

Non-linear aggregation of filters to improve image denoising

Benjamin Guedj, Juliette Rengot

▶ **To cite this version:**

Benjamin Guedj, Juliette Rengot. Non-linear aggregation of filters to improve image denoising. Computing Conference 2020, Jul 2020, London, United Kingdom. hal-02086856v2

HAL Id: hal-02086856
https://inria.hal.science/hal-02086856v2

Submitted on 1 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Non-linear aggregation of filters to improve image denoising

Benjamin Guedj[1] 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. We provide a theoretical bound to support our aggregation scheme, its numerical performance 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.

The present paper introduces a new approach for denoising images, by bringing to the computer vision community ideas developed in the statistical learning literature. The main idea is to combine different classical denoising methods to obtain several predictions of the pixel to denoise. As each classic method has pros and cons and 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, pointing out the "wisdom of the crowd". We adapt the strategy proposed by the algorithm "COBRA - COmBined Regression Alternative" [2, 10]

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. [8] 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, [13] proved that the visual quality of denoised images 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 [18], to create new denoising solutions.

When several denoising algorithms are available, finding the relevant aggregation has been addressed by several works. [16] focused on the analysis of patch-based denoising methods and shed light on their connection with statistical aggregation techniques. [6] 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, [17] showed that usual patch-based denoising methods are less efficient on edge structures.

The COBRA algorithm differs from the aforecited techniques, as it combines preliminary filters in a non-linear way. COBRA has been introduced and analysed by [2].

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 [2, 10]. 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 :

$$f(p) = \frac{\sum_{q \in x} \omega(p,q) x(q)}{\sum_{q \in x} \omega(p,q)} \ , \tag{1}$$

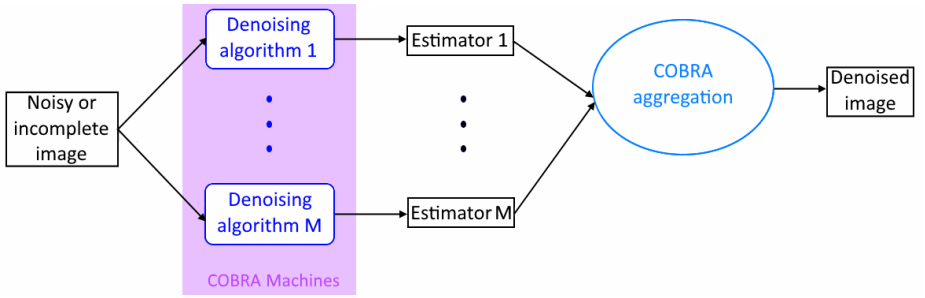


Fig. 1: General model

and we define the weights as

$$\omega(p,q) = \mathbb{1}\left(\sum_{k=1}^{M} \mathbb{1}(|f_k(p) - f_k(q)| \leq \epsilon) \geq M\alpha\right), \tag{2}$$

where ϵ is a confidence parameter and $\alpha \in (0,1)$ a proportion parameter. Note that while f is linear with respect to the intensity x, it is non-linear with respect to each of the preliminary estimators $f_1, \ldots, f_M$.

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 involve this consensus metric). The intensity predicted for each pixel p of the image is $f(p)$ and the COBRA-denoised image is the collection of pixels $\{f(p), p \in x\}$.

This aggregation strategy is implemented in the python library *pycobra* [10]. 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 $(2 \cdot patch_size + 1) \times (2 \cdot patch_size + 1)$. In the experiments section, $patch_size = 2$ is usually a satisfying value. Thus, for each pixel, we construct a vector of nine features.

The COBRA aggregation method has been introduced by [2] in a generic statistical learning framework, and is supported by a sharp oracle bound. For the sake of completeness, we reproduce here one of the key theorems.

Theorem 1 (adapted from Theorem 2.1 in [2]). *Assume we have M preliminary denoising methods. Let $|x|$ denote the total number of pixels in image x. Let $\epsilon \propto |x|^{-\frac{1}{M+2}}$. Let $f^\star$ denote the perfectly denoised image and $\widehat{f}$ denote the COBRA aggregate defined in (1), then we have*

$$\mathbb{E}\left[\widehat{f}(p) - f^\star(p)\right]^2 \leq \min_{m=1,\ldots,M} \mathbb{E}\left[f_m(p) - f^\star(p)\right]^2 + C|x|^{-\frac{2}{M+2}}, \tag{3}$$

Algorithm 1 Image denoising with COBRA aggregation

INPUT:
im_noise = the noisy image to denoise
p_{size} = 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 $(nb_pixels, (2 \cdot p_{\text{size}}+1)^2)$
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

where C is a constant and the expectations are taken with respect to the pixels.

