AN IMPROVED NON-LOCAL DENOISING ALGORITHM

Bart Goossens, Hiêp Luong, Aleksandra Pižurica and Wilfried Philips

Ghent University, Department of Telecommunications and Information Processing (TELIN-IPI-IBBT)
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
bart.goossens@telin.ugent.be

ABSTRACT

Recently, the NLMeans filter has been proposed by Buades et al. for the suppression of white Gaussian noise. This filter exploits the repetitive character of structures in an image, unlike conventional denoising algorithms, which typically operate in a local neighbourhood. Even though the method is quite intuitive and potentially very powerful, the PSNR and visual results are somewhat inferior to other recent state-of-the-art non-local algorithms, like KSVD and BM-3D. In this paper, we show that the NLMeans algorithm is basically the first iteration of the Jacobi optimization algorithm for robustly estimating the noise-free image. Based on this insight, we present additional improvements to the NLMeans algorithm and also an extension to noise reduction of coloured (correlated) noise. For white noise, PSNR results show that the proposed method is very competitive with the BM-3D method, while the visual quality of our method is better due to the lower presence of artifacts. For correlated noise on the other hand, we obtain a significant improvement in denoising performance compared to recent wavelet-based techniques.

1. INTRODUCTION

Digital imaging devices inevitably produce noise, originating from the analog circuitry in these devices. Noise suppression by means of digital post-processing is often desirable, but also very challenging. In this paper, we focus on the design of a noise reduction method for stationary Gaussian noise, that preserves original image details and that has a high visual quality.

During the past decade, numerous and diverse denoising methods have been proposed to this end. Many methods, like total variation [1], bilateral filtering [2] or wavelet-based techniques [3–9] estimate the denoised pixel intensities based on the information provided in a limited surrounding neighbourhood. These methods only exploit the spatial redundancy in a local neighbourhood and are therefore referred to as *local* methods.

Recently, a number of *non-local* methods have been developed. These methods estimate every pixel intensity based on information from the *whole* image thereby exploiting the presence of similar patterns and features in an image. This relatively new class of denoising methods originates from the Non-Local Means (NLMeans), introduced by Buades at al. [10, 11]. Basically, the NLMeans filter estimates a noise-free pixel intensity as a weighted average of all pixel intensities in the image, and the weights are proportional to the similarity between the local neighbourhood of the pixel being processed and local neighbourhoods of surrounding pixels. Other *non-local* denoising methods are exemplar-based (KSVD) [12], or group similar blocks by block-matching and then apply 3D transform-domain filtering to the obtained stacks (BM-3D) [13].

The NLMeans filter, despite being intuitive and potentially very powerful, has two limitations at this moment: first, both the objective quality and visual quality are somewhat inferior to the other recent non-local techniques and second, the NLMeans filter has a complexity that is quadratic in the number of pixels in the image, which makes the technique computationally intensive and even impractical in real applications. For this reason, improvements for enhancing the visual quality and for reducing the computation time have been proposed by different researchers. Some authors investigate better similarity measures [14–16], use adaptive local neighbourhoods [17], or refine the similarity estimates in different iterations [18]. Other authors propose algorithmic acceleration techniques [16, 19–21], based for example on neighbourhood preclassification [16, 19] and FFT-based computation of the neighbourhood similarities [20].

In this paper, we show the connection between the NLMeans filter and robust estimation techniques, similar to the connection made for the bilateral filter in [12]. It turns out that the NLMeans filter is the first iteration of the Jacobi algorithm [12] (also known as the Diagonal Normalized Steepest Descent algorithm) for robustly estimating the noise-free image using the Leclerc loss function. By this observation, it becomes possible to investigate other robust cost functions that are commonly used for robust estimation. Also, this suggests that applying the NLMeans algorithm iteratively in a specific way, further decreases the cost function. Another problem noted by Buades et al. is that the NLMeans filter is not able to suppress any noise for non-repetitive neighbourhoods. In this work, we keep track of the local noise variance during NLMeans filtering and we apply a *local* post-processing

A. Pižurica is a postdoctoral researcher of the Fund for the Scientific Research in Flanders (FWO) Belgium.

filter afterwards, to remove remaining noise in regions with non-repetitive structures. Furthermore, we present an extension of the NLMeans filter to correlated noise and we also present a new acceleration technique that computes the Euclidean distance by a recursive moving average filter. The proposed modifications significantly improve the NLMeans filter, both in PSNR as in computation time, and make the filter competitive to (or even better than) recent *non-local* methods such as BM-3D of Dabov et al. [13].

