MVA master stage report
Study of denoising algorithms and implementation of a new one: NL-means multiscale

M. Colom
miguel.colom@ens-cachan.fr

Co-directed by J.M. Morel and A. Buades

17/09/2010

Contents

1 Summary

This report is the result of my stage at CMLA, co-directed by J.M. Morel and A. Buades. During this stage several denoising techniques and background theories were studied, specially the NL-means method.

The report first formalizes what is the noise model that is used to treat mathematically the images, emphasizing the idea that any image acquired with some captor is nothing more than the realization of a random variable with some distribution. Any raw image is physically a realization of a random Poisson variable with mean $\tau u(x)$ and standard deviation proportional to $\sqrt{\tau u(x)}$, where τ is the exposure time used to obtain the image with the captor.

After discussing the general neighborhood filters and defining the noise model, the report addresses the noise estimation problem from a single image. Some experiments are made over a synthetic test pattern and also over natural images. The so called "method" noise is also defined.

The NL-means original algorithm is presented, and also its signal-dependent version. The report then shows results for images in which the added noise follows a uniform Gaussian distribution, which is, or not, signal-dependent. When the image is sub-scaled (considering the mean of every four pixels), if the noise is uniform, it is reduced by half. The new proposed NL-means multiscale algorithm is based on this simple fact. The algorithm is presented and several tests are shown.

To help understanding in detail how the algorithm works and what are the effects of sub-scaling and denoising on every scale, a natural image is contaminated with non-uniform noise of known parameters and analyzed on three scales.

Finally, new results on images on which the noise is real and not generated by a software are shown. These images were uploaded by users to the IPOL journal [31] and represent a good sample of *realistic* noise.

The method proved to be able to remove noise in such cases, on which the original NL-means fails even to detect it. This is the case of chemical noise on old photographies or old movies and the case of non-uniform noise due to the the characteristic of the captor or the type of application (astronomy, for example).

Many of the existing algorithms expect that the image is affected with some particular and known noise type (for example, Gaussian noise, Uniform noise, Laplace noise, Lorentz

noise, ...). The original NL-means only works with uniform noise, for instance.

Our end objective is to have something like an "image clinic", in which the type of noise that has damaged the image is not known *a priori*, so the image has to be analyzed and the noise classified and estimated before treating the image. The IPOL journal is a good tool for this, because researchers from all the world can upload their own pictures and test the algorithms online by themselves, without specifying the kind of noise they had, which they often ignore anyway.

The modification that turns NL-means into a multiscale approach is a first step, because now it is able to deal with coloured and signal dependent noise too.

All the images presented on this report are available to the readers of this report on request. In fact, it is preferable to look at the images on the screen than on the printed version, which has some compression and printer smoothing.

2 Introduction

The goal of image denoising methods is to recover the original image from a noisy measurement,

$$v(i) = u(i) + n(i), \tag{1}$$

where $v(i)$ is the observed value, $u(i)$ is the "true" value and $n(i)$ is the noise perturbation at a pixel i. The best simple way to model the effect of noise on a digital image is to add a gaussian white noise. In that case, $n(i)$ are i.i.d. gaussian values with zero mean and variance σ^2.

Several methods have been proposed to remove the noise and recover the true image u. Even though they may be very different in tools it must be emphasized that a wide class share the same basic remark : denoising is achieved by averaging. This averaging may be performed locally: the Gaussian smoothing model (Gabor [4]), the anisotropic filtering (Perona-Malik [5], Alvarez et al. [6]) and the neighborhood filtering (Yaroslavsky [7], Smith et al. [8], Tomasi et al. [9]), by the calculus of variations: the Total Variation minimization (Rudin-Osher-Fatemi [10]), or in the frequency domain: the empirical Wiener filters (Yaroslavsky [7]) and wavelet thresholding methods (Coiffman-Donoho [11, 12]).

Formally we define a denoising method D_h as a decomposition

$$v = D_h v + n(D_h, v),$$

where v is the noisy image and h is a filtering parameter which usually depends on the standard deviation of the noise. Ideally, $D_h v$ is smoother than v and $n(D_h, v)$ looks like the realization of a white noise. The decomposition of an image between a smooth part and a non smooth or oscillatory part is a current subject of research (for example Osher et al. [13]). In [14], Y. Meyer studied the suitable functional spaces for this decomposition. The primary scope of this latter study is not denoising since the oscillatory part contains both noise and texture.

The denoising methods should not alter the original image u. Now, most denoising methods degrade or remove the fine details and texture of u.

In this report, the *NL-means* algorithm is first presented. This algorithm is defined by the simple formula

$$NL[u](x) = \frac{1}{C(x)} \int_\Omega e^{-\frac{(G_a * |u(x+.)-u(y+.)|^2)(0)}{h^2}} \, u(y) \, dy,$$

where $x \in \Omega$, $C(x) = \int_\Omega e^{-\frac{(G_a * |u(x+.)-u(z+.)|^2)(0)}{h^2}} dz$ is a normalizing constant, G_a is a Gaussian kernel and h acts as a filtering parameter. This formula amounts to say that the denoised value at x is a mean of the values of all points whose Gaussian neighborhood looks like the neighborhood of x. The main difference of the NL-means algorithm with respect to local filters or frequency domain filters is the systematic use of all possible self-predictions the image can provide, in the spirit of [15]. For a more detailed analysis on the NL-means algorithm and a more complete comparison, see [16].

3 Noise model

Most digital images and movies are obtained by a CCD device. Following [17, 18, 19], CCD's show three kinds of noise. The first one is the *shot noise* proportional to the square root of the number of incoming photons in the captors during the exposure time, namely

$$n_0 = \sqrt{\frac{\Phi}{h\nu} t \cdot A \cdot \eta},$$

where Φ is the light power (W/m^2), $h\nu$ the photon energy (Ws), t the exposure time in seconds (s), A the pixel area (m^2) and η the quantum efficiency. The other constants being fixed we can simply retain $n_0 = c\sqrt{\Phi}$ where Φ is the "true image" and C a constant (see Figure 1).

Figure 1: Simulated shot noise. Left: original image u. Right: noise image $\sqrt{u}n$ where n is the realization of a zero mean white noise with standard deviation $\sigma = 1$. The noise in bright parts is larger than in dark parts, an effect which is corrected and sometimes reversed by gamma-correction.

Second, a *dark* or *obscurity noise* n_1 is due to spurious photons produced by the captor itself. We can assume the dark noise to be white, additive and with zero mean. The zero mean property is due to the subtraction of a *dark frame* from the raw image. The dark frame is obtained by averaging the obscurity noise over a long period of time.

Third, the *read out* noise n_2 is another electronic additive and signal independent noise. This noise can be assumed to have zero mean by the subtraction from the raw image of a *bias frame*.

Digital images eventually undergo a "gamma" correction, i.e. a nonlinear increasing contrast change g: "Gamma correction is the name of an internal adjustment made in the rendering of images through photography, television, and computer imaging. The adjustment causes the spacing of steps of shade between the brightest and dimmest part of an image to appear *appropriate* [19]. Summarizing,

$$u(i) = g\left(\Phi(i) + c\sqrt{\Phi(i)}n_0(i) + n_1(i) + n_2(i)\right),$$

where $u(i)$ is the observed intensity at a pixel i, $\Phi(i)$ the "true physical" light intensity average power sent by the scene to pixel i, c a constant, $n_0(i)$, $n_1(i)$ and $n_2(i)$ three independent and signal independent white noises. In practice $g(s) = s^\alpha$ with $0 < \alpha < 1$. When $\Phi(i)$ is large the shot noise $\sqrt{\Phi(i)}$ dominates n_1 and n_2 and is dominated by the signal $\Phi(i)$. Thus we can expand $u(i)$ as

$$u(i) \simeq g(\Phi(i)) + g'(\Phi(i))\left(c\sqrt{\Phi(i)}n_0(i) + n_1(i) + n_2(i)\right) =: g(\Phi(i)) + n(i). \qquad (2)$$

If instead $\Phi(i)$ is small with respect to $n_1(i) + n_2(i)$,

$$n(i) \simeq u(i) \simeq g(n_1(i) + n_2(i)). \tag{3}$$

Let us mention a case of particular interest. If $g(s) \simeq s^{\frac{1}{2}}$, the noise $n(i)$ reads

$$n(i) \simeq \begin{cases} n_0(i) & \text{in the bright parts of the image} \\ \\ \sqrt{n_1(i) + n_2(i)} & \text{in the dark parts of the image} \end{cases} \tag{4}$$

In all cases the noise is signal dependent but independent at different pixels. Figure 1 displays a simulated shot noise associated to the Lena image. This noise is signal dependent and much stronger in bright regions than in dark regions. In order to apply many computer vision algorithms, the noise parameters must be first estimated. For the study of these parameters for the previous real CCD model we refer the reader to [20].

4 General neighborhood filters

4.1 Local neighborhood filters

The more primitive neighborhood filters replace the color of a pixel with an average of the nearby pixels colors. Thus $J(i)$ is a spatial neighborhood. The filtered value can be written as $\mathcal{M}_\rho u(\mathbf{x}) = \frac{1}{\pi\rho^2} \int_{\mathbb{R}^2} e^{-\frac{|\mathbf{x}-\mathbf{y}|^2}{\rho^2}} u(\mathbf{y})\, d\mathbf{y}$, where the parameter ρ is roughly the size of the spatial neighborhood involved in the filtering. Now, the closest pixels to i have not necessarily the same color as i.

The idea is to average neighbor pixels which also have a similar color value. The filtered value by this strategy can be written as

$$NF_{h,\rho}\, u(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \int_{B_\rho(\mathbf{x})} e^{-\frac{|u(\mathbf{y})-u(\mathbf{x})|^2}{h^2}} u(\mathbf{y}) d\mathbf{y}, \tag{5}$$

where $u(\mathbf{x})$ is the color at $\mathbf{x}$ and $NF_{h,\rho}\, u(\mathbf{x})$ its denoised version. Only pixels inside $B_\rho(\mathbf{x})$ are averaged, h controls the color similarity and $C(\mathbf{x})$ is the normalization factor. SUSAN [8] and the bilateral filter [9] make this process more symmetric by involving a *bilateral* Gaussian depending on both space and gray level. This leads to

$$SNF_{h,\rho}\, u(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \int e^{-\frac{|\mathbf{x}-\mathbf{y}|^2}{\rho^2}} e^{-\frac{|u(\mathbf{y})-u(\mathbf{x})|^2}{h^2}} u(\mathbf{y}) d\mathbf{y}$$

another way to avoid the blurring effect of the spatial filtering $\mathcal{M}_\rho$ by a statistical correction which we are going to use in the sequel. When the Gaussian mean is performed on an edge, the variance of the performed mean can become larger than the variance of the noise. This is a clue that the average is not licit. A statistically optimal correction was proposed by Lee again [21],

$$L\mathcal{M}_\rho u(\mathbf{x}) = \mathcal{M}_\rho u(\mathbf{x}) + \frac{\sigma_{\mathbf{x}}^2}{\sigma_{\mathbf{x}}^2 + \sigma^2}(u(\mathbf{x}) - \mathcal{M}_\rho u(\mathbf{x})),$$

where

$$\sigma_{\mathbf{x}}^2 = \max(0, \frac{1}{\pi\rho^2}\int_{\mathbb{R}^2} e^{-\frac{|\mathbf{x}-\mathbf{y}|^2}{\rho^2}}(u(\mathbf{y}) - \mathcal{M}_\rho u(\mathbf{x}))^2\, d\mathbf{y} - \sigma^2)$$

and σ is the noise standard deviation. The original noisy values are less altered when the variance of the performed mean dominates the variance of the noise. This happens near the edges or in textured regions.

The bilateral filters perform a better denoising than Lee's correction. They maintain sharp boundaries, since they average pixels belonging to the same region as the reference pixel. Bilateral filters fail when the standard deviation of the noise exceeds the edge contrast.

The mean operation can be replaced by nonlinear operator like the median filter. The median filter [22] chooses the median value, that is, the value which has exactly the same number of gray level values above and below in a fixed neighborhood. The median filter preserves the main boundaries, but it tends to remove the details. This filter is optimal for the removal of impulse noise on images and does not blur edges. It is equivalent to an average of the pixels in a direction orthogonal to the gradient, that is to an *anisotropic diffusion* or *mean curvature motion* [23].