What Theorem 1 tells us is that on average on all the image's pixels, the quadratic error between the COBRA denoised image and the perfectly denoised image is upper bounded by the best (*i.e.*, minimal) same error from the preliminary pool of M denoising methods, up to a term which decays to zero as the number of pixels to the $-1/M$. As highlighted in the numerical experiments reported in the next section, M is of the order of 5-10 machines and this remainder term is therefore expected to be small in most useful cases for COBRA. Note that in (3), the leading constant (in front of the minimum) is 1: the oracle inequality is said to be *sharp*. Note also that contrary to more classical aggregation or model selection methods, COBRA mactches or outperforms the best preliminary filter's performance, even though it does not need to identify this champion filter. As a matter of fact, COBRA is adaptive to the pool of filters as the champion is not needed in (1). More comments on this result, and proofs are presented in [2].

3 Numerical experiments

This section illustrates the behaviour of COBRA. All code material (in Python) to replicate the experiments presented in this paper are available at `<hidden url to preserve anonymity>`.

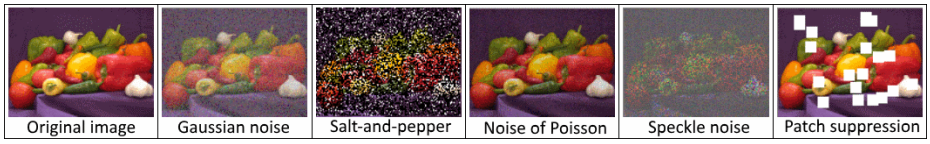


Fig. 2: The different kinds of noise used in our experiments.

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 (summarised in Figure 2).

3.2 Preliminary denoising algorithms

We focus on ten classical denoising methods: the Gaussian filter, the median filter, the bilateral filter, Chambolle's method [5], non-local means [3, 4], the Richardson-Lucy deconvolution [14, 15], the Lee filter [12], K-SVD [1], BM3D [8] and the inpainting method [7, 9]. 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, Lee filers are designed to address Synthetic Aperture Radar (SAR) image despeckling problems, K-SVD and BM3D are state-of-the-art approaches, inpainting is designed to reconstruct lost part, etc.), as the COBRA aggregation scheme is designed to blend together machines with various levels of performance and adaptively use the best local method.

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 $(x_1...x_{125})$. Then two independent copies of each noisy image are created by adding a normal noise: one goes to the data pool to train the preliminary filters, the other one to the data pool to compute the weights defined in (2) and perform aggregation. This separation is intended to avoid over-fitting issues [as discussed in 2]. The whole dataset creation process is illustrated in Figure 3.

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

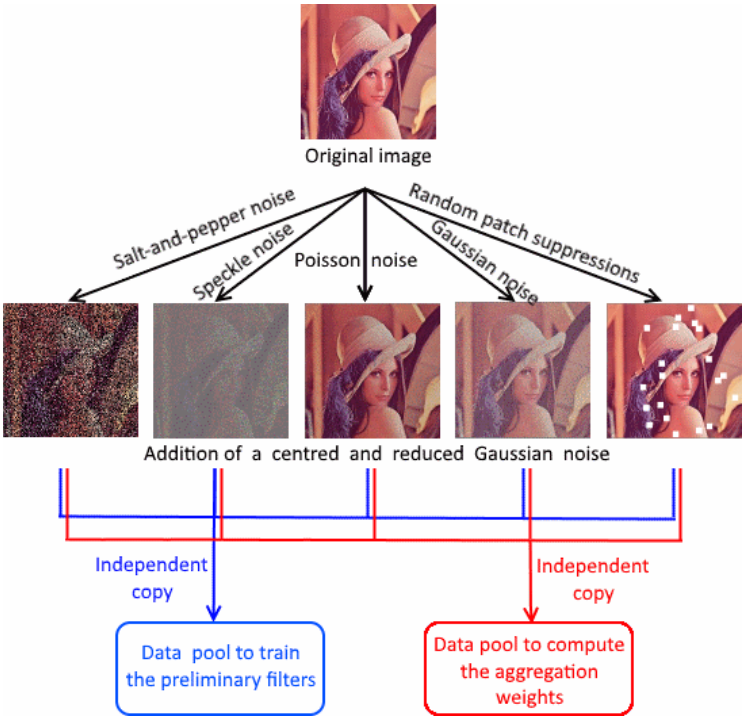


Fig. 3: Data set construction.