The remainder of this article is as follows: in Section 2, the NLMeans algorithm is briefly presented. In Section 3, we investigate the choice of the similarity weighting functions, on probabilistic reasoning and within the robust estimation framework. In Section 4, we present our improvements to the NLMeans algorithm. We discuss how the computation time of the filter can be further reduced in Section 5. Numerical results and visual results together with a discussion, are given in Section 6. Finally, Section 7 concludes this paper.

2. NON LOCAL MEANS

Suppose an unknown signal X_i, $i = 1, ..., N$ is corrupted by an additive noise process V_i, $i = 1, ..., N$, which results in the observed signal:

$$Y_i = X_i + V_i \tag{1}$$

In this work, we stick to definitions for 1-d signals for simplicity of the notations. An extension to 2-d images is straightforward. The denoised value $\hat{X}_i$ of the pixel intensity at position j is computed as the weighted average of *all* pixels in the image:

$$\hat{X}_i = \frac{\sum_{j=1}^{N} w(i,j) Y_j}{\sum_{j=1}^{N} w(i,j)} \tag{2}$$

We will refer to this filter as the *pixel*-based NLMeans. Alternatively, a *vector* (or block)-based NLMeans filter does also exist [10]. Here overlapping blocks are used, resulting in multiple estimates for each pixel in a block. To aggregate the different estimates, an additional weighting function $b(k)$ determines the weight contribution of central pixel to its neighbour at relative position k:

$$\hat{X}_i = \frac{\sum_{j=1}^{N} \sum_{k=-K}^{K} b(k) w(i+k, j+k) Y_j}{\sum_{j=1}^{N} \sum_{k=-K}^{K} b(k) w(i+k, j+k)} \tag{3}$$

The weights $w(i,j)$ depend on the similarity between the neighbourhoods centered at positions i and j. In this paper, neighbourhoods of fixed predefined size are used (e.g. 9×9). Let us denote $\mathbf{x}_i = (X_{i-K}, \ldots, X_{i+K})$, $\mathbf{y}_i = (Y_{i-K}, \ldots, Y_{i+K})$ and $\mathbf{v}_i = (V_{i-K}, \ldots, V_{i+K})$ as vectors containing pixel intensities of the local neighbourhood centered at position i (where for example symmetrical boundary reflection is used at the signal/image boundaries). In the original NLMeans algorithm [10, 11], the weighting function is defined as follows:

$$w(i,j) = \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{2h^2}\right) \tag{4}$$

with $\|\mathbf{y}_i - \mathbf{y}_j\| = \sqrt{(\mathbf{y}_i - \mathbf{y}_j)^T (\mathbf{y}_i - \mathbf{y}_j)}$ the Euclidean distance between the vectors $\mathbf{y}_i$ and $\mathbf{y}_j$. The bilateral filter [2] is closely related to the NLMeans filter. For the bilateral filter, the weighting function is given by:

$$w(i,j) = \exp\left(-\frac{(Y_i - Y_j)^2}{2h^2}\right) \exp\left(-\frac{(i-j)^2}{d^2}\right) \tag{5}$$

where the first factor (called photometric distance) is inversely proportional to the Euclidean distance between the pixel intensities Y_i and Y_j and the second factor (called geometric distance), measures the Euclidean distance between the center sample i and the j-th sample. The NLMeans weighting function can be interpreted as a vector-extension of the bilateral filter weighting function, omitting a geometric distance factor. In [2, 11], the weighting functions as given in equations (5) and (4) are defined on intuition. However it is not guaranteed that these choices are optimal for a given criterion. In [12], it is shown that equation (5) emerges from optimizing a robust M-function. In the next Section, we will derive expressions for $w(i,j)$ based on probabilistic reasoning and we will further extend the result from [12] to the NLMeans filter.

