Non-linear aggregation of filters to improve image denoising

Benjamin Guedj, Juliette Rengot

Non-linear aggregation of filters to improve image denoising

Benjamin Guedj[1][0000−0003−1237−7430] and Juliette Rengot[2]

[1] Inria, France and University College London, United Kingdom
benjamin.guedj@inria.fr
https://bguedj.github.io
[2] Ecole des Ponts ParisTech, France
juliette.rengot@eleves.enpc.fr

Abstract. We introduce a novel aggregation method to efficiently perform image denoising. Preliminary filters are aggregated in a non-linear fashion, using a new metric of pixel proximity based on how the pool of filters reaches a consensus. The numerical performance of the method is illustrated and we show that the aggregate significantly outperforms each of the preliminary filters.

Keywords: image denoising · statistical aggregation · ensemble methods · collaborative filtering

1 Introduction

Denoising is a fundamental question in image processing. It aims at improving the quality of an image by removing the parasitic information that randomly adds to the details of the scene. This noise may be due to image capture conditions (lack of light, blurring, wrong tuning of field depth, ...) or to the camera itself (increase of sensor temperature, data transmission error, approximations made during digitization, ...). Therefore, the challenge consists in removing the noise from the image while preserving its structure. Many methods of denoising already have been introduced in the past decades – while good performance has been achieved, denoised images still tend to be too smooth (some details are lost) and blurred (edges are less sharp). Seeking to improve the performances of these algorithms is a very active research topic.

This paper introduces a new approach for denoising images. The main idea is to use different classical denoising methods to obtain several predictions of the pixel to denoise. Then, these values are aggregated, in the best possible way in order to produce a new denoising. As each classic method has pros and cons and as it is more or less efficient according to the kind of noise or to the image structure, an asset of our method is that is makes the best out of each method's strong points. We adapt the strategy proposed by the algorithm "COBRA - COmBined Regression Alternative" [1, 9] to the specific context of image denoising. This algorithm has been implemented in the python library `pycobra`, available on https://pypi.org/project/pycobra/.

Aggregation strategies may be rephrased as collaborative filtering, since information is filtered by using a collaboration among multiple viewpoints. Collaborative filters have already been exploited in image denoising field. [7] used them to create one of the most performing denoising algorithm: the block-matching and 3D collaborative filtering (BM3D). It puts together similar patches (2D fragments of the image) into 3D data arrays (called "groups"). It then produces a 3D estimate by jointly filtering grouped image blocks. The filtered blocks are placed again in their original positions, providing several estimations for each pixel. The information is aggregated to produce the final denoised image. This method is praised to well preserve fine details. Moreover, [10] proved that the visual quality of denoised image can be increased by adapting the denoising treatment to the local structures. They proposed an algorithm, based on BM3D, that uses different non-local filtering models in edge or smooth regions. Collaborative filters have also been associated to neural network architectures, by [15], to create new denoising solutions.

When several denoising algorithms are available, finding the relevant aggregation has been addressed by several works. [13] focused on the analysis of patch-based denoising methods and shed light on their connection with statistical aggregation techniques. [5] proposed a patch-based Wiener filter which exploits patch redundancy. Their denoising approach is designed for near-optimal performance and reaches high denoising quality. Furthermore, [14] showed that usual patch-based denoising methods are less efficient on edge structures.

The paper is organised as follows. We present our aggregation method, based on the COBRA algorithm in section 2. We then provide a thorough numerical experiments section (section 3) to assess the performance of our method along with an automatic tuning procedure of preliminary filters as a byproduct.

2 The method

We now present an image denoising version of the COBRA algorithm [1, 9]. For each pixel p of the noisy image x, we may call on M different estimators $(f_1 ... f_M)$. We aggregate these estimators by doing a weighted average on the intensities :

$$f(p) = \frac{\sum_{q \in x} \omega(p,q) x(q)}{\sum_{q \in x} \omega(p,q)},$$

and we define the weights as

$$\omega(p,q) = \mathbb{1}\left(\sum_{k=1}^{M} \mathbb{1}(|f_k(p) - f_k(q)| \leq \epsilon) \geq M * \alpha \right), \tag{1}$$

where ϵ is a confidence parameter and $\alpha \in (0,1)$ a proportion parameter.

These weights mean that, to denoise a pixel p, we average the intensities of pixels q such as a proportion at least α, of the preliminary estimators $f_1, \ldots, f_M$ have the same value in p and in q, up to a confidence level ϵ.

