# Introduction to Numerical Imaging

Multi-Scale Denoising

*Authors:*
Leo Paillier
Jules Scholler

*Professors:*
Julie Delon
Yann Gousseau

January 27, 2017

# Contents

**Abstract**

This project is part of the MVA MSc. We have worked in a group of two and we split the work as follows: Jules Scholler wrote the first part of the report (abstract to section 3.2 included), he implemented and tested the first two methods ([2] and [1]). Leo Paillier wrote the second part of the report (section 3.3 up to conclusion), he implemented and tested the last two methods ([3] and [4]). We have worked together for most of the project and know the four different method even if we didn't implemented them.

# 1 Introduction

Images taken with both digital cameras and conventional film cameras will pick up noise from a variety of sources. Further use of these images will often require that the noise be removed for aesthetic purposes such as photography, films, or for practical purposes such as medical imaging or general computer vision. There exists a great variety of noises described by their statistics. In this report we are focusing on the benefit of multi-scales approaches in denoising. We will implement four different multi-scale algorithms and denoise using the same patch-based method: Non-Local Means.

The idea of multi-scale emerged with very noisy image restoration. In that case, artifacts inherent to the denoising algorithm start appearing, low frequency artifacts with NL-means for intance. A natural idea to deal with low frequency noise is to involve a coarse to fine multi-scale procedure denoising the image progressively at all scales and recombining them afterwards. There are several ways for sub-samplings and recombining images and we are comparing four of them in this report.

# 2 Protocol

In order to compare the different multi-scale approaches we used the same protocol with each algorithm. The denoising is performed with NL-means, which is an actual state-of-the-art tool in image processing.

## 2.1 NL-means

Non-local means are a patch-based algorithm. Unlike "local mean" filters, which take the mean value of a group of pixels surrounding a target pixel to smooth the image, NL-means filtering takes a mean of all pixels (or a vicinity of the target pixel) in the image, weighted by how similar the patches around the pixels are to the target pixel patch. The different parameters are the patch width $w$, the size of the vicinity $R$ and the similarity function which compares patches and returns the weights. We used the Gaussian weighting function:

$$S(i,j) = Ae^{\frac{|P_i - P_j|^2}{h^2}} \tag{1}$$

Where $P_i$ is the patch around the pixel target, $P_j$ is another patch in the vicinity $\Omega_R$ of pixel $i$, $h$ is a parameter that penalizes differences, A is a normalization parameters such that

$\sum_{j\in\Omega_R} S(i,j) = 1$. We denote by $I_i$ the value of pixel $i$, NL-means algorithm writes:

$$I_i = \sum_{j\in\Omega_R} S(i,j) I_j \tag{2}$$

The pixel $i$ is non-uniformly (according to the patch likelihood) averaged by all the pixel in $\Omega_R$. Note that, without optimization, the NL-means algorithm can be very greedy. Fortunately, a version of the algorithm that uses image integrals is much more faster and allow us to fine tune our parameters in reasonable times.

### 2.2 Benchmark

For each of the four multi-scale approaches we started by optimizing the traditional (non multi-scale) NL-means algorithm with respect to $w, R, h$. Doing that we obtained the "best" standard denoising achievable with NL-means. We did the same optimization with multi-scale approaches and then compared the results. In order to compare the two methods, we run our tests with a very high noise level.

## 3 Multi-scale approaches

### 3.1 H.-C. Burger and S. Harmeling meta procedure [1]

#### 3.1.1 Principle

The algorithm proposed in [1] is based on Laplacian pyramids. Fig. 1 summarizes the method graphically.

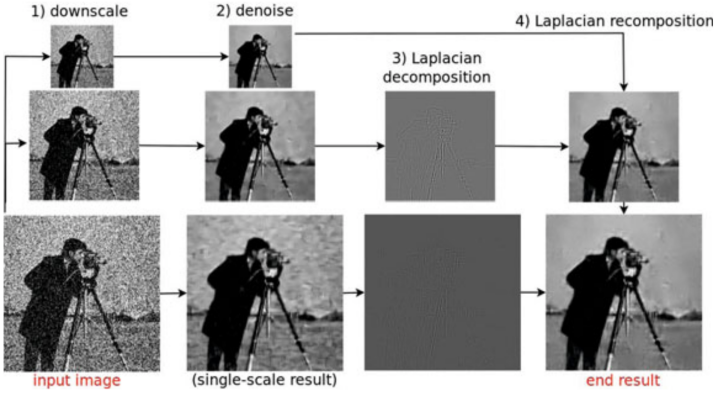



Figure 1: Denoising at different scales and then combines these images similarly to Laplacian pyramids - Figure taken in [1]

The method perform in two steps:

1. Create $n$ down-sampled version of the noisy image (use typically the matlab imresize function with 'lanczos3' kernel) and denoise them with NL-means.
2. Recombine the denoised images in a Laplacian-pyramid fashion to obtain the final denoised image.

