2015/06/16 v0.5.1 IPOL article class

IPOL
IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2012–08–08.
Submitted on 2012–00–00, accepted on 2012–00–00.
ISSN 2105–1232 © 2012 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.2012.l-bm3d

An Analysis and Implementation of the BM3D Image Denoising Method

Marc Lebrun[1]

[1]CMLA, ENS Cachan, France (marc.lebrun@cmla.ens-cachan.fr)

Communicated by Jean-Michel morel *Demo edited by* Miguel Colom

Abstract

BM3D is a recent denoising method based on the fact that an image has a locally sparse representation in transform domain. This sparsity is enhanced by grouping similar 2D image patches into 3D groups. In this paper we propose an open-source implementation of the method. We discuss the choice of all parameter methods and confirm their actual optimality. The description of the method is rewritten with a new notation. We hope this new notation is more transparent than in the original paper. A final index gives nonetheless the correspondence between the new notation and the original notation.

Keywords: denoising, sparsity, adaptive grouping, block-matching 3D-transform shrinkage

1 Introduction

Collaborative filtering is the name of the BM3D grouping and filtering procedure. It is realized in four steps: 1) finding the image patches similar to a given image patch and grouping them in a 3D block 2) 3D linear transform of the 3D block; 3) shrinkage of the transform spectrum coefficients; 4) inverse 3D transformation. This 3D filter therefore filters out simultaneously all 2D image patches in the 3D block.

By attenuating the noise, collaborative filtering reveals even the finest details shared by the grouped patches. The filtered patches are then returned to their original positions. Since these patches overlap, many estimates are obtained which need to be combined for each pixel. *Aggregation* is a particular averaging procedure used to take advantage of this redundancy.

The first collaborative filtering step is much improved by a a second step using Wiener filtering. This second step mimics the first step, with two differences. The first difference is that it compares the filtered patches instead of the original patches. The second difference is that the new 3D group (built with the unprocessed image samples, but using the patch distances of the filtered image) is

processed by Wiener filtering instead of a mere threshold. The final aggregation step is identical to those of the first step.

The proposed method improved on the NL-means [2] method which denoises jointly similar patches, but only by performing a patch average, which amounts to a 1D filter in the 3D block. The 3D filter in BM3D is performed on the three dimensions simultaneously.

The BM3D algorithm detailed here directly comes from the original article [4]. It is generally considered to achieve the best performance bounds in color image denoising. Nevertheless, the authors have pointed out to us more recent and sophisticated versions. Like for NL-means, there is a variant with shape-adaptive patches [5]. In this algorithm denominated BM3D-SAPCA, the sparsity of image representation is improved in two aspects. First, it employs image patches (neighborhoods) which can have data-adaptive shape. Second, the PCA bases are obtained by eigenvalue decomposition of empirical second-moment matrices that are estimated from a group of similar adaptive-shape neighborhoods. This method improves BM3D especially in preserving image details and introducing very few artifacts. The anisotropic shape-adaptive patches are obtained using the 8-directional LPA-ICI techniques [12]. In the very recent developments of BM3D [11, 6], it is generalized to become a generic image restoration tool, including deblurring.

A previous analysis on BM3D [9] studies the sharp drop of the denoising performance when noise standard deviation reaches 40. On the contrary of this present study, only few parameters have been studied, and only for large value of noise. Moreover, this previous study [9] presents a less detailed study of BM3D than the one of the present article. For large value of noise, we reach the same conclusions: the threshold value during the first step needs to be increased, and the best results are achieved by using a 2D bi-orthogonal spline wavelet (denoted by 2D-Bior1.5 in the following) during the second step and by keeping a 8×8 size for patches in both steps.

It is furthermore interesting to notice that recent papers [3, 14] which try to evaluate the inherent bounds of patch-based denoising methods claim that BM3D is really close to those optimality bounds.

2 The Algorithm Step by Step

2.1 Architecture of the Algorithm

We shall first describe how to process gray level images. The extension to color images will be explained later on. In all the following we work in the case of white Gaussian noise where the variance is denoted by σ^2.

The algorithm is divided in two major steps:

1. The first step estimates the denoised image using hard thresholding during the collaborative filtering. Parameters in this step are denoted by the exponent **hard**;

2. The second step is based both on the original noisy image, and on the basic estimate obtained in the first step. It uses Wiener filtering. The second step is therefore denoted by the exponent **wiener**.

The algorithm is summarized in the two next figures 1 and 2.

The figure 3 shows the patches, a search window centered on reference patch P, and illustrates the patch overlapping leading to multiple estimates.

2.2 The First Denoising Step

We denote by P the reference current patch whose size is $k^{\mathbf{hard}} \times k^{\mathbf{hard}}$ of the image loop.

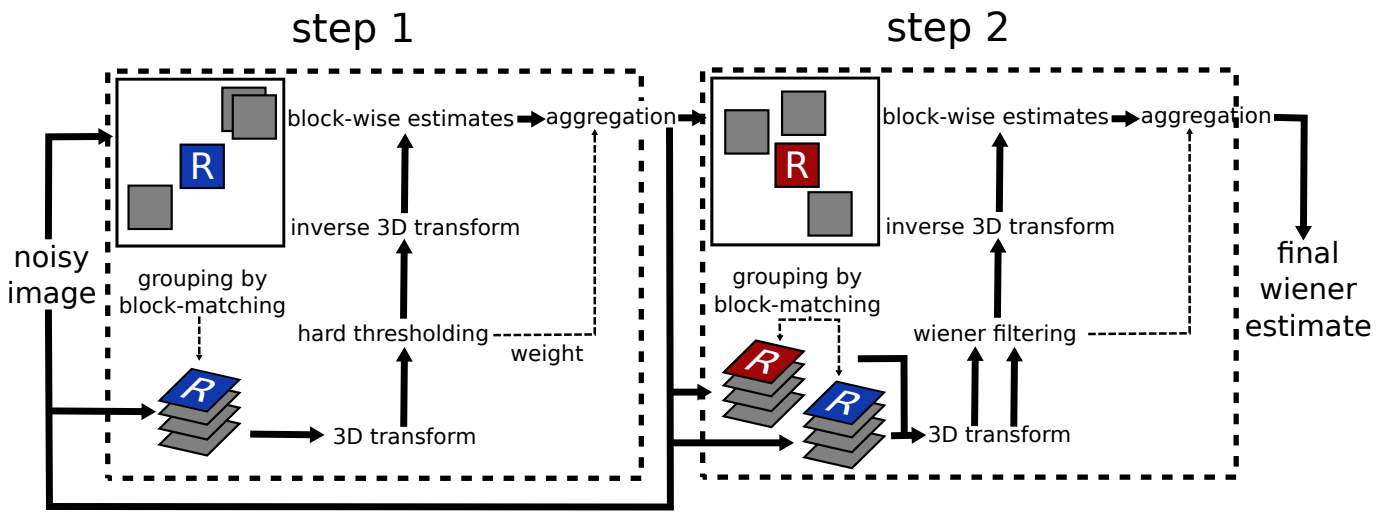


Figure 1: Scheme of the BM3D algorithm.

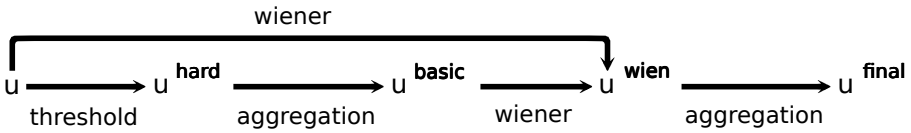


Figure 2: Notations used in this article.

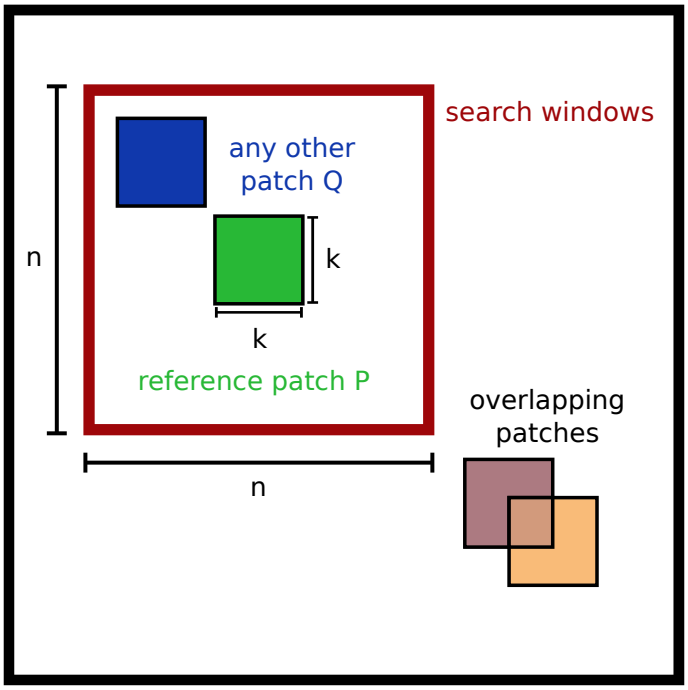


Figure 3: Patches, search window and overlapping.

Grouping

The first sub-step is grouping. The original noisy image u is searched in a P-centered $n^{\mathbf{hard}} \times n^{\mathbf{hard}}$ neighborhood for patches Q similar to the reference patch P. The set of similar patches is simply defined by

$$\mathcal{P}(P) = \{Q : d(P,Q) \leq \tau^{\mathbf{hard}}\} \tag{1}$$

where:

- $\tau^{\mathbf{hard}}$ is the distance threshold for d under which two patches are assumed similar;
- $d(P,Q) = \dfrac{\|\gamma'(P) - \gamma'(Q)\|_2^2}{(k^{\mathbf{hard}})^2}$ is the normalized quadratic distance between patches;
- γ' is a hard thresholding operator with threshold $\lambda_{2D}^{\mathbf{hard}}\sigma$. For $\sigma \leq 40$ one has $\lambda_{2D}^{\mathbf{hard}}\sigma = 0$. It simply puts to zero the coefficients of the patch with an absolute value below the threshold $\lambda_{2D}^{\mathbf{hard}}\sigma$. The other coefficients are unchanged. For $\sigma \leq 40$ all coefficients are unchanged;
- σ^2 is the variance of the zero-mean Gaussian noise.

The 3D group, denoted $\mathbb{P}(P)$, is then built by stacking up the matched patches $\mathcal{P}(P)$. In order to speed up the process, only the $N^{\mathbf{hard}}$ patches in $\mathcal{P}(P)$ that are closest to the reference patch are kept in the 3D group[1]. Patches in $\mathcal{P}(P)$ will be sorted according to their distance to P, in order to take the best $N^{\mathbf{hard}}$ similar patches easily. Thus naturally the first patch will be P because its distance to itself is zero. The order of the patches in the 3D group is not important: the results are similar no matter whether they are ordered according to their distance to the reference patch, or just randomly.

Collaborative Filtering

Once the 3D-block $\mathcal{P}(P)$ is built the collaborative filtering is applied. A 3D isometric linear transform is applied to the group, followed by a shrinkage of the transform spectrum. Finally the inverse linear transform is applied to estimate for each patch

$$\mathbb{P}(P)^{\mathbf{hard}} = {\tau_{3D}^{\mathbf{hard}}}^{-1}(\gamma(\tau_{3D}^{\mathbf{hard}}(\mathbb{P}(P)))) \tag{2}$$

where γ is a hard thresholding operator with threshold $\lambda_{3D}^{\mathbf{hard}}\sigma$:

$$\gamma(x) = \begin{cases} 0 & \text{if} \quad |x| \leq \lambda_{3D}^{\mathbf{hard}}\sigma \\ x & \text{otherwise} \end{cases}$$

For practical purposes, the 3D transform $\tau_{3D}^{\mathbf{hard}}$ of the 3D group $\mathcal{P}(P)$ is made up of two transforms: a 2D transform denoted by $\tau_{2D}^{\mathbf{hard}}$ applied on each patch of $\mathcal{P}(P)$, and a 1D transform denoted by $\tau_{1D}^{\mathbf{hard}}$ applied along the third dimension of the 3D group. The choice of these transforms will be carefully discussed later.