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 [10]. 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 (whose mean is denoted μ_d and standard deviation σ_d) with respect to the original image I_o (whose mean is denoted μ_o and standard deviation σ_o) with four different metrics.

– Mean Absolute Error (MAE - the closer to zero the better) given by

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

– Root Mean Square Error (RMSE - the closer to zero 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 (maximal possible value for a pixel intensity).
– Universal image Quality Index (UQI - the closer to one the better) given by

$$\underbrace{\frac{cov(I_o, I_d)}{\sigma_o \cdot \sigma_d}}_{(i)} \cdot \underbrace{\frac{2 \cdot \mu_o \cdot \mu_d}{\mu_o^2 + \mu_d^2}}_{(ii)} \cdot \underbrace{\frac{2 \cdot \sigma_o \cdot \sigma_d}{\sigma_o^2 + \sigma_d^2}}_{(iii)}$$

where term (i) is the correlation, (ii) is the mean luminance similarity, and (iii) is the contrast similarity [19, Eq. 2].

3.6 Results

Our experiments run on the gray-scale "lena" reference image (range 0 - 255). In all tables, experiments have been repeated 100 times to compute descriptive statistics. The green line (respectively, red) identifies the best (respectively, worst) performance. The yellow line identifies the best performance among the preliminary denoising algorithms if COBRA achieves the best 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 (Figure 4). We add to the reference image "lena" a Gaussian noise of mean $\mu = 127.5$ and of standard deviation $\sigma = 25.5$. Unsurprisingly, the best filter is the Gaussian filter, and the performance of the COBRA aggregate is tailing when the noise level is unknown. When the noise level is known, COBRA outperforms all preliminary filters. Note that the bilateral filter gives better results than non-local means. This is not surprising: [11] reaches the same conclusion for high noise levels.

Results – salt and pepper noise (Figure 5). The proportion of white to black pixels is set to $sp_ratio = 0.2$ and such that the proportion of pixels to replace is $sp_amount = 0.1$. Even if the noise level is unknown, COBRA outperforms all filters, even the champion BM3D.

Results – Poisson noise (Figure 6). COBRA outperforms all preliminary filters.

Results – speckle noise (Figure 7). When confronted with a speckle noise, 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 performance of the preliminary pool of methods.



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

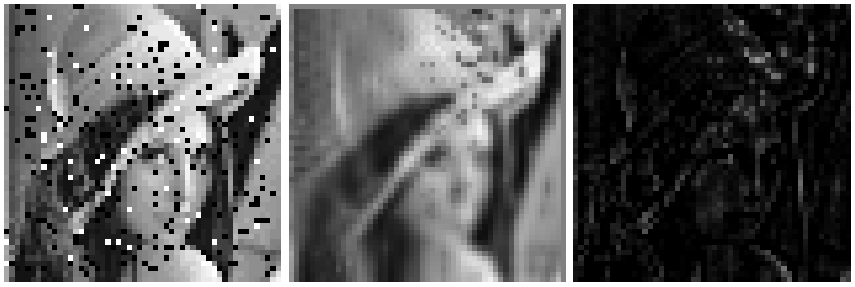
Denoising method		*MAE*		*RMSE*		*PSNR*		*UQI*	
		Average	*std*	*Average*	*std*	*Average*	*std*	*Average*	*std*
COBRA	unknown noise	0.0956	0.0132	0.1173	0.0162	67.1303	0,8916	0.9515	0.0186
	known noise	0,0637	0,0119	0,07817	0,0047	70,9046	0,5367	0,9773	0,0179
Bilateral filter		0,0793	0,0052	0,09764	0,0063	67,3944	1,1590	0,9589	0,0078
Non-local means		0,1018	0,0067	0,1259	0,0080	64,9678	1,5049	0,9393	0,0084
Gaussian filter		0,0833	0,0041	0,1033	0,0049	67,8608	0,4051	0,9587	0,0073
Median filter		0,1299	0,0031	0,1646	0,0036	61,1062	0,4352	0,9095	0,0059
TV-Chambolle		0,0936	0,0054	0,1152	0,0064	64,3309	0,9223	0,9468	0,0076
Richardson-Lucy		0,1276	0,0027	0,1604	0,0034	61,7979	0,1426	0,9271	0,0058
Inpainting		0.0932	0,0033	0,1180	0,0049	65.6594	0,4985	0.9382	0,0073
K-SVD		0,07824	0,0053	0,0966	0,0064	67,4678	1,1359	0,9600	0,0078
BM3D		0,0883	0,0051	0,1138	0,0055	66,1415	0,6796	0,9481	0,0071
Lee		0,1288	0,0033	0,1564	0,0042	62,1361	0,6279	0,9074	0,0062

