Image Denoising

Homework n°3 - Experimental report

1 Exploring Patch Similarity in an Image

We first focus on the experiment from [3].
In some denoising methods, we assume that for any given patch in an image, the image will also contain patches which are similar to the first one, and close to it, and that these patches will follow a Gaussian distribution.

In the demo, the user first selects an initial patch (seed), and the program finds and displays similar patches. Then, the user can also compare these patches by looking at histograms, or by analyzing the results of a Principal Component Analysis (PCA).

1.1 Self-similarity assumption

We first test the validity of the hypothesis of self-similarity.

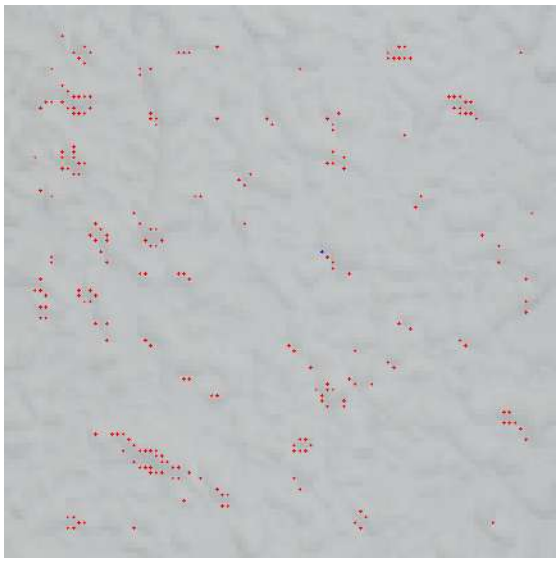
We consider three images, select a patch, and look at the similar patches found by the algorithm.

- The first image is the image of a wall with asperities on it.
- The second image is one of a building, and the seed patch is an edge of this building.
- The third image is the same as the previous one, but this time the initial patch is chosen to be on a tree.

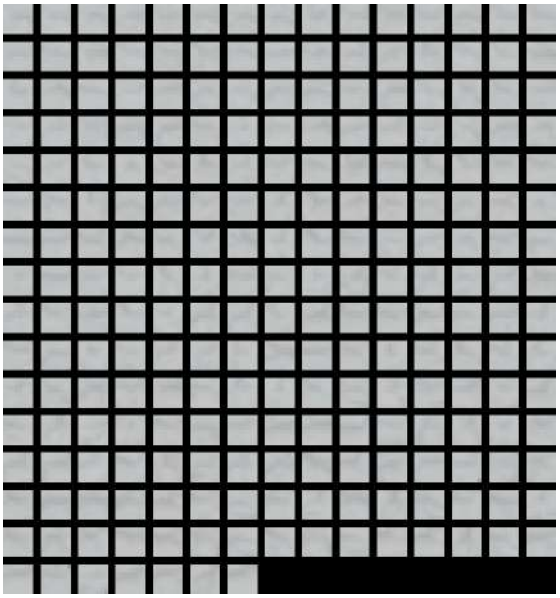
The results can be seen in Figure 1 (the blue point is the center of the seed patch).

- For the image of the wall, the algorithm easily detected similar patches where there are asperities in the wall (because we used Canny points). These patches obviously all look alike.
- For the building, though we initially selected a point on the edge between the building and the sky, many similar patches were found that do not contain the sky at all.
- For the tree, the result is quite satisfying in the sense that the detected patches do represent the tree, but they are globally distinct (because no two patches can really be the same in this case).

Overall, the patches found by the algorithm are indeed very similar to the seed for the case of uniform areas, still similar but potentially less near edges, but in the case of complex textures (the tree), they are not (which is obvious since the patches are essentially all distinct).
Therefore this shows why Non-Local Means works well for uniform areas, but not necessarily so well for textured regions.