4.2 Non local averaging

The most similar pixels to a given pixel have no reason to be close to it. Think of periodic patterns, or of the elongated edges which appear in most images. In 1999 Efros and Leung [15] used non local self-similarities to synthesize textures and to fill in holes in images. Their algorithm scans a vast portion of the image in search of all the pixels that resemble the pixel in restoration. The resemblance is evaluated by comparing a whole window around each pixel, not just the color of the pixel itself. Applying this idea to neighborhood filters leads to a generalized neighborhood filter called non-local means (or NL-means) [24, 25]. NL-means has a formula quite similar to the sigma-filter,

$$NLu(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \int_{\Omega} e^{-\frac{(G_\rho * |u(\mathbf{x}+.) - u(\mathbf{y}+.)|^2)(0)}{h^2}} \; u(\mathbf{y}) \; d\mathbf{y}, \tag{6}$$

where G_ρ is the Gauss kernel with standard deviation ρ, $C(\mathbf{x})$ is the normalizing factor, h acts as a filtering parameter and

$$(G_\rho * |u(\mathbf{x}+.) - u(\mathbf{y}+.)|^2)(0) = \int_{\mathbb{R}^2} G_\rho(\mathbf{t}) |u(\mathbf{x}+\mathbf{t}) - u(\mathbf{y}+\mathbf{t})|^2 d\mathbf{t}.$$

The formula (6) means that $u(\mathbf{x})$ is replaced by a weighted average of $u(\mathbf{y})$. The weights are significant only if a Gaussian window around $\mathbf{y}$ looks like the corresponding Gaussian window around $\mathbf{x}$.

One of the limitations of the NL-means algorithm is the removal of highly structured noise as in jpeg compressed images. The NL-means is able to remove the block artifact due to compression but at the cost of removing some details as the difference between the compressed and restored images shows.

5 Noise estimation

5.1 Introduction

Most noise estimation methods have in common that the noise standard deviation is computed by measuring the derivative or equivalently the wavelet coefficient values of the image, the standard deviation as the median of absolute values of wavelet coefficients at the finest scale.

Olsen [29] and posteriorly Rank et al. [30] proposed to compute the noise standard deviation by taking a robust estimate on the histogram of sample variances of patches in the derivative image. In order to minimize the effect of edges small windows were preferred, with 3×3 or 5×5 pixels. The sample variance of small patches or the pointwise derivatives provide a non robust measure and require a considerable number of samples with few outliers to guarantee the correct selection of the standard deviation. We observed that the opposite point of view, that is, the use of larger windows 15×15 pixels to 21×21 pixels permits a more robust estimation. However, since larger windows may contain more edges a much smaller percentile will be preferred to the median, in practice the 1% or the 0.5%.

Noise in real photograph images is signal dependent. In order to adapt the noise estimation strategies, the gray level image histogram will be divided adaptively into a fixed number of bins having all the same number of samples. This is preferable to classical approaches where the gray range is divided into equal intervals. Such a uniform division can cause many bins to be almost empty.

Before adding noise to a image, a test image was created for which the noise curve showed low values. Finally, a signal dependent noise was added to them, with variance $8 + 2u$ where u was the noiseless gray level.

The uniform and adaptive divisions of the gray level range in a fixed number of 15 bins were compared, and several noise estimation methods were applied to estimate the noise standard deviation inside each bin. The performance of all methods are compared in Table 1 showing the average and standard deviation of the errors between the estimated and original noise curves. The best estimate is obtained by applying the proposed strategy using the variance of large patches rather than small ones or point derivatives. These measurements also confirm that the division of the gray level range into bins with fixed cardinality is preferable to the fixed length interval division. This experiment confirms that a signal dependent noise can be estimated with a high accuracy.

The noise in each channel is estimated independently. Each color range is divided adaptively into a fixed number of bins taking into account the color channel histogram. Inside each bin a percentile is used to estimate the standard deviation.

Fig. 3 displays the ground truth estimated curves with this strategy, both in RAW and JPEG format for two different ISO settings. The ground truth curves are compared with the ones estimated in the first image of the sequence by the proposed single image noise estimation algorithm. For the RAW case, the single image and ground truth estimated curves are nearly identical. Fig. 2 shows a lack of red in the RAW image of the calibration pattern, even if this pattern is actually gray. This effect is corrected by the white balance as observed in the JPEG image.

The ground truth noise curves estimated from the JPEG images do not agree at all with the classical noise model. This is due to the various image range nonlinear transformations applied by the camera hardware during the image formation, which modify the nature and standard deviation of the noise. The ground truth and single image estimated curves in the JPEG case have a similar shape but a different magnitude. The main new feature is that the interpolation and low pass filtering applied to the originally measured values have

	MAD	RMAD	MVPD	MVPD2
$\overline{e}$	1.81	2.87	1.58	0.75
std(e)	1.14	2.59	1.06	0.61

a) Uniform gray division

	MAD	RMAD	MVPD	MVPD2
$\overline{e}$	1.66	1.87	1.36	0.73
std(e)	1.04	1.17	0.90	0.35

b) Adaptive gray division

Table 1: A signal dependent noise with variance $8 + 2u$ is added to 110 noise free images. The uniform and adaptive strategies for dividing the gray level range in a fixed number of 15 bins are compared. For each strategy, the following noise estimation methods in each bin are compared: median of absolute derivatives (MAD), robust median of absolute derivatives (RMAD), median of sample variance of patches 3×3 of the derivative image (MVPD) and 0.005 percentile of sample variance of patches 21×21 of the derivative image (MVPD2). Are displayed the average and standard deviation of the errors between the estimated and original noise curves for the 110 images.

strongly altered the high frequency components of the noise.

The first step before trying to remove noise or extending the NL-means method to signal-depending noise, was to estimate the amount of noise added to the image.

To do it, the noise was estimated with the standard procedure [1], that consists on:

1. Compute the mean μ and standard deviation σ for each $N \times N$ block in the image. N is small (3 or 5, for example).

2. Classify the standard deviations according to their mean.

3. Take the median value of all standard deviations for each mean.

Different authors [2, 3] have proven that taking the image derivative leads to a better estimation of the noise.

When the variance of all pixels in the image is known, one can draw a histogram of the noise for every pixel intensity and every color channel. This histogram is called the *noise curve* of the image.

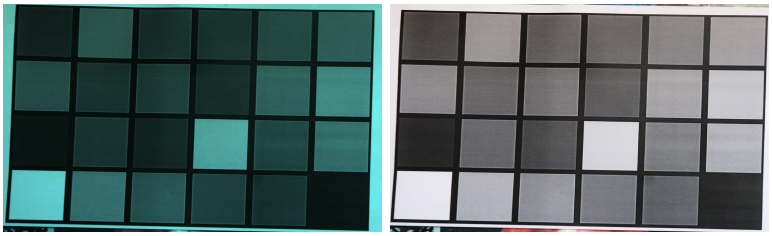


Figure 2: Calibration pattern used for noise ground truth estimation. Left: raw image. Right: JPEG image. Even if the calibration pattern is nearly gray the raw image looks blue because the red is less present. This effect is corrected by the white balance applied by the camera image chain leading to the JPEG image.

To compute the noise curves of several images, the original C/C++ source code by A. Buades was used as a base, but adapted and improved for many of the experiments.

The first modification consisted in adding TIFF and JPEG reading and writing support to the programs, using the libtiff and jpeglib libraries. Originally, the programs were only able to read and write images under the PM format.

In figure 4 we can see a 1024×1024 image that contains rough edges but without any noise.

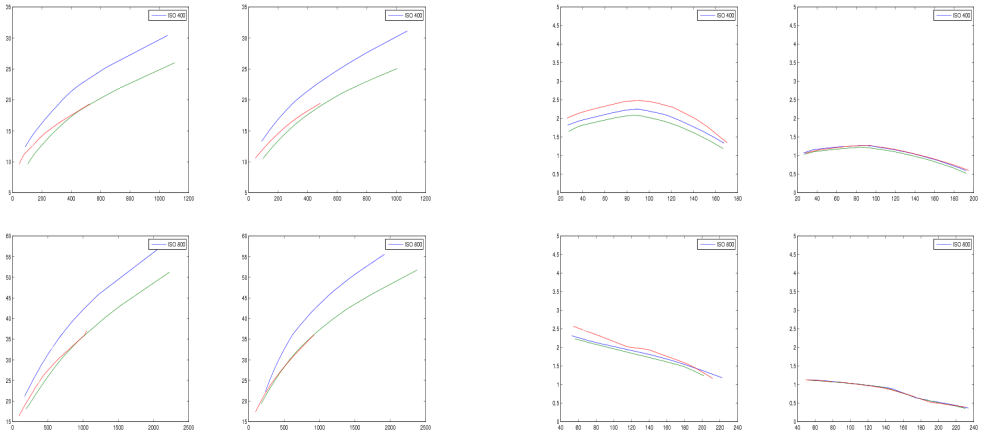
In figure 5 we can see the noise curve histogram corresponding to the image without any noise, computed witn $B = 100$ bins.

We observe that even when the original image does not contain any added noise, the noise curve that is recovered is not equal to zero in all its points, as we could expect.

The conclusion is double:

1. This simple method is not robust to the presence of rough edges in the image, because the variance computed in those blocks that lay over an edge is very high. Not because of the noise, but due to the variance of the data itself.

2. The noise is part of the image data, and therefore it could be impossible to remove it without introducing artifacts in the image. In fact, we can consider that any image u itself is physically a realization of a random Poisson variable with mean $\tau u(x)$ and standard deviation proportional to $\sqrt{\tau u(x)}$, where τ is the exposure time used to obtain the image with some captor. The Poisson noise is pixel dependent and, according to the Law of Large Numbers, we could find out the ideal value of every



a) RAW images b) JPEG images

Figure 3: Ground truth and single image noise estimates for the RAW and JPEG images of Fig. 2. The estimated curve by the temporal average and standard deviation coincide with the one estimated from the first image by the proposed single image noise estimation algorithm. This is not the case for the JPEG images. The ground truth and single image estimated curves in the JPEG case have a similar shape but a different magnitude. The interpolation and low pass filtering applied to the original measured values have altered the high frequency components of the noise and have correlated its low frequencies. This means that the noise statistics are no longer computable from a local patch of the image. The estimation of a noise curve can only be accomplished by computing the temporal variance in a sequence of images of the same scene.

pixel if we could observe a infinite or a large number of realizations of the process. The problem is that we have only one sample per pixel in the image we try to clean.

The noise curve in figure 5 was obtained using the derivative image. The derivative is computed like $d(x, y) = u(x, y) - u(x + 1, y)$ in the original code.

The problem of computing the derivative this way is that only the changes on the horizontal direction are taken into account, but not any other. There is a privileged direction, but the amount of noise should not depend on the orientation of the image, but on its data.

If the image is rotated, that fact should not change the noise curve computed for the image, but with the original derivation method used it depends strongly on the variations

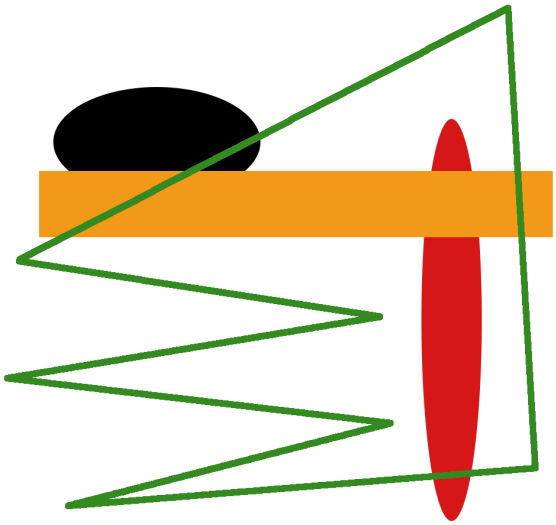


Figure 4: Image without any noise.

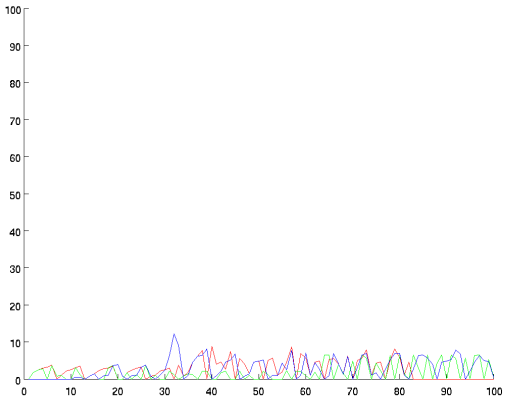


Figure 5: Noise curve for image 4.

of the horizontal direction.

In the figure 6 we consider that same image than in figure 4, but rotated 90 degrees clockwise.

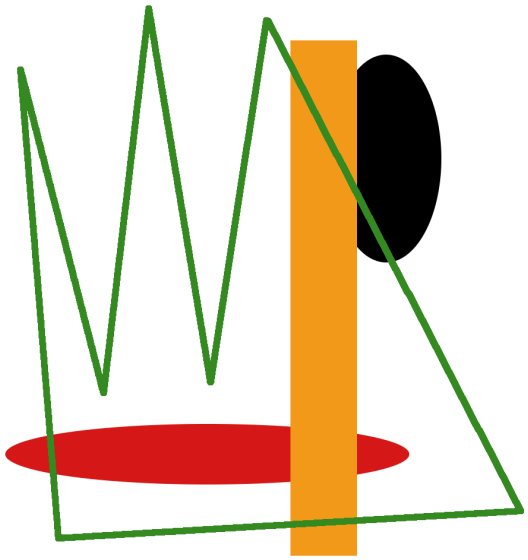


Figure 6: Rotated image without any noise.

In the figure 7 we can see the noise curve computed with that derivative schema.

As we can observe, the noise curves are different, but the images are exactly the same and only differ on the 90 degrees rotation.