Let us emphasize here that our procedure averages the pixels' intensities based on the weights (which involves this involved consensus metric). The intensity predicted for each pixel p of the image is $f(p)$.

This aggregation strategy is implemented in the python library *pycobra* [9]. The general scheme is presented in Figure 1, and the pseudo-code in Algorithm 1. Users can control the number of used features thanks to the parameter "*patch_size*". For each pixel p to denoise, we consider the image patch, centred on p, of size $(2*patch\_size+1)\times(2*patch\_size+1)$. In the experiments section below, $patch\_size = 1$ is a satisfying value. Thus, for each pixel, we construct a vector of nine features.

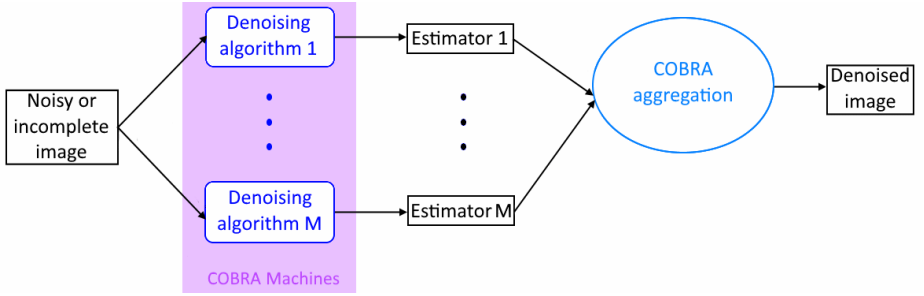


Fig. 1: General model

Algorithm 1 Image denoising with COBRA aggregation

INPUT:
im_noise = the noisy image to denoise
p = the pixel patch size to consider
M = the number of COBRA machines to use
OUTPUT:
Y = the denoised image

Xtrain ← training images with artificial noise
Ytrain ← original training images (ground truth)
cobra ← initial COBRA model
cobra ← to adjust COBRA model parameters with respect to the data (Xtrain, Ytrain)
cobra ← to load M COBRA machines
cobra ← to aggregate the predictions
Xtest ← feature extraction from im_noise in a vector of size $(nb\_pixels, (2 \cdot p + 1)^2)$
Y ← prediction of Xtest by cobra
Y ← to add im_noise values lost at the borders of the image, because of the patch processing, to Y

3 Numerical experiments

All code material (in Python) to replicate the experiments presented in this paper are available at https://github.com/rengotj/cobra_denoising.

3.1 Noise settings

We artificially add some disturbances to good quality images (i.e. without noise). We focus on five classical settings: the Gaussian noise, the salt-and-pepper noise, the Poisson noise, the speckle noise and the random suppression of patches (see Figure 2).

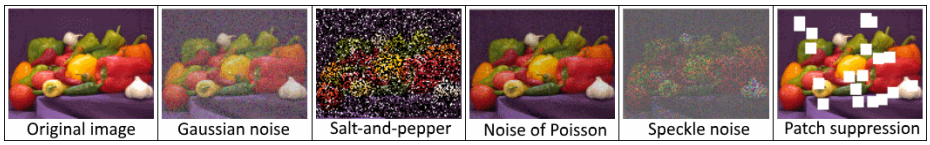


Fig. 2: The different kinds of noise used in our experiments.

3.2 Preliminary denoising algorithms

We focus on seven classical denoising methods: the Gaussian filter, the median filter, the bilateral filter, Chambolle's method [4], non-local means [2, 3], the Richardson-Lucy deconvolution [11, 12] and the inpainting method [6, 8]. This way, we intend to capture different regimes of performance (Gaussian filters are known to yield blurry edges, the median filter is known to be efficient against salt-and-pepper noise, the bilateral filter well preserves the edges, non-local means are praised to better preserve the details of the image, etc.), as the COBRA aggregation scheme allows to make the most of each preliminary filters' strong points.

3.3 Model training

We start with 25 images $(y_1 ... y_{25})$, assumed not to be noisy, that we use as "ground truth". We artificially add noise as described above, yielding 125 noisy images $(x_1 ... x_{125})$. Each noisy image is then used to create two independent copies: one goes to the data pool to train the preliminary filters, the other one to the data pool to compute the weights (1) and perform aggregation. This separation is intended to avoid over-fitting issues [as discussed in 1].