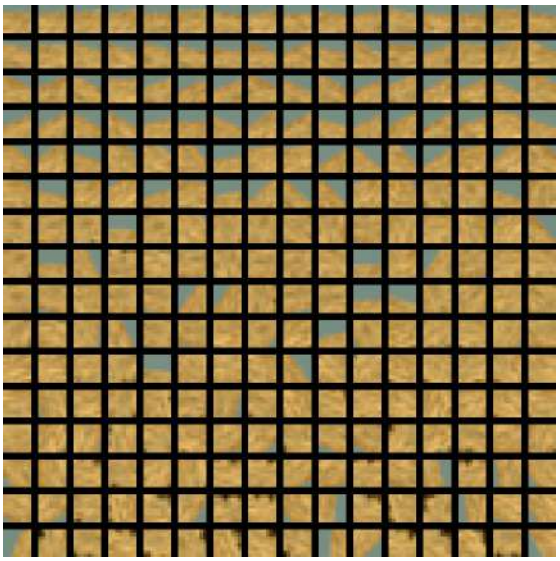


(a) Center of the patches (1st image)

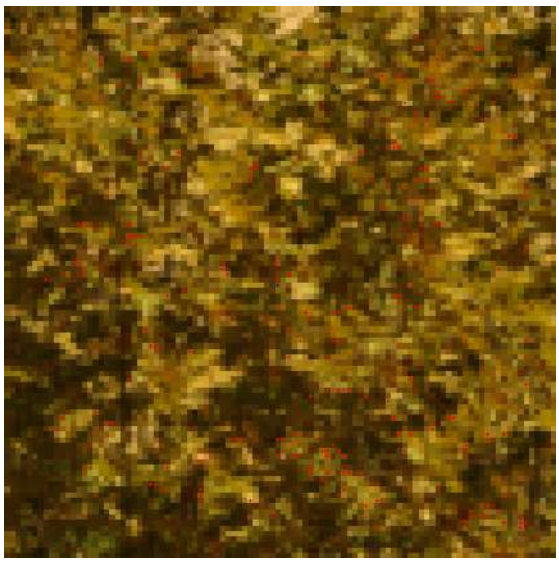


(b) Patches (1st image)

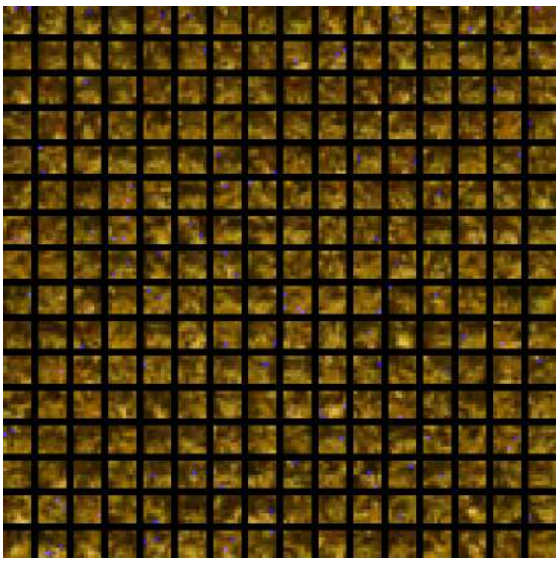
(c) Center of the patches (2nd image)



(d) Patches (2nd image)



(e) Center of the patches (3rd image)



(f) Patches (3rd image)

Figure 1: Similar patches for three images

1.2 Gaussian assumption

Then, we test the Gaussian assumption by performing PCA on the three images.

- For the wall image, the results can be found in Figure 2. We see that the eigenvectors quickly seem to be composed of noise only, but the eigenvalues do not go very quickly to 0 (so this is not really sparse).

  The projections seem to fit quite well with the Gaussian assumption, with p-values given by $0.1887, 0.0026, 0.2061$.

  The two Gaussians seem redundant, and have similar standard deviation (about 2.9).

  From this, we deduce that the Gaussian assumption holds rather well.

  In fact, for this image, there is not much contrast, the patches are similar, so the only variation is basically due to noise. And with the typical Poisson noise, which can be approximated by a Gaussian noise, this is why the Gaussian assumption holds (except maybe for the green channel).

- For the building image, the results can be found in Figure 3. We see that the eigenvalues go faster to 0 (the representation is sparser), and only the first eigenvectors seem to be really relevant.

  The standard deviation of the two Gaussians are approximately 19 and 9, and we see that the two Gaussians are not redundant.

  The p-values are all equal to 0 (up to some precision of course).

  This leads us to reject the Gaussian assumption here.

- For the tree image, the results can be found in Figure 4. The eigenvalues do not go quickly to 0 (so it is not sparse).

  The two Gaussians seem redundant (see Figure 5), and have similar standard deviation (about 20).