The noise curves should be exactly the same, so to avoid the dependence on only one direction, the derivative function was modified to take into account the vertical direction too.

With the proposed new method, the derivatives are computed on both the horizontal and vertical directions and the arithmetic mean of both is assigned instead of just the derivative in the horizontal direction.

Because the derivative in one direction is assumed to keep the variance of the Gaussian noise, if we consider a new random variable made up by the sum of the derivatives on the horizontal and vertical directions, with this new random variable it is possible to recover the noise variance:

$\mathrm{Var}(X) = \sigma \wedge \mathrm{Var}(Y) = \sigma \Rightarrow \mathrm{Var}(\frac{X+Y}{2}) = \frac{1}{4}(\mathrm{Var}(X) + \mathrm{Var}(Y)) = \frac{1}{4}(\sigma + \sigma) = \frac{\sigma}{2}$.

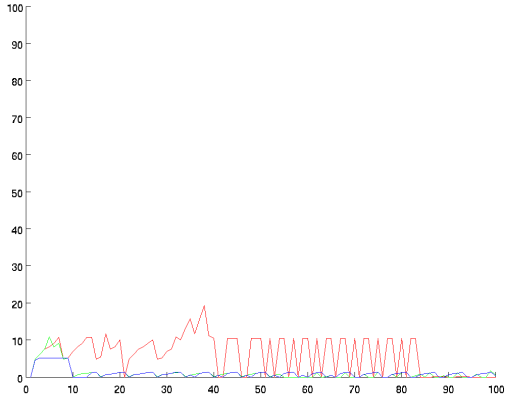


Figure 7: Rotated image curve noise.

In figure 8 we can see the noise curve for the rotated image using the new method. The noise curve is exactly the same for both the original and rotated images.

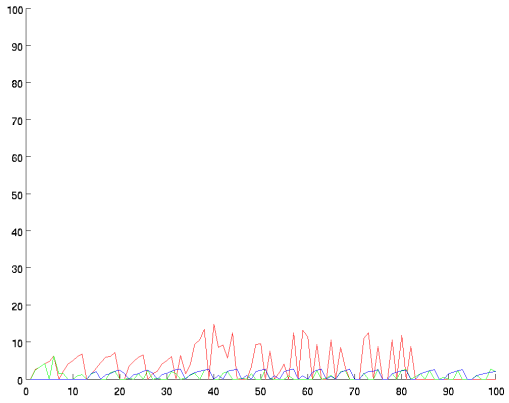


Figure 8: Rotated image curve noise (new method).

Because of that, the new derivation scheme was incorporated in the source code of the programs, and it is the one that is used in all the experiments.

5.2 Experiments with Gaussian noise in synthetic images

In this section we test the algorithm used to generate the noise curves with a synthetic image in which a noise of known standard deviation is added. The objective of this exper-

iment is to confirm that the noise curve matches the known σ in every bin.

The synthetic image consists on 256 blocks of size 100×100 pixels each one. Every one of these blocks has an intensity that varies from 0 to 255. They are ordered randomly.

In figure 9 we can see the image pattern used in this experiment.

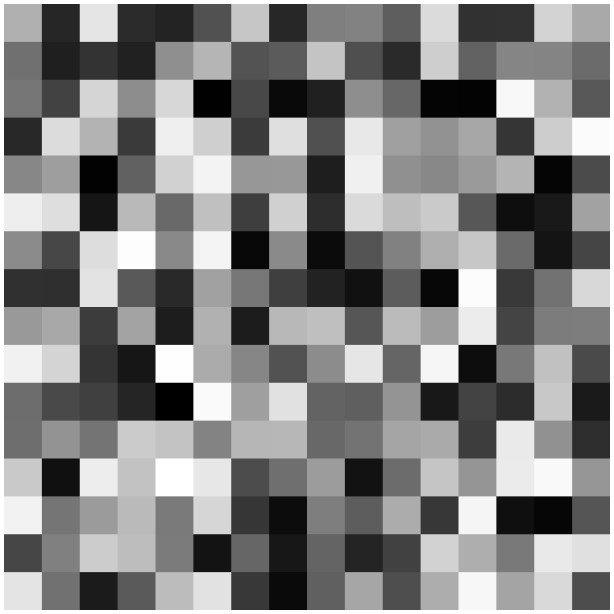


Figure 9: Test pattern image to test the noise curve algorithm.

In figure 10 we can see the same pattern, but contaminated with the addition of Gaussian noise with $\sigma = 30$.

The test pattern is useful to test the noise curve because it consists in several flat regions with the same gray level that can be covered with several $N \times N$ patches of the algorithm. Moreover, the transitions or edges between the blocks are soft[1], so the computed variance of every pixel is due mainly because of the noise.

In figure 11 we can see the computed noise curve for the pattern contaminated with Gaussian noise of $\sigma = 30$. As we can observe, the noise curve is very close to the value $\sigma = 30$ in every bin for every color channel, so the result is correct.

[1]One gray level between two blocks that are adjacent in the same row, and 16 gray levels for blocks that are adjacent in different rows.

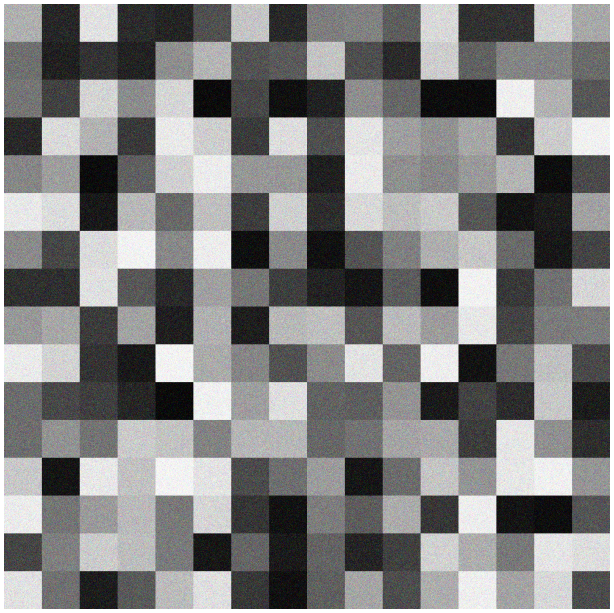


Figure 10: Test pattern contaminated with the addition of Gaussian noise of $\sigma = 30$.

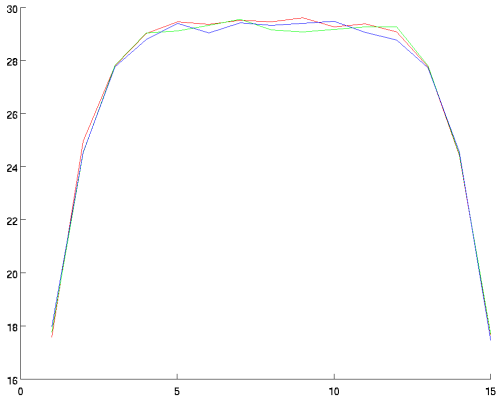


Figure 11: Noise curve for the noisy ($\sigma = 30$) pattern.

5.2.1 Effect of the rotation on the noise curve (synthetic test pattern image)

In this section, the noisy test pattern in figure 10 is rotated and the noise curves of the rotated and original noisy images are compared. In figure 12 we can see the rotated test pattern.

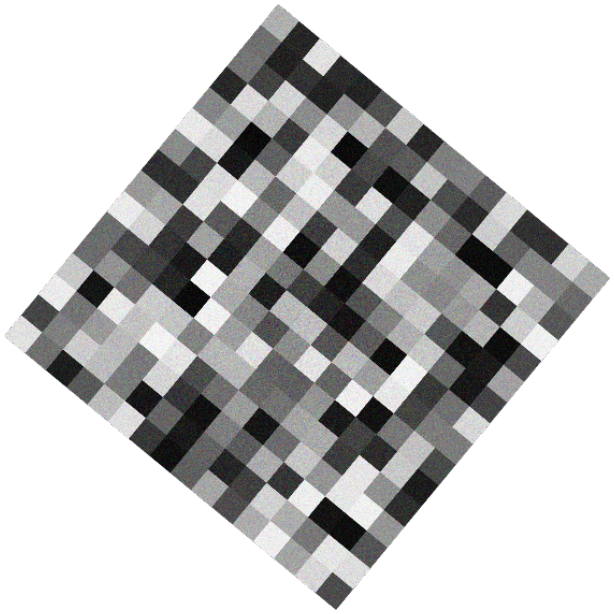


Figure 12: Rotated noisy pattern ($\sigma = 30$).

The noise estimated from both images should be the same, and the aim of the experiments of this section is to show that the noise curves computed with the new method are robust to rotation. It will be shown also that the original method is not.

In figure 13 we can see the noise curves for the rotated noisy pattern for both the original method (left) and the new one (right). The results are similar.

5.2.2 Effect of the rotation on the noise curve (natural image)

The experiments in this section compare the original method noise curve computation with the new one, with a natural image, instead of a synthetic one.

In figure 14 we can see the natural image that is used in this experiment, with Gaussian noise of $\sigma = 3$ (left) and a rotated version of the noisy image.

In figure 15 we can see the noise curves for the original method.

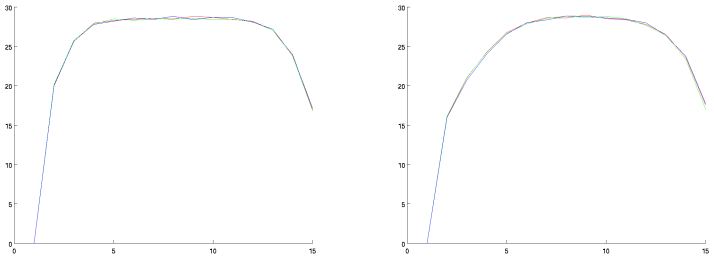


Figure 13: Noise curves for the rotated pattern with the original method (left) and the new one (right).

Figure 14: Noisy natural image with $\sigma = 3$ (left) and a rotated version (right).

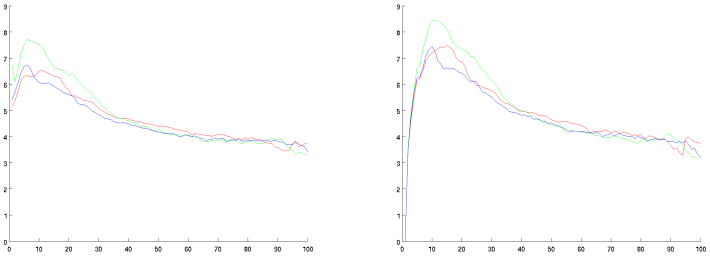


Figure 15: Noise curves computed with the original method. On the right, the rotated image curve noise.

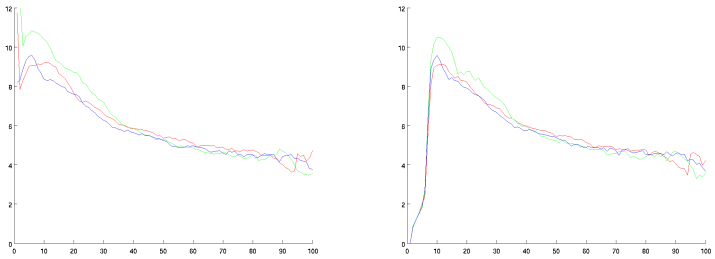


Figure 16: Noise curves computed with the new method. On the right, the rotated image curve noise.

In figure 16, we can see the noise curves computed with the new method.

In figure 17 we can see the noise curve of the original natural image, without any added noise. The image itself contains some noise already.

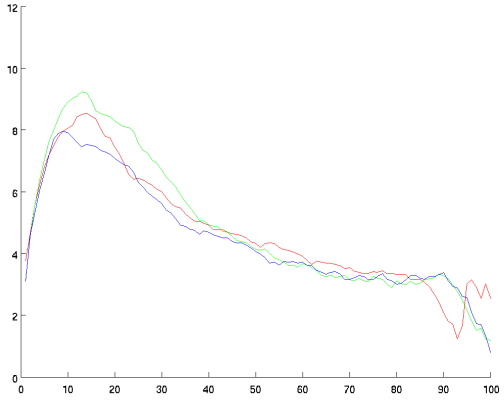


Figure 17: Original image noise curve (without added noise).

Although both methods give similar curves for the non-rotated and rotated noisy images, with the new method the curves are more similar between them, that is, more robust to rotation.

5.3 Method noise

A difference between the original image and its filtered version shows the *noise* removed by the algorithm. This procedure has been introduced recently in [24] and this difference

or residue is called *method noise*. In principle the method noise should look like a noise. Otherwise, the method noise can be filtered again and its deterministic part turned back to the image. Recent denoising methods adopted this recursive strategy to recover image information lost in method noise [26, 27]. When the standard deviation of the noise is higher than the feature contrast a visual exploration of the method noise is not reliable. Image features can be masked in method noise. Thus the evaluation of a denoising method should not rely on experiments where a white noise with standard deviation larger than 5 has been added to the original. The best way is actually not to add noise at all.

