde{P} - \overline{\tilde{P}}^{\mathbf{basic}}\right)_j$$

which retrieves exactly a classical Wiener filter based on the PCA of $\mathcal{P}^{\mathbf{basic}}(\tilde{P})$.

4 Influence of the Parameters on the Performance of NL-Bayes

We applied the previously described algorithm to noiseless images to which a simulated white noise had been added. Many images have been tested, but for the sake of simplicity only one result for each σ will be shown. To evaluate quantitatively the denoising results, two classic measurements have been used:

- The *Root Mean Square Error* (RMSE) between the reference image (noiseless) u_R and the denoised image u_D. The RMSE is

$$RMSE = \sqrt{\frac{\sum_{x \in X} \left(u_R(x) - u_D(x)\right)^2}{|X|}};$$

  the smaller it is, the better the denoising.

- the *Peak Signal to Noise Ratio* (PSNR) evaluated in decibels (dB) by

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE} \right);$$

  the larger it is, the better the denoising.

The question is: how to select the right values for the various parameters in the algorithm as described previously, and to evaluate their influence on the final result? The parameters of the method are:

- $k_{\mathbf{1}}$, $k_{\mathbf{2}}$: size of patches;
- $N_{\mathbf{1}}$, $N_{\mathbf{2}}$: maximum number of similar patches kept;
- $n_{\mathbf{1}}$, $n_{\mathbf{2}}$: search window size;
- γ: used to determine if a patch belongs to an homogeneous area;
- $\beta_{\mathbf{1}}$, $\beta_{\mathbf{2}}$: coefficient used during the collaborative filtering;
- τ_0: minimum threshold to determine similar patches during the second step.

It would be impossible to try all combinations of all parameters. Thus, the principle of the study is to assign to all parameters an optimal or robust value, while only one is varied. The parameters will therefore fixed in the following way:

- $k_{\mathbf{1}}$ and $k_{\mathbf{2}}$ as explained in Section 4.1;
- $N_{\mathbf{1}}$ and $N_{\mathbf{2}}$ as explained in Section 4.6;
- the homogeneous area trick is always used on the first step;
- $n_{\mathbf{1}} = 5 \times k_{\mathbf{1}}$;
- $n_{\mathbf{2}} = 5 \times k_{\mathbf{2}}$;
- $\gamma = 1.05$;
- $\beta_{\mathbf{1}} = 1.0$;
- $\beta_{\mathbf{2}} = \begin{cases} 1.2 & \text{if } \sigma < 50 \\ 1.0 & \text{otherwise.} \end{cases}$;
- $\tau_0 = 4$.

4.1 Influence of the Size of the Patches k_1 and k_2

The size of the patches influences the result, but unlike BM3D, NL-Bayes gives its best results for significantly smaller patch sizes. This fact may be the most surprising result of this comparison.

Unsurprisingly, from the examination of table 2 follows that the size of patches must increase with the noise value. According to these results, the following optimal values will be chosen for the size of patches:

σ	$0 \leq \sigma < 20$	$20 \leq \sigma < 50$	$50 \leq \sigma < 70$	$70 \leq \sigma$
$k_{\mathbf{1}}$	3	5	7	7
$k_{\mathbf{2}}$	3	3	5	7

$\sigma = 2$							$\sigma = 5$					
k_2	3		5		7		3		5		7	
k_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	**46.03**	**1.27**	45.96	1.28	45.88	1.30	**39.89**	**2.58**	39.83	2.60	39.75	2.62
5	**46.03**	**1.27**	45.92	1.29	45.83	1.30	**39.89**	**2.58**	39.77	2.62	39.68	2.65
7	46.00	1.28	45.89	1.29	45.78	1.31	39.87	2.59	39.73	2.63	39.61	2.67
$\sigma = 10$							$\sigma = 20$					
k_2	3		5		7		3		5		7	
k_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	35.38	4.34	35.34	4.36	35.27	4.40	31.16	7.06	**31.20**	**7.02**	31.15	7.06
5	**35.39**	**4.34**	35.29	4.39	35.20	4.43	**31.20**	**7.02**	31.14	7.07	31.08	7.12
7	35.36	4.35	35.24	4.41	35.11	4.48	31.16	7.06	31.10	7.10	30.99	7.20
$\sigma = 30$							$\sigma = 40$					
k_2	3		5		7		3		5		7	
k_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	28.81	9.25	**28.92**	**9.13**	28.90	9.15	27.18	11.16	27.33	10.97	27.34	10.95
5	28.90	9.15	28.89	9.16	28.85	9.21	27.33	10.97	**27.36**	**10.93**	27.34	10.95
7	28.85	9.21	28.85	9.21	28.76	9.30	27.31	10.99	27.35	10.94	27.28	11.03
$\sigma = 60$							$\sigma = 80$					
k_2	3		5		7		3		5		7	
k_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
3	24.90	14.51	25.30	13.85	25.38	13.73	23.15	17.74	23.74	16.58	23.94	16.20
5	25.27	13.90	25.46	13.60	**25.47**	**13.58**	23.78	16.50	24.02	16.05	24.08	15.94
7	25.26	13.92	25.45	13.62	25.44	13.63	23.83	16.41	24.09	15.92	**24.10**	**15.91**
$\sigma = 100$												
k_2	3		5		7							
k_1	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE						
3	21.79	20.75	22.62	18.86	22.88	18.30						
5	22.65	18.79	22.98	18.09	23.06	17.93						
7	22.75	18.58	23.06	17.93	**23.11**	**17.83**						

Table 2: Influence of the size of the patches k_1 and k_2. In **bold** the best result for a given σ.

4.2 Influence of γ

The parameter γ is used to determine if a set of similar patches belongs to a homogeneous area as defined in (10). This parameter must be fixed carefully. If its value is too big, small details in the image may be lost. If instead its value is too small, homogeneous area will not be denoised enough and artifacts can become conspicuous in these regions. Thus, although the PSNR gain is moderate, the visual impact of this step is important.

	$\gamma = 0.95$		$\gamma = 1.0$		$\gamma = 1.05$		$\gamma = 1.1$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**46.00**	**1.28**	45.99	1.28	**46.00**	**1.28**	**46.00**	**1.28**
5	39.88	2.58	**39.89**	**2.58**	**39.89**	**2.58**	**39.89**	**2.58**
10	**35.38**	**4.34**	**35.38**	**4.34**	**35.38**	**4.34**	35.37	4.35
20	31.20	7.02	31.23	7.00	**31.24**	**6.99**	31.23	7.00
30	28.83	9.22	28.90	9.15	28.91	9.14	**28.92**	**9.13**
40	27.25	11.07	27.35	10.94	**27.38**	**10.90**	27.37	10.91
60	25.33	13.80	25.44	13.68	**25.45**	**13.61**	25.40	13.69
80	24.12	15.86	**24.20**	**15.72**	24.16	15.79	24.04	16.01
100	23.15	17.75	**23.23**	**17.58**	23.19	17.65	22.99	18.08

Table 3: Influence of γ. In **bold** best result for a given σ.

One can deduce from table 3 that small variations on γ lead to significant variations for high noise. According to this study, γ can be fixed to 1.05, whatever the value of noise.

4.3 Influence of β_1

This parameter is used in (12) and influences the filtering during the first step. The theoretical value is $\beta_\mathbf{1} = 1.0$, but a study of its influence needs to be done to learn its best empirical value.

	$\beta_\mathbf{1} = 0.8$		$\beta_\mathbf{1} = 0.9$		$\beta_\mathbf{1} = 1.0$		$\beta_\mathbf{1} = 1.1$		$\beta_\mathbf{1} = 1.2$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.92	1.29	45.98	1.28	46.01	1.28	46.02	1.27	**46.03**	**1.27**
5	39.71	2.64	39.80	2.61	39.85	2.59	**39.88**	**2.59**	39.86	2.59
10	35.22	4.42	35.34	4.36	35.39	4.33	**35.40**	**4.33**	35.37	4.35
20	30.96	7.22	31.12	7.08	**31.20**	**7.02**	31.18	7.03	31.09	7.11
30	28.64	9.43	28.85	9.21	**28.92**	**9.13**	28.89	9.16	28.75	9.31
40	27.07	11.30	27.27	11.05	**27.34**	**10.95**	27.30	11.01	27.13	11.22
60	25.07	14.23	25.38	13.72	**25.46**	**13.59**	25.36	13.75	25.08	14.21
80	23.80	16.45	24.13	15.94	**24.19**	**15.73**	24.08	15.93	23.78	16.50
100	22.85	18.37	23.09	17.86	**23.12**	**17.80**	23.00	18.06	22.70	18.67

Table 4: Influence of β_1. In **bold** best result for a given σ.

The theoretical value is confirmed by this study (table 4). Thus $\beta_\mathbf{1}$ is fixed to 1.0, whatever the value of the noise.

4.4 Influence of β_2

This parameter is used in (15) influences the filtering during the first step. The theoretical value is $\beta_\mathbf{2} = 1.0$, but a study of its influence needs to be done to learn its best empirical value.

	$\beta_2 = 0.8$		$\beta_2 = 0.9$		$\beta_2 = 1.0$		$\beta_2 = 1.1$		$\beta_2 = 1.2$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.91	1.29	45.95	1.28	45.98	1.28	46.00	1.28	**46.01**	**1.27**
5	39.75	2.62	39.79	2.61	39.82	2.60	39.84	2.60	**39.86**	**2.59**
10	35.30	4.38	35.34	4.36	35.36	4.35	35.38	4.34	**35.39**	**4.33**
20	31.13	7.08	31.17	7.05	31.19	7.03	31.19	7.03	**31.20**	**7.02**
30	28.85	9.20	28.87	9.18	28.89	9.16	**28.90**	**9.15**	28.89	9.16
40	27.27	11.04	27.28	11.02	**27.31**	**10.99**	27.30	11.00	27.30	11.00
60	25.48	13.57	**25.49**	**13.56**	25.47	13.58	25.46	13.59	25.45	13.61
80	24.16	15.79	**24.17**	**15.77**	24.16	15.79	24.15	15.82	24.12	15.86
100	**23.12**	**17.80**	23.12	17.81	23.10	17.84	23.09	17.87	23.07	17.91

Table 5: Influence of β_2. In **bold** best result for a given σ.