  The p-values are $0.3783, 0.6700, 0.1412$.

  So we do not reject the Gaussian assumption.

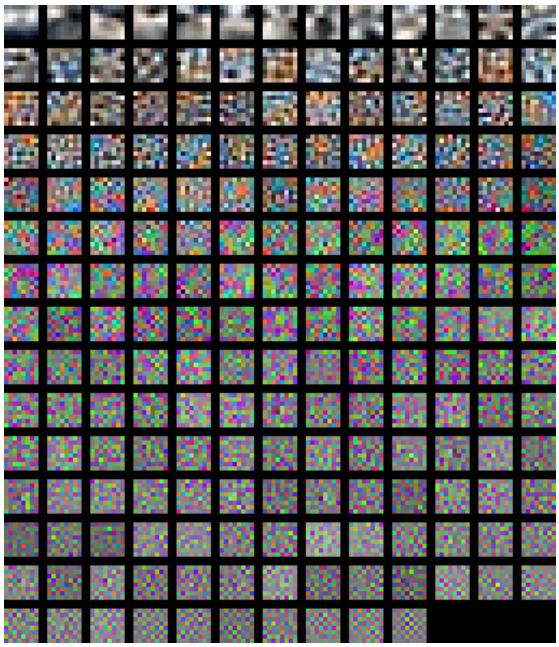
So in the end, the Gaussian assumption does not always hold, as seen for instance for the building image (that is, a region with an edge).

1.3 Impact of the parameters

- Search window size: this highly depends on the image and the seed patch, and whether enough similar patches can be found nearby.

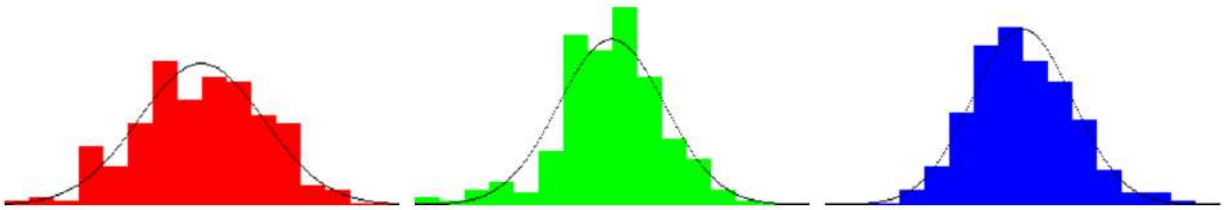
- Size of the patches: if we increase it, the result can be better for the building image (because then, the absence of the sky in another patch will reduce the similarity with the seed patch which contains the sky), but essentially the same for the wall or the tree.

- Number of closest patches: having this parameter too high may not be relevant in the case where the number of similar patches in the image is low, but may smoothen the Gaussians if the number of similar patches is high.



(a) PCA: Eigenvectors (1st image)

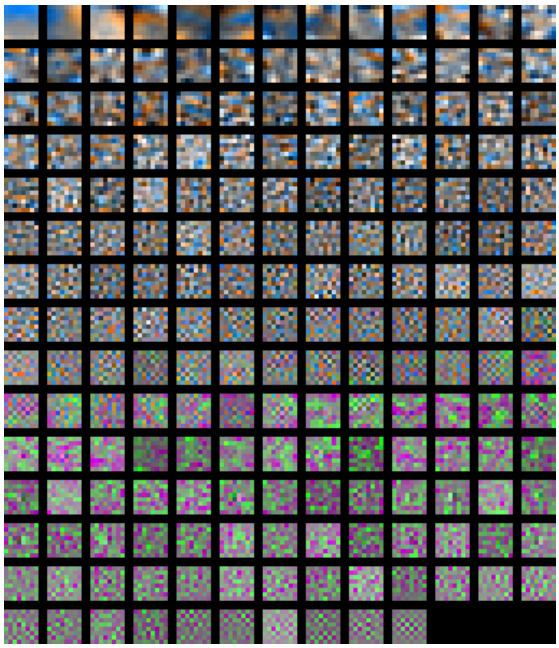
(b) PCA: Eigenvalues (1st image)