Definition 1 (method noise) *Let u be a (not necessarily noisy) image and D_h a denoising operator depending on h. The* method noise of u *is the image difference*

$$n(D_h, u) = u - D_h(u). \tag{7}$$

Principle 1 *For every denoising algorithm, the method noise must be zero if the image contains no noise and should be in general an image of independent zero-mean random variables.*

The gaussian filter method noise highlights all the boundaries and corners of the image. Averages are performed on a radial neighborhood and therefore do not adapt to the geometrical configuration of the image. The anisotropic (median, mean curvature equation) filter averages pixels in the direction of contours and therefore tends to preserve straight edges. However, the corners are not well preserved since they move at the speed of their high curvature. The iteration of the median or the application of the mean curvature motion for larger times would completely modify the image and even straight edges would not be preserved. The total variation minimization [10] is praised for maintaining sharp boundaries. However, most structures are modified and even straight edges are not well preserved. This fact has received a mathematical proof in [14]. The wavelet thresholding [28] method noise is concentrated on the edges and corners. These structures lead to coefficients of large enough value but lower than the threshold and which are erroneously canceled. The method noise of the soft thresholding is not only based on the small coefficients but also on an attenuation of the large ones, leading to a general alteration of the original image.

The bilateral filter preserves the flat zones, but the edges with a low contrast have been modified. The NL-means method noise is the one which looks the more like a white

noise. When applying the algorithms to the non noisy image, the removed features are more noticeable. The corners of the squares can now be seen in the NL-means method noise. These are the only features with a reduced amount of similar samples, since for every corner there are only three similar corners in the image.

6 Signal dependent noise in synthetic test pattern

Natural images are obtained by captors that add noise to the image on a non-uniform way. There's no way to get rid of the noise produced by the operation of the electronic devices that process the signal. Even if infinite resolution were possible, the emission of light is not a continuous process, but the event of emitting a photon by a non-black body can be modelled as a Poisson random variable.

The standard deviation of the noise added to the image depends on the signal. The more intensity has the signal locally, the more standard deviation has the added noise.

To show an example of signal dependant noise, we will use the same test pattern used before (figure 9), and we will add noise with the command *fnoise_ var_ afine patternNoiseTest.tif noiseDep.tif*. This command will produce a new file *noiseDep.tif* with Gaussian noise whose standard deviation depends on the signal. The program adds Gaussian noise of standard deviation $a + bx$. In this experiment $a = 8$, $b = 2$ and x is the original image.

In figure 18 we can see the test pattern with signal-dependent noise added.

In figure 19 we can see the noise curve of the test pattern image with the signal-dependent noise added.

The evolution of the standard deviation is almost linear with the intensity of the pixels (compare with figure 11). To show this fact was the aim of this experiment.

7 The original NL-means algorithm

Given a discrete noisy image $v = \{v(i) \mid i \in I\}$, the estimated value $NL[v](i)$, for a pixel i, is computed as a weighted average of all the pixels in the image,

$$NL[v](i) = \sum_{j \in I} w(i,j)v(j),$$

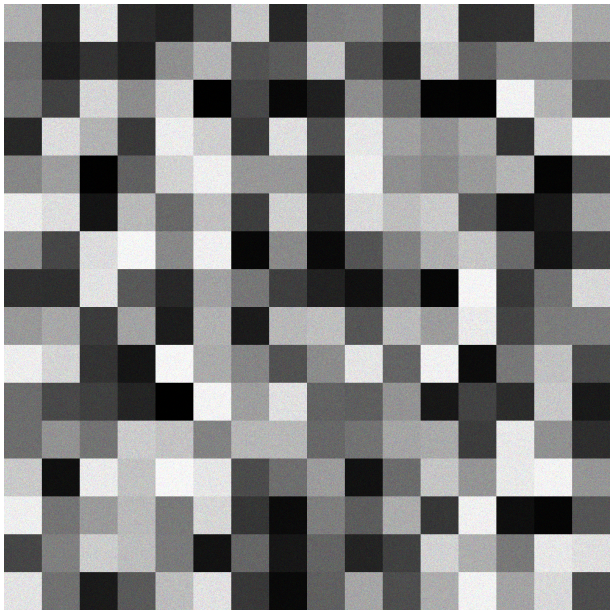


Figure 18: Signal dependent noise with $a = 8$, $b = 2$ added to the test pattern.

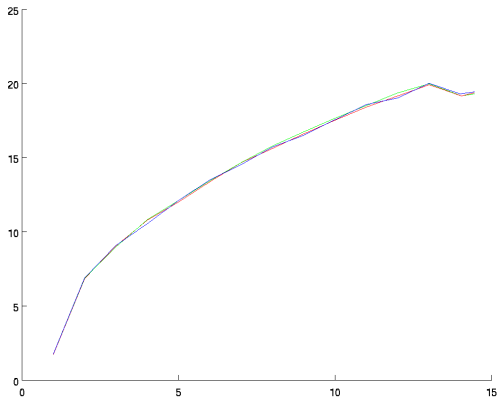


Figure 19: Curve noise for the signal-dependent noise.

where the family of weights $\{w(i,j)\}_j$ depend on the similarity between the pixels i and j, and satisfy the usual conditions $0 \leq w(i,j) \leq 1$ and $\sum_j w(i,j) = 1$.

The similarity between two pixels i and j depends on the similarity of the intensity gray level vectors $v(\mathcal{N}_i)$ and $v(\mathcal{N}_j)$, where $\mathcal{N}_k$ denotes a square neighborhood of fixed size and centered at a pixel k. This similarity is measured as a decreasing function of the weighted Euclidean distance, $\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,a}$, where $a > 0$ is the standard deviation of the Gaussian kernel. The application of the Euclidean distance to the noisy neighbors raises the following equality

$$E||v(\mathcal{N}_i) - v(\mathcal{N}_j)||^2_{2,a} = ||u(\mathcal{N}_i) - u(\mathcal{N}_j)||^2_{2,a} + 2\sigma^2.$$

This equality shows the robustness of the algorithm since in expectation the Euclidean distance conserves the order of similarity between pixels.

The pixels with a similar gray level neighborhood to $v(\mathcal{N}_i)$ have larger weights in the average. These weights are defined as,

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{||v(\mathcal{N}_i) - v(\mathcal{N}_j)||^2_{2,a}}{h^2}},$$

where $Z(i)$ is the normalizing constant

$$Z(i) = \sum_j e^{-\frac{||v(\mathcal{N}_i) - v(\mathcal{N}_i)||^2_{2,a}}{h^2}}$$

and the parameter h acts as a degree of filtering. It controls the decay of the exponential function and therefore the decay of the weights as a function of the Euclidean distances.

The NL-means not only compares the gray level in a single point but the the geometrical configuration in a whole neighborhood. This fact allows a more robust comparison than neighborhood filters.

8 Uniform Gaussian noise in natural images

In this section, we show some results of the NL-means algorithm when denoising natural images with uniform noise.

The first issue that must be solved before adding noise and removing it from the images is that we need images with the lowest noise level possible.

To get them, we averaged every four pixels of high resolution and high quality images to get a new image of half resolution than the original, but in which the noise level is also half the original. This property holds only if the noise is white (i.i.d. variables). If not, the noise level may not reduce to 50%. The size of the images was enough to allow multiple subscales. The last subscale itself presents low values in its noise curve.

On a early approach, it was tried to reduce high-resolution images computing the mean of every four pixel values to create every pixel in the reduced image. In theory, this produces a new image whose noise level is half the original one, but only if the noise is white.

8.1 Adding uniform Gaussian noise

After obtaining images by computing the mean of every four pixels and subscaling as explained before, their noise curves contain low values and are quite uniform, so they are adequate for the tests of adding noise.

With the command *fnoise* it was added uniform Gaussian noise of $\sigma = 2.5$.

In figure 20 we can see the noise curve after adding the noise to the image *blossom*, and in figure 21 the same for image *espresso*.

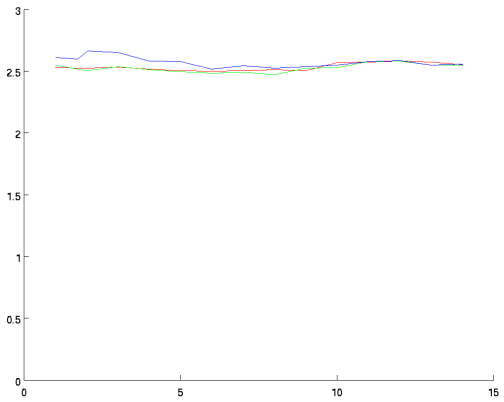


Figure 20: Noise curve after adding uniform Gaussian noise with $\sigma = 2.5$ to *blossom*.

As we can see in figures 20 and 21, the noise estimation is correct, because it is uniform and with the expected value of σ (2.5).

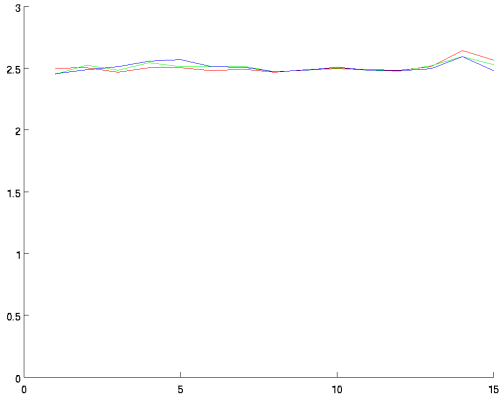


Figure 21: Noise curve after adding uniform Gaussian noise with $\sigma = 2.5$ to *espresso*.

8.2 Removing uniform Gaussian noise

The NL-means is an algorithm that has been proven to be very successful to remove noise from images, but only when the noise distribution is uniform.

In the previous section, that kind of noise was added to the images. Here we will show the denoised images and their noise curves, once denoised by the NL-means algorithm.

The added noise had $\sigma = 2.5$. For the image *blossom*, the NL-means algorithm estimated $\sigma = 2.364879$ and for *espresso* $\sigma = 2.489400$. The estimation of the noise was more accurate for *espresso*.

In figure 22 we can see the denoised versions of *blossom* and *espresso* as well their noise curves. Both noise levels are quite small after the denoising by the NL-means algorith. For *blossom* the standard deviation is always below $\sigma = 0.85$ (most of the values lie at $\sigma = 0.5$ however) and for *espresso* it is always below 0.60. This is a good result (compare with the noise levels of the noisy images in figures 20 and 21).

It is important to note that if the noise is Gaussian and uniform, it should not create any visible structure on the image. Formally, we have that $v = u + n$, where u is the ideal image, n is the added noise and v the noisy image. A denoising algorithm estimates the noise from v and the generates a new image $u^* = v - n^*$ such as the estimated noise n^* is removed. If we subtract $v - u^*$ we have $v - u^* = v - v + n^* = n^*$, that is, the estimated noise. This noise is call the *method noise*.

So, if both the noisy and the denoised images are subtracted, it should be obtained

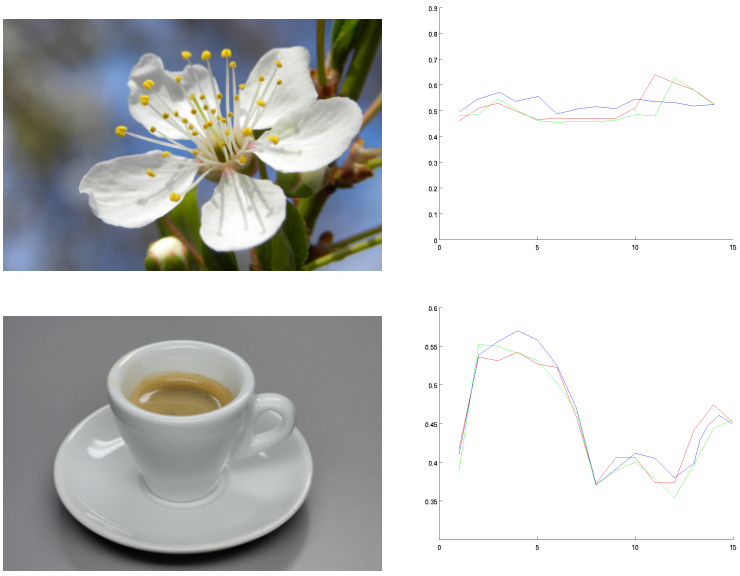


Figure 22: Denoised images and their noise curves.

a new image representing the estimated noise, on which it should not be noticeable any structure.

In figures 23 and 24 we can see the estimated noise image. Although the NL-means performed well and most of the noise was removed, it is possible still to notice the borders of the objects of the original images if looked at. The result is not perfect, but very accurate.

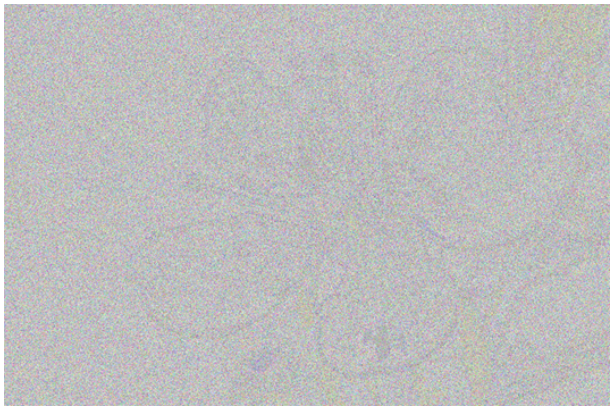


Figure 23: Image of the estimated and removed noise for *blossom* by NL-means.