**Denoising at different scales** We denote by $\alpha_1, ..., \alpha_n$ the scaling factors. Given a noisy image $x_0$, we create $n$ down-sampled images:

$$x_1 = d_{\alpha_1}(x_0) \quad , ..., \quad x_n = d_{\alpha_n}(x_n) \tag{3}$$

where $d$ is the down-sampling operator. The down-sampled images are denoised using NL-means:

$$y_0 = \text{NL-means}(x_0) \quad , ..., \quad y_n = \text{NL-means}(x_n) \tag{4}$$

**Recombination** We decompose the images $y_i$ into low and high frequency components $l_i$ and $h_i$:

$$l_i = d_{\alpha_i/\alpha_j}(y_i) \qquad\qquad h_i = y_i - u_{\alpha_j/\alpha_i}(l_i) \tag{5}$$

where $u$ is the up-sampling operator. Then, $l_i$ is discarded and replaced by $y_{i+1}$. Doing that, the low frequency are replaced with a down-sampled denoised image. This should works because we assume that $y_{i+1}$ contains more accurate low-frequency information. Then, combining $y_{i+1}$ and $h_i$ we obtain:

$$z_i = h_i + u_{\alpha_j/\alpha_i}(y_{i+1}) \tag{6}$$

$$z_i = \underbrace{y_i - u_{\alpha_j/\alpha_i}(d_{\alpha_i/\alpha_j}(y_i))}_{\text{high frequencies of } y_i} + \underbrace{u_{\alpha_j/\alpha_i}(y_{i+1})}_{\text{low frequencies of } y_{i+1}} \tag{7}$$

We will see that with (6) the denoising algorithm performs poorly, indeed, the high-frequency components $z_i$ are beneficial in lower noise settings, but detrimental at higher noise levels. The resulting denoised image appears dull because the high frequencies pixel have very high values, which results in shrinking horizontally the image histogram. Furthermore, the small values of $l_i$ are typically high frequency noise. To reduce these artifacts we implemented a threshold. The threshold operator $\mathcal{T}$ is defined as follows:

$$\mathcal{T}(y, \lambda_{min}) = \begin{cases} y[i] = y[i] & \text{if } y[i] > \lambda_{min} \\ y[i] = 0 & \text{otherwise} \end{cases} \tag{8}$$

The final denoised image is:

$$z_i = \mathcal{T}(h_i, \lambda_{min}) + u_{\alpha_j/\alpha_i}(y_{i+1}) \tag{9}$$

In practice, we keep a certain amount (typically 5%) of the highest coefficients.

### 3.1.2 Results

We present here the result obtained for various images. We added white Gaussian noise on the original images with a standard deviation $\sigma = 0.55$ (intensity pixel $I \in [0, 1]$). The noise repartition function is presented on the following figure.

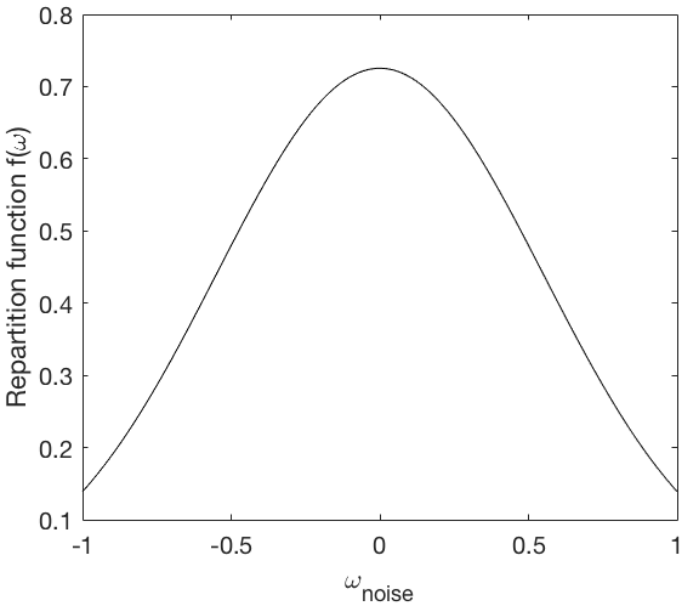


Figure 2: Noise repartition function

Because the level of noise is very high one can see that the noisy image is very deteriorated, it is hard to distinguish the original image. One can see on the following figure that the MS-ML-denoised image is dull and that there is a remaining high noise:

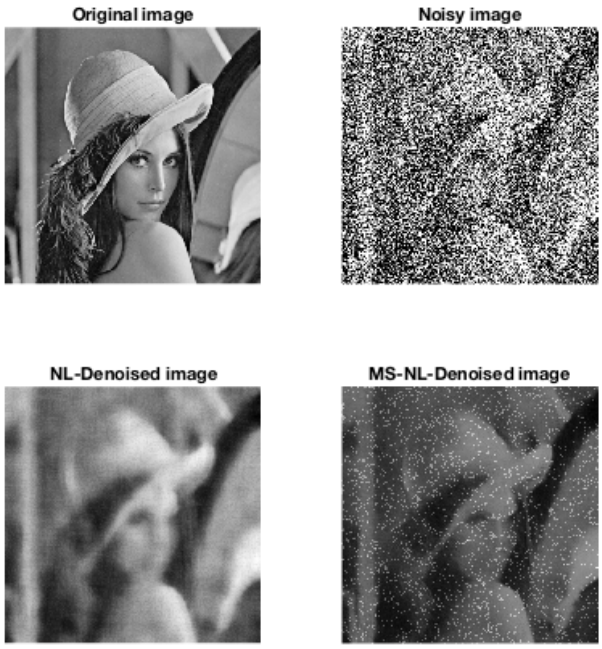
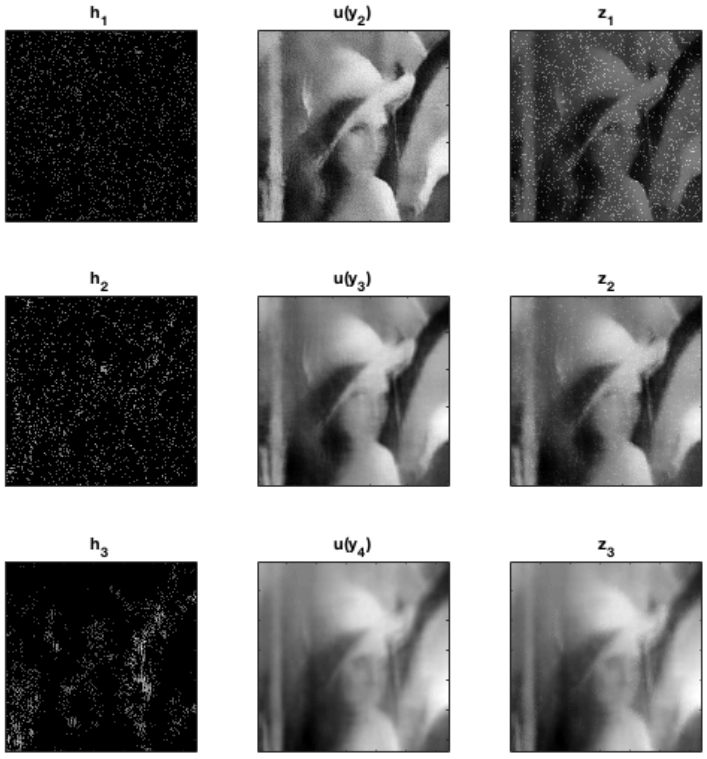
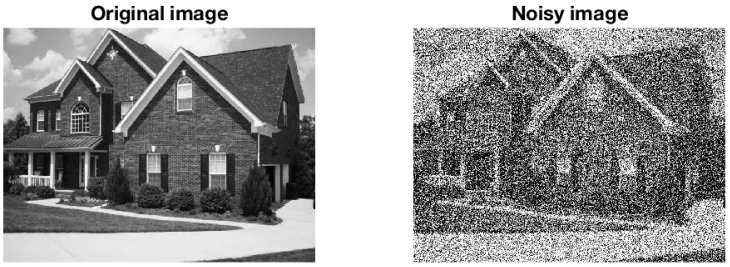


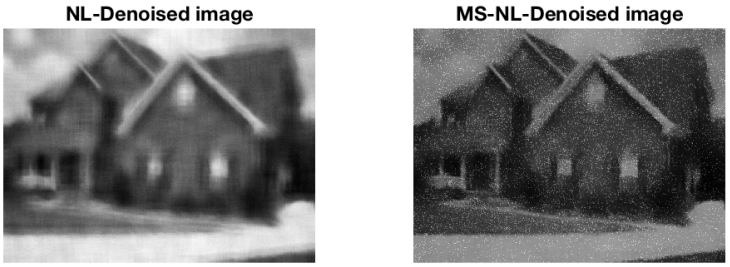

Figure 3: NL-means: $h = 40, R = 15, w = 10$ - MS-NL-means $h = 20, R = 10, w = 5$

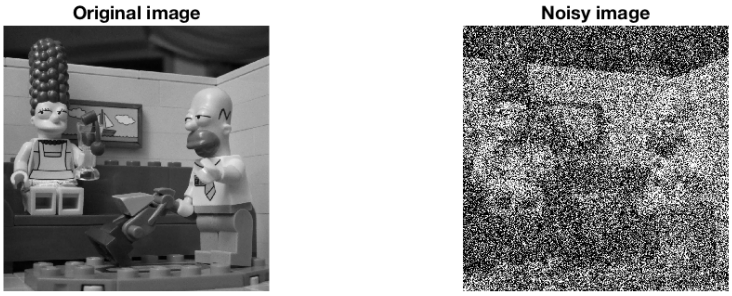
By looking at the construction of the MS denoised image we can see this phenomena:

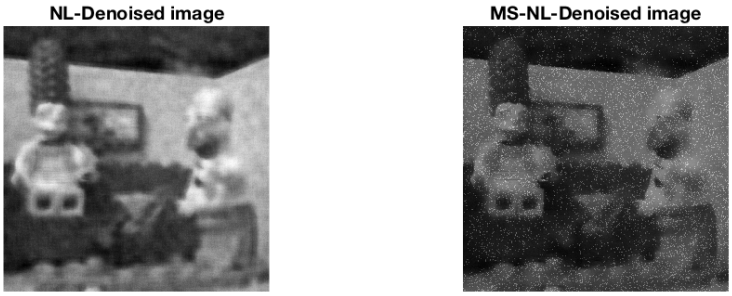


The high frequency $h_i$ are almost only noise at high scale, which result in a noisy final image. We can suppress these artifacts by rejecting the high frequencies for the higher scales but then the image becomes blurry. Moreover, the number of parameters to tune is high and this algorithm use is not systematic. This multi-scales approaches did not convince us compared to others presented here-after (e.g. the noise clinic). Nonetheless, we present some results obtained with this method on the following figures.