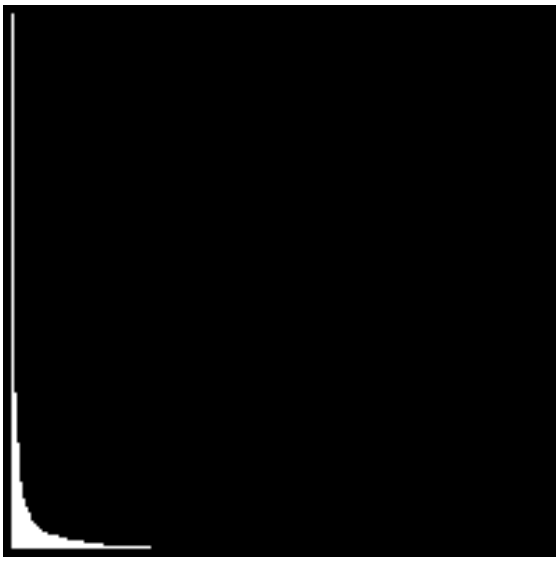
(c) PCA: projections (1st image)

Figure 2: Result of the PCA on the 1st image (wall)

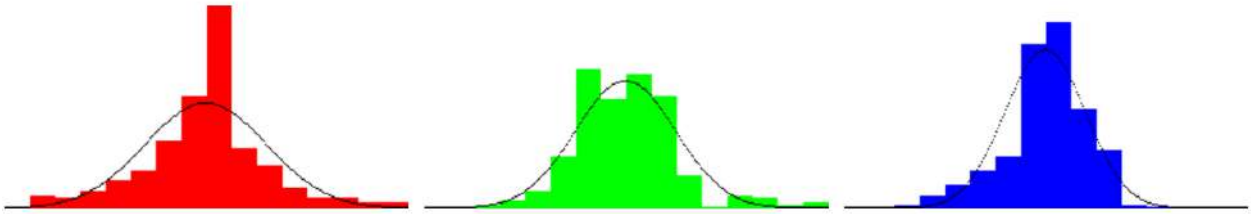
- Normalizing by orientation, mean, variance: it is very useful to normalize by the orientation (that way, a wall or a roof are essentially the same thing), but probably more debatable for the others (should two patches be considered the same if one is much brighter than the other ?).
- Extraction method: in the case of the building, using Canny points leads the algorithm to use points on the roof, for which the corresponding patch does not contain the sky. Using all points allows it to use points in the sky (that are not necessarily considered Canny), which may be more similar to the seed patch. So the method used can depend on the context.



(a) PCA: Eigenvectors (2nd image)

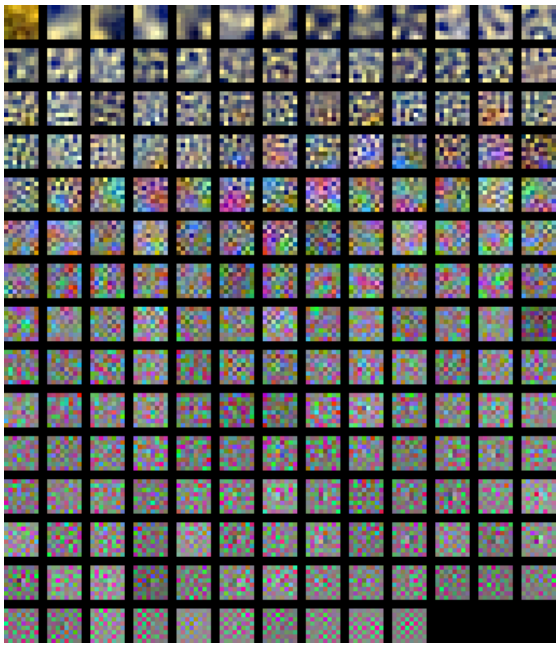


(b) PCA: Eigenvalues (2nd image)

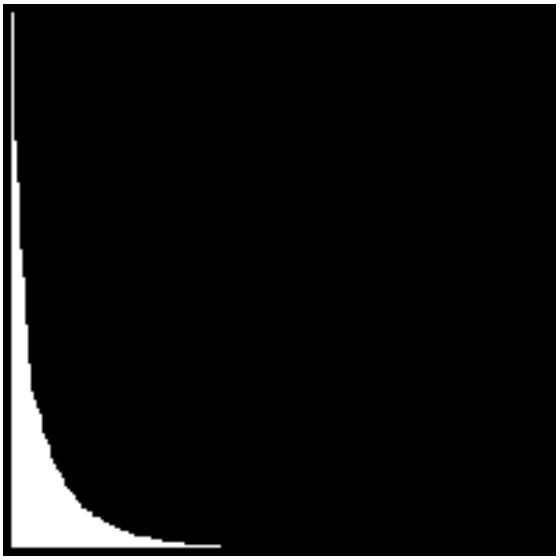


(c) PCA: projections (2nd image)