Fig. 4: Results – Gaussian noise.

Results – random patches suppression (Figure 8). We randomly suppress 20 patches of size (4×4) pixels from the original image. These pixels become white. Unsurprisingly, the best filter is the inpainting method – as a matter of fact this is the only filter which succeeds in denoising the image, as it is quite a specific noise.

Results – images containing several kinds of noise (Figure 9). On all previous examples, 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 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 (see Figure 9a).

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.



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

Denoising method	*MAE*		*RMSE*		*PSNR*		*UQI*	
	Average	*std*	*Average*	*std*	*Average*	*std*	*Average*	*std*
COBRA	0,0678	0,0033	0,0952	0,0051	69,0878	0,0124	0,972	0,0029
Bilateral filter	0,0651	0,002	0,1434	0,0053	64,9947	0,0052	0,9482	0,0044
Non-local means	0,0584	0,002	0,1463	0,0063	64,8317	0,3717	0,9477	0,0047
Gaussian filter	0,0465	0,0018	0,1682	0,0049	63,6197	0,2552	0,9325	0,0048
Median filter	0,1	0,0007	0,1496	0,0012	63,8613	0,0784	0,9286	0,0020
TV-Chambolle	0,0716	0,0021	0,1150	0,0037	66,7517	0,2854	0,9606	0,0028
Richardson-Lucy	0,158	0,002	0,2125	0,0020	61,5832	0,0025	0,4876	0,0179
Inpainting	0,0387	0,0017	0,1500	0,0049	64,6004	0,2521	0,9512	0,0032
K-SVD	0,0733	0,0024	0,1278	0,0039	66,0601	0,2744	0,9536	0,0037
BM3D	0,0674	0,0034	0,0969	0,0057	68,8816	0,1067	0,9707	0,0108
Lee	0,1137	0,0014	0,1520	0,0028	63,7837	0,1454	0,9253	0,0025

Fig. 5: Result – salt-and-pepper noise.

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 the best performance (Figure 10). 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—called COBRA—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