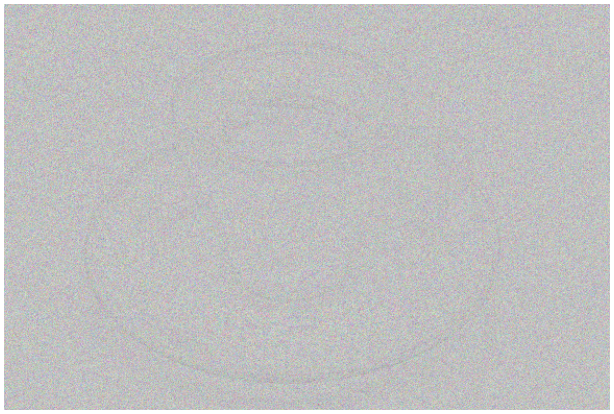


Figure 24: Image of the estimated and removed noise for *espresso* by NL-means.

9 Adding signal-dependent Gaussian noise

In section 8 it was shown that the NL-means algorithm performed very well when the noise in the image was of Gaussian distribution and uniform.

The NL-means SD is a variant of the original NL-means that deals with images whose noise is uniform, but signal-dependent. To achieve it, the algorithm first applies a transform to the image values in order to change their distribution into Gaussian, so the NL-means algorithm can deal with them. After the denoising process, the inverse transform is applied to get a clean image.

An useful transform is the Anscombe transform, that changes Poisson distribution into an approximately Gaussian one. The Anscombe transform is defined as $A : x \to 2\sqrt{x + \frac{3}{8}}$, although there are valid alternatives as $A : x \to \sqrt{x+1} + \sqrt{x}$ or $A : x \to 2\sqrt{x}$.

In this section we will show that when the image noise is signal-dependent, the NL-means SD is able to denoise the images if the noise is uniform.

To generate the noisy images the command *fnoise_ var_ affine* was used to add signal-dependent Gaussian noise of variance $n = a + bu$, where a and b are parameters for the standard deviation and u is the original image.

In figures 26 and 27 we can see the images *blossom* and *espresso* contaminated with signal-dependent Gaussian noise of variance $\sigma = 5 + 10u$, and their noise curves (up). Also the denoised images by the NL-means SD algorithm and their noise curves (down).

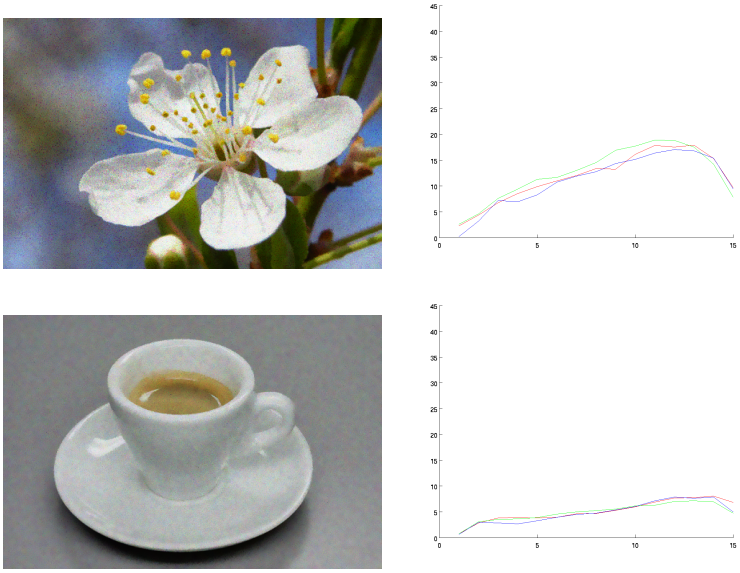


Figure 25: Denoised with SD NL-means *blossom* and *espresso* (signal-dependent noise) and their noise curves.

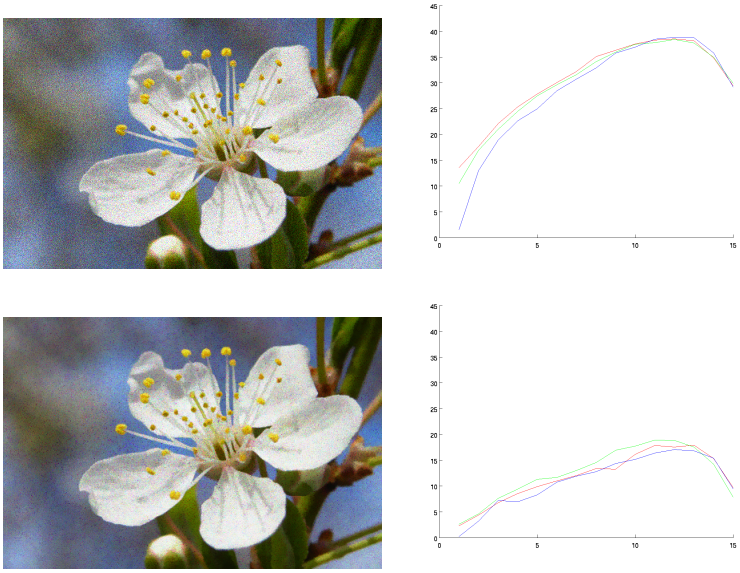


Figure 26: Noisy *blossom* and its noise curve (up). Denoised with NL-means SD and noise curve (down).

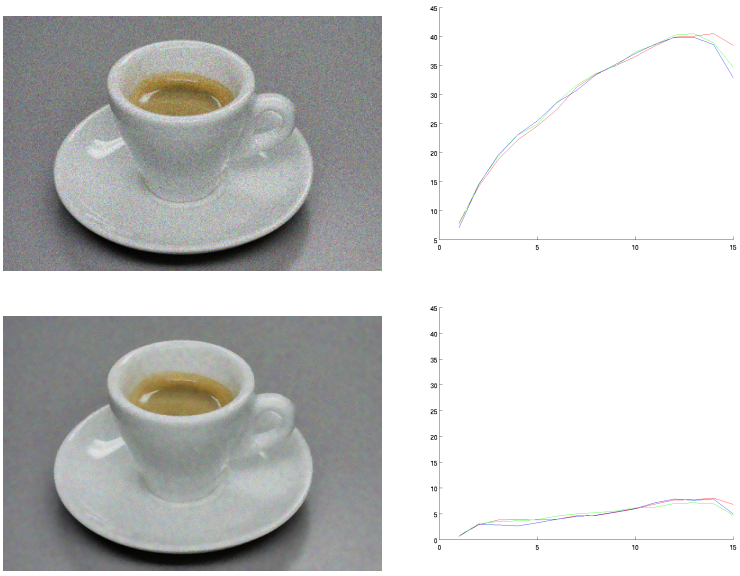


Figure 27: Noisy *espresso* and its noise curve (up). Denoised with NL-means SD and noise curve (down).

9.0.1 Noise reduction over subscales

There is a important property that holds when the noise added is uniform: the noise standard deviation reduces to 50% when the noisy image is subscaled to half its size. If the subscaled image is subscaled again, the noise level also reduces to half. This reductions continue until the scale is so small that the structures of the image are lost. If the images are still subscaled after that point, the noise level increases instead of increasing.

To show this property, in figure 28 we can see on top a noisy version of the *blossom* image (2668×1768) with noise of variance $\sigma = 30u$, and its noise curve. On the middle, the first subscale (1334×884) and its noise curve. On bottom, the second subscale (667×442) and its noise curve.

This experiment checks empirically that the property holds when the noise is uniform, even if signal-dependent.

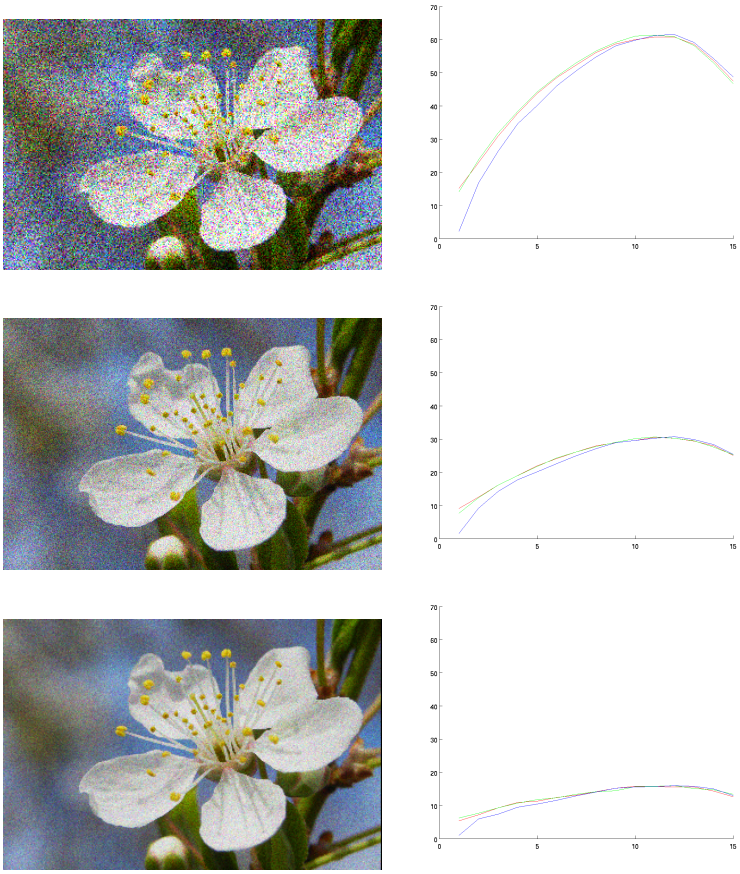


Figure 28: Different subscales of noisy image *blossom* with signal-dependent noise with variance $\sigma = 30u$ added and their noise curves.

10 The new NL-means multiscale algorithm

In this section, a new algorithm based on the NL-means, is presented. As explained before, the original NL-means is useful when the image contains uniform noise or even when the noise-is signal dependent.

When the noise is not uniform, the original NL-means algorithm fails to estimate right the amount of noise and therefore the denoising result is not good, in general.

The objective of my stage at CMLA was not only to study and experiment with noise algorithms, but also to find the way to improve the NL-means algorithm so it was able to denoise images with non-uniform noise.

As explained and showed in section 9.0.1, the noise variance becomes reduces to 50% when the image is subscaled by half. That is, if the image size is $x \times y$ and the variance of the noise is σ, then a subscaled $\frac{x}{y} \times \frac{y}{2}$ version will contain a noise of variance $\sigma/2$. A subscale is create using the procedure proposed in section 8, i.e. averaging every four pixels and substituting those four pixels by their mean in the subscaled version.

If the noise is not uniform, there not exist any reason for which the noise should reduce by half, but is reduced in some quantity that depends on the distribution of the random variable that models the noise.

To simulate the addition of non uniform noise, it was coded a new tool (*fnoise_ non_ uniform*) that convolves signal-dependent noise with a Gaussian kernel. The standard deviation for the SD noise (g) and for the Gaussian kernel (h) are specified as parameters. Convolving the noise with a Gaussian kernel results on a noise with a non-uniform distribution.

In image 29 we added non-uniform noise with $g = 30$ and $h = 1.4$. In image 30 we can see some details in a part of the image, revealing what the noise looks like.

In this case, we can see that convolving the uniform noise with a Gaussian kernel with that parameters add color spots over the image. The bigger is the standard deviation of the Gaussian kernel, the bigger are the spots in the spatial scale.

This is the reason why nor NL-means neither NL-means SD are able to denoise images with non-uniform noise. Even if the patch size in the NL-means algorithm is smaller than the size of the spot, the algorithm is likely to "clean" the spot (i.e., to remove uniform noise over it), rather than to remove the spot from the image.

In a multiscale decomposition, the lower scale reduce the size of the noise spots by

Figure 29: Image with non-uniform noise of $\sigma = 30$ and $h = 1.4$.

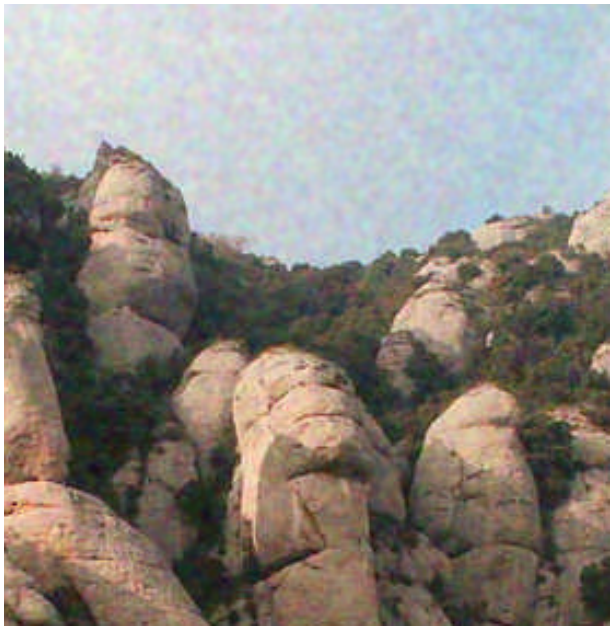


Figure 30: Detail of image with non-uniform noise of $\sigma = 30$ and $h = 1.4$.

half, so this it should be possible to remove those spots in some lower scale.