3.4 Parameters optimisation

The meta-parameters for COBRA are α (how many preliminary filters must agree to retain the pixel) and ϵ (the confidence level with which we declare two pixels identities similar). For example, choosing $\alpha = 1$ and $\epsilon = 0.1$ means that we impose that all the machines must agree on pixels whose predicted intensities are at most different by a 0.1 margin.

The python library `pycobra` ships with a dedicated class to derive the optimal values using cross-validation [9]. Optimal values are $\alpha = 4/7$ and $\epsilon = 0.2$ in our setting.

3.5 Assessing the performance

We evaluate the quality of the denoised image I_d with respect to the original image I_o with two different metrics.

- Root Mean Square Error (RMSE - the smaller the better) given by

$$\sqrt{\Sigma_{x=1}^{N}\Sigma_{y=1}^{M}\frac{(I_d(x,y)-I_o(x,y))^2}{N\times M}}.$$

- Peak Signal to Noise Ratio (PSNR - the larger the better) given by

$$10\cdot\log_{10}\left(\frac{d^2}{\mathrm{RMSE}^2}\right)$$

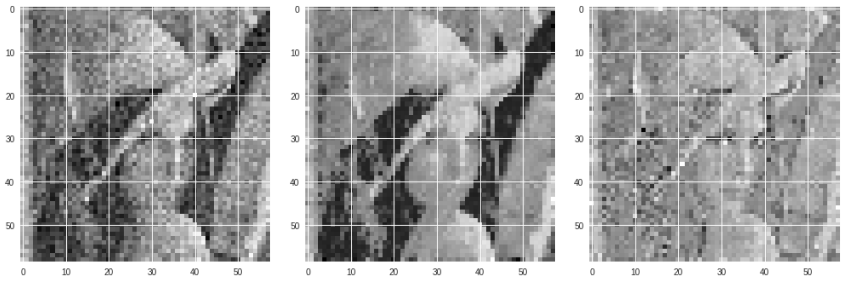
  with d the signal dynamic range (possible values for a pixel intensity).

3.6 Results

In all tables, experiments have been repeated 100 times to compute statistics. The green line (respectively, red) identifies the best (respectively, worst) performance. The first image is noisy, the second is what COBRA outputs, and the third is the difference between the ideal image (with no noise) and the COBRA denoised image.

Results – Gaussian noise We add to the reference image "lena" a Gaussian noise of mean $\mu = 0.5$ and of variance $\sigma = 0.1$ (Figure 3). Unsurprisingly, the best filter is the Gaussian filter, and the performance of the COBRA aggregate are tailing when the noise level is unknown. When the noise level is known, COBRA outperforms all preliminary filters.

Results – salt-and-pepper noise The proportion of white to black pixels is set to $sp\_ratio = 0.2$ and such that the proportion of pixels to replace is $sp\_amount = 0.3$. As for the Gaussian noise setting, COBRA tails the champion (bilateral filter) when the noise level is unknown and outperforms all filters when it is known (Figure 4).



(a) Noisy image (b) COBRA (c) Diff. ideal-COBRA

Denoising method		*PSNR*		*RMSE*	
		Average	*std*	*Average*	*std*
Cobra	**unknown noise**	**67,9970**	**0,8916**	**0,1020**	**0,0103**
	known noise	**70,3529**	**0,5325**	**0,0775**	**0,0048**
Bilateral filter		68,9815	0,6338	0,0909	0,0068
Non-local means		66,5663	0,5885	0,1200	0,0082
Gaussian filter		68,1852	0,4419	0,0995	0,0052
Median filter		64,1589	0,2045	0,1580	0,0037
TV-Chambolle		67,4698	0,5353	0,1081	0,0068
Richardson-Lucy		62,3868	0,1178	0,1937	0,0026
Inpainting		68,1826	0,4408	0,0995	0,0051

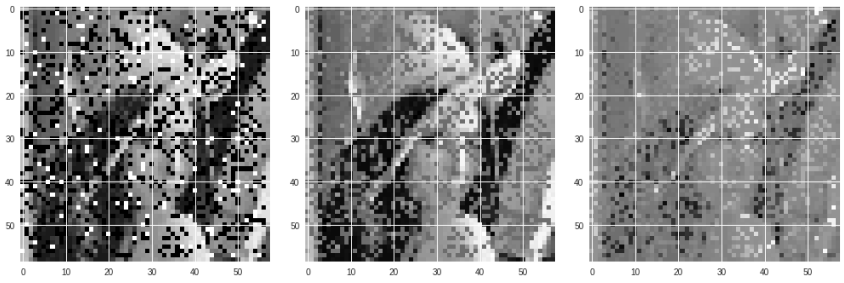
Fig. 3: Results – Gaussian noise.