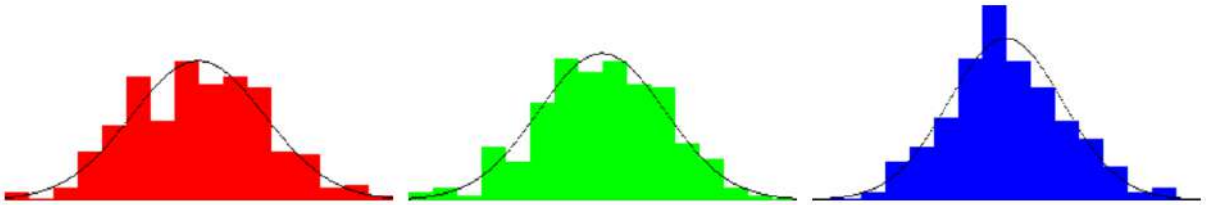
Figure 3: Result of the PCA on the 2nd image (building)



(a) PCA: Eigenvectors (3rd)



(b) PCA: Eigenvalues (3rd)



(c) PCA: projections (3rd image)

Figure 4: Result of the PCA on the 3rd image (tree)

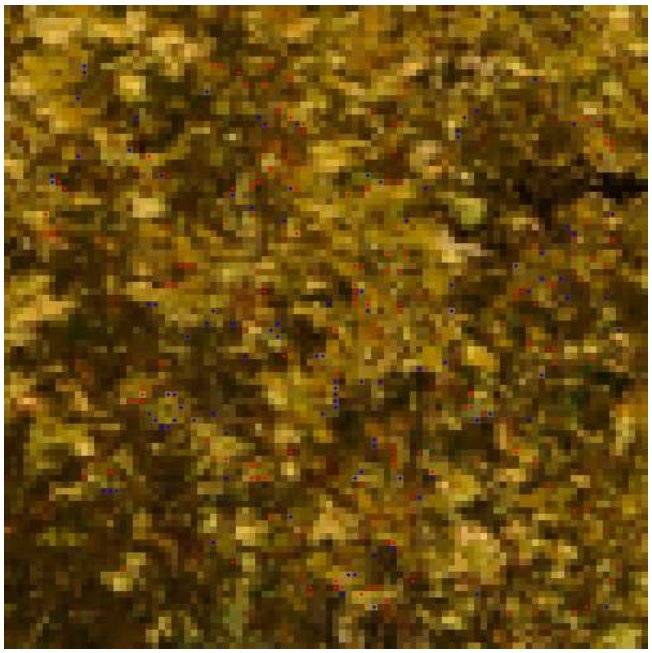


Figure 5: Two Gaussians for the tree image

2 Non-Local Bayes and BM3D

The Non-Local Bayes denoising algorithm [2] is an extension of the Non-Local Means denoising algorithm where, instead of simply using a weighted average of similar patches, we use a Gaussian prior and then compute the *Maximum A Posteriori* estimator.
We compare it to BM3D [1].

2.1 Non-Local Bayes

For NL-Bayes, note that it is possible to use or not the homogeneous area criteria during first step and second step.
For the image in Figure 9, we tested all four possibilities for different values of the noise, and the results are exposed in Figure 6.

σ / PSNR	**Yes / No**	**Yes / Yes**	**No / No**	**No / Yes**
$\sigma = \mathbf{15}$	32.72 dB	32.71 dB	32.56 dB	32.55 dB
$\sigma = \mathbf{30}$	29.03 dB	28.91 dB	28.87 dB	28.75 dB
$\sigma = \mathbf{50}$	26.82 dB	26.56 dB	26.78 dB	26.53 dB
$\sigma = \mathbf{70}$	25.29 dB	24.99 dB	25.22 dB	25.04 dB

Figure 6: PSNR when using homogeneous area criteria (Yes) or not (No) for each of the two steps

We see that on this example, it is always preferable to use the homogeneous area criteria during the first step, but not during the second step. This is what we will therefore do in the next experiments.