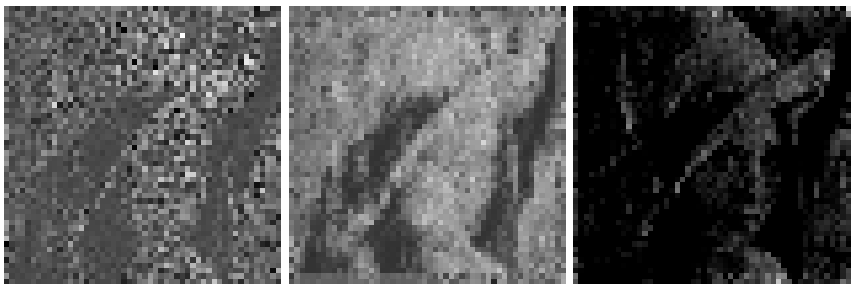


(a) Noisy image (b) COBRA (c) Diff. ideal-COBRA

Denoising method	*MAE*		*RMSE*		*PSNR*		*UQI*	
	Average	*std*	*Average*	*std*	*Average*	*std*	*Average*	*std*
COBRA	0,0818	0,0049	0,0981	0,0053	74,0778	0,1698	0,9630	0,0046
Bilateral filter	0,0791	0,0129	0,0950	0,0143	73,1181	1,1456	0,9643	0,0129
Non-local means	0,0951	0,0131	0,1179	0,0149	68,9004	1,1057	0,9465	0,0157
Gaussian filter	0,0749	0,0122	0,0940	0,0133	68,7468	1,1781	0,9674	0,0121
Median filter	0,1315	0,0087	0,1681	0,0079	63,7143	0,1838	0,9028	0,0128
TV-Chambolle	0,0915	0,0122	0,1109	0,0139	70,5550	0,4018	0,9535	0,0138
Richardson-Lucy	0,1527	0,0056	0,2045	0,0336	61,9173	0,1510	0,9107	0,0111
Inpainting	0,075	0,0123	0,0941	0,0133	69,8320	0,8607	0,9674	0,0122
K-SVD	0,0817	0,0128	0,0984	0,0143	73,2155	1,0839	0,9626	0,0131
BM3D	0,0801	0,0023	0,1006	0,0123	69,8647	0,1403	0,9632	0,0128
Lee	0,1271	0,009	0,1518	0,0104	66,2958	0,3745	0,9234	0,0109

Fig. 6: Results – Poisson noise.

adapts to the unknown noise level. COBRA is supported by a sharp oracle inequality demonstrating its optimality, up to an explicit remainder term which quickly goes to zero. 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	*MAE*		*RMSE*		*PSNR*		*UQI*	
	Average	*std*	*Average*	*std*	*Average*	*std*	*Average*	*std*
COBRA	0,0677	0,0092	0,0870	0,0142	71,1629	0,6794	0,9807	0,0201
Bilateral filter	0,1833	0,0084	0,2246	0,0101	61,1627	0,3829	0,83784	0,0162
Non-local means	0,2032	0,0093	0,2401	0,0106	57,8368	1,7343	0,8082	0,0197
Gaussian filter	0,1876	0,0066	0,2315	0,0091	60,8471	0,3353	0,8335	0,0155
Median filter	0,19521	0,0097	0,2317	0,0116	56,7125	0,6146	0,8214	0,0204
TV-Chambolle	0,1893	0,0093	0,2243	0,0103	60,2860	1,0094	0,8314	0,0017
Richardson-Lucy	0,2191	0,005	0,2689	0,0051	59,5419	0,1637	0,8292	0,0146
Inpainting	0,1877	0,0066	0,2316	0,0092	60,9999	0,2251	0,8333	0,0156
K-SVD	0,1834	0,0091	0,2201	0,0106	61,5374	0,4267	0,8395	0,0164
BM3D	0,187	0,0135	0,2231	0,0131	56,6933	0,2485	0,8352	0,0037
Lee	0,1926	0,0081	0,2321	0,0093	60,6255	0,6187	0,8244	0,0162

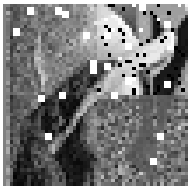
Fig. 7: Results – speckle noise.



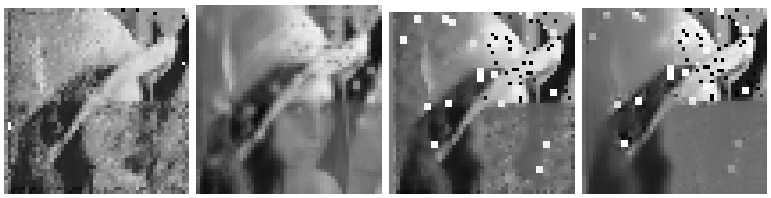
(a) Noisy image (b) COBRA (c) Diff. ideal-COBRA

Denoising method	*MAE*		*RMSE*		*PSNR*		*UQI*	
	Average	*std*	*Average*	*std*	*Average*	*std*	*Average*	*std*
COBRA	0,0283	0.0052	0,0474	0.0052	74,6216	0.8588	0,9902	0.0012
Bilateral filter	0,046	0,0015	0,1011	0,0073	68,0083	0,6135	0,9641	0,0080
Non-local means	0,0525	0,0016	0,1034	0,0081	67,8706	0,6753	0,9622	0,0079
Gaussian filter	0,0154	0,0014	0,0988	0,0075	68,2626	0,6599	0,9677	0,0080
Median filter	0,1009	0,0009	0,1508	0,0015	63,3640	0,1787	0,9311	0,0017
TV-Chambolle	0,0551	0,0023	0,0954	0,0058	67,1800	0,6174	0,9662	0,0063
Richardson-Lucy	0,1499	0,0017	0,2096	0,0020	61,7035	0,0838	0,9229	0,0043
Inpainting	0.0290	0,0003	0.0452	0,0028	75.0200	1,1451	0.9929	0,0003
K-SVD	0,0463	0,0021	0,0977	0,0073	68,3252	0,6376	0,9663	0,0076
BM3D	0,0604	0,0139	0,0915	0,0109	68,8396	0,0765	0,9712	0,0052
Lee	0,1002	0,0011	0,1279	0,0031	63,6801	0,2149	0,9345	0,0037