Here again (table 5) the theoretical value seems to work well, but a slight gain can be obtained if β_2 is tabulated according to σ. The chosen value for β_2 will be 1.2 if $\sigma > 50$, 1.0 otherwise.

4.5 Influence of the Size of the Search Window n_1 and n_2

The size of the search window influences the grouping part of the algorithm. Since the total number of patches $\tilde{Q}$ contained in the neighborhood which needs to be sorted is proportional to the size of the search window, the computational time of the algorithm increases with n_1 and n_2. We would therefore like to minimize these numbers. Nevertheless, if the size of the search window is too small, the "similar" patches will not be that similar to the reference patch $\tilde{P}$. Thus it is necessary to find a good compromise between a good PSNR and a relatively small size for the search window. Moreover, the size of the search window is intuitively dependent on the size of patches k_1 and k_2. This is why n_1 and n_2 will be determined as a factor of k_1 and k_2, i.e. $n_1 = \alpha k_1$ and $n_2 = \alpha k_2$. For this comparison, the others parameters are fixed as usual.

	$\alpha = 3$		$\alpha = 4$		$\alpha = 5$		$\alpha = 6$		$\alpha = 7$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.95	1.28	46.00	1.28	46.01	1.28	**46.02**	**1.27**	46.01	1.28
5	39.74	2.63	39.82	2.60	39.84	2.59	**39.86**	**2.59**	**39.86**	**2.59**
10	35.19	4.43	35.33	4.37	35.36	4.35	35.40	4.33	**35.41**	**5.32**
20	30.98	7.20	31.17	7.05	31.23	6.99	31.30	6.94	**31.32**	**6.93**
30	28.63	9.44	28.85	9.20	28.92	9.13	28.99	9.06	**29.02**	**9.02**
40	27.05	11.33	27.30	11.00	27.38	10.90	27.46	10.80	**27.50**	**10.75**
60	25.13	14.12	25.34	13.79	25.42	13.66	25.52	13.51	**25.55**	**13.46**
80	23.85	16.37	24.09	15.93	24.18	15.77	24.25	15.63	**24.28**	**15.57**
100	22.85	18.35	23.04	17.97	23.13	17.79	23.19	17.67	**23.23**	**17.57**

Table 6: Influence of the size of the search window n_1 and n_2. In **bold** best result for a given σ.

It follows from the comparison table 6 that increasing the size of the search window improves the result by finding more similar patches. Accordingly the parameters are fixed to $n_1 = 7k_1$ and $n_2 = 7k_2$.

4.6 Influence of the Minimal Number of Closest Neighbors, N_1 and N_2

As we have to invert a matrix in (12) and (15), a minimal number of similar patches is needed. Otherwise this matrix will not been invertible. Thus, N_1 and N_2 have to be determined empirically. Moreover, they depend on the noisy image. For this reason, only the final chosen values are given here

$$N_1 = \begin{cases} 30 & \text{if } k_1 = 3 \\ 60 & \text{if } k_1 = 5 \\ 90 & \text{if } k_1 = 7 \end{cases} \qquad N_2 = \begin{cases} 30 & \text{if } k_2 = 3 \\ 60 & \text{if } k_2 = 5 \\ 90 & \text{if } k_2 = 7 \end{cases}$$

4.7 Influence of τ_0

This parameter is used only in the second step, to fix the minimum threshold between two similar patches. Indeed in the second step we got an estimate $u^{\mathbf{basic}}$ and the distances between patches are better estimated on $u^{\mathbf{basic}}$ than on $\tilde{u}$. Thus, in homogeneous areas we can allow for many more similar patches than n_2. The parameter τ_0 is voluntarily kept small, because otherwise patches which differ significantly from the reference patch would be considered similar.

	$\tau_0 = 0$		$\tau_0 = 2$		$\tau_0 = 4$		$\tau_0 = 8$		$\tau_0 = 16$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.99	1.28	**46.01**	**1.27**	46.00	1.28	45.94	1.29	45.81	1.30
5	39.88	2.58	**39.93**	**2.57**	39.91	2.58	39.85	2.59	39.68	2.64
10	35.36	4.35	35.44	4.31	**35.45**	**4.30**	35.39	4.34	35.21	4.42
20	31.17	7.05	31.28	6.96	**31.29**	**6.95**	31.25	6.99	31.08	7.12
30	28.82	9.23	29.01	9.03	29.02	9.02	**29.03**	**9.02**	28.89	9.16
40	27.18	11.16	27.40	10.87	27.44	10.82	**27.46**	**10.80**	27.34	10.95
60	25.43	13.64	25.55	13.46	**25.58**	**13.42**	25.54	13.47	25.44	13.64
80	24.08	15.94	24.18	15.75	**24.21**	**15.71**	24.17	15.78	24.00	16.10
100	23.16	17.72	23.28	17.48	**23.33**	**17.37**	23.31	17.42	23.08	17.88

Table 7: Influence of τ_0. In **bold** best result for a given σ.

Using the minimum threshold with a small value ($\tau_0 = 2$ or 4) is always better than ($\tau_0 = 0$). Moreover, as expected, a too large value ($\tau_0 = 16$) gives really worse results. According to this comparison (table 7), we shall set $\tau_0 = 4$.

4.8 Summary Table of the Best Parameters

Here is the summary table with the final chosen values for all parameters, depending on the value of the noise:

σ	$0 \leq \sigma < 20$	$20 \leq \sigma < 50$	$50 \leq \sigma < 70$	$70 \leq \sigma$
k_1	3	5	7	7
k_2	3	3	5	7
γ	1.05	1.05	1.05	1.05
β_1	1.0	1.0	1.0	1.0
β_2	1.2	1.2	1.0	1.0
n_1	21	35	49	49
n_2	21	21	35	49
N_1	30	60	90	90
N_2	30	30	60	90
τ_0	4	4	4	4

5 A Detailed Study of the Algorithm

This part discusses several sound variants for each step of the algorithm. It gives experimental evidence that the choices taken in the algorithm are the best in terms of PSNR.

5.1 Grouping

5.1.1 Color Space Transform

The $Y_0U_0V_0$ space will be compared to RGB, for both steps. When RGB is chosen in the algorithm,

- the distance is computed on all channels like in (13);
- the collaborative filtering is done on each channel separately;
- $N_{\mathbf{1}}$ is increased to avoid having a non-invertible matrix.

Step 1 / Step 2	RGB / RGB		$Y_0U_0V_0$ / RGB		RGB / $Y_0U_0V_0$		$Y_0U_0V_0$ / $Y_0U_0V_0$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**46.10**	**1.26**	46.04	1.27	45.83	1.30	45.80	1.31
5	**39.89**	**2.58**	39.86	2.59	39.66	2.65	39.64	2.66
10	35.39	4.33	**35.42**	**4.32**	35.22	4.42	35.25	4.41
20	31.10	7.10	**31.30**	**6.94**	31.00	7.19	31.18	7.03
30	28.73	9.33	**28.93**	**9.11**	28.65	9.41	28.85	9.20
40	27.23	11.08	**27.47**	**10.78**	27.18	11.15	27.43	10.84
60	*23.87*	*16.34*	**25.54**	**13.48**	*23.82*	*16.43*	25.52	13.50
80	*22.96*	*18.13*	24.24	15.66	*22.92*	*18.23*	**24.24**	**15.65**
100	*22.14*	*19.93*	23.24	17.56	*22.10*	*20.02*	**23.25**	**17.54**

Table 8: Color space transform. In **bold** the best PSNR for a given σ. In *italic* results with many observed non-invertible matrices in the first step.

Despite the fact that $N_{\mathbf{1}}$ has been increased in the case where RGB is chosen for the first step, in practice the matrix $\mathbf{C}_{\tilde{P}}$ is often not invertible. This explains why the result is that bad for high values of the noise when RGB is chosen for the first step (table 8).

5.2 Collaborative Filtering

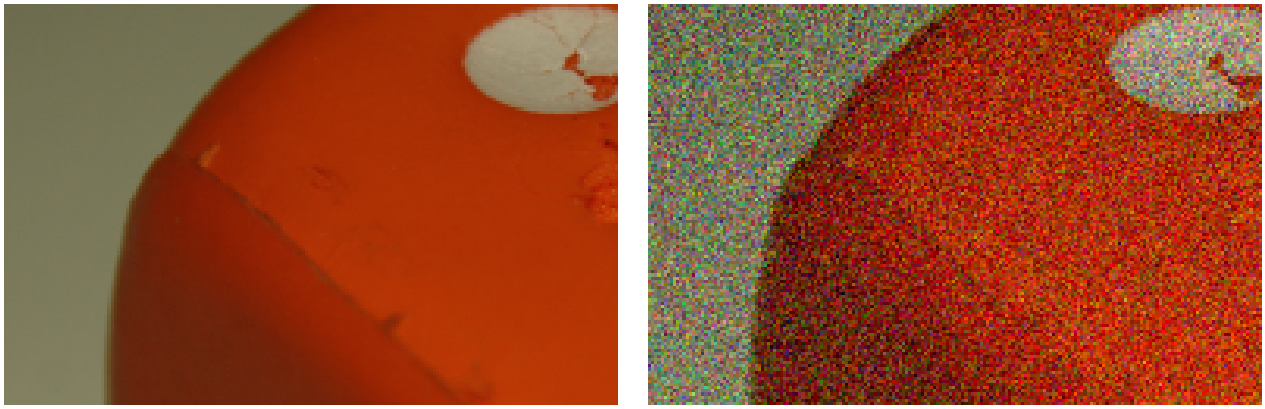
5.2.1 The Homogeneous Area Criterion

One of the innovations in NL-Bayes is the detection of the homogeneous areas in the first step. In order to show its relevance, table 9 has some results with and without this criterion, in both steps:

One can see that the homogeneity detection is useful for medium and high values of the noise variance. On an image with many homogeneous areas, the application of this detection avoids artifacts, as one can see in figure 1 for a noise standard deviation equal to 20.