3. COMPUTING SIMILARITY WEIGHTS

One of the key elements for designing a high performance NLMeans filter, is the selection of the similarity weights. Clearly, the similarity weights should be adapted to the image in order to achieve maximal improvement. Because only a noisy version of the image is available, the weights should also take the noise properties into account; the presence of noise generally degrades the estimate of the similarity between two neighbourhoods in the image. The question now becomes: how to determine the similarity weights in an *"optimal"* sense for a given criterion? Luckily, estimation theory can give an answer to this problem.

3.1. Best Linear Unbiased Estimator (BLUE)

We now only consider the estimation of the noise-free patch $\mathbf{x}_i$ as a function of surrounding noisy patches . To take the correlation between different $\mathbf{x}_i$ into account, we model the residual $\mathbf{r}_{i,j}$ between patches (centered at position i and j) as:

$$\mathbf{x}_i = \mathbf{x}_j + \mathbf{r}_{i,j} \tag{6}$$

with $\mathbf{r}_{i,j}$ a zero-mean Gaussian random variable with position-dependent covariance matrix $\sigma_{i,j}^2 \mathbf{I}$ (the disadvantages of this choice will be discussed further on) and with $\sigma_{i,i} = 0$. Due to the additivity of the noise, we have:

$$\mathbf{y}_i = \mathbf{x}_j + \mathbf{r}_{i,j} + \mathbf{v}_i \tag{7}$$

For white Gaussian noise, $\mathbf{v}_i \sim \mathrm{N}(\mathbf{0}, \sigma_w^2 \mathbf{I})$. The Best Linear Unbiased Estimator (BLUE) for this problem is given

by:

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \sum_{j=1}^{N} -\log f_{\mathbf{y}}(\mathbf{y}_j; \mathbf{x}) \quad (8)$$

$$= \arg\min_{\mathbf{x}} \sum_{j=1}^{N} \frac{1}{2} \left\| \frac{\mathbf{y}_j - \mathbf{x}}{\sqrt{\sigma_w^2 + \sigma_{i,j}^2}} \right\|^2 \quad (9)$$

$$= \frac{\sum_{j=1}^{N} \frac{1}{\sigma_w^2 + \sigma_{i,j}^2} \mathbf{y}_j}{\sum_{j=1}^{N} \frac{1}{\sigma_w^2 + \sigma_{i,j}^2}} \quad (10)$$

which here also corresponds to the maximum likelihood (ML) estimator. The variances $\sigma_{i,j}^2$, $i \neq j$ are unknown and need also to be estimated. Noting that from:

$$(\mathbf{y}_j - \mathbf{y}_i) = (\mathbf{r}_{j,j} - \mathbf{r}_{i,j}) + (\mathbf{v}_j - \mathbf{v}_i)$$

it follows that $\mathrm{Var}[\mathbf{y}_j - \mathbf{y}_i] = \left(2\sigma_w^2 + 2\sigma_{i,j}^2\right)\mathbf{I}$, the following estimate can be obtained:

$$\widehat{\sigma_{i,j}^2} = \max\left(0, \frac{1}{2(2K+1)} \|\mathbf{y}_i - \mathbf{y}_j\|^2 - \sigma_w^2\right) \quad (11)$$

This solution suggests the use of the following weight matrix:

$$w(i,j) = \begin{cases} \frac{4K+2}{\|\mathbf{y}_i - \mathbf{y}_j\|^2} & \frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{4K+2} > \sigma_w^2 \\ \frac{1}{\sigma_w^2} & \frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{4K+2} \leq \sigma_w^2 \end{cases} \quad (12)$$