2.2 Theoretical comparison between NL-Bayes and BM3D

The main idea of both NL-Bayes and BM3D is to group similar patches together, filter them together, and then aggregate them. The output is then used as an (oracle) input to a second step.

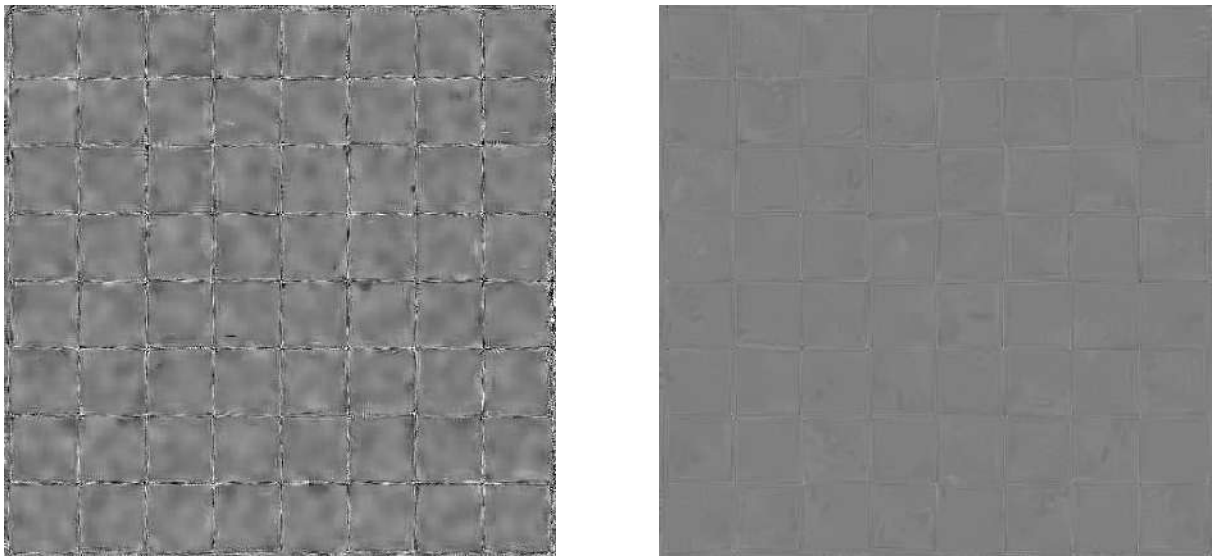
The main difference between the two algorithms is the way the patches are filtered together: for NL-Bayes, the idea is to consider the initial patch, and detect whether it belongs to an homogeneous area or not, by computing a standard deviation. In the homogeneous case, a basic average is computed; in the non-homogeneous case, the Gaussian distribution we mentioned above is used, using empirically estimated parameters.

Except for this difference, the other steps are mainly similar. [2] details the differences in Table 1, among which:

- NL-Bayes does not process the patches that were already used;
- As explained above, the collaborative filtering part is quite different;
- During the second step, BM3D only uses channel Y^o of the image obtained after the first step, while NL-Bayes uses all the channels.