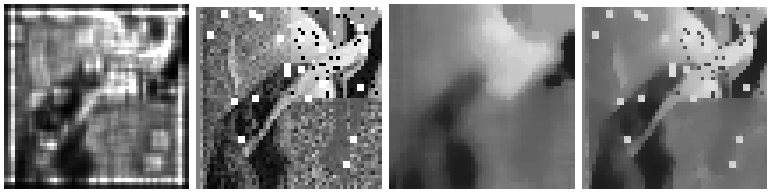
Fig. 8: Results – random suppression of patches.



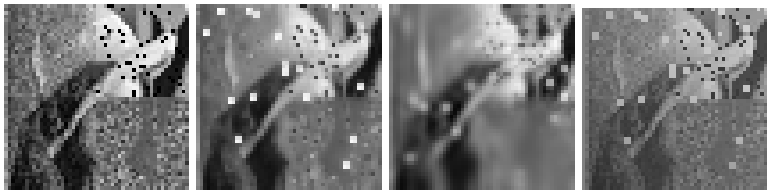
(a) Noisy image



(b) COBRA (Unknown noise) (c) COBRA (Known noise) (d) Bilateral filter (e) Non-local means



(f) Richardson-Lucy deconvolution (g) Gaussian filter (h) Median filter (i) TV Chambolle



(j) Inpainting (k) K-SVD (l) BM3D (m) Lee filter

Denoising method		*MAE*		*RMSE*		*PSNR*		*UQI*	
		Average	*std*	*Average*	*std*	*Average*	*std*	*Average*	*std*
COBRA	unknown noise	0,0802	0,0041	0,1155	0,0050	66,8771	0.2741	0,9557	0,0084
	known noise	0,0801	0,0039	0,1083	0,0048	67,7980	0,2699	0,9601	0,0085
Bilateral filter		0,1074	0,0058	0,1786	0,0082	63,0962	0,4001	0,9139	0,0095
Non-local means		0,1154	0,0062	0,1844	0,0091	62,8274	0,4302	0,9038	0,0100
Gaussian filter		0,1051	0,0048	0,1873	0,0076	62,6849	0,3536	0,9109	0,0091
Median filter		0,1334	0,0049	0,1769	0,0058	63,1068	0,2333	0,8991	0,0073
TV-Chambolle		0,1124	0,0056	0,1616	0,0070	63,8476	0,3609	0,9190	0,0080
Richardson-Lucy		0,1741	0,0033	0,2294	0,0036	60,9190	0,1415	0,8940	0,0065
Inpainting		0,0898	0,0045	0,1576	0,0069	64,2497	0,3682	0,9398	0,0061
K-SVD		0,1096	0,0055	0,1623	0,0070	63,9382	0,3702	0,9208	0,0083
BM3D		0,1024	0,0029	0,1392	0,0105	65,2288	0,0399	0,9378	0,0024
Lee		0,1369	0,0036	0,1778	0,0048	62,9373	0,2711	0,9001	0,0046

Fig. 9: Denoising an image afflicted with multiple noises types.