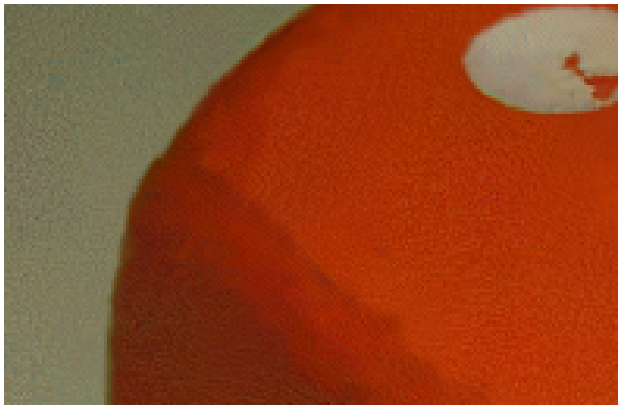
5.2.2 Diagonalizing the Covariance Matrix

As presented in section 3.5, $\mathbf{C}_{\tilde{P}}$ and $\mathbf{C}_{\tilde{P}}^{\mathbf{basic}}$ can be diagonalized with the use of a PCA on the 3D block $\mathcal{P}(\tilde{P})$ and $\mathcal{P}^{\mathbf{basic}}(\tilde{P})$, which leads to an algorithm which we called NL-PCA. The principal difference between NL-PCA and NL-Bayes is in the collaborative filtering part, as detailed in 3.5.



Original image

Noisy image $\sigma = 30$



No / No | PSNR = 33.93 / RMSE = 5.13

Yes / No | PSNR = 38.82 / RMSE = 2.92

No / Yes | PSNR = 38.69 / RMSE = 2.96

Yes / Yes | PSNR = 38.36 / RMSE = 3.08

Figure 1: Results with and without the homogeneous area criterion.

Step 1 / Step 2	No / No		Yes / No		No / Yes		Yes / Yes	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**46.04**	**1.27**	**46.04**	**1.27**	**46.04**	**1.27**	**46.04**	**1.27**
5	39.89	2.58	**39.90**	**2.57**	39.88	2.58	39.89	2.58
10	35.36	4.35	**35.39**	**4.33**	35.37	4.34	35.35	4.35
20	31.05	7.14	31.27	6.96	**31.29**	**6.95**	31.26	6.98
30	28.68	9.38	**29.01**	**9.04**	**29.01**	**9.04**	28.97	9.08
40	27.05	11.32	**27.48**	**10.78**	27.47	10.79	27.42	10.85
60	25.39	13.70	**25.58**	**13.41**	25.55	13.45	25.45	13.62
80	24.14	15.83	**24.26**	**15.62**	24.16	15.80	23.98	16.12
100	23.09	17.86	**23.27**	**17.49**	23.15	17.73	22.93	18.20

Table 9: The homogeneous area criterion. In **bold** the best PSNR for a given σ.

All the rest is exactly the same. We compare here the results of NL-PCA to NL-Bayes on three (noiseless) images (figure 2), to which noise was added.

Dice

Flowers

Traffic

Figure 2: Images used for the comparison of NL-PCA and NL-Bayes.

Table 10 shows that NL-Bayes is slightly better than NL-PCA. Nevertheless, the results of NL-PCA could be improved by adapting the parameters, and adding a weight to the aggregation part, like BM3D does. Indeed, the values of N_1 and N_2 are not adapted to NL-PCA: this parameter has large values to avoid non-invertible matrices, but the PCA can be hedged if patches in $\mathcal{P}(\tilde{P})$ are not that similar. Moreover, β_1 and β_2 need to be adapted too. NL-Bayes is faster than NL-PCA by an

	Dice				Flowers				Traffic			
	NL-PCA		NL-Bayes		NL-PCA		NL-Bayes		NL-PCA		NL-Bayes	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	48.85	0.92	**49.08**	**0.90**	47.59	1.06	**47.77**	**1.04**	45.00	1.43	**45.25**	**1.39**
5	45.84	1.30	**45.92**	**1.29**	43.35	1.73	**43.41**	**1.72**	39.42	2.73	**39.59**	**2.67**
10	43.17	1.77	**43.31**	**1.74**	39.66	2.65	**39.83**	**2.60**	35.40	4.33	**35.49**	**4.29**
20	**40.68**	**2.36**	40.56	2.39	36.33	3.89	**36.42**	**3.85**	31.28	6.96	**31.48**	**6.80**
30	**38.83**	**2.92**	38.81	2.92	34.16	5.00	**34.20**	**4.97**	29.20	8.84	**29.34**	**8.70**
40	**37.35**	**3.46**	**37.35**	**3.46**	32.52	6.03	**32.59**	**5.98**	27.80	10.39	**27.87**	**10.30**
60	35.60	4.23	**35.62**	**4.22**	30.72	7.42	**30.84**	**7.32**	25.77	13.12	**26.01**	**12.77**
80	34.53	4.79	**34.60**	**4.75**	29.28	8.76	**29.47**	**8.57**	24.62	14.98	**24.75**	**14.76**
100	33.46	5.41	**33.48**	**5.40**	28.15	9.98	**28.26**	**9.85**	23.76	16.54	**23.85**	**16.37**

Table 10: Diagonalizing the covariance matrix. In **bold** the best PSNR for a given σ.

average factor of 50%, due to the fact that there is only one matrix inversion, and not a PCA.

5.2.3 Ideal Wiener and Upper Bounds for the Performance

There is a significantly PSNR improvement by using the second step, because the covariance matrix is better estimated. To judge the contribution of this second step better, it is possible to compare it with an ideal Wiener filter, which is obtained when the original noise-free image is taken as oracle reference, i.e. as the output of the first step. This ideal Wiener filter is the best possible estimate for the second step of this algorithm. It is therefore interesting to see how far we stand from this ideal estimate with the current one, in table 11.

	Algorithm		Ideal Wiener	
σ	PSNR	RMSE	PSNR	RMSE
2	45.98	1.28	**47.40**	**1.09**
5	39.89	2.58	**41.50**	**2.15**
10	35.45	4.30	**37.28**	**3.49**
20	31.26	6.97	**33.42**	**5.44**
30	28.96	9.09	**31.42**	**6.85**
40	27.46	10.81	**30.13**	**7.95**
60	25.58	13.41	**27.97**	**10.19**
80	24.29	15.57	**26.66**	**11.85**
100	23.25	17.55	**25.80**	**13.08**

Table 11: Ideal Wiener and upper bounds for the performance. In **bold** the best PSNR for a given σ.

Of course, the Wiener filter using the noise-free image as oracle is better than the filter using the basic estimate obtained after the first step. One can see that there is a large room for improvement for this first step, of 2 to 3 dBs. The images in figures 3, 4 and 5 compare the visual performance for different values of the noise.

5.3 The "Paste" Option

To speed up the algorithm, a paste trick has been used. Whenever a patch $\tilde{Q} \in \mathcal{P}(\tilde{P})$ has an estimate, it is no more processed as a reference patch. One could think that this trick produces artifacts and has an impact on the PSNR. To check that, we will compare for both steps this paste option,

Original image

Noisy image ($\sigma = 10$)

NL-Bayes ($\sigma = 10$)

Ideal Wiener ($\sigma = 10$)

Figure 3: NL-Bayes and ideal Wiener, $\sigma = 10$.

Original image

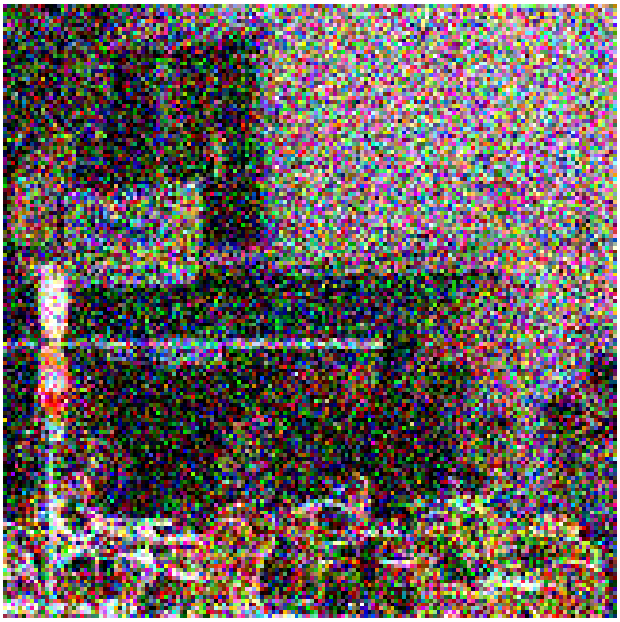
Noisy image ($\sigma = 30$)

NL-Bayes ($\sigma = 30$)

Ideal Wiener ($\sigma = 30$)

Figure 4: NL-Bayes and ideal Wiener, $\sigma = 30$.

Original image



Noisy image ($\sigma = 80$)

NL-Bayes ($\sigma = 80$)

Ideal Wiener ($\sigma = 80$)

Figure 5: NL-Bayes and ideal Wiener, $\sigma = 80$.

denoted by "*paste*" in the following, and another trick used in BM3D. This other trick, denoted by "*step*" divides approximately the number of processed reference patches by nine, by taking a 3 pixels scanning step row and column.

Step 1 / Step 2	*Step* / *Step*		*Paste* / *Step*		*Step* / *Paste*		*Paste* / *Paste*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.99	1.28	46.00	1.28	**46.03**	**1.27**	**46.03**	**1.27**
5	39.81	2.61	39.84	2.59	**39.87**	**2.59**	**39.87**	**2.59**
10	35.33	4.36	35.40	4.33	35.41	4.32	**35.43**	**4.31**
20	31.25	6.98	31.27	6.96	31.30	6.94	**31.31**	**6.93**
30	28.97	9.08	28.98	9.06	29.01	9.03	**29.02**	**9.02**
40	27.35	10.94	27.39	10.89	27.40	10.87	**27.43**	**10.84**
60	25.47	13.58	25.52	13.51	25.51	13.53	**25.53**	**13.48**
80	24.16	15.79	24.22	15.68	24.20	15.73	**24.24**	**15.66**
100	23.24	17.55	**23.28**	**17.47**	23.26	17.52	23.27	17.50

Table 12: The "paste" option. In **bold** the best PSNR for a given σ.

There is no significant PSNR loss by using both acceleration tricks (table 12). The fact that the use of the "*paste*" option improves things could be explained by the fact that in a area with many details, patches are very different in a very small area. Then almost every patch will be treated as a reference patch, while with the "*step*" trick only 1 patch out of 9 will be treated, and then very small details will be processed worse. The use of this "*paste*" trick speeds up a bit more the algorithm than the "*step*" trick, by almost 30%, depending on the percentage of homogeneous areas in the noisy image.