In this section, a multiscale modification is proposed in order to present a new NL-means algorithm that is able to deal with non-uniform noise.

The algorithm does the following:

1. Check if at least the image size allows to subscale it three times. I.e., its dimensions are multiples of 8. If not, black columns or rows are added. This ensures that it will processed at least three subscales in the image.
2. Create a *detail mask* for the input image.
3. Consider as many as subscales as possible. Each subscale is made by averaging every four pixels in the upper scale. If the width or height of a subscale are not multiple of two, no more subscales are generated.
4. Obtain the details of every scale. We define $Z(u)$ as the inverse operation of the subscale, that is, quadruplicating every pixel to create a new zoom in $\times 2$ image. If we denote s_i as the subscale at level i, the details D_i are obtained as $D_i = s_{i-1} - si$. Level 0 is the first subscale of the image.
5. For every scale of the image...
6. Denoise current scale with the NL-means SD algorithm.
7. Superscale (quadruplicate pixels) of current scale and add to it the details. If the current scale is the upper scale (output), then add the details according to the detail mask.
8. Clean image of previous step with the NL-means SD algorithm.
9. Replace upper scale with image in previous step.
10. ...repeat until all scales have been processed.

The objective of the detail mask is to be able to separate (in the upper scale) noise from real structures in the noisy image. If a zone is flat, it is more convenient to fetch the pixels from the lower scale (before superscaling it) because the noise level will be divided by half on that region. If a zone contains a texture or a border, the details are added to the output image in order to avoid a pixelated look.

Using the detail mask improved greatly the results of the multiscale approach, because many images contain flat zones and edges represent a small (but important) part of the image structure. The detail mask tries to preserve the details, edges and textures, while taking data from lower scales in flat zones to reduce the noise level.

The detail mask is computed using a 2×2 patch that is moved over all possible locations in the input image. The standard deviation of the patch is computed, normalized to be between 0 and 1 and its value assigned for every location. Once the value of the standard deviation for all the pixels in the original image is known, it is compared to a threshold to determine if the pixel is a detail (value over the threshold) or belongs to a flat zone (value below the threshold).

To determine the threshold, many values are tested. For every value, a mask is computed. If the mask contains more than a 25% of pixels labelled as details, a larger threshold is tried (it is incremented by 0.01). The value of 25% has been chosen experimentally by testing various natural images.

To explain the multiscale algorithm, every scale will be analyzed in detail.

To the original image (figure 31) it was added signal-dependent noise of standard deviation $g = 30$, convolved with a Gaussian kernel of $h = 1.4$. This convolution results in non-uniform noise. This noise was added to the original image (size 1136×852).

Figure 31: Original image.

In figure 32 we can see the noisy image.

In figure 33 we can see a detail of the noisy image that allows to observe the spots

Figure 32: Image with non-uniform noise ($g = 30$, $h = 1.4$).

caused by the noise.

The size of the image was changed to $1136x856$ by the algorithm, in order to process at least three subscales.

In image 34 we can observe the detail mask computed for this input image. The algorithm chose the value 0.08 as the threshold. As we can see, this value is adequate for this input image, because it detects the flat zones (the sky and the facade of the buildings) and designates as details the borders of the structures.

In image 35 we can see the result of the NL-means SD algorithm when trying to remove the non-uniform noise form the image, and the noise curve. The noise is still high noticeable.

In image 36 we can see a detail of the NL-means SD denoising result.

10.1 Scale 2

In figure 37 we can observe scale 2 (142×107) of the original input, its noise curve, the denoised version zoomed 2X with details added in order to substitute the upper scale, and the noise curve of the denoised image.

We can observe that the noise level is dramatically reduced on the subscale 2.

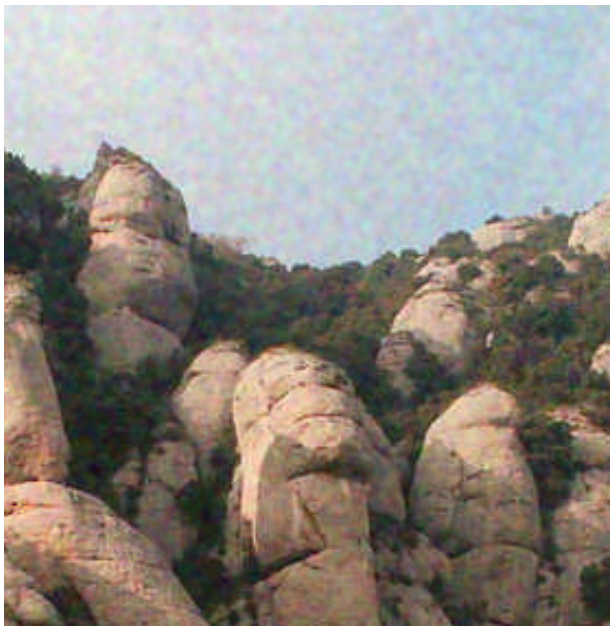


Figure 33: Detail of the noisy image.

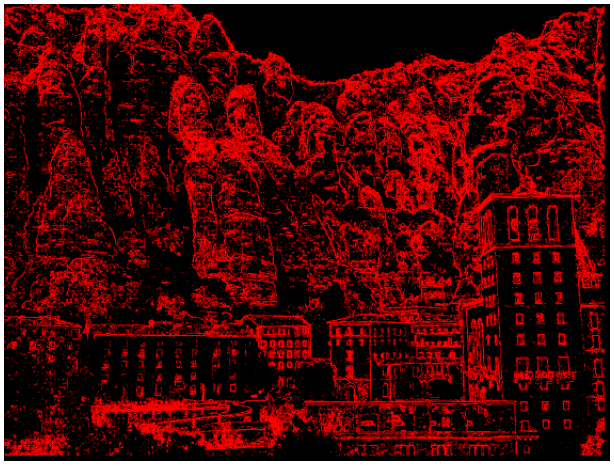


Figure 34: Detail mask of the input image.

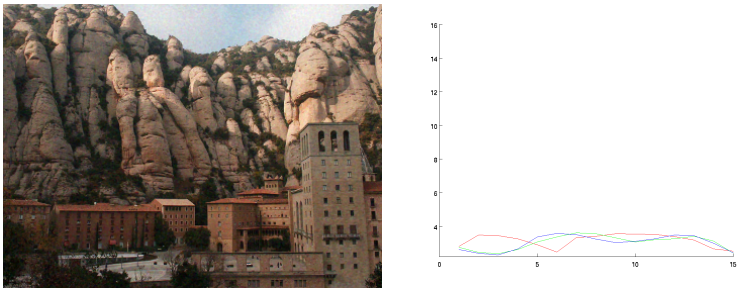


Figure 35: NL-means SD denoising result.

Figure 36: Detail of the NL-means SD denoising result.

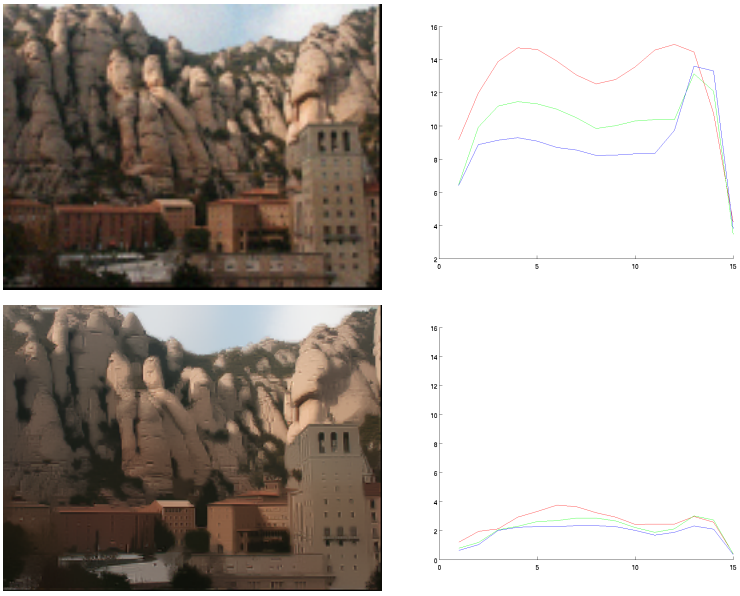


Figure 37: Scale 2 (up) and zoomed X2 version denoised (down).

10.2 Scale 1

In figure 38 we can observe scale 1 (284×214) of the original input, its noise curve, the denoised version zoomed 2X with details added in order to substitute the upper scale, and the noise curve of the denoised image.

10.3 Scale 0

In figure 39 we can observe scale 0 (568×428) of the original input, its noise curve, the denoised version zoomed 2X with details added in order to substitute the upper scale, and the noise curve of the denoised image.

10.4 Upper scale

This scale corresponds to the upper scale, that is the denoised output image with the details added from the lower scale using the detail mask. In image 40 we can see the denoising result.

In image 41 we can compare a detail of a zone of the noisy image, the denoise result using the NL-means SD algoritm and finally with the NL-means multiscale algorithm.

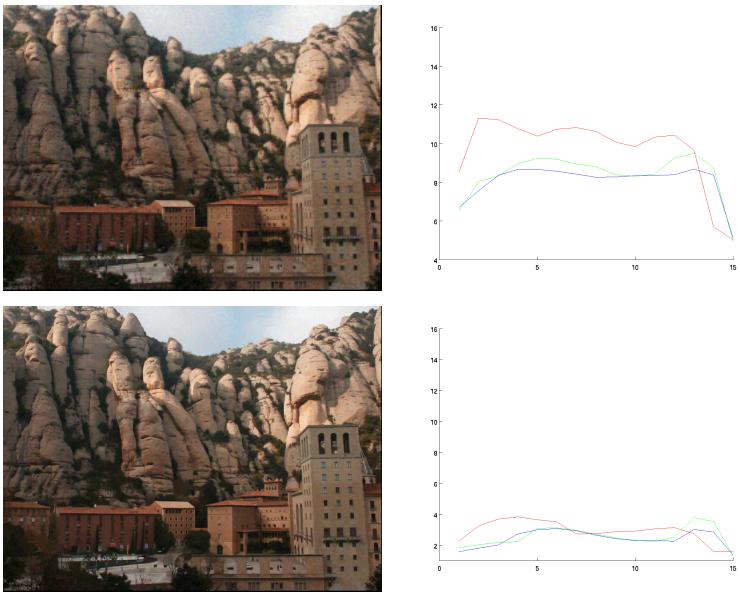


Figure 38: Scale 1 (up) and zoomed X2 version denoised (down).

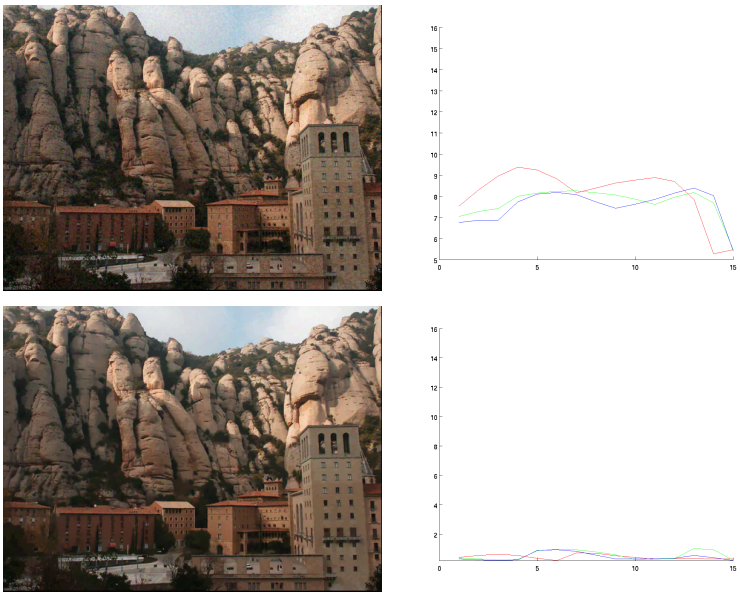


Figure 39: Scale 0 (up) and zoomed X2 version denoised (down).

Figure 40: NL-means multiscale output.

Figure 41: Noisy image (left), NL-means SD (right) and NL-means multiscale output details.

In figure 42 we can see the method noise of the algorithm. Although some structures are visible, it is because the detail mask, that prevents that the noise is removed on the edges to keep the details of the image. We can see that, out of the edges, no structure is visible and the algorithm has extracted the color spots (the non-uniform noise added) from the image.