Aggregation

When the collaborative filtering is done, we get an estimate for each used patch and then a variable number of estimates for every pixel. These estimates are saved in a buffer:

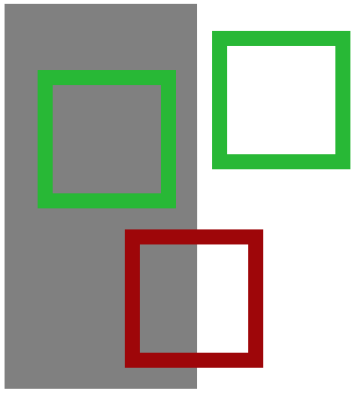
$$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + w_P^{\mathbf{hard}} u_{Q,P}^{\mathbf{hard}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + w_P^{\mathbf{hard}} \end{cases} \tag{3}$$

where:

[1]Moreover, when the applied 1D transform $\tau_{1D}^{\mathbf{hard}}$ is a Walsh-Hadamard transform it is necessary to have a power of 2 for the number of similar patches. Then $N^{\mathbf{hard}}$ is always chosen as a power of 2, and if there are less similar patches than $N^{\mathbf{hard}}$ for a given reference patch, only a power of 2 inferior or equal to this number of similar patches will be kept.

- $\boldsymbol{\nu}$ (resp. $\boldsymbol{\delta}$) designates the numerator (resp. denominator) part of the basic estimate of the image obtained at the end of the grouping step;
- $u_{Q,P}^{\mathbf{hard}}(x)$ is the estimate of the value of the pixel x belonging to the patch Q obtained during collaborative filtering of the reference patch P;
- $w_P^{\mathbf{hard}} = \begin{cases} \left(N_P^{\mathbf{hard}}\right)^{-1} & \text{if} \quad N_P^{\mathbf{hard}} \geq 1 \\ 1 & \text{otherwise} \end{cases}$
- $N_P^{\mathbf{hard}}$ is the number of retained (non-zero) coefficients in the 3D block after hard-thresholding: $\gamma\left(\tau_{3D}^{\mathbf{hard}}\left(\mathbb{P}(P)\right)\right)$.

The interest of this weighting is that it gives a priority to homogeneous patches (where there are many canceled coefficients). Patches containing an edge will be less taken into account than homogeneous ones on the border of the edge. The next figure illustrates this fact: priority is given to the green patches during the aggregation.



Green patches will have a weight superior to the red patch, because they are more sparse (have less nonzero transform coefficients).

The result is an artifact reduction around the edges and it avoids the classic ringing effects observable with transform threshold methods. In order to reduce more the border effects which can appear, a $k^{\mathbf{hard}} \times k^{\mathbf{hard}}$ Kaiser window is used as part of the weights. It is simply done as a element-by-element multiplication between the Kaiser window and the estimated patch after the inverse 3D transformation. How the Kaiser window can be obtained will be discussed in the section 4.4.

The basic estimate after this first step is given by

$$u^{\mathbf{basic}}(x) = \frac{\displaystyle\sum_P w_P^{\mathbf{hard}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x) u_{Q,P}^{\mathbf{hard}}(x)}{\displaystyle\sum_P w_P^{\mathbf{hard}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x)} \tag{4}$$

which is simply obtained by dividing the two buffers (numerator and denominator) element-by-element, with $\chi_Q(x) = 1$ if and only if $x \in Q$, 0 otherwise.

2.3 The Second Denoising Step

In this second part of the algorithm a basic estimate $u^{\mathbf{basic}}$ of the denoised image is available. The second step performs a Wiener filter of the original image u, but uses as oracle the basic estimate $u^{\mathbf{basic}}$. It is observed in the experiments that this second step restores more details and improves the denoising performance, as will be clear in the experiments and tables below.

Grouping

The patch-matching is only processed on the basic estimate. When a set of similar patches

$$\mathcal{P}^{\mathbf{basic}}(P) = \{Q : d(P,Q) \leq \tau^{\mathbf{wien}}\} \tag{5}$$

has been obtained two 3D groups are formed:

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

Once again, for optimization a maximum number of $N^{\mathbf{wien}}$ patches is kept in the two 3D groups. They have been chosen exactly as described in the first denoising step.

Collaborative Filtering

When the two 3D groups are obtained the collaborative filtering can be launched. To do so empirical Wiener coefficients are defined by

$$\omega_P(\xi) = \frac{\left|\tau_{3D}^{\mathbf{wien}}\left(\mathbb{P}^{\mathbf{basic}}(P)\right)(\xi)\right|^2}{\left|\tau_{3D}^{\mathbf{wien}}\left(\mathbb{P}^{\mathbf{basic}}(P)\right)(\xi)\right|^2 + \sigma^2}. \tag{6}$$

The Wiener collaborative filtering of $\mathbb{P}(P)$ is realized as the element-by-element multiplication (denoted with a dot) of the 3D transform of the noisy image $\tau_{3D}^{\mathbf{wien}}(\mathbb{P}(P))$ with the Wiener coefficients ω_P. Through this sub-step an estimate of the 3D group is obtained as

$$\mathbb{P}^{\mathbf{wien}}(P) = {\tau_{3D}^{\mathbf{wien}}}^{-1}\left(\omega_P \,.\, \tau_{3D}^{\mathbf{wien}}(\mathbb{P}(P))\right). \tag{7}$$

Aggregation

When collaborative filtering is achieved, the estimates for every pixel are stored in a buffer:

$$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \boldsymbol{\nu}(x) = \boldsymbol{\nu}(x) + w_P^{\mathbf{wien}} u_{Q,P}^{\mathbf{wien}}(x) \\ \boldsymbol{\delta}(x) = \boldsymbol{\delta}(x) + w_P^{\mathbf{wien}} \end{cases} \tag{8}$$

where:

- $\boldsymbol{\nu}$ (resp. $\boldsymbol{\delta}$) designates the numerator (resp. denominator) part of the final estimation of the image obtained at the end of the previous step;
- $u_{Q,P}^{\mathbf{wien}}(x)$ is the estimation of the value of the pixel x belonging to the patch Q obtained during the collaborative filtering of the reference patch P;
- $w_P^{\mathbf{wien}} = \|\omega_P\|_2^{-2}$.

As for the first step, a $k^{\mathbf{wien}} \times k^{\mathbf{wien}}$ Kaiser window is applied to reduce border effects.

The final estimate obtained after the second step is given by

$$u^{\mathbf{final}}(x) = \frac{\displaystyle\sum_P w_P^{\mathbf{wien}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x) u_{Q,P}^{\mathbf{wien}}(x)}{\displaystyle\sum_P w_P^{\mathbf{wien}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x)} \tag{9}$$

which is simply obtained by dividing both buffers (numerator and denominator) element-by-element. Here $\chi_Q(x) = 1$ if and only if $x \in Q$, 0 otherwise.

In the original article, during the patch aggregation sub-step, a Kaiser window is applied to each patch in order to slightly attenuate the patch borders. Nevertheless, experimental results show that the Kaiser windows do not improve the PSNR, and are visually less efficient than the weighting. For the sake of simplicity, these windows are not mentioned in this algorithmic description. By fidelity to the original article, they are nevertheless implemented in the code published with this article.

3 A Study of the Optimal Parameters

3.1 Comparison Criteria and Parameters Under Study

The results shown hereafter are obtained from the previously described algorithm applied to noiseless images with a simulated white noise added. Many images have been tested, but for the sake of simplicity only one result by σ will be shown. All shown results has been obtained on the noise-free image shown in figure 4.

To describe quantitatively denoising results, two classic measures will be used:

The Root Mean Square Error (RMSE) between the reference image (noiseless) u_R and the denoised image u_D is computed as:

$$RMSE = \sqrt{\frac{\sum_{x \in X} \left(u_R(x) - u_D(x)\right)^2}{|X|}}. \tag{10}$$

The smaller the RMSE, the better the denoising.

The Peak Signal to Noise Ratio (PSNR) is evaluated in decibels (dB):

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE} \right). \tag{11}$$

The larger the PSNR, the better the denoising.

Choosing the right values for the different parameters in the algorithm and their influence has to be discussed. The set of the parameters is:

- $k^{\mathbf{hard}}$ and $k^{\mathbf{wien}}$: size of patches;
- $N^{\mathbf{hard}}$ and $N^{\mathbf{wien}}$: maximum number of similar patches kept;
- $p^{\mathbf{hard}}$ and $p^{\mathbf{wien}}$: In order to speed up the processing, the loop over the pixels of the image is done with a step p (integer) in row and column. For example if $p = 3$ the algorithm is accelerated by a 9 factor;
- $n^{\mathbf{hard}}$ and $n^{\mathbf{wien}}$: search window size;
- $\tau^{\mathbf{hard}}$ and $\tau^{\mathbf{wien}}$: maximum thresholds for the distance between two similar patches;
- $\lambda_{2D}^{\mathbf{hard}}$ and $\lambda_{3D}^{\mathbf{hard}}$.

Figure 4: Valldemossa noiseless image.

Several of these parameters have actually little influence on the final result, namely $n^{\mathbf{hard}}$ and $n^{\mathbf{wien}}$. Therefore the following values will be fixed and used throughout the study:

- $n^{\mathbf{hard}} = 39$;
- $n^{\mathbf{wien}} = 39$.

Moreover, when parameters values are not explicitly given, we use the default values shown in section 3.7.

3.2 Influence of $N^{\mathbf{hard}}$ and $N^{\mathbf{wien}}$

Parameters used for this study:

- $\tau^{\mathbf{hard}} = 2500$ if $\sigma < 40$, and 5000 otherwise. Moreover, if $N^{\mathbf{hard}} \geq 32$, this threshold is multiplied by a factor 5;
- $\tau^{\mathbf{hard}}_{2D}$ is a Bior1.5 transform whatever the value of σ;
- $\tau^{\mathbf{wien}}_{2D}$ is a 2D DCT transform, whatever the value of σ.