5.4 Influence of the Second Step

Using a second step to obtain a better covariance matrix improves a lot the final result, as we can see in table 13.

	First step		Second step	
σ	PSNR	RMSE	PSNR	RMSE
2	45.65	1.33	46.03	1.27
5	39.41	2.73	39.88	2.58
10	34.84	4.62	35.45	4.30
20	30.47	7.64	31.24	6.99
30	28.15	9.97	28.99	9.06
40	26.58	11.95	27.46	10.80
60	24.15	15.80	25.50	13.53
80	22.87	18.32	24.25	15.64
100	21.92	20.43	23.27	17.51

Table 13: Influence of the second step.

6 Comparison with Several Classic and Recent Methods

In order to evaluate the real capacity of NL-Bayes, a fair and precise comparison with other state-of-the-art methods had to be done. The four first considered methods have public and commented

implementations in Image Processing On Line [13, 22, 14, 3]. Thus, the experiments below can be verified on line for five of the compared algorithms.

- BM3D;
- DCT denoising;
- NL-means;
- K-SVD;
- BM3D-SAPCA (only for grey level images) as described by Dabov et al. [6];
- BLS-GSM as described bu Portilla et al. [17].

The following study has been led on the noise-free color images ($\sigma_{real} << 1$) shown in figure 6. All algorithms have been processed on the same noisy images obtained from noiseless images (saved in real values and not sampled on $[0, 255]$)):

6.1 Comparative Table

6.1.1 Color Images

According to the results (tables 14 and 15), one can observe that the comparative performance of the methods is quite independent of the noise value. The summary table below shows a mean of the scores over all test images[3].

Methods	NL-Bayes	BM3D	DCT	K-SVD	NL-means	BLS-GSM
Mean	**33.52**	32.98	31.77	32.17	31.70	NA

6.1.2 Grey Level Images

For grey level images, the size of patches needs to be increased for both steps in NL-Bayes. Then k_1 and k_2 are set to 5 for small values of noise, instead of 3. The other parameters are identical to the color case.

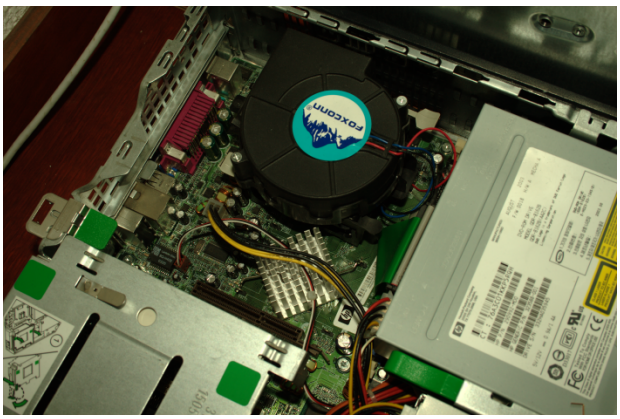
According to the results in tables 16 and 17, one can observe again that the comparative performance of the methods is quite independent of the noise value. Here is a summary table showing a mean of the PSNR scores over all test images:

Methods	NL-Bayes	BM3D	DCT	K-SVD	NL-means	BLS-GSM	BM3D-SAPCA
Mean	**32.02**	31.79	30.96	30.40	30.59	NA	NA

For slight values of the noise, BM3D-SAPCA performs better in terms of PSNR, and also visually. Thus, it appears that using shape adaptive patches slightly improves the performance. Edges appear more "straight" with BM3D-SAPCA than with NL-Bayes. Unfortunately one can also observe several oscillating artifacts (cosine outliers) on the results in figure 7. Nevertheless, for medium and large values of the noise, due to its inherent artifacts, BM3D-SAPCA has slightly less convincing results. We also noticed that using the same homogeneous area criterion for grey level images than for color images, but with three times less samples blurs the result of NL-Bayes. This homogeneity criterion should be refined for grey level images.

[3] As Alley and Trees are too big for BLS-GSM, summary results can not be shown for the BLS-GSM method

Alley



Computer

Dice

Flowers

Girl

Traffic

Trees

Valldemossa

Figure 6: Noise-free color images.

$\sigma = 2$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	**45.39**	44.95	44.54	41.22	42.79	NA
Computer	**45.88**	45.21	44.55	44.80	44.07	44.69
Dice	**49.07**	48.85	48.48	48.03	48.52	48.59
Flowers	**47.77**	47.30	47.14	47.32	46.41	47.12
Girl	**47.64**	47.38	46.94	47.24	46.95	47.14
Traffic	**45.26**	44.58	44.22	43.52	43.58	44.15
Trees	**43.51**	43.08	42.89	37.46	42.23	NA
Valldemossa	**45.14**	44.71	44.42	38.96	43.34	44.41
Mean	**46.21**	45.76	45.40	43.57	44.74	NA
$\sigma = 5$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	**39.35**	38.96	38.29	38.42	37.28	NA
Computer	**40.73**	39.98	38.98	39.58	38.99	39.30
Dice	**45.95**	45.81	45.06	45.29	45.11	45.21
Flowers	**43.39**	43.00	42.58	43.11	42.10	42.76
Girl	**44.23**	44.05	43.36	43.57	43.48	43.70
Traffic	**39.59**	38.67	38.09	38.78	37.61	38.10
Trees	**36.67**	36.09	35.66	35.57	34.69	NA
Valldemossa	**38.77**	38.35	37.89	37.91	35.97	38.02
Mean	**41.08**	40.61	39.99	40.28	39.40	NA
$\sigma = 10$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	**35.12**	34.83	33.92	34.31	33.59	NA
Computer	**36.98**	36.26	35.00	35.76	35.54	35.47
Dice	**43.32**	43.05	41.84	41.72	41.97	42.21
Flowers	**39.82**	39.48	38.61	39.35	38.46	39.10
Girl	**41.72**	41.45	40.35	40.31	40.48	41.14
Traffic	**35.47**	34.56	33.76	34.72	33.98	33.92
Trees	**31.89**	31.25	30.68	31.05	29.56	NA
Valldemossa	**34.14**	33.79	33.17	33.31	32.13	33.41
Mean	**37.31**	36.83	35.92	36.32	35.71	NA
$\sigma = 20$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	**31.37**	31.21	30.13	30.55	29.94	NA
Computer	**33.27**	32.69	31.08	31.94	31.61	31.89
Dice	**40.55**	39.94	37.94	37.30	38.20	39.00
Flowers	**36.33**	35.86	34.41	35.28	34.38	35.34
Girl	**39.10**	38.72	36.74	36.42	36.92	38.49
Traffic	**31.49**	30.83	29.79	30.70	30.11	30.14
Trees	**27.51**	26.91	26.15	26.87	26.35	NA
Valldemossa	**29.86**	29.59	28.69	29.08	28.44	29.17
Mean	**33.68**	33.22	31.87	32.27	31.99	NA

Table 14: Comparative results on color images from $\sigma = 2$ to $\sigma = 20$. In **bold** the best PSNR for a given image.

$\sigma = 30$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	**29.37**	29.34	28.20	28.60	27.82	NA
Computer	**31.12**	30.70	28.75	29.88	29.24	29.90
Dice	**38.80**	37.95	34.69	36.44	36.85	37.05
Flowers	**34.22**	33.77	31.98	33.58	32.26	33.19
Girl	**37.32**	36.98	33.73	35.40	35.55	36.91
Traffic	**29.35**	28.85	27.64	28.59	27.71	28.20
Trees	**25.22**	24.67	23.81	24.53	23.78	NA
Valldemossa	**27.53**	27.30	26.34	26.79	25.87	26.97
Mean	**31.62**	31.19	29.39	30.48	29.89	NA
$\sigma = 40$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	28.05	**28.07**	26.89	27.27	26.47	NA
Computer	**29.50**	29.12	26.92	28.22	27.52	28.52
Dice	**37.26**	36.35	32.42	34.60	35.16	35.50
Flowers	**32.63**	32.11	30.23	31.92	30.51	31.68
Girl	**36.05**	35.70	31.26	33.81	34.13	35.61
Traffic	**27.86**	27.46	26.14	27.14	26.20	26.93
Trees	**23.67**	23.18	22.30	23.06	22.40	NA
Valldemossa	**26.02**	25.80	24.88	25.32	24.44	25.50
Mean	**30.13**	29.72	27.63	28.92	28.35	NA
$\sigma = 60$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	26.37	**26.40**	25.49	25.66	24.66	NA
Computer	**27.52**	27.04	25.57	26.28	25.26	26.68
Dice	**35.71**	34.09	32.08	32.94	33.26	33.61
Flowers	**30.80**	29.94	28.81	30.04	28.27	29.70
Girl	**34.71**	33.71	31.94	32.53	32.61	34.08
Traffic	**25.98**	25.75	24.74	25.36	24.27	25.26
Trees	**21.78**	21.19	20.68	21.27	20.45	NA
Valldemossa	**24.12**	23.87	23.05	23.40	22.40	23.69
Mean	**28.37**	27.75	26.55	27.19	26.40	NA
$\sigma = 80$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	25.22	**25.39**	24.47	24.50	23.39	NA
Computer	**25.97**	25.88	24.16	24.91	23.68	25.41
Dice	**34.57**	32.37	30.04	31.04	31.38	32.16
Flowers	**29.41**	28.53	27.27	28.47	26.58	28.29
Girl	**33.64**	32.20	29.83	30.83	31.10	32.93
Traffic	**24.70**	24.68	23.53	24.13	22.94	24.19
Trees	**20.58**	20.43	19.70	20.17	19.30	NA
Valldemossa	22.83	**22.87**	21.96	22.20	21.09	22.54
Mean	**27.11**	26.54	25.12	25.78	24.93	NA
$\sigma = 100$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	**24.42**	24.37	23.63	23.59	22.52	NA
Computer	**24.87**	24.40	23.08	23.85	22.55	24.48
Dice	**33.42**	30.42	28.53	29.52	29.91	31.06
Flowers	**28.20**	27.04	26.09	27.29	25.43	27.23
Girl	**32.73**	30.56	28.13	29.43	30.07	32.02
Traffic	**23.84**	23.53	22.69	23.20	21.96	23.41
Trees	**19.82**	19.48	19.03	19.38	18.56	NA
Valldemossa	**21.91**	21.91	21.18	21.24	20.08	21.72
Mean	**26.15**	25.21	24.05	24.69	23.88	NA

Table 15: Comparative results on color images from $\sigma = 30$ to $\sigma = 100$. In **bold** the best PSNR for a given image.