## 3.2 The Noise Clinic algorithm [2]

### 3.2.1 Principle

The multiscale algorithm proposed in [2] sub-samples recursively the image into four channels that are partly redundant. The four channels are obtained by moving the sub-sampling grid by respectively (0,0), (1,0), (0,1), (1,1). In that way there is enough information for up-sampling after denoising the denoised images at the lower scale. Doing that, there is no aliasing when reconstructing the image at the higher scale.

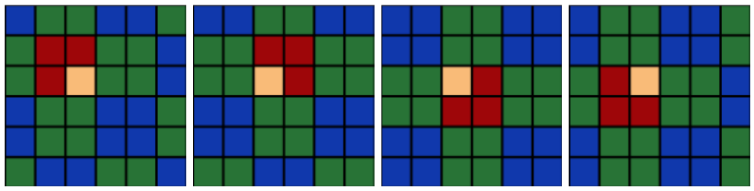


Figure 4: Four different origins for sub-sampling a grid. The origin is represented by the yellow pixel.

As explained on Fig. 3, the four sub-sampled images can be seen as an averaging of the original image with different origins. The sub-sampled images are denoised with NL-means and up-sampled by re-combining the four denoised channels. The optimal denoising was achieved using 2 sub-scales.

### 3.2.2 Results

We present here the result obtained for various images. We added white Gaussian noise on the original image with a standard deviation $\sigma = 0.55$ (intensity pixel $I \in [0, 1]$). We can see that the noisy image is very deteriorated, it is hard to distinguish the original image. In the caption we give for each image the optimal parameters for the classic NL-means and for the multi-scale approach. Note that for multi-scales approaches the size of the patches $w$ and the size of the vicinity $R$ are smaller in the original image but become greater on sub-scales (it is multiplied by a factor of 2 for each sub-sampling).

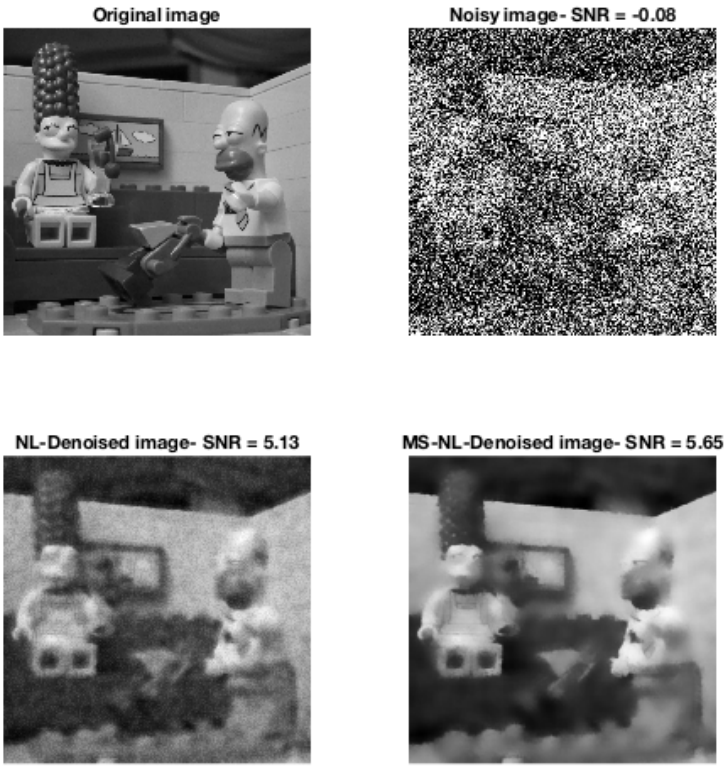



Figure 5: NL-means: $h = 40, R = 15, w = 10$ - MS-NL-means (2 scales) $h = 12, R = 6, w = 3$