	$N^{\mathbf{hard}} = 8$							
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.54	1.35	**45.56**	**1.34**	**45.56**	**1.34**	45.55	1.35
5	39.41	2.73	39.44	2.72	**39.45**	**2.72**	39.44	2.72
10	35.03	4.52	35.07	4.50	**35.08**	**4.49**	35.07	4.50
20	30.90	7.27	30.94	7.24	**30.96**	**7.22**	30.95	7.23
30	28.70	9.37	28.76	9.30	**28.78**	**9.28**	28.78	9.28
40	27.18	11.16	27.22	11.11	27.25	11.07	**27.26**	**11.05**
60	25.28	13.88	25.35	13.77	25.39	13.71	**25.42**	**13.66**
80	24.05	16.00	24.14	15.83	24.22	15.69	**24.24**	**15.65**
100	22.90	18.26	23.04	17.97	23.14	17.76	23.20	17.64
	$N^{\mathbf{hard}} = 16$							
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.53	1.35	45.55	1.35	45.55	1.35	45.54	1.35
5	39.41	2.73	39.44	2.72	39.44	2.72	39.42	2.73
10	35.04	4.51	35.07	4.50	35.07	4.50	35.06	4.50
20	30.92	7.25	**30.96**	**7.22**	**30.96**	**7.22**	30.95	7.23
30	28.72	9.34	28.76	9.30	**28.78**	**9.28**	28.77	9.29
40	27.20	11.13	27.23	11.09	27.24	11.08	27.24	11.08
60	23.30	17.44	25.35	13.77	25.39	13.71	25.41	13.68
80	24.06	15.98	24.15	15.81	24.21	15.71	24.23	15.67
100	22.93	18.20	23.07	17.91	23.17	17.70	23.22	17.60

$N^{\mathbf{hard}} = 32$								
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.50	1.35	45.51	1.35	45.51	1.35	45.50	1.35
5	39.37	2.74	39.39	2.74	39.39	2.74	39.37	2.74
10	35.01	4.53	35.03	4.52	35.01	4.53	35.01	4.53
20	30.91	7.26	30.93	7.25	30.92	7.25	30.90	7.27
30	28.72	9.34	28.75	9.31	28.75	9.31	28.73	9.33
40	27.19	11.14	27.21	11.12	27.22	11.11	27.20	11.13
60	25.34	13.79	25.39	13.71	25.41	13.68	**25.42**	**13.66**
80	24.07	15.96	24.13	15.85	24.17	15.78	24.16	15.80
100	23.14	17.76	23.21	17.62	23.27	17.50	**23.28**	**17.48**
$N^{\mathbf{hard}} = 64$								
	$N^{\mathbf{wien}} = 8$		$N^{\mathbf{wien}} = 16$		$N^{\mathbf{wien}} = 32$		$N^{\mathbf{wien}} = 64$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.47	1.36	45.48	1.36	45.47	1.36	45.46	1.36
5	39.33	2.75	39.35	2.75	39.34	2.75	39.32	2.76
10	34.97	4.55	34.99	4.54	34.98	4.55	34.95	4.56
20	30.88	7.29	30.90	7.27	30.88	7.29	30.85	7.31
30	28.70	9.37	28.73	9.33	28.72	9.34	28.69	9.38
40	27.15	11.20	27.18	11.16	27.17	11.17	27.15	11.20
60	25.32	13.82	25.36	13.76	25.38	13.73	25.38	13.73
80	24.06	15.98	24.11	15.89	24.13	15.85	24.12	15.87
100	23.12	17.80	23.19	17.66	23.23	17.58	23.23	17.58

In **bold***: best result for a given* σ*.*

One can see that those parameters have a very small influence on the result, and values given in the original paper ($N^{\mathbf{hard}} = 16$ and $N^{\mathbf{wien}} = 32$) are very close to the best result, whatever the value of the noise. Then, according to this study, the final algorithm will kept original parameters, i.e. $N^{\mathbf{hard}} = 16$ and $N^{\mathbf{wien}} = 32$.

3.3 Influence of $\lambda_{3D}^{\mathbf{hard}}$

This parameter is important because it defines the coefficient thresholding level of the 3D group in the transform domain during the first filtering sub-step. The chosen value in the original article is 2.7 and it turns out to be the best choice. Here is a table evaluating its influence, in the case where $\tau_{2D}^{\mathbf{hard}} =$ Bior1.5 and $\tau_{2D}^{\mathbf{wien}} =$ DCT whatever the value of the noise. The error results are given after application of the entire algorithm.

$\lambda_{3D}^{\mathbf{hard}}$	2.5		2.7		3.0		3.2	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.51	1.35	**45.55**	**1.34**	**45.55**	**1.34**	**45.55**	**1.34**
5	39.39	2.73	**39.45**	**2.72**	**39.45**	**2.72**	39.41	2.73
10	34.96	4.55	**35.04**	**4.51**	35.02	4.52	34.96	4.55
20	30.88	7.29	**30.97**	**7.21**	30.93	7.25	30.82	7.34
30	28.63	9.44	**28.74**	**9.32**	28.65	9.41	28.52	9.56
40	27.12	11.23	**27.26**	**11.06**	27.19	11.14	27.07	11.30
60	25.15	14.10	**25.33**	**13.80**	25.26	13.92	25.07	14.23
80	24.00	16.09	**24.26**	**15.61**	24.24	15.64	24.08	15.95
100	23.00	18.05	**23.21**	**17.62**	23.00	18.05	22.77	18.53

In **bold** *the best result for a given* σ*.*

One can see that small variations on this parameter have strong influence on the result. Then this parameter needs to be carefully chosen, and according to the original article, the final algorithm will kept original parameter, i.e. $\lambda_{3D}^{\mathbf{hard}} = 2.7$.

3.4 Influence of the Thresholds $\tau^{\mathbf{hard}}$ and $\tau^{\mathbf{wien}}$

The thresholds $\tau^{\mathbf{hard}}$ and $\tau^{\mathbf{wien}}$ are highly dependent on σ and very influential. One can lose many dBs in PSNR by choosing wrong values for these thresholds. Their correct evaluation is crucial.

However, by picking them large enough, it is possible to keep a constant value for these thresholds for a wide range of σ's. In the original article a value of (2500, 400) is proposed for the thresholds pair $(\tau^{\mathbf{hard}}, \tau^{\mathbf{wien}})$ for $\sigma \in [0, 40]$. These values give good results.

	$\tau^{\mathbf{wien}} = 20$			$\tau^{\mathbf{hard}} = 600$	
$\tau^{\mathbf{hard}}$	PSNR	RMSE	$\tau^{\mathbf{wien}}$	PSNR	RMSE
100	37.02	3.59	5	37.31	3.47
200	37.43	3.43	10	37.50	3.40
300	37.59	3.36	15	37.58	3.37
400	37.61	3.35	20	37.63	3.35
600	37.63	3.35	30	37.68	3.33
1200	37.63	3.35	45	37.71	3.32
2500	37.63	3.35	60	37.73	3.31

Taking low values for the thresholds allows us to keep a limited number of similar patches during the patch-matching. But if this number is too limited, not enough denoising is being done. As it is preferable to work with few similar patches, for some values of the threshold this limit on the number of similar patches is reached for many reference patches. This explains the PSNR stagnation when the thresholds are increased. On the other hand, if these values increase too much, patches significantly different from the reference patch will bring spurious details to the final result. Then, there will be a fall in the PSNR. A good balance must be found. Yet, the breathing space is quite substantial, and it is possible to get generic thresholds for a wide range of σ's.

In the original article, a distinction is done for high value of noise, i.e. $\sigma > 40$. For high noise, $\tau^{\mathbf{hard}}$ is increased to a value of 5000. Here is a study of this parameter for high value of noise, with $\tau_{2D}^{\mathbf{hard}}$ = Bior1.5 and $\tau_{2D}^{\mathbf{wien}}$ = 2D DCT:

$\tau^{\mathbf{hard}}$	2500		5000		10000		25000	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
40	**27.27**	**11.03**	**27.27**	**11.03**	**27.27**	**11.03**	**27.27**	**11.03**
50	26.27	12.38	**26.30**	**12.35**	**26.30**	**12.35**	26.29	12.35
60	25.40	13.69	25.43	13.64	**25.44**	**13.63**	**25.44**	**13.63**
70	24.61	15.00	**24.74**	**14.77**	24.73	14.79	24.73	14.79
80	23.77	16.52	**24.21**	**15.70**	24.19	15.73	24.19	15.73
90	23.18	17.67	23.68	16.69	23.71	16.64	**23.72**	**16.62**
100	22.81	18.45	23.26	17.51	23.34	17.35	**23.35**	**17.34**

In **bold***: best results for a given* σ*.*

One can see that an increasing of $\tau^{\mathbf{hard}}$ is necessary for a very high noise. But increasing a lot is useless because there is a stagnation of the result, due to the maximum number $N^{\mathbf{hard}}$ of similar patches kept. Then once again the value of the original article will be kept for high noise.

3.5 Influence of the Size of the Patches: $k^{\mathbf{hard}}$ and $k^{\mathbf{wien}}$

The window size of the patches influences the result and must be adapted to σ. Indeed, for low values of σ the window size must be relatively small to be well adapted to the details, whereas for large values of σ larger window sizes are better, because most of the details of the image are anyway lost in noise. Since much of the noise is canceled in the first step, we can work with smaller patches in the second step.

In the original article, a patch size of $k^{\mathbf{hard}} = k^{\mathbf{wien}} = 8$ (resp. $k^{\mathbf{hard}} = k^{\mathbf{wien}} = 12$, but only if the 2D DCT is chosen as τ_{2D}) is proposed for $\sigma \leq 40$ (resp. $\sigma > 40$).

As Bior1.5 can not be applied if the patch size is not a power of 2, τ_{2D} is always chosen equal to 2D DCT for this study.

	$k^{\mathbf{hard}} = 4$									
	$k^{\mathbf{wien}} = 4$		$k^{\mathbf{wien}} = 6$		$k^{\mathbf{wien}} = 8$		$k^{\mathbf{wien}} = 10$		$k^{\mathbf{wien}} = 12$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.58	1.34	**45.59**	**1.34**	**45.59**	**1.34**	45.54	1.35	45.51	1.35
5	39.42	2.73	39.46	2.71	39.45	2.72	39.40	2.73	39.36	2.74
10	34.97	4.55	35.05	4.51	35.05	4.51	35.00	4.53	34.96	4.56
20	30.65	7.48	30.83	7.33	30.86	7.30	30.82	7.34	30.80	7.35
30	28.25	9.86	28.52	9.56	28.58	9.50	28.55	9.53	28.54	9.54
40	26.67	11.83	27.05	11.33	27.13	11.22	27.12	11.23	27.10	11.26
60	24.32	15.51	24.81	14.66	24.96	14.41	24.97	14.39	24.97	14.39
80	22.79	18.49	23.47	17.10	23.60	16.85	23.61	16.83	23.59	16.87
100	21.51	21.43	22.31	19.55	22.51	19.10	22.52	19.08	22.53	19.06

	$k^{\mathbf{hard}} = 8$									
	$k^{\mathbf{wien}} = 4$		$k^{\mathbf{wien}} = 6$		$k^{\mathbf{wien}} = 8$		$k^{\mathbf{wien}} = 10$		$k^{\mathbf{wien}} = 12$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.54	1.35	45.56	1.34	45.55	1.35	45.51	1.35	45.48	1.36
5	**39.47**	**2.71**	39.45	2.72	39.43	2.72	39.39	2.74	39.35	2.75
10	**35.10**	**4.48**	35.09	4.49	35.06	4.50	35.01	4.53	34.98	4.55
20	**31.01**	**7.18**	31.00	7.19	30.96	7.22	30.93	7.25	30.89	7.28
30	28.78	9.28	**28.79**	**9.27**	28.77	9.29	28.74	9.32	28.72	9.34
40	27.32	10.98	**27.36**	**10.93**	27.34	10.95	27.34	10.95	27.31	10.99
60	25.19	14.03	25.31	13.84	25.36	13.76	**25.38**	**13.73**	25.37	13.74
80	24.11	15.89	**24.27**	**15.60**	**24.27**	**15.60**	**24.27**	**15.60**	24.24	15.65
100	22.67	18.75	23.06	17.93	23.17	17.71	23.21	17.62	**23.23**	**17.58**
	$k^{\mathbf{hard}} = 12$									
	$k^{\mathbf{wien}} = 4$		$k^{\mathbf{wien}} = 6$		$k^{\mathbf{wien}} = 8$		$k^{\mathbf{wien}} = 10$		$k^{\mathbf{wien}} = 12$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.45	1.36	45.50	1.35	45.50	1.35	45.46	1.36	45.43	1.36
5	39.38	2.74	39.39	2.74	39.36	2.74	39.31	2.76	39.28	2.77
10	35.01	4.53	35.00	4.53	34.97	4.55	34.91	4.58	34.87	4.60
20	30.90	7.27	30.87	7.30	30.83	7.33	30.78	7.37	30.74	7.41
30	28.65	9.42	28.64	9.43	28.60	9.47	28.56	9.52	28.52	9.56
40	27.14	11.21	27.16	11.18	27.11	11.25	27.09	11.27	27.04	11.34
60	25.05	14.26	25.19	14.03	25.24	13.95	25.24	13.95	25.22	13.98
80	23.92	16.24	24.07	15.96	24.05	16.00	24.01	16.07	23.95	17.93
100	22.51	19.10	22.92	18.22	23.02	18.01	23.05	17.95	23.06	17.93

In **bold***: best results for a given* σ*.*

An other influence of the size of the patch is on the processing time. Indeed smaller patches give faster algorithm. Then, as results for $\sigma > 40$ for $k^{\mathbf{wien}} = 8$ or $k^{\mathbf{wien}} = 12$ are really close, we will prefer to kept $k^{\mathbf{wien}} = 8$ because of the processing time.