$\sigma = 2$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM	BM3D-SAPCA
Alley	**43.49**	43.27	42.90	43.08	42.58	43.01	**43.49**
Computer	**45.02**	44.65	44.24	44.45	44.01	44.06	44.98
Dice	49.43	49.73	49.28	48.79	48.47	49.31	**49.91**
Flowers	47.79	48.15	48.18	47.69	46.04	48.10	**48.38**
Girl	47.81	47.95	47.44	47.19	46.95	47.59	**48.11**
Traffic	**44.28**	44.00	43.78	43.81	43.53	43.48	**44.28**
Trees	**42.72**	42.51	42.43	42.44	42.27	42.42	42.65
Valldemossa	**43.86**	43.66	43.54	43.49	43.29	43.31	43.83
Mean	45.55	45.49	45.22	45.12	44.64	45.16	**45.70**
$\sigma = 5$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM	BM3D-SAPCA
Alley	37.10	36.83	36.15	36.57	36.32	36.47	**37.12**
Computer	39.23	38.73	37.89	38.24	38.01	38.05	**39.25**
Dice	45.37	45.69	45.00	42.75	43.39	45.03	**46.13**
Flowers	42.27	42.83	42.56	41.64	40.33	42.76	**43.03**
Girl	43.46	43.54	42.87	41.63	42.07	43.23	**43.78**
Traffic	37.87	37.48	37.03	37.18	36.73	37.00	**37.89**
Trees	**35.36**	35.06	34.77	34.99	34.46	35.00	35.29
Valldemossa	**36.85**	36.59	36.23	36.32	35.74	36.26	36.80
Mean	39.69	39.59	39.06	38.66	38.38	39.22	**39.91**
$\sigma = 10$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM	BM3D-SAPCA
Alley	32.93	32.72	31.90	32.35	32.23	32.28	**33.00**
Computer	35.11	34.60	33.31	33.80	33.73	33.87	**35.19**
Dice	42.45	42.45	41.47	37.58	38.99	41.65	**43.07**
Flowers	38.42	38.85	38.25	36.74	35.95	38.68	**38.97**
Girl	40.65	40.61	39.64	37.09	38.01	40.21	**40.87**
Traffic	33.30	32.95	32.16	32.50	32.40	32.42	**33.36**
Trees	**30.22**	29.84	29.21	29.76	29.54	29.77	30.14
Valldemossa	**31.98**	31.69	30.97	31.29	31.18	31.30	31.90
Mean	35.63	35.47	34.61	33.89	34.00	35.02	**35.81**
$\sigma = 20$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM	BM3D-SAPCA
Alley	29.42	29.34	28.47	28.47	28.71	28.84	**29.57**
Computer	31.09	30.86	29.21	29.40	30.03	30.10	**31.36**
Dice	**39.58**	39.02	37.60	31.97	36.88	38.17	**39.58**
Flowers	34.65	34.91	33.83	31.62	32.84	34.71	**34.94**
Girl	37.80	37.72	36.19	31.88	35.74	37.36	**37.99**
Traffic	29.27	29.07	28.17	28.30	28.51	28.53	**29.36**
Trees	**25.78**	25.34	24.50	25.30	25.23	25.22	25.55
Valldemossa	**27.68**	27.41	26.56	26.90	27.07	27.03	27.59
Mean	31.91	31.71	30.57	29.23	30.63	31.24	**31.99**

Table 16: Comparative results on grey level images from $\sigma = 5$ to $\sigma = 20$. In **bold** the best PSNR for a given image.

$\sigma = 30$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM	BM3D-SAPCA
Alley	27.60	27.63	26.78	26.97	26.73	27.15	**27.86**
Computer	28.88	28.74	27.04	27.79	27.68	28.00	**29.10**
Dice	**37.84**	37.03	35.22	32.70	34.23	36.25	37.27
Flowers	32.59	32.64	31.30	31.07	30.40	32.57	**32.72**
Girl	**36.10**	35.80	34.11	32.31	33.39	35.69	36.03
Traffic	27.20	27.13	26.27	26.69	26.38	26.56	**27.33**
Trees	**23.55**	23.21	22.45	23.15	23.08	23.08	23.27
Valldemossa	**25.54**	25.32	24.53	24.98	24.90	25.02	25.49
Mean	**29.91**	29.69	28.46	28.21	28.35	29.29	29.88

$\sigma = 40$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM	BM3D-SAPCA
Alley	26.36	26.55	25.69	25.65	25.51	26.02	**26.65**
Computer	27.38	27.27	25.60	26.20	26.10	26.64	**27.70**
Dice	**36.26**	35.41	33.26	30.42	33.70	34.74	35.46
Flowers	31.01	31.03	29.59	29.17	28.83	30.94	**31.10**
Girl	**34.81**	34.47	32.38	30.19	32.67	34.40	34.28
Traffic	25.82	25.86	25.06	25.31	24.89	25.34	**26.03**
Trees	**22.20**	21.99	21.30	21.89	21.66	21.84	21.99
Valldemossa	**24.21**	24.08	23.35	23.67	23.53	23.72	24.14
Mean	**28.51**	28.33	27.03	26.56	27.11	27.95	28.42

$\sigma = 60$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	24.91	**25.07**	24.32	24.35	23.73	24.63
Computer	**25.46**	25.39	23.88	24.59	23.88	24.88
Dice	**34.47**	33.16	31.97	29.73	31.88	32.67
Flowers	28.92	28.83	28.14	27.83	26.71	**29.03**
Girl	**33.30**	32.45	31.72	29.66	31.21	32.79
Traffic	24.19	**24.31**	23.46	23.82	23.11	23.83
Trees	**20.64**	20.57	19.95	20.42	19.96	20.43
Valldemossa	**22.60**	22.52	21.72	22.12	21.61	22.15
Mean	**26.81**	26.54	25.64	25.32	25.26	26.30

$\sigma = 80$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	23.74	**24.02**	23.39	23.14	22.66	NA
Computer	23.98	**24.08**	22.71	23.21	22.47	23.69
Dice	**33.09**	31.21	30.16	27.35	29.92	31.23
Flowers	27.40	27.31	26.62	26.30	25.26	**27.63**
Girl	**32.18**	30.99	30.27	27.70	30.02	31.64
Traffic	23.08	**23.25**	22.53	22.66	21.92	22.84
Trees	19.75	**19.80**	19.26	19.64	19.10	19.63
Valldemossa	21.47	**21.49**	20.84	21.01	20.45	21.17
Mean	**25.59**	25.27	24.47	23.88	23.98	NA

$\sigma = 100$	NL-Bayes	BM3D	DCT denoising	K-SVD	NL-means	BLS-GSM
Alley	22.97	**23.06**	22.71	22.14	21.84	NA
Computer	**22.89**	22.86	21.88	22.17	21.35	22.81
Dice	**31.57**	29.14	28.84	25.53	28.44	30.16
Flowers	26.31	26.09	25.59	25.02	24.17	**26.56**
Girl	**31.07**	29.24	28.95	26.16	28.73	30.71
Traffic	22.25	**22.36**	21.79	21.70	20.99	22.11
Trees	19.00	**19.17**	18.79	19.07	18.51	19.11
Valldemossa	20.61	**20.66**	20.18	20.17	19.51	20.46
Mean	**24.58**	24.07	23.59	22.75	22.94	NA

Table 17: Comparative results on grey level images from $\sigma = 30$ to $\sigma = 100$. In **bold** the best PSNR for a given image.

Original image

Noisy image, $\sigma = 30$

NL-Bayes

BM3D-SAPCA

Figure 7: Comparison on grey level images.

The better visual results with BM3D-SAPCA for moderate noise seem to be due to the adaptive shapes, but also to the larger patch size (8×8 against 5×5). Nonetheless, everything has a price. NL-Bayes is significantly simpler than BM3D-SAPCA. Indeed BM3D-SAPCA employs image patches (neighborhoods) which can have data-adaptive shape. The PCA bases are obtained by eigenvalue decomposition of empirical second-moment matrices that are estimated from groups of similar shape-adaptive neighborhoods. The anisotropic shape-adaptive patches are obtained using the 8-directional LPA-ICI techniques. The principal steps of this algorithm are:

1. Obtain shape-adaptive neighborhood centered at the current pixel using the 8-directional LPA-ICI;

2. Find patches similar to the reference patch using block-matching, and extract an shape-adaptive neighborhood from each of these matched blocks using the shape obtained in Step 1;

3. Determine the transform to be applied on the shape-adaptive neighborhoods (depending on the number of similar blocks), which can be eigenvectors of a second-moment matrix, or a shape-adaptive DCT;

4. Form a 3-D array by stacking together the shape-adaptive neighborhoods with highest similarity to the reference one;

5. Apply the transform obtained in Step 3 on each of the grouped shape-adaptive neighborhoods. Subsequently, apply a 1-D orthogonal transform (e.g., Haar wavelet decomposition) along the third dimension of the 3-D group;

6. Perform shrinkage (hard-thresholding or empirical Wiener filtering) on the 3-D spectrum;

7. Invert the 3-D transform of Step 5 to obtain estimates for all of the grouped shape-adaptive neighborhoods;

8. Return the obtained estimates to their original locations using weighted averaging in case of overlapping.

In terms of complexity, NL-Bayes is also significantly faster than BM3D-SAPCA. On an i5 processor compiling with GCC with optimization mode -O3, in the Ubuntu 12.10 system the average CPU time on this paper's image database was of 20 seconds for NL-Bayes and 900 seconds for the BM3D-SAPCA binary. The comparison cannot be made more complete as we do not dispose of the source code of BM3D-SAPCA, and ignore its degree of parallelism.

6.2 Images

In addition to the PSNR/RMSE results, it is really interesting to compare visually all methods, especially to remark (and regret) the inherent and characteristic artifacts of each one.

A relatively low noise shows each method at its best (figure 8).

The noise standard deviation limit beyond which artifacts appear with **all methods** is $\sigma = 40$ (figure 11). It is the limit between moderate noise, where we get visually acceptable results, and high noise where no method gives so far visually acceptable results.