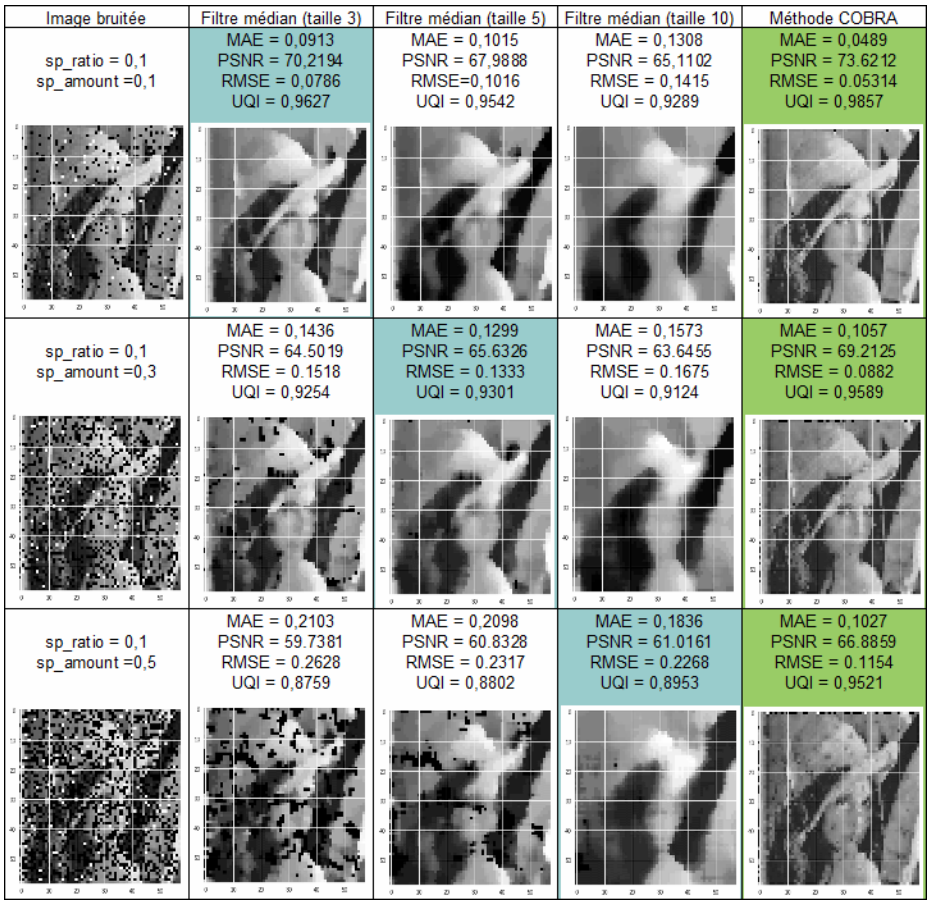


Fig. 10: Automatic tuning of the median filter using COBRA.

Bibliography

[1] Aharon, M., Elad, M., Bruckstein, A., et al.: K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on signal processing **54**(11) (2006) 4311 5

[2] Biau, G., Fischer, A., Guedj, B., Malley, J.D.: Cobra: A combined regression strategy. Journal of Multivariate Analysis **146** (2016) 18 – 28 1, 2, 3, 4, 5

[3] 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). Volume 2. (2005) 60–65 vol. 2 5

[4] Buades, A., Coll, B., Morel, J.M.: Non-local means denoising. Image Processing On Line **1** (2011) 208–212 5

[5] Chambolle, A.: Total variation minimization and a class of binary mrf models. Energy Minimization Methods in Computer Vision and Pattern Recognition **3757** (2005) 132–152 5

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

[7] Chuiab, C., Mhaskar, H.: Mra contextual-recovery extension of smooth functions on manifolds. Applied and Computational Harmonic Analysis **28** (01 2010) 104–113 5

[8] 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) (2007) 2080–2095 2, 5

[9] Damelin, S., Hoang, N.: On surface completion and image inpainting by biharmonic functions: Numerical aspects. International Journal of Mathematics and Mathematical Sciences **2018** (01 2018) 8 5

[10] Guedj, B., Srinivasa Desikan, B.: Pycobra: A python toolbox for ensemble learning and visualisation. Journal of Machine Learning Research **18**(190) (2018) 1–5 1, 2, 3, 6

[11] Kumar, B.S.: Image denoising based on non-local means filter and its method noise thresholding. Signal, image and video processing **7**(6) (2013) 1211–1227 7

[12] Lee, J.S., Jurkevich, L., Dewaele, P., Wambacq, P., Oosterlinck, A.: Speckle filtering of synthetic aperture radar images: A review. Remote sensing reviews **8**(4) (1994) 313–340 5

[13] 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

[14] Lucy, L.: An iterative technique for the rectification of observed distributions. Astronomical Journal **19** (06 1974) 745 5

[15] Richardson, W.H.: Bayesian-based iterative method of image restoration. Journal of the Optical Society of America **62** (1972) 55–59 5

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

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

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

[19] Wang, Z., Bovik, A.C.: A universal image quality index. IEEE signal processing letters **9**(3) (2002) 81–84 7