3.6 Influence of $p^{\mathbf{hard}}$ and $p^{\mathbf{wien}}$

In order to speed up the algorithm, it is possible to use a step p in both rows and columns to go from one reference patch to the next. For example, a step of 3 theoretically divides by 9 the processing time. In the original article a step of 3 is proposed. The PSNR loss due to this step use is negligible for large values of noise. Nevertheless, for low noise, it is better to keep a step of 1 or 2. Moreover, since the second step works on a first basic estimate assumed to be noiseless, or at least with lower noise, a bigger step for the first step than for the second can be used.

$p^{\mathbf{hard}} = 1$						
	$p^{\mathbf{wien}} = 1$		$p^{\mathbf{wien}} = 3$		$p^{\mathbf{wien}} = 5$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**45.59**	**1.34**	45.56	1.34	45.50	1.35
5	**39.53**	**2.69**	39.49	2.70	39.41	2.73
10	**35.10**	**4.48**	35.07	4.50	34.89	4.59
20	**31.01**	**7.18**	30.97	7.21	30.88	7.29
30	**28.79**	**9.27**	28.76	9.30	28.66	9.41
40	**27.32**	**10.98**	27.31	10.99	27.26	11.05
60	25.37	13.74	25.35	13.77	25.31	13.84
80	**24.23**	**15.67**	**24.23**	**15.67**	24.21	15.71
100	22.87	18.32	22.87	18.32	22.86	18.35
$p^{\mathbf{hard}} = 3$						
	$p^{\mathbf{wien}} = 1$		$p^{\mathbf{wien}} = 3$		$p^{\mathbf{wien}} = 5$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.58	1.34	45.55	1.35	45.49	1.36
5	39.51	2.70	39.47	2.71	39.39	2.74
10	35.09	4.49	35.05	4.51	34.96	4.56
20	30.99	7.20	30.92	7.25	30.85	7.31
30	28.76	9.30	28.73	9.33	28.64	9.43
40	27.31	10.99	28.29	9.82	27.24	11.08
60	**25.44**	**13.63**	25.42	13.66	25.36	13.76
80	24.19	15.74	24.19	15.74	24.17	15.78
100	**23.20**	**17.64**	23.19	17.66	23.19	17.66
$p^{\mathbf{hard}} = 5$						
	$p^{\mathbf{wien}} = 1$		$p^{\mathbf{wien}} = 3$		$p^{\mathbf{wien}} = 5$	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.54	1.35	45.51	1.35	45.44	1.36
5	39.47	2.71	39.43	2.72	39.34	2.75
10	35.04	4.51	35.00	4.53	34.90	4.59
20	30.93	7.25	30.89	7.28	30.78	7.37
30	28.70	9.37	28.67	9.40	28.55	9.53
40	27.22	11.11	27.20	11.13	27.15	11.20
60	25.31	13.84	25.29	13.87	25.22	13.98
80	24.09	15.92	24.08	15.94	24.07	15.96
100	23.03	17.99	23.02	18.01	23.01	18.03

In **bold***: best results for a given* σ*.*

As the loss in PSNR is negligible compared to the gain in speed, and for the sake of fidelity to the original article, we keep $p = 3$ for both steps in the provided code.

3.7 Summary Table

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

	$\sigma \leq 40$	$\sigma > 40$
$N^{\mathbf{hard}}$	16	16
$N^{\mathbf{wien}}$	32	32
$\lambda_{3D}^{\mathbf{hard}}$	2.7	2.7
$\tau^{\mathbf{hard}}$	2500	5000
$\tau^{\mathbf{wien}}$	400	3500
$k^{\mathbf{hard}}$	8	8
$k^{\mathbf{wien}}$	8	8
$p^{\mathbf{hard}}$	3	3
$p^{\mathbf{wien}}$	3	3

4 A Detailed Study of Possible Variants

This part examines some ambiguous choices of the original method and experimentally decides for the best choice in terms of PSNR. Unless otherwise specified, in the following $\tau_{2D}^{\mathbf{hard}} = \tau_{2D}^{\mathbf{wien}}$ refer to transform thresholds with the 2D DCT, and $\tau_{1D}^{\mathbf{hard}} = \tau_{1D}^{\mathbf{wien}}$ to the 1D DCT. Moreover shown results have been obtained on the image shown on figure 4, and when it is not explicitly said, default parameter values shown in section 3.7 have been used.

4.1 Variants of the First Denoising Step

Grouping

Distance: To calculate the distance, we can choose between doing it on the patches obtained after τ_{3D} or directly on the patches of the image. If the 3D transform is an isometry, the distances are equal, so the distance will be calculated directly on the image.

Normalization: Several classic DCT definitions do not normalize the first coefficient. For the 1D case, the first coefficient is for example not divided by $\sqrt{2}$. If we do not take into account this fact during the hard-thresholding, the same threshold will be applied, whatever the coefficient. The table below shows that this makes a significant difference. Thus, some care must be taken to use a normalized DCT.

	non-normalized DCT		normalized DCT	
σ	PSNR	RMSE	PSNR	RMSE
2	41.70	2.10	41.53	2.14
5	38.02	3.20	38.00	3.21
10	35.10	4.48	35.21	4.42
20	31.65	6.67	32.36	6.14
30	29.13	8.91	30.29	7.80
40	27.21	11.11	28.70	9.36

Fixed parameters:
distance: image, aggregation: with weighting, step: 3, patches: 8×8*, collaborative Filtering: DCT.*

Thresholding $\tau_{2D}^{\mathbf{hard}}$: In the original article, this threshold only appears from $\sigma > 40$ since we initialized $\lambda_{2D} = 0$ for $\sigma \leq 40$. Applying a threshold to the 2D transform coefficients for low values of σ is useless since there is no improvement after the second step. Moreover for a noise lower than 5 the results are degraded.

Collaborative Filtering

For the 3D group, a choice must be done between:

- Simply averaging image patches along the third dimension, which would be a primitive version of NL-means. In NL-means there also is a weighting of the patches according to their distance. This simple average solution is denoted by *basic NL-means*;
- Applying a hard-thresholding on a 1D transform along the third dimension, or in other terms apply τ_{3D} as described before, which is denoted *Hard Thresholding*.

The next table shows the considerable amelioration obtained by adding a thresholded 3D transform along patches compared to a simple average of the patches in the case where $\tau_{3D}^{\mathbf{hard}} = \tau_{3D}^{\mathbf{wien}} = DCT$:

	Collaborative Filtering			
	basic NL-means		Hard Thresholding	
σ	PSNR	RMSE	PSNR	RMSE
2	40.96	2.28	41.36	2.18
5	36.40	3.86	37.80	3.28
10	33.54	5.36	34.72	4.68
20	29.54	8.50	31.41	6.86
30	27.01	11.36	28.96	9.09
40	25.04	14.26	27.14	11.21

Fixed parameters:
distance: image, aggregation: with weighting, step: 3, normalized DCT, patches: 8×8.

Aggregation

During the aggregation the weighting is based on the number of coefficients canceled during the hard thresholding. However, as shown by the next table, this weighting works, but does not play such an important part in the PSNR improvement. Nevertheless, artifacts on edges are visually attenuated by using a weighted aggregation (see figure 2.2).

	Without weighting		With weighting	
σ	PSNR	RMSE	PSNR	RMSE
2	41.26	2.20	41.53	2.14
5	37.83	3.27	38.00	3.21
10	34.83	4.62	35.21	4.42
20	32.09	6.33	32.36	6.14
30	30.04	8.02	30.29	7.80
40	28.60	9.47	28.70	9.36

Fixed Parameters:
distance: image, normalized DCT, step: 3, collaborative filtering: DCT, patches: 8×8.

Because they have many coefficients that are thresholded, the aggregation weighting enforces the role of homogeneous patches compared to patches containing edges. The issue with this weighting is: how to choose the right homogeneity indicator?

Another natural indicator would be the standard deviation of the patches. Indeed, for homogeneous patches, the standard deviation would be very small, whereas for patches containing edges, the standard deviation would be much bigger. Then the weighting would be the inverse of the standard deviation, which would mean for a 3D group

$$w_P^{\mathbf{hard},\mathbf{wien}} = \left(\frac{1}{N-1} \sum_{k=1}^{N} \sum_{i=1}^{M} \left(\mathbb{P}(P)(x_{i,k})\right)^2 \right)^{-\frac{1}{2}} \tag{12}$$

where:

- N is the number of similar patches to P;
- $M = k^{\mathbf{hard}} \times k^{\mathbf{hard}}$ or $M = k^{\mathbf{wien}} \times k^{\mathbf{wien}}$.

Then homogeneous patches would have more weights than patches containing edges.
One can also wonder if it would not be better to calculate this standard deviation on the 3D group rather than on its estimate. In the following we denote:

- *H.T.* the original weighting in step 1 (i.e., the inverse of the number of coefficients different from zero during the hard thresholding);
- *STD* the weighting using the standard deviation processed on the original image;
- *STD C.F.* the weighting using the standard deviation processed on the estimate of the 3D group.

The following table compares the new weighting results and shows that H.T. wins:

	Without weighting		*H.T.*		*STD*		*STD C.F.*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.13	1.41	45.13	1.41	**45.14**	**1.41**	45.13	1.41
5	40.68	2.36	**40.71**	**2.35**	**40.71**	**2.35**	40.69	2.35
10	37.26	3.50	**37.31**	**3.47**	37.29	3.48	37.27	3.49
20	33.58	5.34	**33.65**	**5.30**	33.58	5.34	33.57	5.35
30	31.10	7.10	**31.17**	**7.04**	31.10	7.11	31.08	7.12
40	29.35	8.68	**29.41**	**8.62**	29.35	8.69	29.32	8.71

In **bold***: best result for a given* σ*.*

Another question is: why is the weight computed for the whole 3D group when a priori every 2D patch of the 3D group could be given a different weight? This would yield more differentiated estimates for each pixel.

The answer is experimental. The next table shows that giving a single weight to the whole 3D group is better than giving a weight to each patch:

	2D weighting		3D weighting	
σ	PSNR	RMSE	PSNR	RMSE
2	45.08	1.42	45.13	1.41
5	40.44	2.42	40.71	2.35
10	36.94	3.63	37.31	3.47
20	33.16	5.61	33.65	5.30
30	30.64	7.49	31.17	7.04
40	28.86	9.19	29.41	8.62

4.2 Variants of the Second Denoising Step

Wiener Filtering

The second step adds details for large σ values, as illustrated in the experiments below. Using a Wiener filter in this second step rather than a hard thresholding avoids losing details. Nevertheless the weighting does not visually improve much the image near edges, and the gain in PSNR is negligible.