This means that for $i \neq j$ the weights are inversely proportional to the Euclidean distance between the vectors at position i and j. Unfortunately, the Gaussian distribution for modeling the residuals is not a good choice: for natural images we may expect that for each vector $\mathbf{x}_i$ there are many other vectors $\mathbf{x}_j$ that are very similar to $\mathbf{x}_i$. The residual or difference between $\mathbf{x}_j$ and $\mathbf{x}_i$ would from this perspective rather have a Laplacian distribution than a Gaussian distribution. Moreover, dissimilar patches $\mathbf{x}_i$ tend to cluster, which causes multiple modes in the histogram of the residual. One possibility is e.g. to use a better suited multivariate (multimodal) distribution for $\mathbf{r}_{i,j}$ that also incorporates correlations between the components of $\mathbf{r}_{i,j}$. Because the estimators for such a distribution are much more complicated and the model training is computationally more intensive, an attractive alternative is to use robust statistics instead and to treat the deviations from the model as outliers.

3.2. Robust M-estimator

The Robust M-estimator is derived from the ML estimator by replacing the quadratic term in the negative log-likelihood functional by a multivariate robust loss function $\rho(\mathbf{x})$, as follows:

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \sum_{j=1}^{N} \rho(\mathbf{x} - \mathbf{y}_j) \quad (13)$$

The minimum can be found iteratively, e.g. by gradient descent. Applying only one iteration of this algorithm yields:

$$\hat{\mathbf{x}}_i = \mathbf{y}_i - \lambda_i \sum_{j=1}^{N} \rho'(\mathbf{y}_i - \mathbf{y}_j) \quad (14)$$

with $\rho'(\mathbf{x})$ the gradient of $\rho(\mathbf{x})$. An interesting case is obtained by considering a robust function with derivative of the form:

$$\rho'(\mathbf{x}) = \mathbf{x} g(\mathbf{x}),$$

In this case, equation (14) becomes:

$$\hat{\mathbf{x}}_i = \mathbf{y}_i - \lambda_i \sum_{j=1, j\neq i}^{N} g(\mathbf{y}_i - \mathbf{y}_j)(\mathbf{y}_i - \mathbf{y}_j)$$

$$= \left(1 - \lambda_i \sum_{j=1, j\neq i}^{N} g(\mathbf{y}_i - \mathbf{y}_j)\right)\mathbf{y}_i + \lambda_i \sum_{j=1, j\neq i}^{N} g(\mathbf{y}_i - \mathbf{y}_j)\mathbf{y}_j \quad (15)$$

To speed up the first iteration, the step-size λ_i can be adapted based on the Jacobi algorithm. Analogous to [12], this leads to:

$$\lambda_i = \frac{1}{1 + \sum_{j=1, j\neq i}^{N} g(\mathbf{y}_i - \mathbf{y}_j)} \quad (16)$$

Equation (15) becomes:

$$\hat{\mathbf{x}}_i = \frac{\mathbf{y}_i + \sum_{j=1, j\neq i}^{N} g(\mathbf{y}_i - \mathbf{y}_j)\mathbf{y}_j}{1 + \sum_{j=1, j\neq i}^{N} g(\mathbf{y}_i - \mathbf{y}_j)} \quad (17)$$

which is the NLMeans filter with weight function given by:

$$w(i,j) = \begin{cases} g(\mathbf{y}_i - \mathbf{y}_j) & i \neq j \\ 1 & i = j \end{cases} \quad (18)$$

Example: for the Leclerc robust function defined by:

$$\rho(\mathbf{r}) = h^2 - h^2 \exp\left(-\frac{\mathbf{r}^T\mathbf{r}}{2h^2}\right) \quad (19)$$

we have:

$$\rho'(\mathbf{r}) = \mathbf{r} \exp\left(-\frac{\mathbf{r}^T\mathbf{r}}{2h^2}\right) \quad (20)$$

and

$$w(i,j) = \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{2h^2}\right) \quad (21)$$

Hence, the Leclerc robust function leads us to the weighting function from equation (4). Interestingly, the weight function (12) derived in Section (3.1) can also be interpreted to be associated to a robust function:

$$\rho(\mathbf{r}) = \begin{cases} \frac{1}{2} + \log\|\mathbf{r}\| & \|\mathbf{r}\| > h \\ \frac{\|\mathbf{r}\|^2}{2h^2} & \|\mathbf{r}\| \leq h \end{cases} \quad (22)$$

This interpretation makes it easier to compare the properties of the resulting estimator with other robust estimators. Note that the robust estimation framework gives us a much wider variety of weighting functions. Different robust functions will be compared more in detail in Section 4.3.