Results – Poisson noise COBRA outperforms all preliminary filters, see Figure 5.

Results – speckle noise When confronted with a speckle noise (Figure 6), COBRA outperforms all preliminary filters. Note that this is a difficult task and most filters have a hard time denoising the image. The message of aggregation is that even in adversarial situations, the aggregate (strictly) improves on the preliminary performance.

Results – random patches suppression We randomly suppress 20 patches of size (4×4) pixels from the original image. These pixels become white (Figure 7). Unsurprisingly, the best filter is the inpainting method – as a matter of fact this is the only filter which succeeds in denoising the image.

Results – images containing several kinds of noise. So far, COBRA matches or outperforms the performance of the best filter for each kind of noise (to the notable exception of missing patches, where inpainting methods are superior). Finally, as the type of noise is usually unknown and even hard to infer from images, we are interested in putting all filters and COBRA to test when facing multiple types of noise levels. We apply a Gaussian noise in the upper left-hand



(a) Noisy image (b) COBRA (c) Diff. ideal-COBRA

Denoising method	*PSNR*		*RMSE*	
	Average	*std*	*Average*	*std*
Cobra	**67,8723**	**0,3787**	**0,1031**	**0,0045**
Bilateral filter	61,1750	0,1784	0,2227	0,0045
Non-local means	59,5938	0,1370	0,2672	0,0042
Gaussian filter	59,4825	0,1292	0,2706	0,0040
Median filter	64,0278	0,1593	0,1604	0,0029
TV-Chambolle	63,0854	0,1851	0,1787	0,0038
Richardson-Lucy	60,7194	0,1099	0,2348	0,0029
Inpainting	60,2957	0,1248	0,2464	0,0035

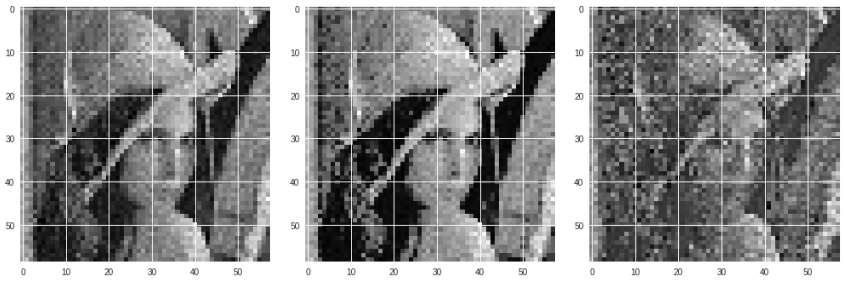
Fig. 4: Result – salt-and-pepper noise.

corner, a salt-and-pepper noise in the upper right-hand corner a noise of Poisson in the lower left-hand corner and a speckle noise in the lower right-hand corner. In addition, we randomly suppress small patchs on the whole image.

In this now much more adversarial situation, none of the preliminary filters can achieve proper denoising. This is the kind of setting where aggregation is the most interesting, as it will make the best of each filter's abilities. As a matter of fact, COBRA significantly outperforms all preliminary filters (see Figure 8).

3.7 Automatic tuning of filters

Clearly, internal parameters for the classical preliminary filters may have a crucial impact. For example, the median filter is particularly well suited for salt-and-pepper noise, although the filter size has to be chosen carefully as it should grow with the noise level (which is unknown in practice). A nice byproduct of our aggregated scheme is that we can also perform automatic and adaptive tuning of those parameters, by feeding COBRA with as many machines as possible values for these parameters. Let us illustrate this on a simple example: we train our model with only one classical method but with several values of the parameter to tune. For example, we can define three machines applying median filters with different filter sizes : 3, 5 or 10. Whatever the noise level our approach achieves