	Step 1 only		Step 1 (fixed) + Step 2 (collaborative filtering)					
			H. T. + weight.		Wiener Filtering		W. F. + weighting	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.13	1.41	43.18	1.77	45.21	1.40	45.22	1.40
5	40.70	2.35	39.12	2.82	40.86	2.31	40.90	2.29
10	37.31	3.47	35.97	4.06	37.61	3.36	37.66	3.34
20	33.65	5.30	32.75	5.87	34.23	4.96	34.27	4.92
30	31.17	7.04	30.64	7.49	32.01	6.40	32.03	6.38
40	29.41	8.62	29.08	8.96	30.44	7.66	30.47	7.64

Fixed parameters for:
step 1 — distance: image, aggregation: with weighting, collaborative filtering: DCT, patches: 8×8*, step: 3,*
step 2 — distance: image, patches: 8×8*, step: 3.*

This table shows that the second step is worthwhile and important, especially for large noise values. However, the weighting in the second step is much less useful than the weighting in the first step. Using a Wiener filter during the second step and not having to repeat the hard thresholding shows the importance of this improvement. The gain is important both in PSNR and visually since it attenuates the noise again and improves/adds details at the same time.

Ideal Wiener Filtering

It is also possible to compare the ordinary Wiener filter with an ideal Wiener filter, which is obtained when the original noise-free image is taken as oracle reference. This ideal Wiener filter is the best possible estimate for the second step of this algorithm. *It is therefore interesting to see how far we stand from this ideal estimate with the current one.*

	Step 1 + Step 2		Ideal Wiener	
σ	PSNR	RMSE	PSNR	RMSE
2	45.24	1.39	47.81	1.04
5	40.87	2.31	43.19	1.77
10	37.64	3.34	39.98	2.56
20	34.25	4.94	36.99	3.60
30	32.07	6.35	35.21	4.43
40	30.40	7.70	33.71	5.26

Fixed parameters:
step 1 — distance: image, aggregation: with weighting, collaborative filtering: DCT, patches: 8×8*, step: 3,*
step 2 — distance: image, patches: 8×8*, step: 3.*

As expected, the ideal Wiener filter gives a better result. A Wiener filter in the second step seems to be some 3 dB away from the ideal result.

Aggregation

Wien denotes the original weighting in the second step (i.e., the inverse of the empirical Wiener coefficients norm). As for the first step, would it be possible to improve the weighting by using the standard deviation instead of the norm of the empirical Wiener coefficients?

	H.T. | *Wien*		*STD* | *STD*		*STD C.F.* | *STD C.F.*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	**45.22**	**1.40**	45.12	1.41	**45.22**	**1.40**
5	**40.89**	**2.30**	40.71	2.35	40.88	2.30
10	**37.66**	**3.34**	37.45	3.42	37.64	3.35
20	**34.27**	**4.93**	34.10	5.03	34.26	4.94
30	**32.03**	**6.38**	31.92	6.46	**32.03**	**6.38**
40	30.47	7.64	30.36	7.73	**30.48**	**7.63**

In **bold***: best result for a given* σ*.*

Moreover, it is possible to switch the weightings?

	H.T. | *STD*		*H.T.* | *STD C.F.*		*STD* | *Wien*		*STD C.F.* | *Wien*	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.18	1.40	**45.23**	**1.40**	45.18	1.40	45.22	1.40
5	40.82	2.32	**40.89**	**2.30**	40.81	2.32	40.88	2.30
10	37.57	3.37	**37.66**	**3.34**	37.55	3.38	37.64	3.44
20	34.20	4.97	**34.28**	**4.93**	34.18	4.98	34.26	4.94
30	32.00	6.40	**32.06**	**6.36**	31.96	6.43	32.02	6.39
40	30.43	7.67	**30.50**	**7.61**	30.42	7.68	30.47	7.64

In **bold***: best result for a given* σ*.*

Even though the improvement is minor, it is still possible to improve the result a little bit by working on the aggregation weighting. The weighting in this second step seems useless. Yet, better results are obtained by modifying the weighting in the second step. Thus, this study underlines that it is still possible to gain a bit by working on the weighting in the aggregation sub-step.

4.3 Influence of the 3D Transform

The choice of the 3D transform (τ_{2D} and τ_{1D}) is crucial. Here are some choices:

- τ_{2D}: This 2D transform is applied on each patch of the 3D group. We have the choice between a normalized 2D DCT and a bi-orthogonal spline wavelet, where the vanishing moments of the decomposing and reconstructing wavelet functions are 1 and 5 respectively. We shall denote by Bior1.5 this bi-orthogonal spline wavelet;

- τ_{1D}: This 1D transform is applied along the third dimension of the 3D group, after the 2D transform has been applied. We have the choice between a normalized 1D DCT and a Walsh-Hadamard transform.

In order to clarify the use of those transforms, here is a brief explanation of the practical implementation of the Bior1.5 and the Walsh-Hadamard transforms, as they are used in this implementation of BM3D.

4.3.1 Walsh-Hadamard Transform

This 1D transform is very simple. Since it recursively processes the sums and differences between pair of values of the vector it is applied to, the size of this vector must be a power of 2. This forces

the number of similar patches in a 3D group to be a power of 2 itself. Let us denote for instance the first four coordinates of the processed vector by $V_0 = [a\ b\ c\ d]$. The first step computes the sums and the differences of the pairs of values and regroups them, which leads to

$$V_1 = [(a+b)\ (c+d)\ (a-b)\ (c-d)]$$

The process is iterated on each pair: $V_1^h = [(a+b)\ (c+d)]$ and $V_1^l = [(a-b)\ (c-d)]$, which yields the final vector

$$V_f = [(a+b+c+d)\ (a+b-c-d)\ (a-b+c-d)\ (a-b-c+d)].$$

Moreover, in order to keep the norm of the initial vector, it is necessary to normalize V_f by $\sqrt{N}$, where N is the vector dimension. In that case, the inverse Walsh-Hadamard transform is exactly the same as the forward transform.

4.3.2 Bior1.5

To obtain this transform, it is necessary to have four filters, which values were taken from a previous article by David Donoho [7]:

- low frequency filter for the forward transform:

$$lpd = \frac{\sqrt{2}}{256}[3;\ -3;\ -22;\ 22;\ 128;\ 128;\ 22;\ -22;\ -3;\ 3]$$

- high frequency filter for the forward transform:

$$hpd = \frac{\sqrt{2}}{2}[0;\ 0;\ 0;\ 0;\ -1;\ 1;\ 0;\ 0;\ 0;\ 0]$$

- low frequency filter for the backward transform:

$$lpr = \frac{\sqrt{2}}{2}[0;\ 0;\ 0;\ 0;\ 1;\ 1;\ 0;\ 0;\ 0;\ 0]$$

- high frequency filter for the backward transform:

$$hpr = \frac{\sqrt{2}}{256}[3;\ 3;\ -22;\ -22;\ 128;\ -128;\ 22;\ 22;\ -3;\ -3]$$

The 2D transform is separable, first done in column, then in row (or vice versa). Thus it is enough to explain the 1D transform only. Let us denote by V_0 a vector of size $N = 2^n$. Then V_1^l and V_1^h — which sizes are 2^{n-1} — are obtained by the following steps.

- First of all, V is periodically extended beyond its boundaries. Denote for instance $V_0 = [a\ b\ c\ d\ e\ f\ g\ h]$, of size $N = 8$. Then the extension is

$$\tilde{V_0} = [d\ e\ f\ g\ h\ a\ b\ c\ d\ e\ f\ g\ h\ a\ b\ c\ d\ e]$$

  The periodization is done symmetrically in order to have at any position from $M/2$ to $M/2+N$ all values of V_0 and only those values for the discrete convolution of V_0 with *lpd* or *hpd*. Here M is the size of the transform, *lpd* or *hpd*. For example, at the first position $M/2$, the vector with which the discrete convolution is done has for values: $[d\ e\ f\ g\ h\ a\ b\ c]$. If the periodization was done anti-symmetrically, this same vector would be $[f\ e\ d\ c\ b\ a\ b\ c]$ and then the value g would not be represented and the values b and c would be over-represented.

- The discrete convolution of $\tilde{V_0}$ with *lpd* and *hpd* is computed:

$$\forall i \in [0, 2^{n-1} - 1], V_1^l[i] = \sum_{k=0}^{9} \tilde{V_0}[2i + k] lpd[k]$$

$$\forall i \in [0, 2^{n-1} - 1], V_1^h[i] = \sum_{k=0}^{9} \tilde{V_0}[2i + k] hpd[k]$$

Then we get $V_1 = [V_1^h \; V_1^l]$. The process is only iterated on high frequencies part of the sub-vector V_1: V_1^h, which size is 2^{n-1}. The backward transform is done in a similar way:

- Let it be started for instance from $V_2 = [V_2^h \; V_2^l V_1^l]$. For a sake of clarity, we remind that the size of V_2^h and V_2^l is 2^{n-2} and the size of V_1^l is 2^{n-1} if the size of $V_2 = 2^n$.
- V_2^h and V_2^l are periodically extended;
- V_1^h is obtained by the following convolutions:

$$\forall i \in [0, 2^{n-2} - 1], V_1^h[i] = \sum_{k=0}^{4} \left(hpr[2k] V_2^h[k + i] + hpr[2k + 1] V_2^l[k + i] \right)$$

$$\forall i \in [0, 2^{n-2} - 1], V_1^h[i + 2^{n-2}] = \sum_{k=0}^{4} \left(lpr[2k] V_2^h[k + i] + lpr[2k + 1] V_2^l[k + i] \right)$$

- The iteration is done on $V_1 = [V_1^h \; V_1^l]$.

In order to show the importance of the choice of the transforms τ_{2D} and τ_{1D}, the table 1 contains the the result of a comparative study led on the different possible choices.

In all cases the size of patches is 8×8, except when we use a 2D DCT for $\sigma \geq 40$: in this case the size is 12×12. One reason which could explain the utility of not using the same 2D transform for both steps is that the artifacts created by the chosen transform in the first step are not enhanced during the second step, which is the case if the same transform is used twice.

According to those results, the best combination of transforms is $\tau_{2D}^{\mathbf{hard}}$ = Bior1.5 and $\tau_{1D}^{\mathbf{hard}}$ = Hadamard for the step 1, and $\tau_{2D}^{\mathbf{wien}}$ = 2D DCT and $\tau_{1D}^{\mathbf{wien}}$ = Hadamard for the step 2.

4.4 Influence of the Kaiser Window

According to the original article, a Kaiser window (with parameter $\alpha = 2.0$) is applied during the weighting aggregation in order to reduce border effects which can appear when certain 2D transforms (typically 2D DCT) are used. Then an element-by-element multiplication is done between the Kaiser window and the estimated patch during the aggregation of block-wise estimates. Of course, the weight of each coefficient of the Kaiser window is taken into account.

The Kaiser window of length k is defined in 1-D by the formula:

$$K_n = \begin{cases} \dfrac{I_0\left(\pi\alpha\sqrt{1 - \left(\frac{2n}{k-1} - 1\right)^2}\right)}{I_0(\pi\alpha)} & if \; 0 \leq n \leq k - 1 \\ 0 & otherwise \end{cases}$$

	$\tau_{2D}^{\mathbf{hard}}$ | $\tau_{1D}^{\mathbf{hard}}$ | $\tau_{2D}^{\mathbf{wien}}$ | $\tau_{1D}^{\mathbf{wien}}$									
	DCT|DCT|DCT|DCT		DCT|Hadamard|DCT|Hadamard		DCT|Hadamard|Bior|Hadamard		Bior|Hadamard|DCT|Hadamard		Bior|Hadamard|Bior|Hadamard	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	45.11	1.41	45.49	1.35	38.76	2.94	**45.59**	**1.33**	38.58	3.00
5	38.79	2.93	39.29	2.77	36.68	3.74	**39.35**	**2.75**	36.48	3.82
10	34.21	4.97	34.83	4.62	33.94	5.12	**35.00**	**4.53**	33.80	5.21
20	29.89	8.17	30.69	7.44	30.59	7.53	**30.94**	**7.24**	30.51	7.61
30	27.78	10.41	28.44	9.65	28.57	9.51	**28.73**	**9.34**	28.50	9.58
40	26.27	12.39	26.57	11.96	26.96	11.45	**27.23**	**11.10**	27.06	11.31
60	24.52	15.15	24.75	14.75	25.09	14.19	**25.28**	**13.88**	24.91	14.49
80	23.46	17.12	23.67	16.71	24.03	16.02	**24.21**	**15.70**	24.20	15.71
100	22.45	19.22	22.61	18.88	22.74	18.60	**23.11**	**17.83**	22.96	18.14

In **bold***: best PSNR for a given* σ*.*

Table 1: Comparative study on τ_{2D} and τ_{1D}.

where I_0 is the zero$^{\text{th}}$ order modified Bessel function of the first kind:

$$I_0(x) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(m+1)} \left(\frac{x}{2}\right)^{2m}$$

where Γ is the gamma function, a generalization of the factorial function to non-integer values.