Above $\sigma = 40$ (figure 12), inherent and characteristic artifacts become obvious. These limitations are important to explore. They open up the question of denoising high noise.

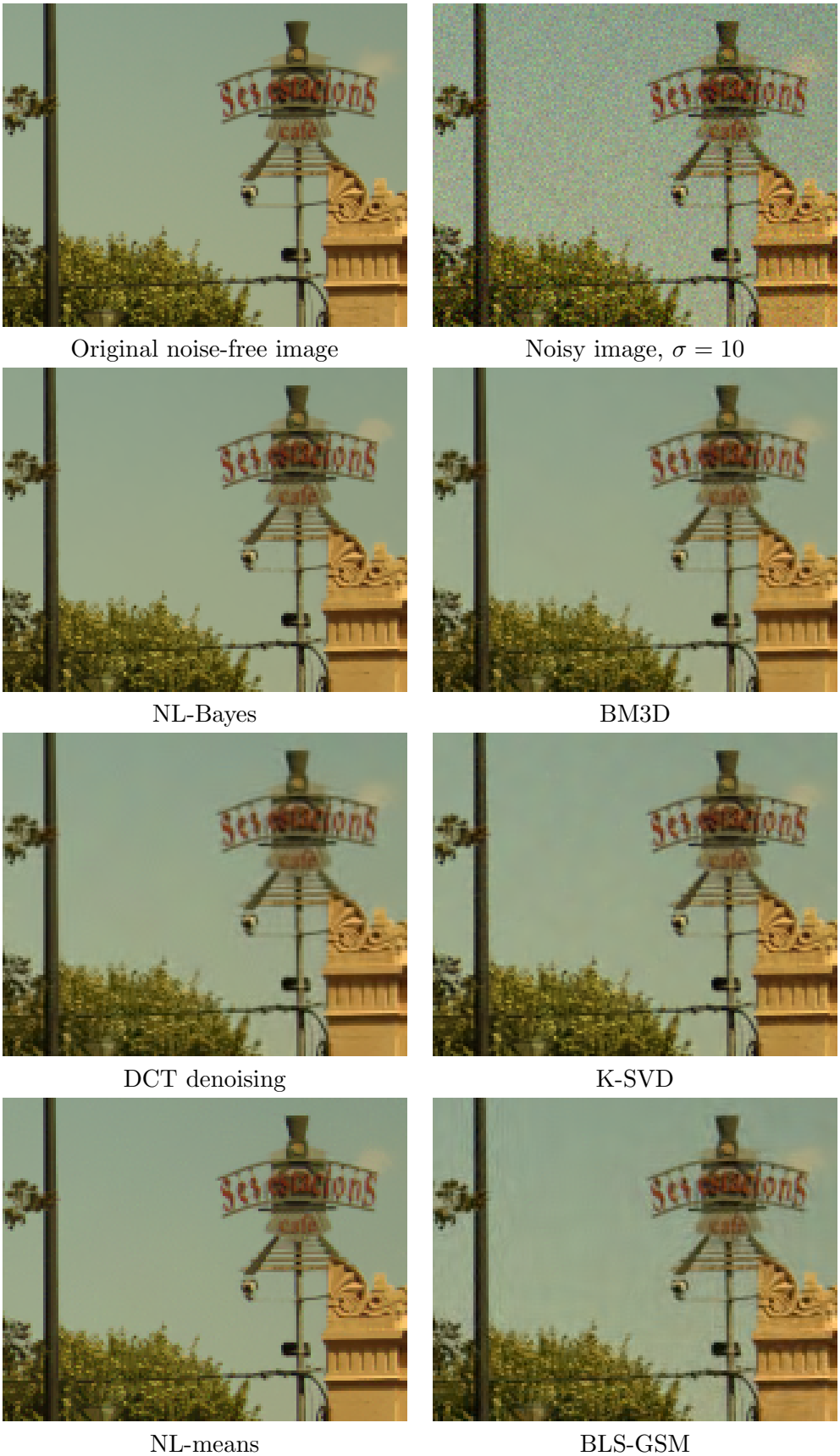


Figure 8: Visual comparison of all methods, $\sigma = 10$.

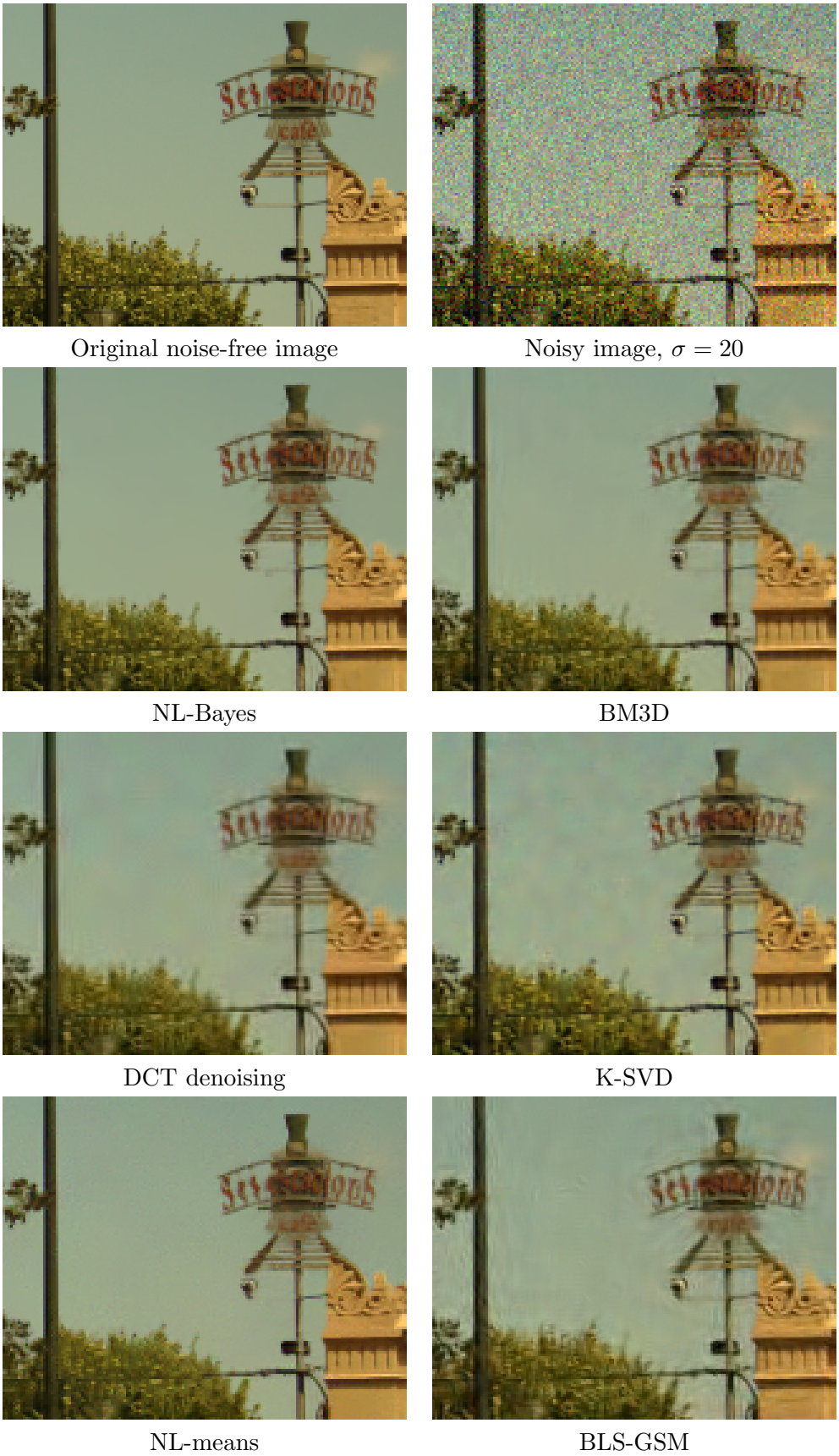
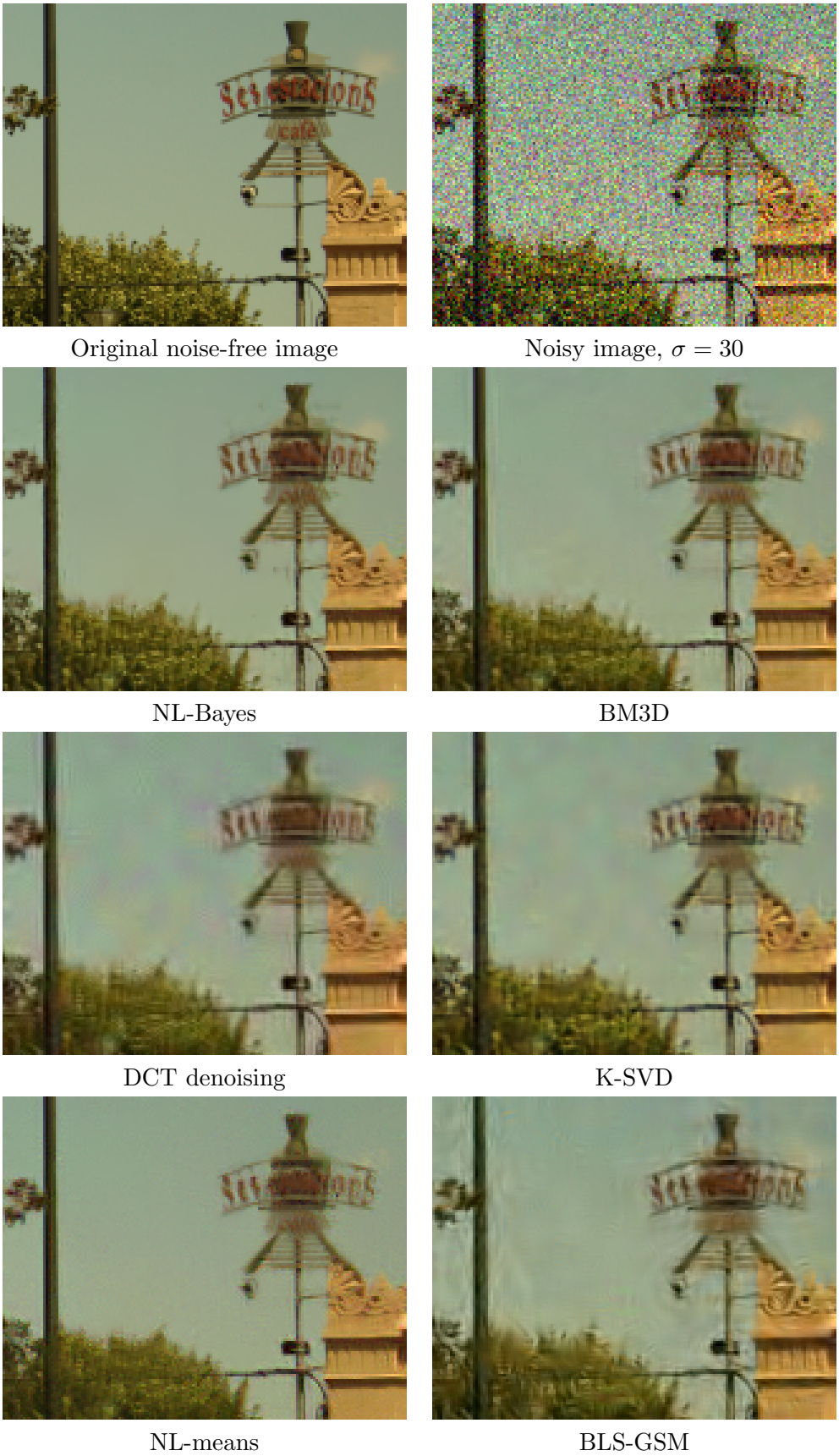