(a) Noisy image (b) COBRA (c) Diff. ideal-COBRA

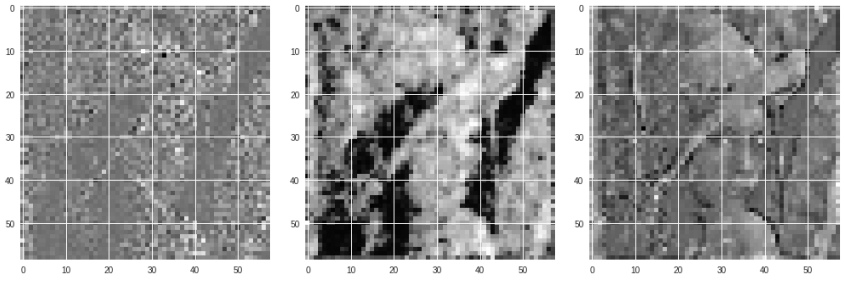
Denoising method	*PSNR*		*RMSE*	
	Average	*std*	*Average*	*std*
Cobra	**69,5769**	**1,2691**	**0,0855**	**0,0126**
Bilateral filter	69,2636	1,3314	0,0888	0,0135
Non-local means	67,3015	1,1171	0,1109	0,0141
Gaussian filter	69,3744	1,1854	0,0874	0,0119
Median filter	64,0544	0,3874	0,1600	0,0071
TV-Chambolle	67,8956	1,0731	0,1035	0,0127
Richardson-Lucy	62,3801	0,1567	0,1939	0,0035
Inpainting	69,3663	1,1887	0,0875	0,0119

Fig. 5: Results – Poisson noise.

the best performance (Figure 9). This casts our approach onto the adaptive setting where we can efficiently denoise an image regardless of its (unknown) noise level.

4 Conclusion

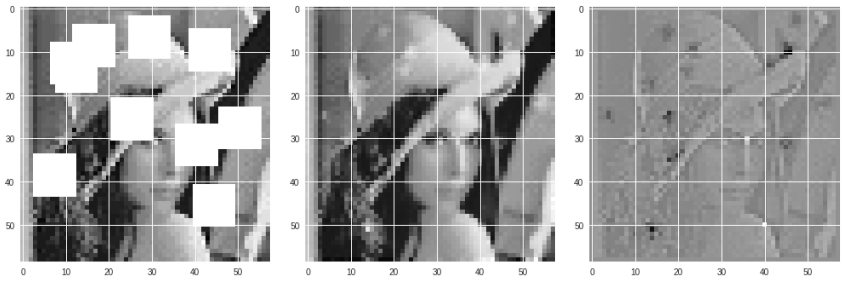
We have presented a generic aggregated denoising method which improves on the performance of preliminary filters, makes the most of their abilities (e.g., adaptation to a particular kind of noise) and automatically adapts to the unknown noise level. Numerical experiment suggests that our method achieves the best performance when dealing with several types of noise. Let us conclude by stressing that our approach is generic in the sense that *any* preliminary filters could be aggregated, regardless of their nature and specific abilities.



(a) Noisy image (b) COBRA (c) Diff. ideal-COBRA

Denoising method	*PSNR*		*RMSE*	
	Average	*std*	*Average*	*std*
Cobra	**65,3273**	**0,6790**	**0,1381**	**0,0147**
Bilateral filter	61,6701	0,4949	0,2107	0,0130
Non-local means	61,1187	0,4900	0,2245	0,0136
Gaussian filter	61,3401	0,4467	0,2188	0,0120
Median filter	61,4221	0,5537	0,2169	0,0150
TV-Chambolle	61,7254	0,4899	0,2094	0,0127
Richardson-Lucy	60,1506	0,2342	0,2507	0,0070
Inpainting	61,3367	0,4463	0,2189	0,0120