Practically $k \times k$ Kaiser windows are hard-coded for $k = 8$ and $k = 12$. If the size of patches is different from those values, Kaiser windows are not used (i.e. they are set to 1). Here an example of a 2-D Kaiser window for $k = 8$ and $\alpha = 2.0$ used in our algorithm:

$$\begin{pmatrix} 0.1924 & 0.2989 & 0.3846 & 0.4325 & 0.4325 & 0.3846 & 0.2989 & 0.1924 \\ 0.2989 & 0.4642 & 0.5974 & 0.6717 & 0.6717 & 0.5974 & 0.4642 & 0.2989 \\ 0.3846 & 0.5974 & 0.7688 & 0.8644 & 0.8644 & 0.7688 & 0.5974 & 0.3846 \\ 0.4325 & 0.6717 & 0.8644 & 0.9718 & 0.9718 & 0.8644 & 0.6717 & 0.4325 \\ 0.4325 & 0.6717 & 0.8644 & 0.9718 & 0.9718 & 0.8644 & 0.6717 & 0.4325 \\ 0.3846 & 0.5974 & 0.7688 & 0.8644 & 0.8644 & 0.7688 & 0.5974 & 0.3846 \\ 0.2989 & 0.4642 & 0.5974 & 0.6717 & 0.6717 & 0.5974 & 0.4642 & 0.2989 \\ 0.1924 & 0.2989 & 0.3846 & 0.4325 & 0.4325 & 0.3846 & 0.2989 & 0.1924 \end{pmatrix}$$

One can see that borders of the patch are smoothly decreased and then allow to reduce border effects during the aggregation part.

Here is a study of the influence of the use of a Kaiser window, by using 8×8 patches and $p^{\mathbf{hard}} = p^{\mathbf{wien}} = 3$:

	With		Without	
σ	PSNR	RMSE	PSNR	RMSE
2	**45.61**	**1.34**	45.59	1.34
5	**39.44**	**2.72**	39.42	2.72
10	**35.05**	**4.51**	35.04	4.52
20	**30.97**	**7.21**	30.96	7.22
30	**28.76**	**9.30**	28.75	9.32
40	27.27	11.04	**27.28**	**11.03**
60	25.43	13.65	**25.43**	**13.64**
80	24.23	15.66	**24.24**	**15.64**
100	23.13	17.78	**23.14**	**17.77**

One can see that the use of the Kaiser window has no influence of the PSNR result. Visually, there is no difference between using the Kaiser window or not for all values of noise, even near edges. However, according to the authors of the original BM3D article [4], the use of the Kaiser window is useful for large values of $p^{\mathbf{hard}}$ and $p^{\mathbf{wien}}$. But as we are working with patches of size 8×8, this step between two references patches can not be greater than 4. Then a step greater than 3 has not been tested for studying the influence of the Kaiser window.

However, for the sake of fidelity with the original method, we keep the use of a Kaiser window in the provided code.

5 Extending BM3D to Color Images

Adapting the algorithm to color images is easy and can be done in the following steps:

1. First a transformation to a luminance-chrominance space from the RGB noisy image is applied. Y denotes the luminance channel and by U and V the chrominance channels;

2. For each step:

   - Grouping is only performed with the Y channel;
   - The 3D block built on Y is used for all three channels;
   - Collaborative filtering is applied to each channel separately as well as the weighted aggregation.

3. Return to the RGB space by applying the inverse transformation.

Three classic transformations were tested: the YUV transform denoted by the matrix A_{YUV}, the YCbCr space transform denoted by A_{YCbCr} and, last but not least, a more intuitive transform introduced in the original article, A_{opp}:

$$\begin{pmatrix} 0.30 & 0.59 & 0.11 \\ -0.15 & -0.29 & 0.44 \\ 0.61 & -0.51 & -0.10 \end{pmatrix} \quad \begin{pmatrix} 0.30 & 0.59 & 0.11 \\ -0.17 & -0.33 & 0.50 \\ 0.50 & -0.42 & -0.08 \end{pmatrix} \quad \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

As those matrix are not normalized, the value of σ in each channel must be carefully adapted according to these transforms. The compared denoising performance with these different color transforms is shown in the table below:

	A_{YUV}		A_{YCbCr}		A_{opp}	
σ	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
2	43.43	1.72	43.58	1.69	43.73	1.66
5	36.63	3.76	36.84	3.67	37.14	3.54
10	31.81	6.54	32.03	6.38	32.44	6.09

The choice of the color space transform is therefore important and causes important variations. In conclusion, also verified on the experiments below, for JPEG images the A_{opp} transform is the best choice.

6 Image Denoising Experiments

This experimental study was conducted on noiseless images on which a white Gaussian noise was added. Overall, they show the excellent performance of the denoising algorithm, giving back definitely a better image. The results shown below constitute a qualitative study of the influence of each parameter and of each step on the final result. They demonstrate the very slight improvement obtained by weighting, the definite visual gain obtained by the second step, and prove that for JPEG images the A_{opp} transform is the best choice.

6.1 Grey Level Images

The images hereafter show the result of the weighting for each step of the algorithm:

Step 1, without weighting

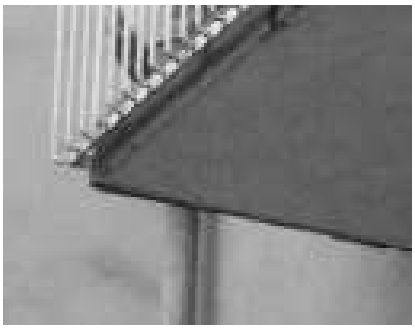
Step 1 with weighting

Step 2, without weighting

Step 2 with weighting

Difference between with and without weighting in both steps.

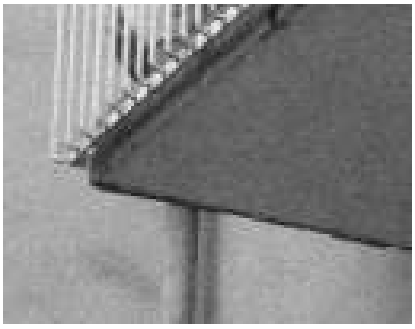
On the next set of images one can see the results for different values of σ for each step of the algorithm:



Noisy image $\sigma = 2$

Basic estimate

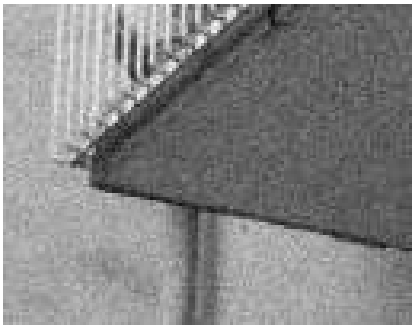
Final estimate



Noisy image $\sigma = 5$

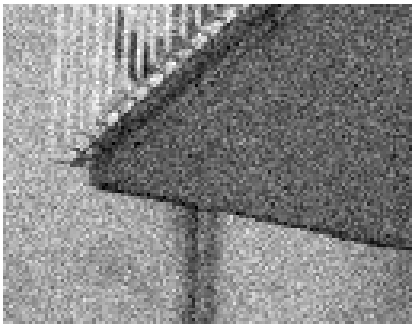
Basic estimate

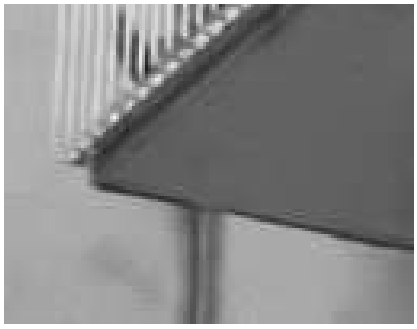
Final estimate


Noisy image $\sigma = 10$

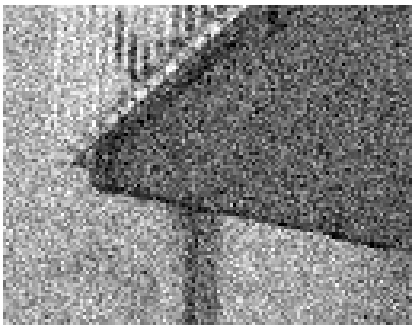

Basic estimate

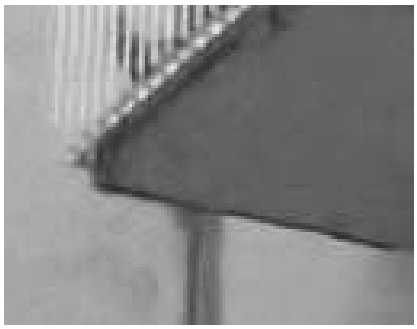

Final estimate

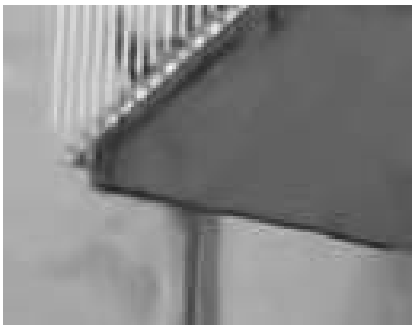

Noisy image $\sigma = 20$

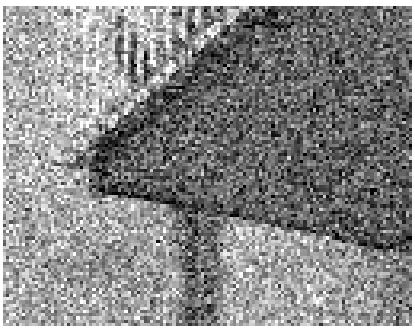

Basic estimate

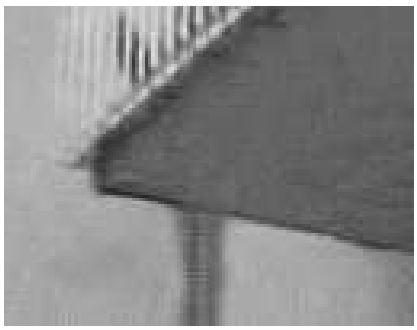

Final estimate

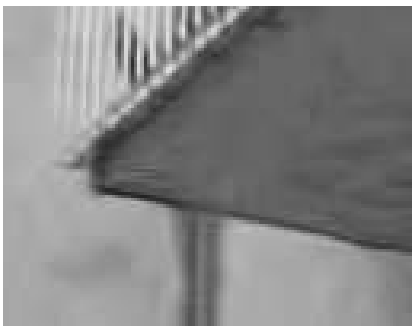

Noisy image $\sigma = 30$


Basic estimate


Final estimate


Noisy image $\sigma = 40$

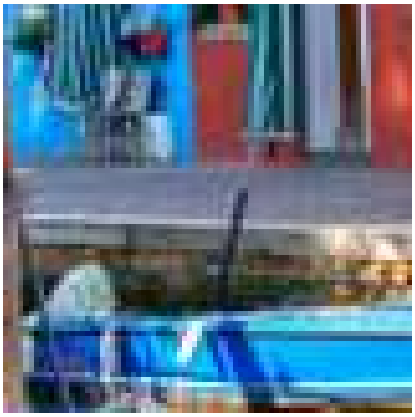

Basic estimate


Final estimate

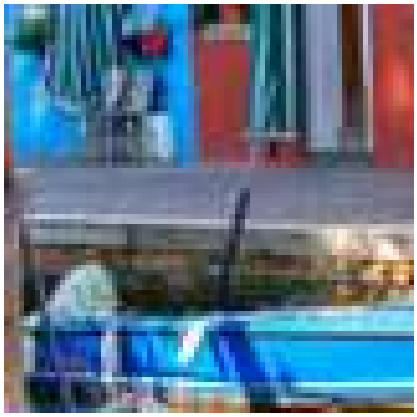
6.2 Color Images

The next image has been used to show the importance of the choice of the color space transform:

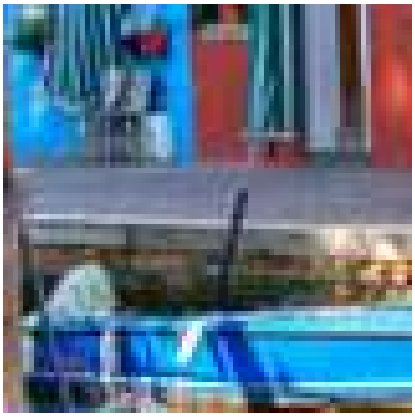
The following set of sub-images shows the results for different choices of the color space transform and different values of the noise:



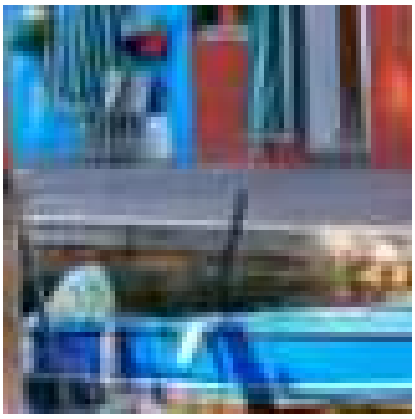
$\sigma = 5$, A_{opp}



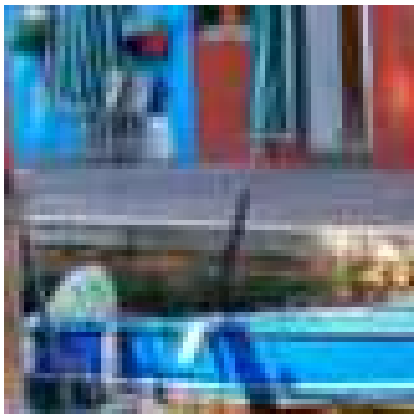
$\sigma = 5$, A_{YUV}



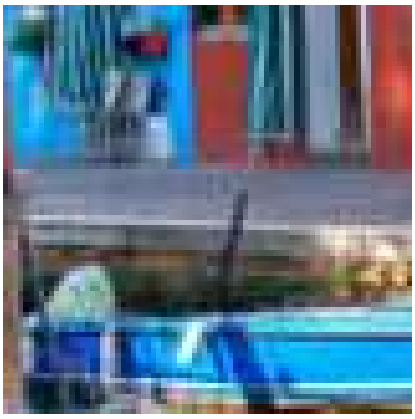
$\sigma = 5$, A_{YCbCr}



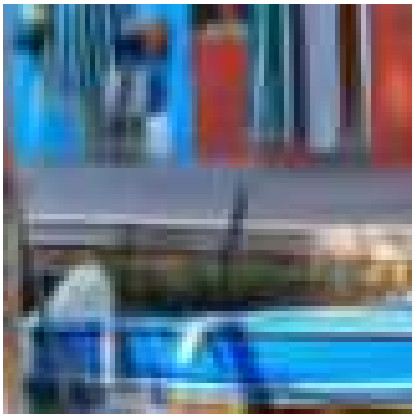
$\sigma = 15$, A_{opp}



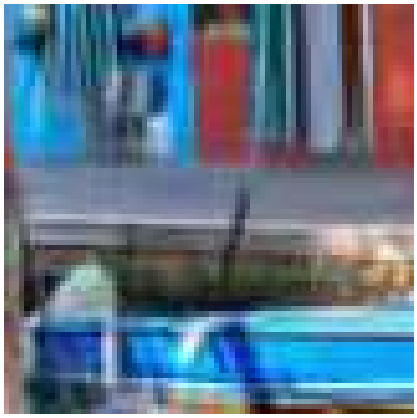
$\sigma = 15$, A_{YUV}



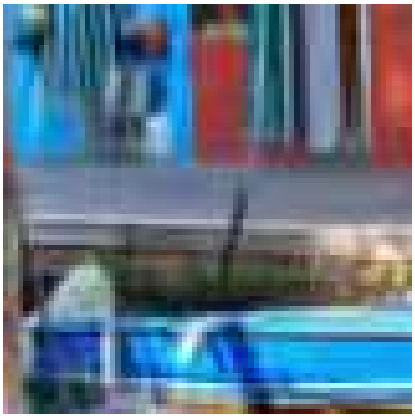
$\sigma = 15$, A_{YCbCr}



$\sigma = 30$, A_{opp}



$\sigma = 30$, A_{YUV}



$\sigma = 30$, A_{YCbCr}

This new set of sub-images shows the results for the same choice of color space transform and values of the noise, but on a different detail of the image:

$\sigma = 5$, A_{opp}

$\sigma = 5$, A_{YUV}

$\sigma = 5$, A_{YCbCr}

$\sigma = 15$, A_{opp}

$\sigma = 15$, A_{YUV}

$\sigma = 15$, A_{YCbCr}

$\sigma = 30$, A_{opp}

$\sigma = 30$, A_{YUV}

$\sigma = 30$, A_{YCbCr}

The images hereafter show for different values of σ the result for each step of the algorithm:

Noisy image $\sigma = 2$

Basic estimate

Final estimate

Noisy image $\sigma = 5$

Basic estimate

Final estimate


Noisy image $\sigma = 10$


Basic estimate


Final estimate


Noisy image $\sigma = 20$


Basic estimate


Final estimate


Noisy image $\sigma = 30$


Basic estimate


Final estimate


Noisy image $\sigma = 40$


Basic estimate


Final estimate

7 Comparison with Several Classic and Recent Methods

In order to evaluate the real capacity of this state-of-the-art denoising method, a fair and precise comparison with other classic and recent methods needs to be done. The other considered methods are: DCT denoising [15], NL-means [1], TV denoising [8], KSVD [13], and BLS-GSM.

More than a dozen of images have been processed for six values of noise between 2 and 40. Moreover, an average on every image for each value of sigma is given to help readers make up their own idea on the relative performance of all methods.

7.1 Images,

These images can be found on the demonstration part of this article on IPOL [10]. They can be considered noise-free, since their real noise is significantly below 1. They have been obtained by applying a drastic zoom out on already good quality outdoor images. Indeed, apply a zoom out by a n factor reduces the noise by a $\sqrt{n}$ factor.

Computer

Dice

Flowers

Girl

Traffic

Valldemossa

7.2 Results

The results of the algorithm as described in this article applied on the six images shown in section 7.1 are summarized in the following tables 7.2, 7.2 and 7.2. One can see that the BM3D algorithm gives the best results for all values of noise on all tested images.

8 Conclusion

This detailed study carried out on BM3D, has led us to the following conclusions:

- This method delivers excellent results.
- The main elements explaining this improvement with respect to former methods are:

$\sigma = 2$												
	BM3D		BLS-GSM		K-SVD		NL-means		DCT denoising		TV denoising	
Image	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
Computer	**45.22**	**1.40**	44.69	1.49	44.52	1.52	44.03	1.60	44.56	1.51	41.68	2.10
Dice	**48.86**	**0.92**	48.59	0.95	47.79	1.04	48.51	0.96	48.49	0.96	45.45	1.36
Flowers	**47.31**	**1.10**	47.12	1.12	47.09	1.13	46.36	1.23	47.14	1.12	43.33	1.74
Girl	**47.40**	**1.09**	47.14	1.12	47.28	1.10	46.96	1.14	46.95	1.15	43.13	1.78
Traffic	**44.56**	**1.51**	44.15	1.58	43.80	1.65	43.55	1.69	44.20	1.57	41.46	2.16
Valldemossa	**44.68**	**1.49**	44.41	1.53	40.08	2.53	43.33	1.74	44.40	1.54	41.32	2.19
Mean	**46.34**	**1.25**	46.02	1.30	45.09	1.49	45.46	1.39	45.96	7.85	42.73	1.89

$\sigma = 5$												
	BM3D		BLS-GSM		K-SVD		NL-means		DCT denoising		TV denoising	
Image	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
Computer	**39.98**	**2.56**	39.30	2.76	39.58	2.68	38.86	2.91	38.98	2.87	36.35	3.88
Dice	**45.80**	**1.31**	45.21	1.40	45.27	1.39	45.12	1.41	45.03	1.43	43.81	1.64
Flowers	**42.99**	**1.81**	42.76	1.86	43.09	1.79	42.05	2.01	42.57	1.90	40.21	2.49
Girl	**44.03**	**1.60**	43.70	1.67	43.59	1.69	43.44	1.72	43.34	1.74	41.21	2.22
Traffic	**38.67**	**2.97**	38.10	3.17	38.75	2.94	37.50	3.40	38.09	3.18	35.56	4.25
Valldemossa	**38.33**	**3.09**	38.02	3.20	37.87	3.26	35.94	4.07	37.87	3.26	34.66	4.72
Mean	**41.63**	**2.22**	41.18	2.34	41.36	2.29	40.48	2.59	40.98	2.40	38.63	3.20

Table 2: Comparisons for $\sigma = 2$ and 5.

$\sigma = 10$												
	BM3D		BLS-GSM		K-SVD		NL-means		DCT denoising		TV denoising	
Image	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
Computer	**36.28**	**3.91**	35.47	4.30	35.79	4.14	35.44	4.31	35.00	4.53	32.95	5.74
Dice	**43.02**	**1.80**	42.21	1.98	41.71	2.09	42.06	2.01	41.82	2.07	41.74	2.09
Flowers	**39.49**	**2.70**	39.10	2.83	39.31	2.76	38.49	3.03	38.58	3.00	37.63	3.35
Girl	**41.45**	**2.16**	41.14	2.24	40.29	2.47	40.42	2.43	40.32	2.46	39.63	2.66
Traffic	**34.54**	**4.78**	33.92	5.13	34.69	4.70	33.89	5.15	33.75	5.24	31.85	6.52
Valldemossa	**33.79**	**5.21**	33.41	5.44	33.31	5.51	32.02	6.39	33.17	5.60	30.39	7.71
Mean	**38.85**	**3.07**	38.31	3.28	38.28	3.65	37.05	3.89	37.11	3.82	35.70	4.68

$\sigma = 20$												
	BM3D		BLS-GSM		K-SVD		NL-means		DCT denoising		TV denoising	
Image	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
Computer	**32.71**	**5.90**	31.89	6.49	31.96	6.43	31.59	6.71	31.09	7.11	29.66	8.39
Dice	**39.93**	**2.57**	39.00	2.86	37.23	3.51	38.17	3.15	37.89	3.25	38.31	3.10
Flowers	**35.85**	**4.11**	35.34	4.36	35.24	4.41	34.56	4.77	34.37	4.88	34.41	4.85
Girl	**38.71**	**2.96**	38.49	3.03	36.36	3.88	36.81	3.68	36.68	3.74	37.03	3.59
Traffic	**30.83**	**7.33**	30.14	7.93	30.70	7.44	30.12	7.95	29.79	8.26	28.54	9.54
Valldemossa	**29.57**	**8.47**	26.97	11.43	29.08	8.96	28.37	9.73	28.67	9.40	26.67	11.83
Mean	**34.60**	**5.22**	33.64	6.02	33.43	5.77	33.27	6.00	33.08	6.11	32.44	6.88

Table 3: Comparisons for $\sigma = 10$ and 20.