Figure 9: Visual comparison of all methods, $\sigma = 20$.



Original noise-free image | Noisy image, $\sigma = 30$

NL-Bayes | BM3D

DCT denoising | K-SVD

NL-means | BLS-GSM

Figure 10: Visual comparison of all methods, $\sigma = 30$.

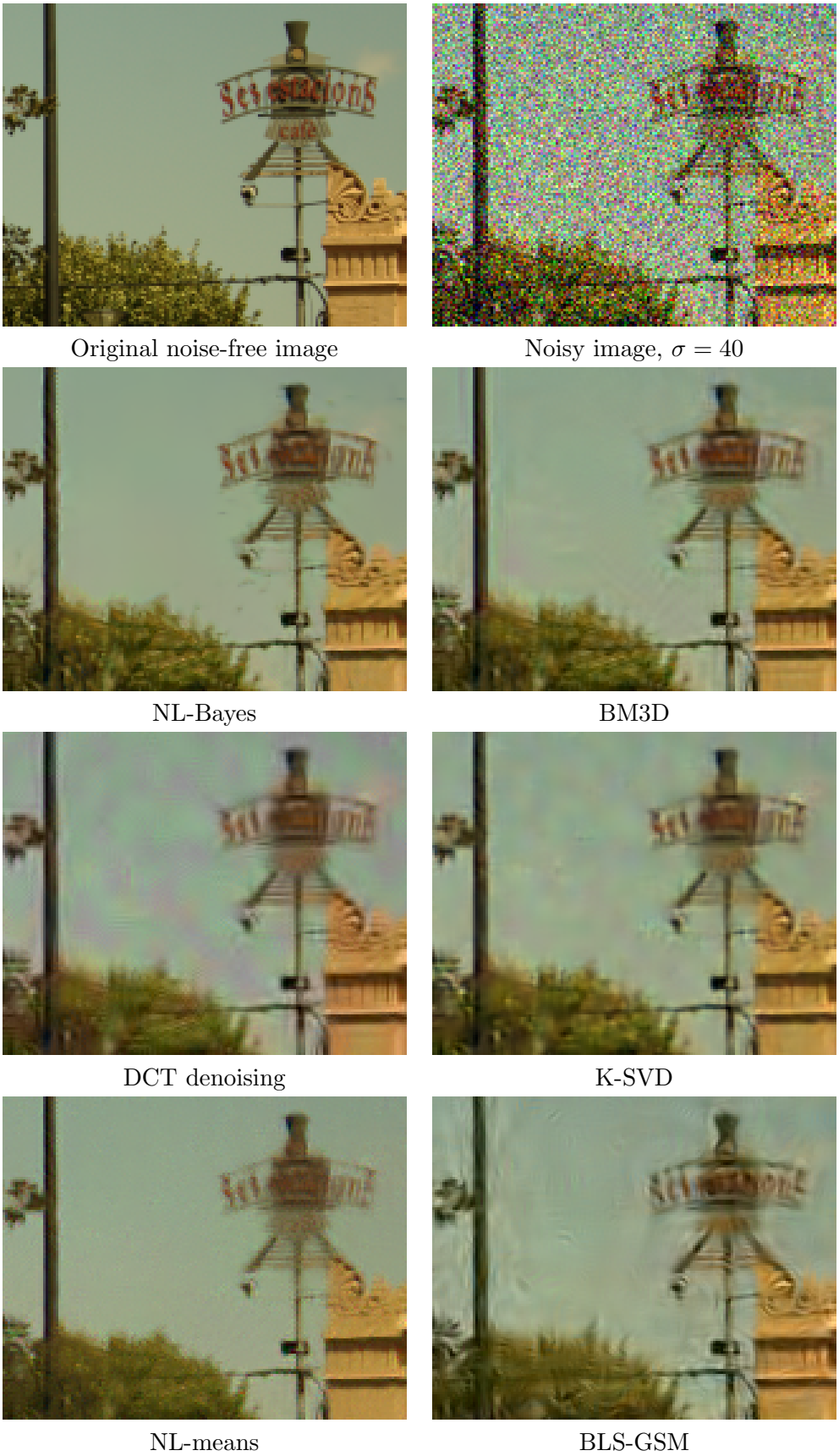
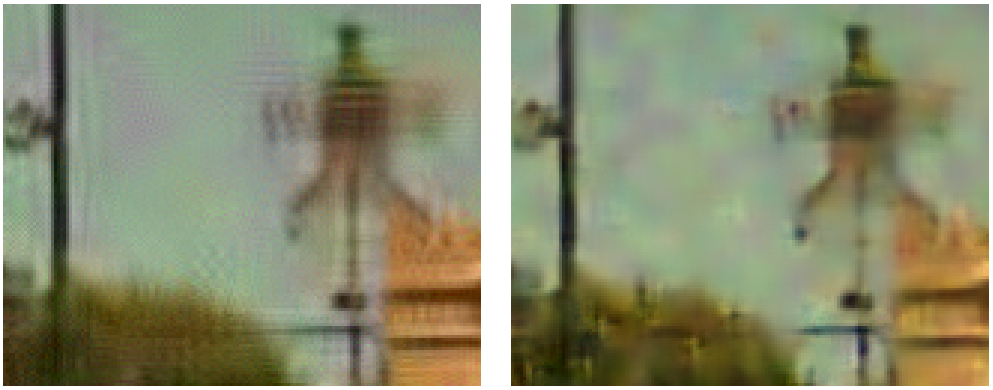


Figure 11: Visual comparison of all methods, $\sigma = 40$.

Original noise-free image | Noisy image, $\sigma = 80$

NL-Bayes | BM3D



DCT denoising | K-SVD

NL-means | BLS-GSM

Figure 12: Visual comparison of all methods, $\sigma = 80$.

6.3 Experimental Conclusion

This detailed study carried out on NL-Bayes had led us to the following conclusions:

- working on very small size of patches, this method is really fast for small and medium noise, even more thanks to the acceleration trick that used patches are no more processed again as reference patches;
- The main elements which allow for real improvements of the results are:
  - working in the $Y_0U_0V_0$ space color for the first step;
  - making a second step with the result of the first step as "oracle";
  - aggregating the estimates. This aggregation is improved significantly by working with a 3D group and by keeping all estimates obtained of the similar 2D-patches like BM3D;
  - using the homogeneous area criteria to remove almost all artifacts in homogeneous area.
- Nevertheless, there is still room for improvement, by, perhaps:
  - using a multi-scale approach to remove low frequency noise;
  - using a more complex model than the Gaussian patch model.

7 Comparison to PCA Based Methods

7.1 Comments About "Image Denoising with Patch Based PCA: Local Versus Global"

The use of the PCA (*Principal Component Analysis*) has been treated in detail by Deledalle et al. [7]. This article proposes to use a PCA to learn an orthonormal basis adapted locally or globally to the image. The interesting part in this article is the confrontation of global versus local PCA. Indeed, a first simple approach is to learn the PCA over the whole image, and then to denoise patches by applying a hard thresholding to the coefficients of the patch on the new learned orthonormal basis. This works quite well, but for patches with a rare structure, the denoising is not optimal, because this structure is under-represented in the whole image. Thus, to adapt the method to every kind of structure, even the rarest, the solution is to learn the PCA more locally, on small windows around the current patch. Nevertheless, working on a small windows, there is a risk of not finding enough patches to learn an unbiased PCA.

Another interesting point of this article is that it shows the first eigenvectors and the last ones of each PCA. One can easily see that the first eigenvectors describe the orientation of the gradient of the patches, whereas the last ones present only noise. This illustrates experimentally how canceling the coefficients of the last eigenvectors (those with the lowest corresponding eigenvalues) permits to remove the noise in the image without affecting structure or details. Despite this interesting study, one can finger-point some defaults of this article. In fact, its PSNR results are worse than BM3D and the algorithm is slower. These defaults are partly due to the fact that this paper does not use all tricks that improve the denoising performance: only gray-level images are treated, no second oracular step is applied. The size of the patches can probably be reduced. Nevertheless, the algorithm applies a uniform aggregation. A puzzling result of their study is that applying a hard threshold gives better results than a Wiener filter. But a second step would demonstrate that with the right oracle given by the first step, the Wiener filter is in fact better, being optimal in theory. Regrettably, only small noise values $\sigma \in [5, 20]$ have been tested.

7.2 Analysis and Comments About TSID

This part considers the original method described by Zhang et al. [24]. As this algorithm is really close to NL-PCA, it will be compared to it stepwise, and for this purpose the same notation is adopted.