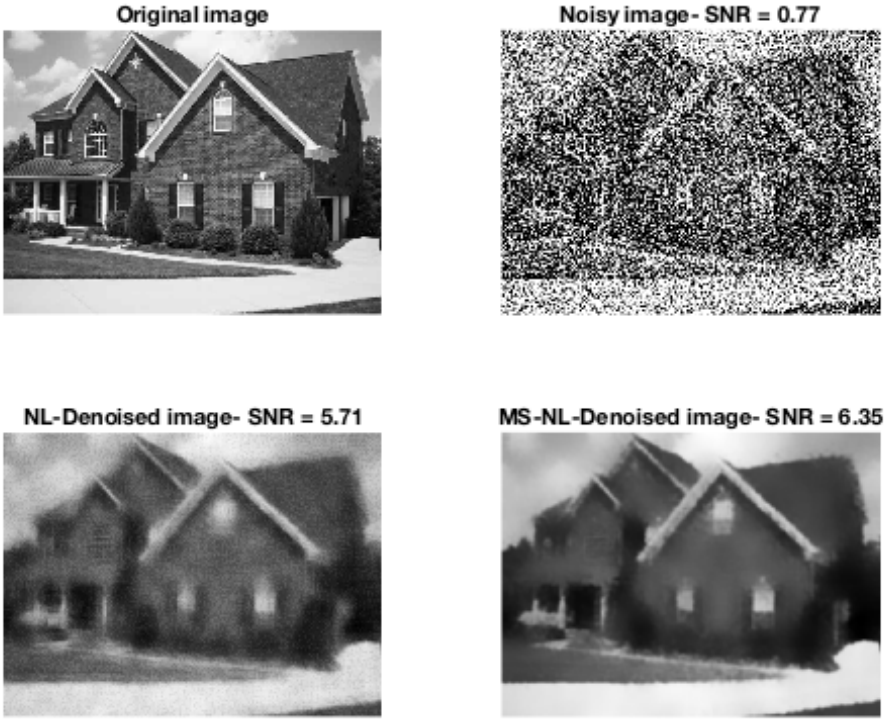



Figure 6: NL-means: $h = 40, R = 15, w = 10$ - MS-NL-means (2 scales) $h = 12, R = 6, w = 3$


Figure 7: NL-means: $h = 40, R = 15, w = 10$ - MS-NL-means (2 scales) $h = 12, R = 6, w = 3$

We can see that the multi-scale approach is clearly better at denoising very noisy images. It suppresses almost all the low frequency artifacts and the restored image has an overall better quality. The main advantage of this multi-scale algorithm is that the optimal parameters found were the same. This method is therefore universal with respect to the multi-scale parameters.

## 3.3 Ray Histogram Fusion [3]

### 3.3.1 Principle

This algorithm was initially developed for image rendering using Monte Carlo path tracing. The idea is the following: to generate a 2D image from a 3D scene we randomly cast light rays from the camera to the scene and follow their paths taking into consideration reflections, refractions, light sources, object textures and materials, etc. For each pixel we then end up with an histogram of lights rays which are usually averaged to get the pixel value.

Although this technique is still more computationally efficient than solving the true illumination distribution it is still very greedy. In order to limit the number of rays that have to be simulated, the global illumination algorithm presentend in this paper uses a similarity measure on the rays histogram to allow pixel to share their histograms thus virtually boosting the number of rays with a small marginal cost instead of plain averaging of the histogram.

Below is an example of different histogram and the similarity between pixels. Both the Figure 8 and the caption paragraph are direclty taken from the paper.

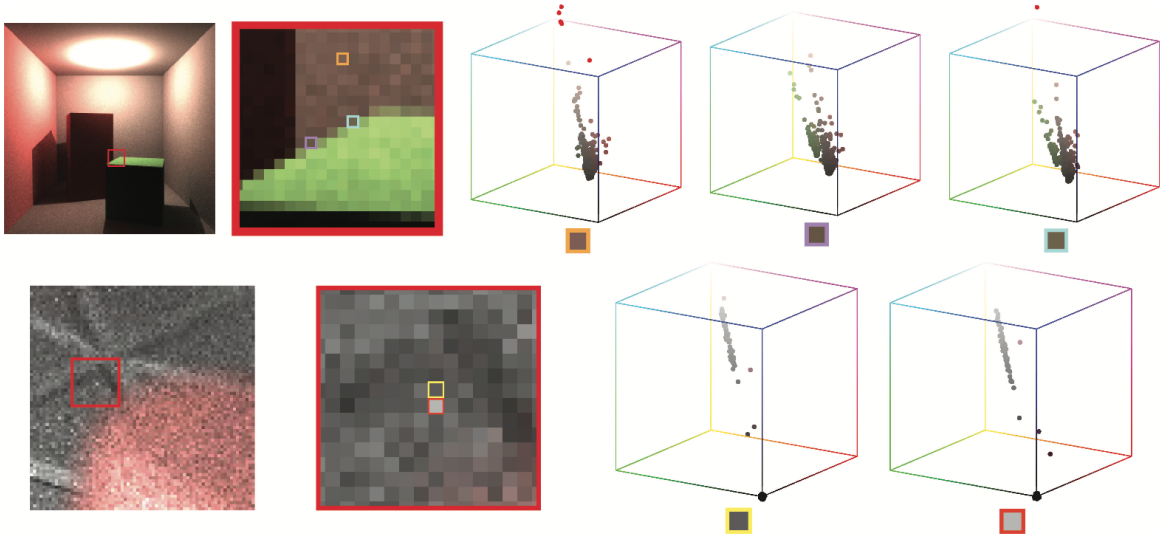


Figure 8: Ray histogram similarity between pixels.

The top row singles out three pixels in the Cornell Box scene and their sample color distributions. (The samples with color values falling out of the $[0,1]^3-$box are by convention colored in red.) The first pixel, situated on the brown wall, has a unimodal sample color distribution. The other two pixels belong to an occlusion boundary showing a bimodal green-brown distribution. This feature is shared by many pixels on the same boundary, which can therefore share their samples. The bottom row shows two pixels of the `toasters` scene with different colors. Their sample color distributions are nevertheless very similar and will therefore be merged as well.

Alas in our situation we don't have access to the light rays histogram for each pixel as all the information we have is the pixel color. Thus most of the paper can not be directly applied to the denoising problem that we are presented with.