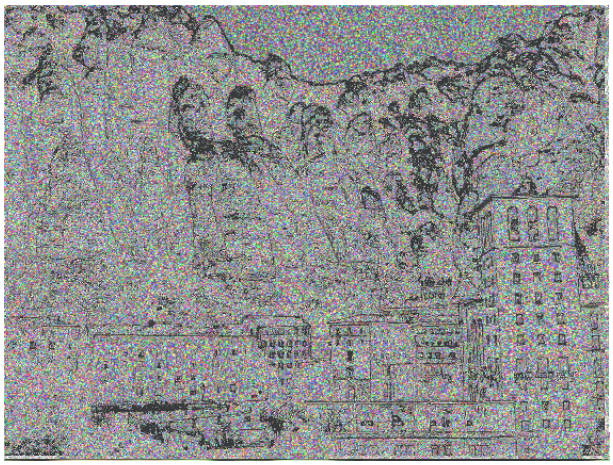


Figure 42: NL-means difference image.

11 More results of the NL-means multiscale algorithm with real noise

In this section more results of the NL-means multiscale algorithm are showed. The following images are examples of noisy images that users uploaded to the IPOL web page in order to test NL-means.

11.1 Image "man"

In figure 43 we can see the input image, the output given by the NL-means SD algorith and the output by the new NL-mean multiscale algorithm. In figure 44 the noise curves for these three images. In figure 45 the difference image and in figure 46 the detail mask.

Figure 43: Input image, NL-means SD and NL-means multiscale output for "man".

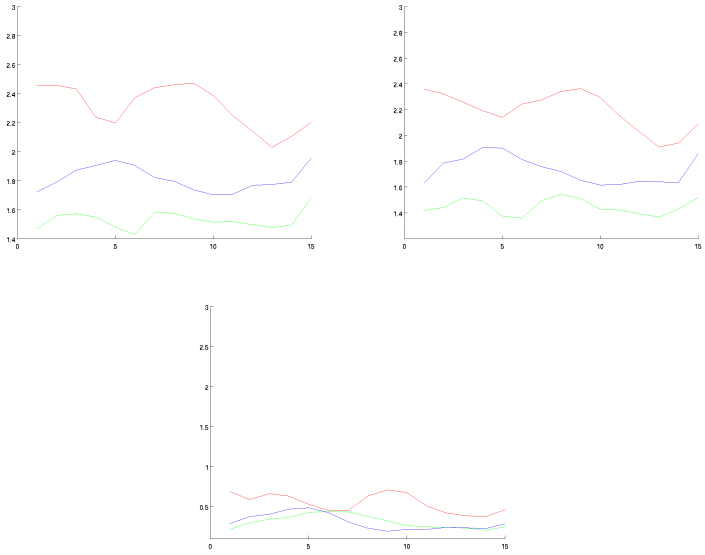


Figure 44: Input, NL-means SD and NL-means multiscale output noise curves for "man".

Figure 45: NL-means difference image for "man".

Figure 46: Detail mask for "man".

11.2 Image "war"

In figure 47 we can see the input image, the output given by the NL-means SD algorith and the output by the new NL-mean multiscale algorithm. In figure 48 the noise curves for these three images. In figure 49 the difference image and in figure 50 the detail mask.

11.3 Image "factory"

In figure 51 we can see the input image, the output given by the NL-means SD algorith and the output by the new NL-mean multiscale algorithm. In figure 52 the noise curves for these three images. In figure 53 the difference image and in figure 54 the detail mask.

11.4 Image "singer"

In figure 55 we can see the input image, the output given by the NL-means SD algorith and the output by the new NL-mean multiscale algorithm. In figure 56 the noise curves for these three images. In figure 57 the difference image and in figure 58 the detail mask.

11.5 Image "kitchen"

In figure 59 we can see the input image, the output given by the NL-means SD algorith and the output by the new NL-mean multiscale algorithm. In figure 60 the noise curves for these three images. In figure 61 the difference image and in figure 62 the detail mask.

Figure 47: Input image, NL-means SD and NL-means multiscale output for "war".

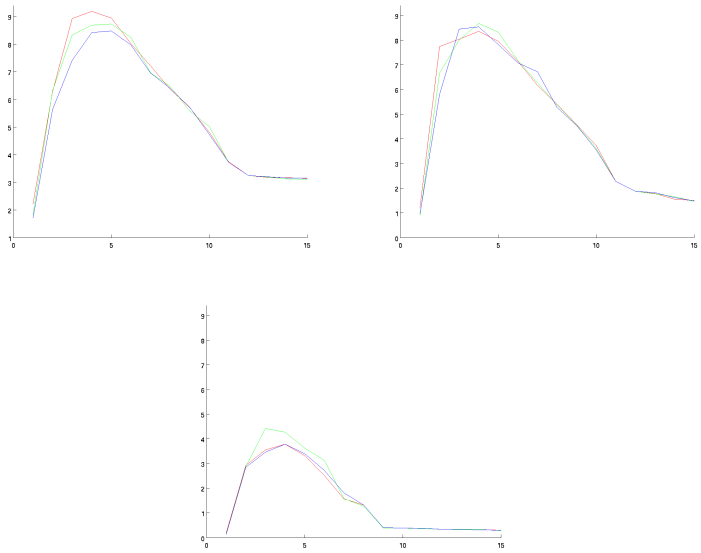


Figure 48: Input, NL-means SD and NL-means multiscale output noise curves for "war".

12 Conclusion

During my stage at CMLA I could explore existing denoising methods and specially the general neighborhood filters (local neighborhood filters and non-local averaging). As a particular case of the non-local averaging I studied in detail the NL-means [16].

Using as a base the algorithms coded by A. Buades with the C++ language, these codes were modified to test new improvements of the NL-means denoising.

The first modification was to use a multiscale approach. The noise level is reduced by half if the noise is uniform in a subscale. If the noise is not uniform, a noise reduction is noticeable, although it may not be divided by two, because it depends on the statistical distribution of the noise.

The multiscale approach was proven to give great results, both in the noise curves and by observation by the human eye.

The second modification was the use of a detail mask. This mask helped to split the image details into real details (edges, textures) and noise. The use of the mask gave very good results also, especially on the flat regions of the image. Over these zones, the noise is almost completely removed.

These modifications mean an important improvement of the NL-means algorithm, and

Figure 49: NL-means difference image for "war".

Figure 50: Detail mask for "war".

the next step is its publication on the IPOL journal.

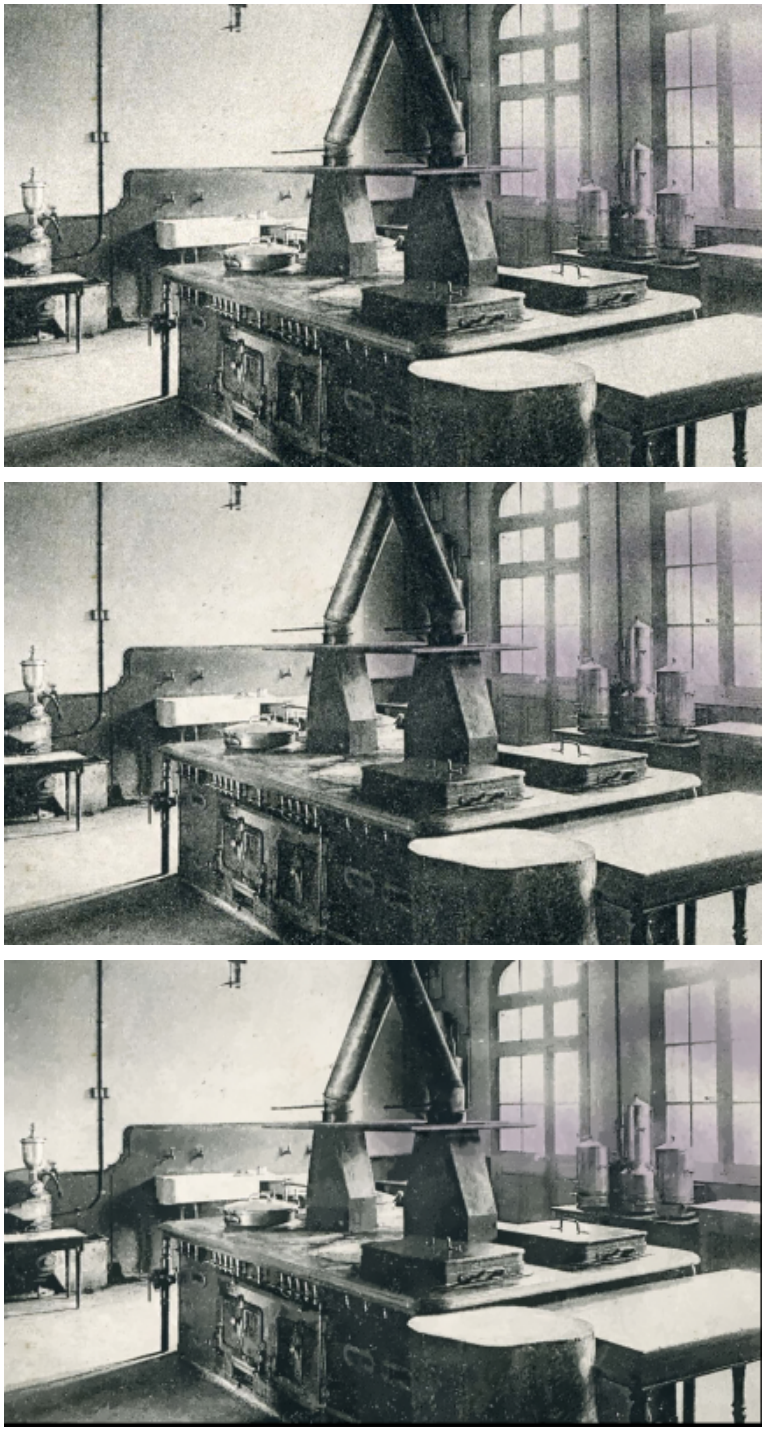


Figure 51: Input image, NL-means SD and NL-means multiscale output for "factory".

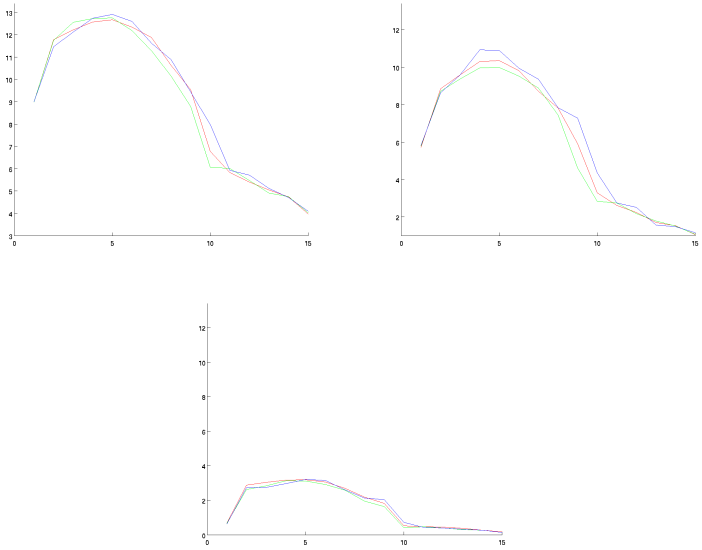


Figure 52: Input, NL-means SD and NL-means multiscale output noise curves for "factory".

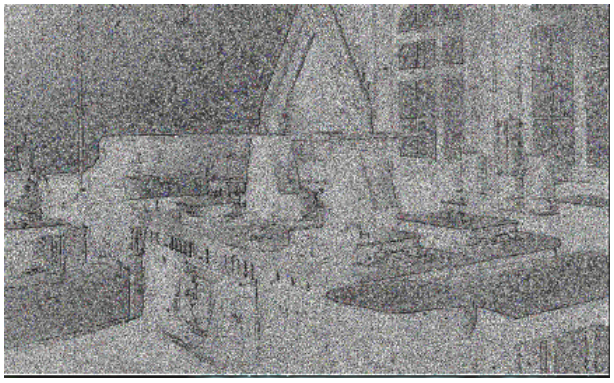


Figure 53: NL-means difference image for "factory".

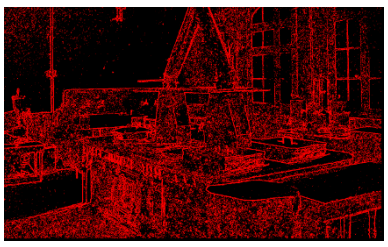


Figure 54: Detail mask for "factory".

Figure 55: Input image, NL-means SD and NL-means multiscale output for "singer".

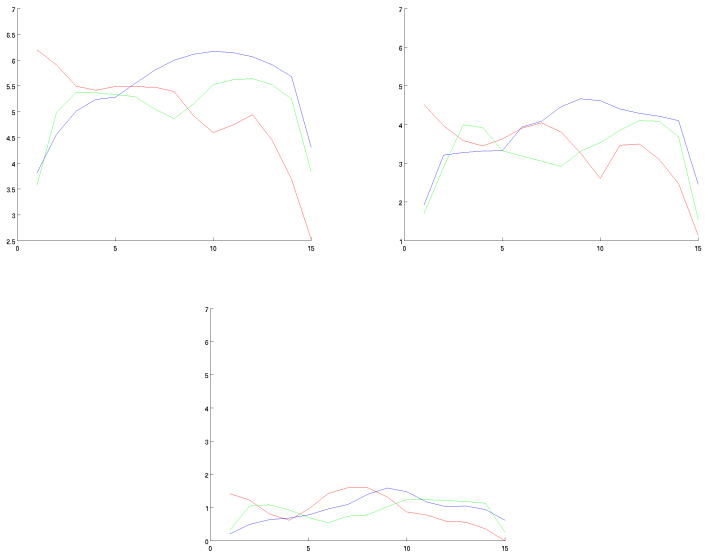


Figure 56: Input, NL-means SD and NL-means multiscale output noise curves for "singer".