7.2.1 Implementation

The algorithm is divided in two separate steps. Contrarily to NL-PCA, these steps are identical, except for the used value of σ which is upgraded after the first step. Thus, only one step will be described. This algorithm was originally developed for gray level images. In the following we therefore examine the application to gray level images. The image is scanned pixel per pixel. Let us denote by $\tilde{P}$ the current reference patch which size is $k_{\mathbf{1}} \times k_{\mathbf{1}}$ (with $k_{\mathbf{1}} = 5$) and x_r the current central pixel of $\tilde{P}$. The loop on the image is done on x_r. Like for NL-PCA and BM3D, each step is divided into 3 parts:

Grouping : Similar patches $\tilde{Q}$ are found by block-matching with the same distance as the one used for NL-Bayes. A patch is considered similar if its distance to $\tilde{P}$ is below a fixed threshold, depending on σ. Moreover, to stabilize the process of the PCA on the obtained 3D-block, a minimum number (denoted by N_P) of similar patches is requested. In this case the N_P best patches obtained during block-matching are used.

Collaborative Filtering : When the set of similar patches has been obtained, a matrix is built, containing all similar patches seen as vectors. Then, as for NL-PCA, the columns of the matrix are first centered around their common center of mass. After that, a singular value decomposition (SVD) of the centered matrix is obtained. Then, a LMMSE (Wiener filter) is applied to the new coefficients, and the patch is finally reconstructed with these new coefficients on the PCA.

Getting the estimate : The principal difference with NL-PCA, is that only the estimate of the central pixel of the reference patch x_r is kept, which clearly decreases the PSNR, no aggregation step being possible.

7.2.2 Extending to Color Images

In BM3D the extension to color images is done by using $Y_0U_0V_0$ instead of RGB; the grouping part is done only on the Y_0 channel and the rest of the algorithm is performed independently on the three channels. For TSID instead this extension is done in the simplest way; the whole algorithm is applied independently on the three channels R, G and B. This is another point that could have been very easily improved, but was not envisaged by the authors.

7.2.3 Comparisons Between TSID and NL-PCA

Table 18 shows some average comparative results over both sets of color and grey images in terms of PSNR for TSID and NL-PCA.

Acknowledgements

Research partially financed by the MISS project of Centre National d'Etudes Spatiales, the Office of Naval research under grant N00014-97-1-0839, by the European Research Council, advanced grant

	NL-PCA		TSID	
σ	PSNR	RMSE	PSNR	RMSE
2	45.29	1.39	44.06	1.60
5	40.84	2.31	39.09	2.83
10	37.30	3.48	35.72	4.17
20	33.35	5.48	32.07	6.35
30	30.74	7.40	29.70	8.35
40	28.86	9.19	27.92	10.24
60	27.57	10.66	25.55	13.46
80	25.97	12.83	23.93	16.21
100	24.72	14.81	22.87	18.32

Grey level images

	NL-PCA		TSID	
σ	PSNR	RMSE	PSNR	RMSE
2	46.03	1.27	43.76	1.65
5	39.84	2.60	36.97	3.61
10	35.28	4.39	32.18	6.27
20	30.92	7.25	28.03	10.11
30	28.61	9.46	25.94	12.87
40	27.08	11.29	24.59	15.03
60	24.94	14.43	22.74	18.59
80	23.63	16.78	21.45	21.57
100	22.60	18.90	20.48	24.14

Color images

Table 18: Comparisons between TSID and NL-PCA.

"Twelve labours" and the Spanish government under TIN2011-27539. The first author acknowledges support by DxO Labs through a CIFRE PhD scholarship.

Glossary

Subscripts of variables in **bold** are semantic subscripts.

Parameters

- $k_{\mathbf{1}} \times k_{\mathbf{1}}$ (resp. $k_{\mathbf{2}} \times k_{\mathbf{2}}$): dimension of patches for the first (resp. second) step of the algorithm;
- $n_{\mathbf{1}} \times n_{\mathbf{1}}$ (resp. $n_{\mathbf{2}} \times n_{\mathbf{2}}$): dimension of search zone for similar patches for the first (resp. second) step of the algorithm;
- $N_{\mathbf{1}}$ (resp. $N_{\mathbf{2}}$): number of retained similar patches $\tilde{Q}$ (resp. $Q^{\mathbf{basic}}$) to the reference one $\tilde{P}$ (resp. $P^{\mathbf{basic}}$) for the first (resp. second) step of the algorithm;
- τ_0: fixed threshold used to estimate the similarity threshold τ in the second step of the algorithm;
- γ: used to determine if a patch belongs to an homogeneous area;
- $\beta_{\mathbf{1}}$ (resp. $\beta_{\mathbf{2}}$): parameter of conservative attenuation close to 1 used in the first (resp. second) step of the algorithm.

Notations

- **basic**: designates the first step of the algorithm;
- **final**: designates the second step of the algorithm;
- **1**: designates parameters used in the first step of the algorithm;
- **2**: designates parameters used in the second step of the algorithm;
- u: noiseless ideal image;
- $\tilde{u}$: noisy image;

- $u^{\mathbf{basic}}$: basic estimate of the denoised image, obtained at the end of the first step of the algorithm;
- $u^{\mathbf{final}}$: final denoised image obtained at the end of the second step of the algorithm;
- n: Gaussian noise of standard deviation σ;
- σ: standard deviation of white noise at each pixel;
- N: total number of pixels or patches in the image;
- P: noiseless reference patch of u;
- Q: second patch compared to P;
- $\tilde{P}$, $\tilde{Q}$: noisy patches;
- $P^{\mathbf{basic}}$ (resp. $P^{\mathbf{final}}$): restored patch after the first (resp. second) step of the algorithm;
- $\mathbb{P}(G)$: probability of an event G (in the image and noise stochastic models);
- EQ: expectation (of a random patch Q);
- $\overline{P}$: empirical expectation of the patches similar to P;
- $d(\tilde{P}, \tilde{Q})$: Euclidean distance between patches (considered as vectors of their values);
- $\mathbf{C}_P$: covariance matrix (of patches similar to P or $\tilde{P}$);
- $\mathbf{I}$: identity matrix;
- $\mathcal{P}(\tilde{P})$: set of similar patches $\tilde{Q}$ to the reference one $\tilde{P}$ for the first step of the algorithm;
- $\mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}})$: set of similar patches $Q^{\mathbf{basic}}$ to the reference one $P^{\mathbf{basic}}$ in the basic estimation $u^{\mathbf{basic}}$ for the second step of the algorithm;
- $\mathcal{P}^{\mathbf{basic}}(\tilde{P})$: set of similar patches $\tilde{Q}$ to the reference one $\tilde{P}$ in the noisy image $\tilde{u}$, obtained by stacking up patches together in the same order as $\mathcal{P}^{\mathbf{basic}}(P^{\mathbf{basic}})$.

Image Credits


by A. Buades, CC-BY


by M. Colom, CC-BY

References

[1] A. Buades, B. Coll, and J.M. Morel. A non-local algorithm for image denoising. *IEEE Computer Society*, 2005. http://dx.doi.org/10.1109/CVPR.2005.38.

[2] A. Buades, B. Coll, and J.M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2006. http://dx.doi.org/10.1137/040616024.

[3] A. Buades, B. Coll, and J.M. Morel. Non-local means denoising. *Image Processing On Line*, 2011. http://dx.doi.org/10.5201/ipol.2011.bcm_nlm.

[4] A. Buades, M. Lebrun, and J.M. Morel. A Non-local Bayesian image denoising algorithm. *SIAM Journal on Imaging Science*, 2013. (to appear).

[5] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(82):3736–3745, 2007. http://dx.doi.org/10.1109/TIP.2007.901238.

[6] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, et al. BM3D image denoising with shape-adaptive principal component analysis. *Proceedings of the Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS), Saint-Malo, France*, April 2009.

[7] C.A. Deledalle, J. Salmon, and A. Dalalyan. Image denoising with patch based PCA: local versus global. In *Proceedings of the British Machine Vision Conference*, pages 25.1–25.10. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.25.

[8] D.L. Donoho and J.M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. http://dx.doi.org/10.2307/2337118.

[9] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, volume 2, pages 1033–1038. Corfu, Greece, 1999. http://dx.doi.org/10.1109/ICCV.1999.790383.

[10] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006. http://dx.doi.org/10.1109/TIP.2006.881969.

[11] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984. http://dx.doi.org/10.1109/TPAMI.1984.4767596.

[12] P. Getreuer. Rudin-Osher-Fatemi total variation denoising using split Bregman. *Image Processing On Line*, 2012, 2012. http://dx.doi.org/10.5201/ipol.2012.g-tvd.

[13] M. Lebrun. An analysis and implementation of the BM3D image denoising method. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.l-bm3d.

[14] M. Lebrun and A. Leclaire. An implementation and detailed analysis of the K-SVD image denoising algorithm. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.llm-ksvd.

[15] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008. http://dx.doi.org/10.1109/TIP.2007.911828.

[16] J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Modeling and Simulation*, 7(1):214–241, 2008. http://dx.doi.org/10.1137/070697653.

[17] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003. http://dx.doi.org/10.1109/TIP.2003.818640.

[18] W.H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, 1972. http://dx.doi.org/10.1364/JOSA.62.000055.

[19] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992. http://dx.doi.org/10.1016/0167-2789(92)90242-F.

[20] L.P. Yaroslavsky. Local adaptive image restoration and enhancement with the use of DFT and DCT in a running window. In *Proceedings of SPIE*, volume 2825, pages 2–13, 1996. http://dx.doi.org/10.1117/12.255218.

[21] L.P. Yaroslavsky, K.O. Egiazarian, and J.T. Astola. Transform domain image restoration methods: review, comparison, and interpretation. In *Proceedings of SPIE*, volume 4304, page 155, 2001. http://dx.doi.org/10.1117/12.424970.

[22] G. Yu and G. Sapiro. DCT image denoising: a simple and effective image denoising algorithm. *Image Processing On Line*, 2011. http://dx.doi.org/10.5201/ipol.2011.ys-dct.

[23] G. Yu, G. Sapiro, and S. Mallat. Image modeling and enhancement via structured sparse model selection. In *2010 17th IEEE International Conference on Image Processing (ICIP)*, pages 1641–1644, 2010. http://dx.doi.org/10.1109/ICIP.2010.5653853.

[24] L. Zhang, W. Dong, D. Zhang, and G. Shi. Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition*, 43(4):1531–1549, 2010. http://dx.doi.org/10.1016/j.patcog.2009.09.023.