4. IMPROVEMENTS TO THE NLMEANS FILTER

Based on the theoretical framework explained in Section 2 and Section 3, we are now able to present a number of improvements for the NLMeans algorithm:

- If only one iteration of the NLMeans filter is applied, the solution has generally not converged to a (local) optimum. Practically, local neighbourhoods that only have few similar neighbourhoods may still contain noise after one iteration (unless h is chosen large enough; but this would cause oversmoothing in other regions). In this paper, we propose to keep track of the noise variance at every location in the image and to remove the remainder of the noise as a post-processing step using a local filter (see Section 4.1).

- As discussed in Section 3.2, the NLMeans algorithm can be considered to be the first iteration of the Jacobian optimization algorithm for the cost function in equation (13). This would suggest that applying the NLMeans algorithm iteratively would further decrease the cost function.

- The choice of the Leclerc robust function is somewhat arbitrary. We will see that further improvements can be achieved by using the Bisquare robust function.

- The existing NLMeans algorithms assume that the image noise is white (uncorrelated), while in most practical denoising applications the noise is correlated. In this case, computation of the similarity based on the Euclidean distance is hampered which eventually leads to a poor denoising performance. By extending the reasoning from Section 3, it now becomes possible to devise a NLMeans filter for *coloured* (correlated) noise.

These modifications will be described more in detail in the remainder of this Section.

4.1. Local filtering of remaining noise

In some circumstances (e.g. non-repetitive structures), one iteration of the NLMeans filter may not remove all of the noise. Theoretically, an infinite number of neighbourhoods is required in order to completely suppress the noise, which is not possible for finite image dimensions. However, it is possible to compute the noise variance of each estimated vector, based on equations (15) and (17):

$$\begin{aligned}\sigma^2_{x,i} &= \mathrm{Var}\left[\hat{\mathbf{x}}_i\right] \\ &= \sum_{j=1,j\neq i}^{N} \left(\lambda_i g\left(\mathbf{y}_i-\mathbf{y}_j\right)\right)^2 \mathrm{Var}\left[\mathbf{y}_j\right] + \\ &\quad \lambda_i^2 \mathrm{Var}\left[\mathbf{y}_i\right] \\ &= \sigma_w^2\lambda_i^2\left(\sum_{j=1,j\neq i}^{N} g^2\left(\mathbf{y}_i-\mathbf{y}_j\right)+1\right) \quad (23)\end{aligned}$$

with λ_i given by equation (16). Clearly, the noise variance depends on the position in the image, which means that we are dealing with non-stationary noise. In theory, any algorithm for non-stationary noise can be used, such as the ones presented in [6, 22]. Because before the aggregation, the output of the NLMeans filter is a set of vectors, it is beneficial to use a vector-based denoising algorithm and to aggregate afterwards. In this paper, we adopt a locally adaptive basis of Principle Components, as in [23]. This method assumes that the Gaussian noise is stationary, but an extension to non-stationary noise is straightforward, as we will show next. First, we define $\mathbf{C}_{y,i}$ as the local covariance matrix of $\mathbf{y}_i$, estimated as follows[1]:

$$\hat{\mathbf{C}}_{y,i} = \frac{1}{2R+1}\sum_{n=-R}^{R}(\mathbf{y}_{i+n}-\boldsymbol{\mu}_i)(\mathbf{y}_{i+n}-\boldsymbol{\mu}_i)^T$$

$$\text{with} \quad \hat{\boldsymbol{\mu}}_i = \frac{1}{2R+1}\sum_{n=-R}^{R}\mathbf{y}_{i+n}$$