Nonetheless the gist of the multiscale part can be easily transcribed to denoising:

1. From a noisy image input $\tilde{u}$, generate a gaussian multiscale sequence $(\tilde{u}_s)_{1\leq s\leq N}$ by recursively applying a gaussian filter to avoid aliasing followed by a decimation-by-2.

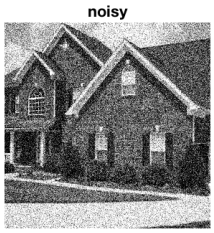
2. Apply the denoising algorithm separately to each scale to recover $(\bar{u}_s)_{1\leq s\leq N}$.

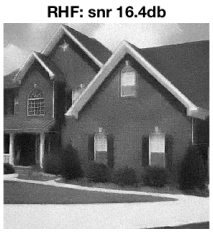
3. Compute the final image $\bar{u}=\hat{u}_0$ using the recursion $\hat{u}_i=\bar{u}_i-U_1D_1\bar{u}_i+U_1\hat{u}_{i+1}$ initialized with $\hat{u}_N=\bar{u}_N$ and where $U_1$ is a bicubic interpolator by a factor 2 and $D_1$ is the gaussian filtering plus a decimation by 2.

Basically we are using the low frequencies from the coarser scales and combining it with the high frequencies from the finer scales.

### 3.3.2 Results

First we will start by presenting the result of both the NL-mean and RHF algorithm applied on the 3 images corrupted with a gaussian noise of $\sigma = 50$ in the figures below.





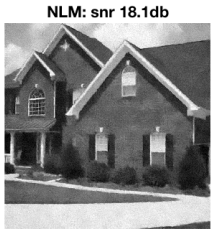
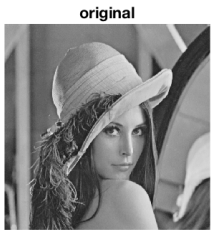
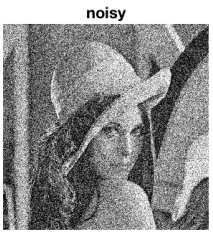
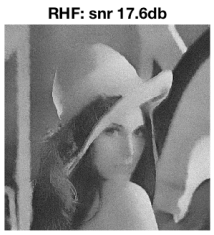



Figure 9: RHF vs NL-mean with $\sigma = 50$, house.







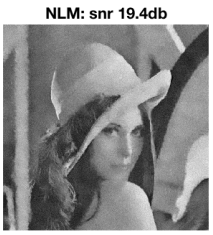
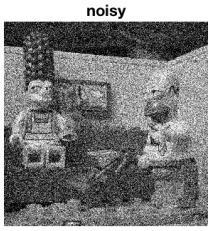



Figure 10: RHF vs NL-mean with $\sigma = 50$, Lenna.



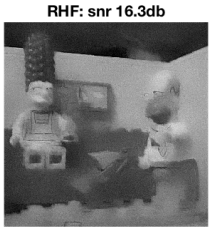



Figure 11: RHF vs NL-mean with $\sigma = 50$, Simpson.

In the previous pictures we only used 2 scales. As can be seen in the figure below, increasing the number of scales doesn't increase the SNR, on the contrary.

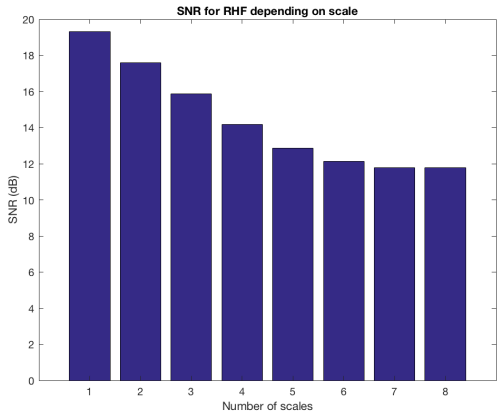



Figure 12: SNR for the RHF algorithm at different scales.

This might be due to the bicubic interpolation which leads to overshooting or haloing. This can be seen easily when increasing the number of scales and displaying the image in color. By looking at the edges of the roof in the picture below we can clearly observe the overshoot effect.