$\sigma = 30$												
	BM3D		BLS-GSM		K-SVD		NL-means		DCT denoising		TV denoising	
Image	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
Computer	**30.67**	**7.47**	29.90	8.16	29.84	8.21	28.98	9.07	28.75	9.31	27.89	10.28
Dice	**37.88**	**3.25**	37.05	3.58	36.52	3.81	37.18	3.53	34.74	4.67	36.23	3.94
Flowers	**33.73**	**5.25**	33.19	5.59	33.54	5.36	32.66	5.94	31.95	6.44	32.49	6.05
Girl	**36.97**	**3.61**	36.91	3.64	35.38	4.34	35.54	4.26	33.65	5.30	35.28	4.39
Traffic	**28.87**	**9.18**	28.20	9.92	28.60	9.47	27.40	10.88	27.65	10.57	26.84	11.60
Valldemossa	**27.30**	**11.00**	26.97	11.43	26.80	11.66	25.55	13.46	26.34	12.29	24.79	14.69
Mean	**32.57**	**6.63**	32.04	7.05	31.78	7.14	31.22	7.86	30.51	8.10	30.59	8.50

$\sigma = 40$												
	BM3D		BLS-GSM		K-SVD		NL-means		DCT denoising		TV denoising	
Image	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR	RMSE
Computer	**29.15**	**8.89**	28.52	9.56	28.25	9.86	27.31	10.99	26.94	11.47	26.68	11.82
Dice	**36.28**	**3.91**	35.50	4.28	34.49	4.81	35.31	4.38	32.37	6.14	34.66	4.72
Flowers	**32.10**	**6.33**	31.68	6.65	31.90	6.48	30.99	7.20	30.23	7.85	31.08	7.12
Girl	**35.62**	**4.22**	35.61	4.23	33.73	5.25	34.03	5.07	31.22	7.01	33.94	5.12
Traffic	**27.50**	**10.75**	26.93	11.48	27.19	11.14	26.01	12.77	26.17	12.53	25.76	13.14
Valldemossa	**25.78**	**13.11**	25.50	13.54	25.28	13.88	24.10	15.91	24.84	14.61	23.60	16.85
Mean	**31.07**	**7.87**	30.62	8.29	30.14	8.57	29.62	9.39	28.63	9.93	29.29	9.79

Table 4: Comparisons for $\sigma = 30$ and 40.

  - 3D group processing is not only done along the third dimension (this would simply average similar patches) but also includes the spatial dimensions. This is the main difference with NL-means.

  - The second step retrieves details lost in the first step and improves edge contrast.

  - Keeping only a relatively low maximum number of similar patches during the patch-matching permits to be flexible concerning the choice of values for the parameters ($\tau^{\mathbf{hard}}$ in particular).

  - Denoising all the 2D patches included in $\mathcal{P}(P)$ during the collaborative filtering is a considerable gain for the aggregation, since it gives to each pixel a large number of good estimates.

- On the other hand the weighting of these estimates makes little contribution in the end. Visually useful in the first step, it does not modify the results of the second step in PSNR or even visually (except in rare cases with special pictorial details).

- As shown in the study, even though the algorithm is a set of very judicious choices, it is still possible to improve it, particularly the following sub-steps:

  - In spite of the fact that theoretically the weighting would be significant, practically it is not, as shown in our study on the influence of parameters. However, it is still possible to make it more efficient by using a weighting contingent on the standard deviation of the 3D group estimated in the second step.

  - The ideal Wiener filter study proves that it would be possible to get even better results by taking a more judicious first step, which is even good for the second step treatment. Any other denoising result could be seen as a basic estimate.

- Despite excellent results this method has several drawbacks worth mentioning:

  - There are still many artifacts with high noise ($\sigma > 30$).

  - This method is more complex, less flexible, and slower than basic methods such the DCT threshold denoising. It is nonetheless easy to parallelize it.

– Visually the method often flattens out micro-textured zones and therefore may give poor visual results, in particular for skin textures. A visual gain would be obtained in that case by adding back a slight noise after the denoising to recover the grain. For such artifacts see for example the staircase effects in the Barbara skin:

9 Glossary

Subscripts of variables in bold are semantic subscripts. For clarity the notation has been changed from the original article. A correspondence table confronting the original notation with the one used in the present article is provided below.

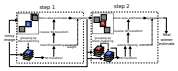
- **hard**: designates the first step where collaborative filtering is performed by hard thresholding;
- **wien**: designates the second step where collaborative filtering is performed by Wiener filtering;
- $\mathcal{P}(P)$ (resp. $\mathcal{P}^{\mathbf{basic}}(P)$): set of patches from the original image (resp. the basic estimate) similar to patch P;
- $\mathbb{P}(P)$: 3D group associated to $\mathcal{P}(P)$;
- $\mathbb{P}^{\mathbf{hard}}(P)$ (resp. $\mathbb{P}^{\mathbf{wien}}(P)$): estimate of $\mathbb{P}(P)$ due to collaborative filtering in the step 1 (resp. step 2);
- τ_{2D}: 2D transform applied on each patch of the 3D group $\mathcal{P}(P)$. $\tau_{2D}^{\mathbf{hard}}$ (resp. $\tau_{2D}^{\mathbf{wien}}$) denotes this 2D transform for the first (resp. second) step of the algorithm;
- τ_{1D}: 1D transform applied along the third dimension of the 3D group $\mathcal{P}(P)$. $\tau_{1D}^{\mathbf{hard}}$ (resp. $\tau_{1D}^{\mathbf{wien}}$) denotes this 1D transform for the first (resp. second) step of the algorithm;
- $\omega_P^{\mathbf{hard}}$ (resp. $\omega_P^{\mathbf{wien}}$): weighting obtained during the aggregation of step 1 (resp. step 2). It only depends on the reference patch P;
- P: reference patch;
- Q: any other patch;
- $u^{\mathbf{hard}}$ (resp. $u^{\mathbf{wien}}$): estimate of $u(x)$ after collaborative filtering during step 1 (resp. step 2);

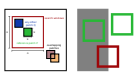
- $u^{\mathbf{basic}}$ (resp. $u^{\mathbf{final}}$): estimate of the denoised image at the end of the first step (resp. second step);
- γ': hard thresholding operator with threshold $\sigma\lambda_{2D}^{\mathbf{hard}}$;
- γ: hard thresholding operator with threshold $\sigma\lambda_{3D}^{\mathbf{hard}}$.
- $k^{\mathbf{hard}}$ (resp. $k^{\mathbf{wien}}$): size of the patches used during step 1 (resp. step 2);
- $N^{\mathbf{hard}}$ (resp. $N^{\mathbf{wien}}$): maximum number of similar patches retained during the grouping part of step 1 (resp. step 2);
- $n^{\mathbf{hard}}$ (resp. $n^{\mathbf{wien}}$): size of the search window of step 1 (resp. step 2) centered on reference patch P;
- $p^{\mathbf{hard}}$ (resp. $p^{\mathbf{wien}}$): step in both rows and columns between two reference patches of step 1 (resp. step 2);
- $\tau^{\mathbf{hard}}$ (resp. $\tau^{\mathbf{wien}}$): threshold of the maximum distance below which two patches are similar during step 1 (resp. step 2).

Our notations	Original notations
$\mathcal{P}(P)$	$Z_{S_{x_R}^{\mathbf{ht}}}$
$\mathcal{P}^{\mathbf{basic}}(P)$	$Y_{S_{x_R}^{\mathbf{wie}}}^{basic}$
$\mathbb{P}^{\mathbf{hard}}(P)$	$Y_{S_{x_R}^{\mathbf{ht}}}^{\mathbf{ht}}$
$\mathbb{P}^{\mathbf{wien}}(P)$	$Y_{S_{x_R}^{\mathbf{wie}}}^{\mathbf{wie}}$
$w_P^{\mathbf{hard}}$	$w_{x_R}^{\mathbf{ht}}$
$w_P^{\mathbf{wien}}$	$w_{x_R}^{\mathbf{wie}}$
P	Z_{x_R}
Q	Z
$u^{\mathbf{hard}}$	$y(x)$
$u^{\mathbf{basic}}$	y^{basic}
$u^{\mathbf{final}}$	$y^{\mathbf{final}}$
$k^{\mathbf{hard}}$	$N_1^{\mathbf{ht}}$
$k^{\mathbf{wien}}$	$N_1^{\mathbf{wie}}$
$N^{\mathbf{hard}}$	$N_2^{\mathbf{ht}}$
$N^{\mathbf{wien}}$	$N_2^{\mathbf{wie}}$
$n^{\mathbf{hard}}$	$N_S^{\mathbf{ht}}$
$n^{\mathbf{wien}}$	$N_S^{\mathbf{wie}}$
$p^{\mathbf{hard}}$	$N_{step}^{\mathbf{ht}}$
$p^{\mathbf{wien}}$	$N_{step}^{\mathbf{wie}}$
$\tau^{\mathbf{hard}}$	$\tau_{match}^{\mathbf{ht}}$
$\tau^{\mathbf{wien}}$	$\tau_{match}^{\mathbf{wie}}$

Correspondence with the original BM3D article

Image Credits

 by M. Lebrun and N. Limare, CC-BY, adapted from the original BM3D article [4]

 by M. Lebrun and N. Limare, CC-BY

 by H. Bry, Flickr, CC-BY-NC-SA

 by A. Buades, CC-BY

 by M. Colom, CC-BY

References

[1] 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.

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

[3] P. Chatterjee and P. Milanfar. Is denoising dead? *Image Processing, IEEE Transactions on*, 19(4):895–911, 2010. http://dx.doi.org/10.1109/TIP.2009.2037087.

[4] 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.

[5] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, et al. Bm3d image denoising with shape-adaptive principal component analysis. *Proc. of the Workshop on Signal Processing with Adaptive Sparse Structured Representations, Saint-Malo, France*, April 2009.

[6] A. Danielyan, V. Katkovnik, and K. Egiazarian. Bm3d frames and variational image deblurring. *IEEE Transactions on Image Processing*, 2012. http://dx.doi.org/10.1109/TIP.2011.2176954.

[7] Donoho. Smooth wavelet decompositions with blocky coefficient kernels. *SCHUMAKER, L. L., and WEBB, G., Eds. Recent Advances in Wavelet Analysis. Academic Press, New York*, pages 259–308, 1993.

[8] P. Getreuer. Rudin-osher-fatemi total variation denoising using split bregman. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.g-tvd.

[9] Yingkun Hou, Chunxia Zhao, Deyun Yang, and Yong Cheng. Comments on "image denoising by sparse 3-d transform-domain collaborative filtering". *IEEE Transactions on Image Processing*, 20(1), January 2011. http://dx.doi.org/10.1109/TIP.2010.2052281.

[10] Image Processing On Line. http://www.ipol.im/. ISSN:2105-1232, http://dx.doi.org/10.5201/ipol.

[11] V. Katkovnik, A. Danielyan, and K. Egiazarian. Decoupled inverse and denoising for image deblurring: variational BM3D-frame technique. In *Proceedings of IEEE International Conference on Image Processing (ICIP, 2011)*, 2011. http://dx.doi.org/10.1109/ICIP.2011.6116455.

[12] V.I.A. Katkovnik, V. Katkovnik, K. Egiazarian, and J. Astola. *Local approximation techniques in signal and image processing*, volume PM157. Society of Photo Optical, 2006. http://dx.doi.org/10.1117/3.660178.

[13] 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.

[14] A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2833–2840. IEEE, 2011. http://dx.doi.org/10.1109/CVPR.2011.5995309.

[15] 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.