To find the PCA basis, we apply the diagonalization:

$$\hat{\mathbf{C}}_{y,i} = \mathbf{U}_i\boldsymbol{\Lambda}_i\mathbf{U}_i^T \quad (24)$$

The local covariance matrix of $\mathbf{x}_i$, denoted as $\mathbf{C}_{x,i}$ can be estimated as follows:

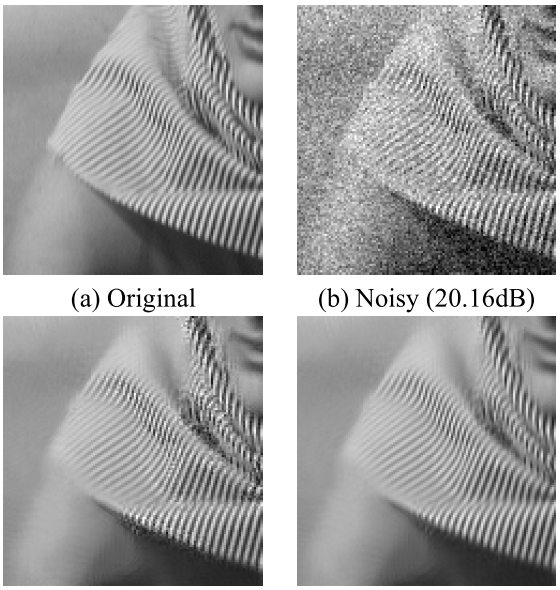
$$\begin{aligned}\hat{\mathbf{C}}_{x,i} &= \left(\hat{\mathbf{C}}_{y,i}-\sigma_w^2\mathbf{I}\right)_+ \\ &= \mathbf{U}_i\left(\boldsymbol{\Lambda}_i-\sigma_{\mathrm{w}}^2\mathbf{I}\right)_+\mathbf{U}_i^T \quad (25)\end{aligned}$$

where $(\cdot)_+$ replaces possible negative eigenvalues by a small positive number, such that the resulting matrix is positive definite (due to estimation errors of $\hat{\mathbf{C}}_{y,i}$, it is possible that the difference $\hat{\mathbf{C}}_{y,i}-\sigma_w^2\mathbf{I}$ has negative eigenvalues). The linear MMSE estimator is given by:

$$\begin{aligned}\hat{\hat{\mathbf{x}}}_i &= \hat{\boldsymbol{\mu}}_i + \hat{\mathbf{C}}_{x,i}\left(\hat{\mathbf{C}}_{x,i}+\sigma_{x,i}^2\mathbf{I}\right)^{-1}(\hat{\mathbf{x}}_i-\hat{\boldsymbol{\mu}}_i) \\ &= \hat{\boldsymbol{\mu}}_i + \mathbf{U}_i\mathbf{A}_i\mathbf{U}_i^T(\hat{\mathbf{x}}_i-\hat{\boldsymbol{\mu}}_i) \quad (26)\end{aligned}$$

with $\mathbf{A}_i = \left(\boldsymbol{\Lambda}_i-\sigma_{\mathrm{w}}^2\mathbf{I}\right)_+\left(\left(\boldsymbol{\Lambda}_i-\mathbf{I}\sigma_w^2\right)_+ + \sigma_{x,i}^2\mathbf{I}\right)^{-1}$ a diagonal (Wiener filter) matrix. Because a direct implementation of the above formulas has a high computational cost (the diagonalization in equation (24) needs to

[1] This expressions given here are for 1-d signals, but can be easily extended to images by using a double summation.



(a) Original (b) Noisy (20.16dB)

(c) NLMeans-W without post-processing filter (29.18dB) (d) NLMeans-W with post-processing filter (30.53dB)

Figure 1. Denoising example of Barbara: the effect of using the proposed post-processing filter (a) Crop out of the original image (b) Image with white Gaussian noise ($\sigma = 25$). (c) The result of the NLMeans filter *without* post-processing filter (d) The result of the NLMeans filter *with* post-processing filter. PSNR values are between parentheses.

be performed for every pixel in the image), we only estimate the covariance matrix $\hat{\mathbf{C}}_{y,i}$ at subsampled positions i and assume this covariance matrix is piecewise constant. Also, for neighbourhoods with a sufficient number of similar blocks, the variance $\sigma^2_{x,i}$ will be very small after the NLMeans filtering stage. In this case do not apply the local filtering step. An example for the Barbara image is given in Figure 1. It can be seen that the NLMeans filter removes most of the noise, except in regions with limited repetitivity. In this regions, the post-processing filter works excellent in removing the remainder of the noise in Figure 1c, while retaining the stripes.