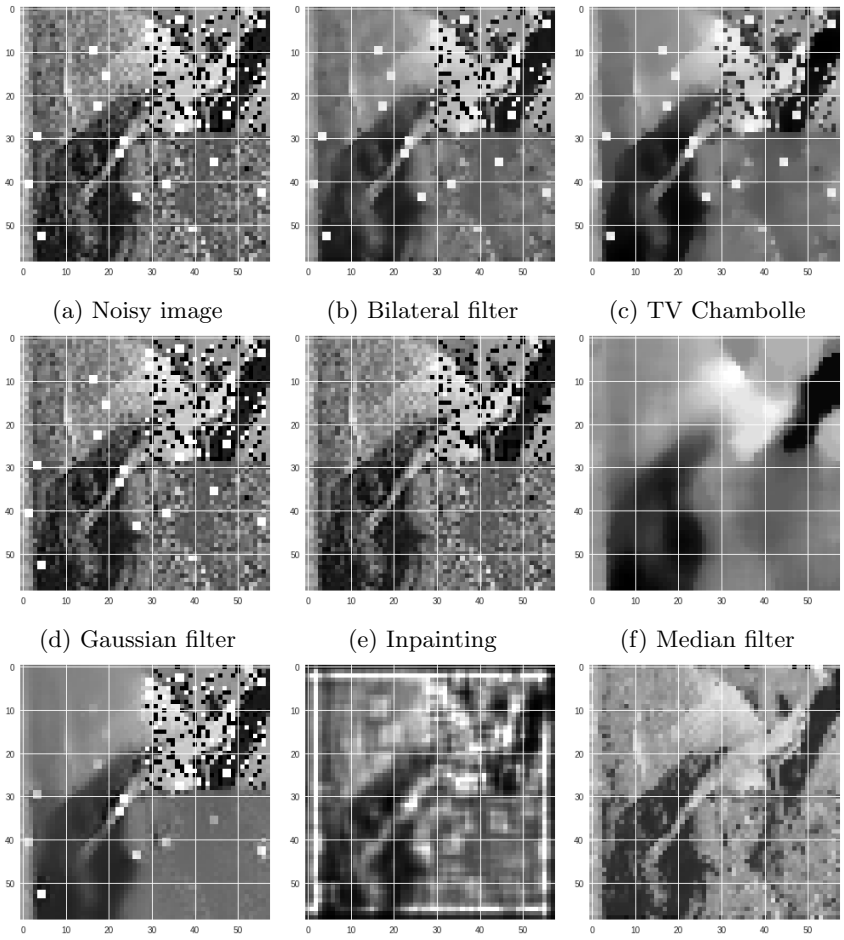
Fig. 6: Results – speckle noise.



(a) Noisy image (b) COBRA (c) Diff. ideal-COBRA

Denoising method	*PSNR*		*RMSE*	
	Average	*std*	*Average*	*std*
Cobra	**73,6217**	**0,5816**	**0,0533**	**0,0035**
Bilateral filter	68,7393	0,6827	0,0935	0,0072
Non-local means	68,4178	0,6507	0,0970	0,0072
Gaussian filter	69,1059	0,7130	0,0896	0,0072
Median filter	64,9597	0,0910	0,1440	0,0015
TV-Chambolle	68,9456	0,4978	0,0911	0,0051
Richardson-Lucy	62,2937	0,0864	0,1958	0,0019
Inpainting	84,6220	1,0184	0,0150	0,0017

Fig. 7: Results – random suppression of patches.



(a) Noisy image (b) Bilateral filter (c) TV Chambolle

(d) Gaussian filter (e) Inpainting (f) Median filter

(g) Non-local means (h) Richardson-Lucy deconvolution (i) COBRA

Denoising method		*PSNR*		*RMSE*	
		Average	*std*	*Average*	*std*
Cobra	**unknown noise**	**65,1860**	**0,3128**	**0,1404**	**0,0050**
	known noise	**66,1642**	**0,5372**	**0,1256**	**0,0080**
Bilateral filter		62,0859	0,2470	0,2006	0,0056
Non-local means		61,6465	0,1948	0,2110	0,0047
Gaussian filter		61,5469	0,1944	0,2134	0,0047
Median filter		63,4152	0,3016	0,1722	0,0060
TV-Chambolle		63,4025	0,2707	0,1724	0,0053
Richardson-Lucy		60,9305	0,1385	0,2291	0,0036
Inpainting		62,5344	0,2553	0,1905	0,0055

Fig. 8: Denoising an image afflicted with multiple noises types.