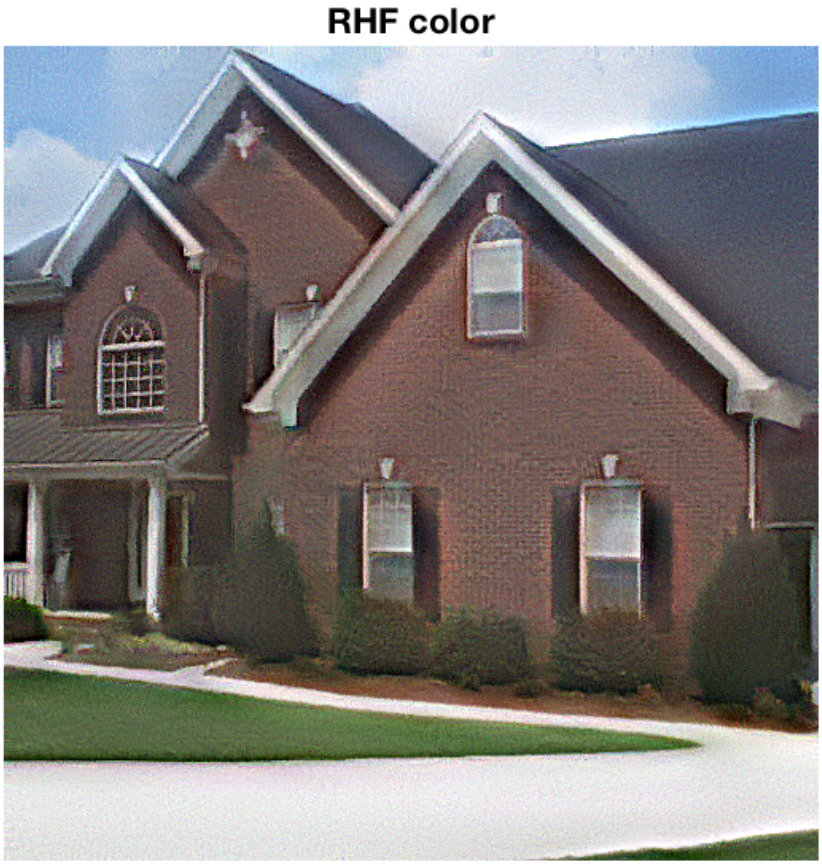



Figure 13: Overshoot or haloing effect with bicubic interpolation with 5 scales.

## 3.4 Multi-Scale Expected Patch Log Likelihood [4]

### 3.4.1 Principle

Most of the latest patch-based approaches do the following: the image is decomposed in fully overlapping patches, then each of those patches is restored and finally the patches are averaged to recompose the original image. What it means is that the prior is imposed at the patch-level but is lost when the global averaging is being done which leads to artifacts in the final image. Thus the idea of this paper is to impose the prior on every scale patches of the final image.

Formally if we denote by $X$ the original image and $Y$ the noisy image we want to find $X^*$ that maximizes:

$$\max_X P(X \mid Y) = \min_X -\log P(Y \mid X) - \log P(X)$$

The multiscale patch priors from the article in the case of 2 scales are defined as follow:

$$\log P(X) = w_1 \sum_i \log P_1(R_i X) + w_2 \sum_i \log P_2(\hat{R}_i S X)$$

Where $P_1$ and $P_2$ follow the GMM model defined in [5] which is trained using 200 mixtures components with zero means and full covariance matrices on $2 \times 10^6$ patches of $8 \times 8$ from 50000 natural images:

$$blaTumefaisdeserreursdecompilation$$

$R_i$ is a patch extractor operator,  is a low-pass filter $H$ (that is fine-tuned on the dataset) followed by a downsampling $D$ and $\hat{R}_i$ extracts the patch from the decimated signal $SX$. Besides, the weights $w_1$ and $w_2$ represent the importance of the different scales and are also fine-tune depending on the dataset. This equation can be easily generalized for any number of scales. It is also to be noted that to avoid artifacts all possible downsampling grid should be used and then recombined. Thus using the noise model we obtain:

$$\min_X \frac{\lambda}{2}\|X - Y\|_2^2 - w_1 \sum_i \log P_1(R_i X) + w_2 \sum_i \log P_2(\hat{R}_i S X)$$

Where $\lambda = \frac{p}{\sigma^2}$ and $p$ is the patch size. This can be solved recursively using Half Quadratic Splitting by introducing a set of auxiliary variables $z_i$ and $\hat{z}_i$:

$$\begin{aligned}\min_X \frac{\lambda}{2}\|X - Y\|_2^2 + w_1 \sum_i \min_{z_i} \left( \frac{\beta}{2}\|R_i X - z_i\|_2^2 - \log P_1(z_i) \right) \\ + w_2 \sum_i \min_{\hat{z}_i} \left( \frac{\hat{\beta}}{2}\|\hat{R}_i S X - \hat{z}_i\|_2^2 - \log P_2(\hat{z}_i) \right)\end{aligned}$$

When $\beta \to +\infty$ and $\hat{\beta} \to +\infty$ then we obtain the desired results.

To solve this equation first we consider $X$ fixed and we solve for $z_i$ and $\hat{z}_i$ using the GMM. We chose the most likely Gaussian:

$$\begin{aligned}\max_k P(k \mid R_i X) &= \max_k P(R_i X \mid k) P(k) \\ &= \min_k -\log P(R_i X \mid k) - \log P(k) \\ &= \min_k \frac{1}{2} \log\left(\left|\Sigma_k + \frac{1}{\beta} I\right|\right) + \frac{1}{2}(R_i X)^\intercal \left(\Sigma_k + \frac{1}{\beta} I\right)^{-1} (R_i X) - \log \pi_k\end{aligned}$$

Once we have found the best Gaussian $\hat{k}$ we can apply a Wiener filter to restore the patch:

$$z_i = \Sigma_{\hat{k}} \left(\Sigma_{\hat{k}} + \frac{1}{\beta} I\right)^{-1} R_i X$$

Then we consider $z_i$ and $\hat{z}_i$ fixed and solve for $X$ which yields the closed form solution:

$$X = \left(\lambda I + w_1 \beta \sum_i R_i^\intercal R_i + w_2 \hat{\beta} \sum_i S^\intercal \hat{R}_i^\intercal \hat{R}_i S\right)^{-1} \left(\lambda Y + w_1 \beta \sum_i R_i^\intercal z_i + w_2 \hat{\beta} \sum_i S^\intercal \hat{R}_i^\intercal \hat{z}_i\right)$$

After this iteration, $\beta$ and $\hat{\beta}$ are increased and we start the cycle again. Empirically it is shown that we can stop the algorithm after 4-5 iterations. Practically $\beta$ and $\hat{\beta}$ are equal up to a constant and only one of them needs to be fine-tuned.