4.2. An iterative NLMeans filter

An iterative NLMeans filter can be obtained by performing the optimization in equation (13) iteratively. Therefore, we choose the observed image as an initial estimate, i.e. $\hat{\mathbf{x}}_j^{(0)} = \mathbf{y}_j$, $j = 1, ..., N$. For the n-th iteration ($n > 0$), we find:

$$\hat{\mathbf{x}}_i^{(n)} = \hat{\mathbf{x}}_i^{(n-1)} - \lambda_i^{(n)} \sum_{j=1}^{N} g\left(\hat{\mathbf{x}}_i^{(n-1)} - \hat{\mathbf{x}}_j^{(n-1)}\right)\left(\hat{\mathbf{x}}_i^{(n-1)} - \hat{\mathbf{x}}_j^{(n-1)}\right)$$

with, following the Jacobi algorithm for determining the step size:

$$\lambda_i^{(n)} = \frac{1}{1 + \sum_{j=1, j\neq i}^{N} g\left(\hat{\mathbf{x}}_i^{(n-1)} - \hat{\mathbf{x}}_j^{(n-1)}\right)} \tag{27}$$

Implementation of this estimate is analogous to the implementation of (17), except that the estimates from the previous iterations are used as input. At first sight, the reader may incorrectly have the impression that the iterative NLMeans filter increases the computational cost by the number of iterations. Instead, the iterative filter offers a number of advantages:

- A small search window (e.g. 31×31) can be used instead of the whole image, which dramatically reduces the computation time. When applying more iterations, information from outside the search window will also be used, resulting in a complete *non-local* denoising technique.
- Potentially a better end solution can be found, because the cost function is further reduced.
- A limitation of the non-iterative NLMeans filter is that the weights are computed directly based on the observed noisy image. As a result, the weights are very sensitive to the image noise. In [13], this problem is avoided by applying a rough denoising preprocessing step before selecting similar neighbourhoods. However, this technique has the problem that by rough denoising, some details may be lost, potentially resulting in an incorrect selection of dissimilar blocks. In the iterative NLMeans algorithm, every iteration reduces the average noise variance, resulting in better weight estimates, thereby increasing the overall denoising performance.

4.3. The choice of the robust loss function

As said before, the NLMeans algorithm proposed in [10, 11] can be interpreted as the first Jacobian iteration of a robust estimation method, that uses the Leclerc loss function. In our earlier work [16] we noted that the exponential form of $g(\mathbf{x})$ still assigns positive weights to *dissimilar* neighbourhoods. Even though these weights are very small, the estimated pixel intensities can be severely biased due to many small contributions. We therefore proposed a preclassification based on the first three statistical moment to exclude dissimilar blocks [16]. An alternative is to change the shape of the robust function. An overview of some *relevant* robust functions are given in Table 1. We will now look at the characteristics of the robust weighting functions more in detail.

- The weighting function associated with the *BLUE* estimator derived in Section 3.1 and the *Cauchy* weighting function have a very slow decay (see Figure 2a). They assign larger weights to dissimilar blocks than the Leclerc robust function, which will eventually lead to oversmoothing.

- The *Leclerc* weighting function has a faster decay, but still assigns positive weights to dissimilar blocks.
- The *Andrews* weighting function imposes a hard threshold to compare neighbourhoods (the weight is 0 as soon as a given threshold is exceeded), while the *Tukey* and *Bisquare* weighting functions rather use a soft threshold (Figure 2b). Experimentally we found that applying a soft threshold often improves the visual quality, in analogy to wavelet thresholding.
- To further improve upon the Tukey and Bisquare weighting functions, we also modified the Bisquare robust function in order to have a steeper slope (see Table 1 and Figure 2b). In Section 6 we will report the improvement in PSNR by using this robust function.