2.3 Comparison between NL-Bayes and BM3D in practice

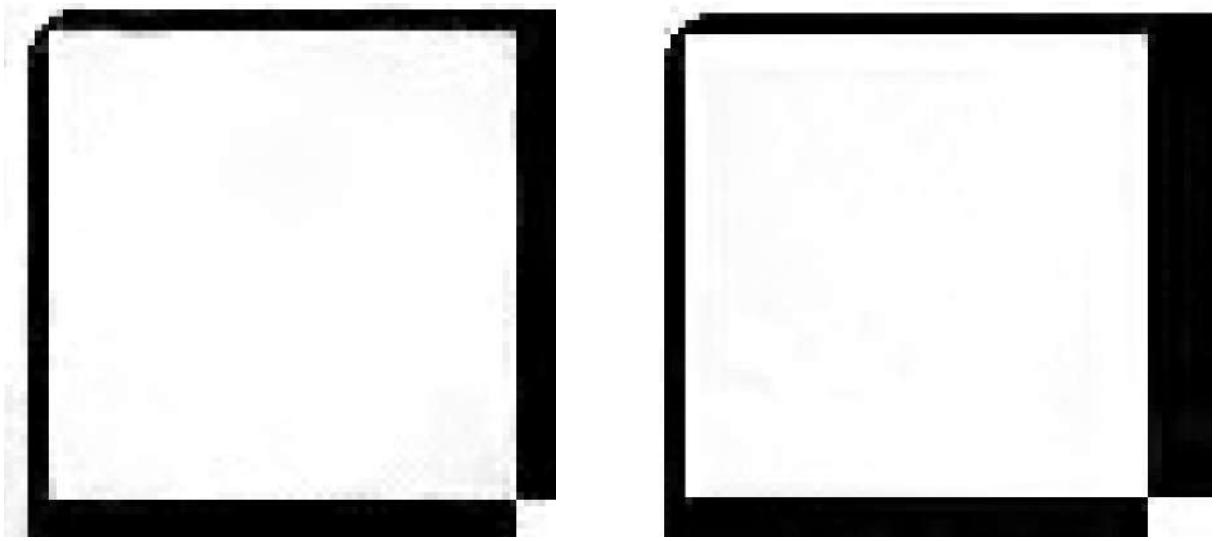
We first compare the two methods on a simple chessboard in Figure 7, where we display the differences.



(a) Non-Local Bayes (34.43 dB) (b) BM3D (38.07 dB)

Figure 7: Result (difference) of NL-Bayes and BM3D (chessboard)

We see a large difference for the edges, which are much stronger for NL-Bayes than for BM3D. Figure 8 shows a zoomed view for both methods, where we can see the difference near the edges (and the corners !).
However overall, we do not see a huge difference between the two denoised images, despite the gap between the PSNR (BM3D being better).

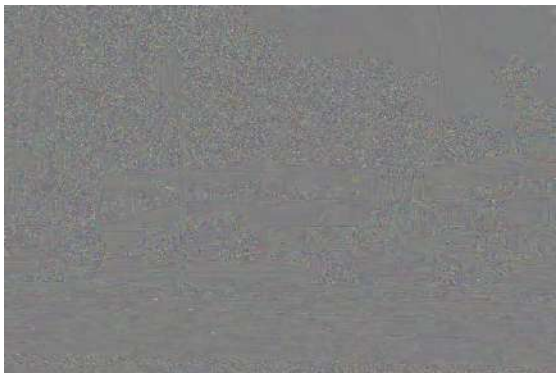


(a) Non-Local Bayes (34.43 dB) (b) BM3D (38.07 dB)

Figure 8: Result (difference) of NL-Bayes and BM3D (chessboard)

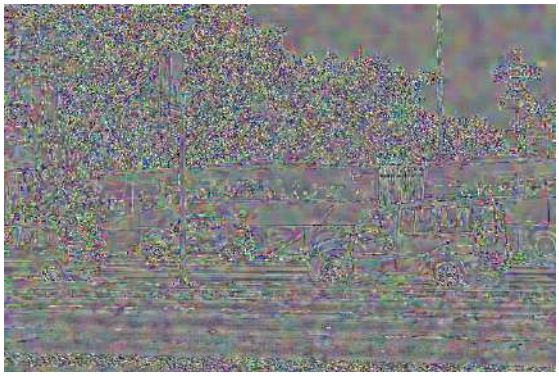
In Figure 9, we consider a "real" image.

We see that the edges are sharper with NL-Bayes (for instance the edge of the building on the right).
However, the homogeneous areas, such as the sky or the road, tend to be less good than with BM3D (green spots appear).
But, when looking at the trees, NL-Bayes shows details that are totally unseen with BM3D.



(a) BM3D (difference)

(b) BM3D (denoised, 28.85 dB)



(c) NLBayes (difference)

(d) NLBayes (denoised, 29.03 dB)

Figure 9: Comparison between BM3D and NLBayes (real image)

And in terms of PSNR, the two have similar results.

References

[1] Marc Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2:175–213, 2012. https://doi.org/10.5201/ipol.2012.l-bm3d.

[2] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. *Image Processing On Line*, 3:1–42, 2013. https://doi.org/10.5201/ipol.2013.16.

[3] Jose-Luis Lisani and Jean-Michel Morel. Exploring Patch Similarity in an Image. *Image Processing On Line*, 11:284–316, 2021. https://doi.org/10.5201/ipol.2021.325.