After several iterations, every patch extracted from the reconstructed image is likely given the local model, and similarly, every patch extracted from a decimated version of it is likely as well.

### 3.5 Results

We could not get a version of this algorithm running because the trained GMM model described in the paper [5] wasn't made available and we didn't succeed in implementing the EM algorithm to train it in a reasonnable time. We thus defer to the paper [4] for results.

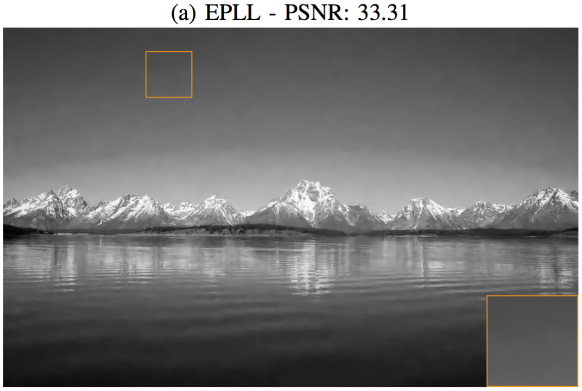
The first thing to realize is that the scale weights $w_1$ and $w_2$ plus the filtering $H$ and downsampling $D$ are learning on the dataset and thus vary depending on the noise level. This means that depending on the noise level we have different values for those parameter as can be seen below. In particular this means that a prior estimation of the noise level is needed before running the algorithm (which takes in the order of 5 minutes to complete).

| $\sigma$ | EPLL | MS-EPLL | Filter | DS | Weight |
|---|---|---|---|---|---|
| 15 | 32.17 | 32.30 | Identity | 1 | 1 |
| | | | Identity minus Gaussian with $\sigma = 0.6$ | 1 | 0.2 |
| | | | Gaussian with $\sigma = 0.8$ | 2 | 0.2 |
| 25 | 29.74 | 29.89 | Identity | 1 | 1 |
| | | | Gaussian with $\sigma = 0.8$ | 2 | 0.15 |
| 50 | 26.60 | 26.77 | Identity | 1 | 1 |
| | | | Gaussian with $\sigma = 0.8$ | 2 | 0.05 |
| | | | Gaussian with $\sigma = 1.5$ | 4 | 0.05 |
| 100 | 23.56 | 23.80 | Identity | 1 | 1 |
| | | | Gaussian with $\sigma = 1.1$ | 2 | 0.04 |
| | | | Gaussian with $\sigma = 1.7$ | 4 | 0.05 |

Figure 14: Fine-tuned parameters depending on the noise level.

Below with display results from the article showing the improvement of the multiscale EPLL over the traditional EPLL. We observe that the low-frequency artifacts are completely removed.

(a) EPLL - PSNR: 33.31



(b) MSEPLL - PSNR: 33.61

Figure 15: Traditional Expected Patch Log Likehood vs Multiscale version.

Last but not least it's important to note that if you consider the algorithm in its entirety its complexity is the highest of the 4. Basically you first have to train a patch prior, which raises several question on which prior to chose, and then you have to solve the complete

log-likelihood equation iteratively using Half Quadratic Splitting to both optimize the MAP problem and the quadratic function on $X$ with an optimization algorithm. Theoretically this algorithm seems very promising but one has to keep in mind that the computation time is significant, especially when the patch prior is taken into consideration.

# 4 Conclusion

[1] and [3] both follow the same general algorithm: downsampled version of the noisy images are created, then each of them is denoised and finally the final image is recomposed using the high-frequency details of the finer scale and the low-frequency details of the coarser scale. We have observed that those two approaches don't perform very well compared to other techniques.
[2] is built on the same premises but instead of taking one downsampled image, it considers all possible decimation grids and then average them when reconstructing the final image.
Finally [4] takes a whole different approach and focus more on a statistical approach by essentially learning a patch prior in the form of a Gaussian Mixture Model and then iteratively find a solution such that all extracted patch at every scale are likely given the patch prior. Although this algorithm was not implemented it is very promising if one can compute the aforementioned patch prior.

## References

[1] H. C. Burger and S. Harmeling. Improving denoising algorithms via a multi-scale meta-procedure. 2011.

[2] Marc Lebrun, Miguel Colom, and Jean-Michel Morel. The noise clinic: a blind image denoising algorithm. *Image Processing On Line*, 5:1–54, 2015.

[3] A.Buadès J.Chauvier N.Phelps J-M.Morel M.Delbraccio, P.Musé. Boosting monte carlo rendering by ray histogram fusion. *ACM TOG*, 2014.

[4] V. Papyan and M. Elad. Multi-scale patch-based image restoration. *Image Processing On Line*, 2016.

[5] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. *2011 IEEE International Conference on Computer Vision (ICCV 2011)*, 00(undefined):479–486, 2011.