Figure 57: NL-means difference image for "singer".

Figure 58: Detail mask for "singer".

Figure 59: Input image, NL-means SD and NL-means multiscale output for "kitchen".

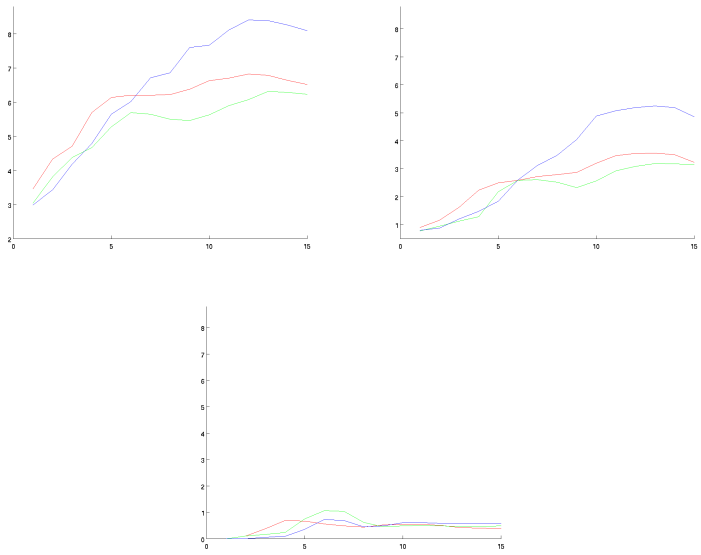


Figure 60: Input, NL-means SD and NL-means multiscale output noise curves for "kitchen".

Figure 61: NL-means difference image for "kitchen".

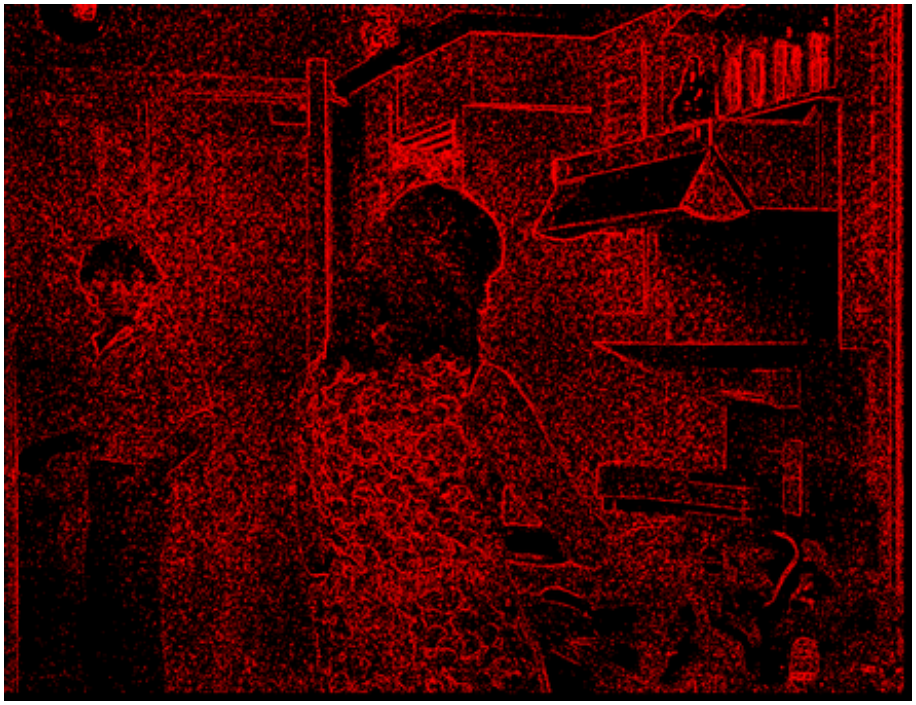


Figure 62: Detail mask for "kitchen".

A Appendix. NL-means multiscale pseudocode

Here it follows a pseudocode description for the NL-means multiscale algorithm. The complete C++ source code is available to the reader by request.

```
// Subscale image I half size by averaging every 4 pixels.
outImage subScale(I) {
  width = width of I
  height = height of I
  nchannels = number of channels in I

  outImage = image of half size than input

  // compute new width and new height.
  newWidth = width / 2
  newHeight = height / 2

  for every pixel (x,y) in inputImage and for every channel:
    outImage(x, y) = (I(2x,2y) + I(2x+1,2y) + I(2x,2y+1) + I(2x+1,2y+1))/4
}

// Superscale image I double size by quadruplicating every 4 pixels.
output superScale(input) {
  width = width of input
  height = height of input
  nchannels = number of channels in input

  outImage = image of double size than input

  // compute output
  for every pixel (x,y) in inputImage and for every channel:
    outImage(x, y) = I(floor(x/2), floor(y/2))
}
```

```
// Computes the standard deviation of the input image using a NxN patch.
output computeStdMask(input, threshold) {
  patch = square of NxN

  width = width of input
  height = height of input
  nchannels = number of channels in input

  for every pixel (x,y) in inputImage {
    1) fill patch with NxN values from input starting at (x,y),
      averaging the values from different channels

    2) compute the standard deviation of the patch (std)

    3) Assign std the output(x,y)

    4) Normalize output so SUM(output) = 1

    5) Set value 255 to those pixels in output that are over the threshold,
       and zero otherwise.
  }
}
.
// Get the details substracting the data a low resolution
// image from a high resolution image
details getDetails(highRes, lowRes) {
  details = highRes - lowRes
}

// Adds given details to the specified image using a mask
output addDetails(details, image, mask) {
  width = width of image
  height = height of image
  nchannels = number of channels in image
```

```
  for every pixel (x,y) in inputImage and for every channel:
    if mask(x,y) != 0
      output(x,y) = image(x,y) + details(x,y)
    else
      output(x,y) = image(x,y)
  }
}

// Returns the number of pixels labelled as "detail" in the given matrix.
countDetails(I) {
  return #pixels P in I for which P != 0
}

// Next number divisible by div
nextDivisible(value, div)
  if value is divisible by div, return value,
    if not, return next (higher) value divisible by div
}

BEGIN NL-MEANS-MULTISCALE
  // 1) Load input
  Load input image

  // 2) Get size and number of channels in input
  width = width of image
  height = height of image
  nchannels = number of channels in image

  // 3) Set initial values for the parameters
  patchNx = 2
  patchNy = 2
  thStd = 0.05
```

```
// 4) Obtain size that allows at least 3 scales
if width or height not divisible by 8 {
  width = nextDivisible(width, 8)
  height = nextDivisible(height, 8)

  enlarge image to the new size
}

stdMask = new image of the same width, height and number of channels

// 5) Look for a threshold that leaves no more than 25% detail pixels
detailPercert = 100
while detailPercert > 25 {
  stdMask = computeStdMask(input, patchNx, patchNy, thStd);
  detailPercent = (100*countDetails(stdMask)) / (width*height);
  thStd = thStd + 0.01;
}

// 6) Denoise lower scales and substitute high scales with then
for every possible scale sub of image (from smallest to biggest size)
  // 6.1) Get details of current scale
  details[scale] = getDetails(sub[scale-1], sub[scale])

  // 6.2) Denoise current scale
  Denoise sub[scale] with NL-means_SD and save result into cleanScale

  // 6.3) Superscale denoised scale
  superCleanScale = superScale(cleanScale)

  // 6.4) Add details to superCleanScale only if current scale is the top scale
  if current scale is top
    superCleanScale = addDetails(details, superCleanScale, stdMask)

  // 6.5) Denoise superCleanScale into cleanSuperCleanScale
```

```
    Denoise superCleanScale with NL-means_SD and save result into cleanSuperCleanScale

    // 6.6) Replace upper scale with cleanSuperCleanScale
    if current scale is top scale
      Denoise top scale with NL-means_SD and save result to disk
    else
      sub[scale-1] = cleanSuperCleanScale // Replace upper scale
END NL-MEANS-MULTISCALE
```

List of Figures

References

[1] A. Buades, Y. Lou, J.M. Morel, Z. Tang, *A Note on Multi-Image Denoising.*

[2] S. I. Olsen, *Estimation of noise in images: an evaluation.* CVGIP: Graph. Models Image Process., vol. 55, no. 4, pp. 319-323, 1993.

[3] M. Lendl, K. Rank and R. Unbehauen *Estimation of image noise variance.* Vision, Image and Signal Processing, 1999, vol. 146, pp. 80-84.

[4] M. Lindenbaum and M. Fischer and A.M. Bruckstein, *On Gabor contribution to image enhancement*, Pattern Recognition, 1994, 27, 1–8.

[5] P. Perona and J. Malik, *Scale space and edge detection using anisotropic diffusion*, IEEE Trans. Patt. Anal. Mach. Intell., 1990, 29, 845–866.

[6] L. Alvarez and P-L. Lions and J-M. Morel, *Image selective smoothing and edge detection by nonlinear diffusion (II)*, Journal of numerical analysis, 1992, 29, 845–866.

[7] L.P. Yaroslavsky, *Digital Picture Processing - An Introduction*, Springer Verlag, 1985.

[8] S.M. Smith and J.M. Brady, *Susan - a new approach to low level image processing*, International Journal of Computer Vision, 1997, 23, 1, 45–78.

[9] C. Tomasi and R. Manduchi, *Bilateral filtering for gray and color images*, Proceedings of the Sixth Internatinal Conference on Computer Vision, 1998, 839-845.

[10] L. Rudin and S. Osher and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, 1992, 60, 259–268.

[11] D. Donoho, *De-noising by soft-thresholding*, IEEE Transactions on Information Theory, 1995, 41, 613–627.

[12] R.R. Coifman and D. Donoho, *Wavelets and Statistics*, Translation-invariant denoising, 1995, 125–150.

[13] S.J. Osher and A. Sole and L.A. Vese, *Image decomposition and restoration using total variation minimization and the H^{-1} norm*, Multiscale Modeling and Simulation, 2003, 1, 3, 349–370.

[14] Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*, AMS University Lecture Series, 2002, 22.

[15] A. Efros and T. Leung, *Texture synthesis by non parametric sampling*, Proc. Int. Conf. Computer Vision, 1999, 2, 1033–1038.

[16] A. Buades and B. Coll and J.M. Morel, *On image denoising methods*, CMLA, 2004, 2004–15.

[17] M. Colleen Gino, "Noise, Noise, Noise", *http : //www.astrophys − assist.com/educate/noise/noise.htm.*

[18] S. B. Howell, "Handbook of CCD Astronomy", Cambridge University Press, 2000.

[19] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", 2nd edition, Prentice Hall, 2002.

[20] C. Liu, W.T. Freeman, R. Szeliski and S.B. Kang, "Noise estimation from a single image", CVPR 2006.

[21] J.S. Lee, "Digital image enhancement and noise filtering by use of local statistics", IEEE Trans. Patt. Anal. Machine Intell., vol. 2, pp. 165-168, 1980.

[22] J. Tukey, "Exploratory Data Analysis", Addison-Wesley, 1977.

[23] B. Merriman, J. Bence, and S. Osher, "Diffusion generated motion by mean curvature", In Proc. of the Geometry Center Workshop, 1992.

[24] A. Buades, B. Coll and J.M. Morel, "A review of image denoising methods, with a new one", Multiscale Modeling and Simulation, vol 4 (2), pp 490-530, 2005.

[25] A. Buades, B. Coll and J.M. Morel, "A non-local algorithm for image denoising", IEEE Int. Conf. on Computer Vision and Pattern Recognition, 2005.

[26] S. Osher, M. Burger, D. Goldfarb, J. Xu and W. Yin, "An iterative regularization method for total variation based image restoration", Multiscale Modelling and Simulation, vol. 4, 460-489, 2005.

[27] E. Tadmor, S. Nezzar, and L. Vese, "A multiscale image representation using hierarchical (BV, L^2) decompositions", Multiscale Modeling and Simulation, vol 2, pp.554–579, 2004.

[28] D. Donoho and I. Johnstone, "Ideal spatial adaptation via wavelet shrinkage", Biometrika, 81 pp. 425-455, 1994.

[29] S. I. Olsen. "Estimation of noise in images: an evaluation". CVGIP: Graph. Models Image Process., 55(4):319–323, 1993.

[30] : M. Lendl K. Rank and R. Unbehauen. "Estimation of image noise variance". In Vision, Image and Signal Processing, volume 146, pages 80–84, 1999.

[31] : Image Processing On-Line; http://www.ipol.im