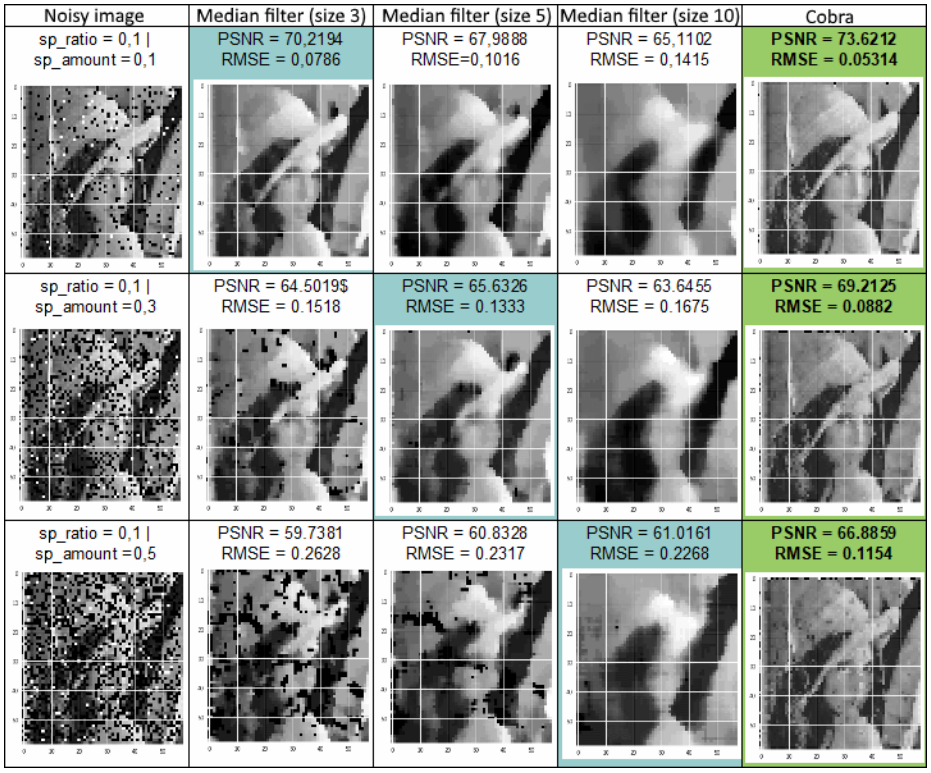


Fig. 9: Automatic tuning of the median filter using COBRA.

Bibliography

[1] Biau, G., Fischer, A., Guedj, B., Malley, J.D.: Cobra: A combined regression strategy. Journal of Multivariate Analysis **146**, 18 – 28 (2016). https://doi.org/https://doi.org/10.1016/j.jmva.2015.04.007 1, 2, 4

[2] Buades, A., Coll, B., Morel, J..: A non-local algorithm for image denoising. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 2, pp. 60–65 vol. 2 (2005) 4

[3] Buades, A., Coll, B., Morel, J.M.: Non-local means denoising. Image Processing On Line **1**, 208–212 (2011) 4

[4] Chambolle, A.: Total variation minimization and a class of binary mrf models. Energy Minimization Methods in Computer Vision and Pattern Recognition **3757**, 132–152 (2005) 4

[5] Chatterjee, P., Milanfar, P.: Patch-based near-optimal image denoising. IEEE Transactions on Image Processing **21**(4), 1635–1649 (2012) 2

[6] Chuiab, C., Mhaskar, H.: Mra contextual-recovery extension of smooth functions on manifolds. Applied and Computational Harmonic Analysis **28**, 104–113 (01 2010) 4

[7] Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE Transactions on image processing **16**(8), 2080–2095 (2007) 2

[8] Damelin, S., Hoang, N.: On surface completion and image inpainting by biharmonic functions: Numerical aspects. International Journal of Mathematics and Mathematical Sciences **2018**, 8 (01 2018) 4

[9] Guedj, B., Srinivasa Desikan, B.: Pycobra: A python toolbox for ensemble learning and visualisation. Journal of Machine Learning Research **18**(190), 1–5 (2018), http://jmlr.org/papers/v18/17-228.html 1, 2, 3, 5

[10] Liu, J., Liu, R., Chen, J., Yang, Y., Ma, D.: Collaborative filtering denoising algorithm based on the nonlocal centralized sparse representation model. In: 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI) (2017) 2

[11] Lucy, L.: An iterative technique for the rectification of observed distributions. Astronomical Journal **19**, 745 (06 1974) 4

[12] Richardson, W.H.: Bayesian-based iterative method of image restoration. Journal of the Optical Society of America **62**, 55–59 (1972) 4

[13] Salmon, J., Le Pennec, E.: Nl-means and aggregation procedures. In: 2009 16th IEEE International Conference on Image Processing (ICIP). pp. 2977–2980 (Nov 2009). https://doi.org/10.1109/ICIP.2009.5414512 2

[14] Salmon, J.: Agrégation d'estimateurs et méthodes à patch pour le débruitage d'images numériques. Ph.D. thesis, Université Paris-Diderot-Paris VII (2010) 2

[15] Strub, F., Mary, J.: Collaborative filtering with stacked denoising autoencoders and sparse inputs. In: NIPS workshop on machine learning for eCommerce (2015) 2