4.4. NLMeans filter for correlated noise

In the previous Sections, we assumed that the Gaussian noise is *uncorrelated* (white). Applying the NLMeans filter without modifications to images corrupted with *correlated* noise often yields a poor denoising performance (see Section 6). Fortunately, a robust estimator for correlated noise can be obtained by replacing the Euclidean distance $\|\mathbf{y}_i - \mathbf{y}_j\|$ by the Mahalanobis distance $\sqrt{(\mathbf{y}_i - \mathbf{y}_j)^T \mathbf{C}_w^{-1} (\mathbf{y}_i - \mathbf{y}_j)}$ that takes the noise covariance matrix $\mathbf{C}_w$ into account (see e.g. [24]), giving the following estimator:

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \sum_{j=1}^{N} \rho\left(\mathbf{C}_w^{-1/2}(\mathbf{x} - \mathbf{y}_j)\right)$$

with $(\cdot)^{1/2}$ the square root of a positive definite matrix. The weight function becomes:

$$w(i,j) = \begin{cases} g\left(\mathbf{C}_w^{-1/2}(\mathbf{y}_i - \mathbf{y}_j)\right) & i \neq j \\ 1 & i = j \end{cases} \quad (28)$$

Clearly, the correlatedness of the noise only affects the weight function and not the final averaging. However, the matrix multiplication in equation (28) is still computationally expensive, given the large number of weights to be computed. Therefore, we apply a prewhitening linear filter operation to the noisy image. If the noise Power Spectral Density is given by $|H(l)|^2$, then we compute the prewhitened image $Y_i^{\text{prewhit}}, i = 1, ..., N$ as:

$$\tilde{Y}^{\text{prewhit}}(l) = \tilde{Y}(l) \frac{1}{\max(\epsilon, |H(l)|)}$$

with $\tilde{Y}^{\text{prewhit}}(l)$, $\tilde{Y}(l)$ the discrete Fourier transform of respectively Y_i^{prewhit} and Y_i, and ϵ a small positive number to ensure stability (for the results in Section 6, $\epsilon = 10^{-4}$). Next, this prewhitened image is used to compute the weights based on the Euclidean distance (see also Figure 3).

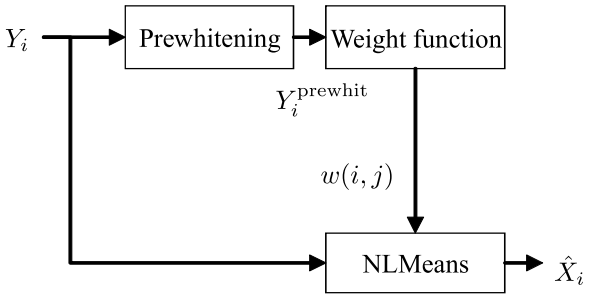


Figure 3. Block diagram for the suppression of *correlated* noise

5. SPEEDING UP THE NLMEANS FILTER

Because the algorithmic complexity of the brute force NLMeans filter on images is $\mathcal{O}(N^2(2K+1)^2)$, with N the number of pixels in the image, an efficient implementation is desirable. Our previous analysis in [16] revealed that the main part of the computation time is taken by the weight computation.

5.1. Exploiting weight symmetry

As in [16], we reduce the computation time by approximately a factor 2 by exploiting the fact that weight functions are symmetrical (i.e. $w(i,j) = w(j,i)$). Therefore, we keep track of a weight normalization matrix and a accumulated contribution matrix (both of the same size as the input image), and at the beginning of the algorithm, initialized with zeros. When processing a pixel i with the contribution of a pixel j, we add the products $w(i,j)Y_i$ and $w(i,j)Y_j$ to the accumulated contribution matrix at the pixel positions j and i respectively. The weight normalization matrix is also accumulated at the same pixel positions with $w(i,j)$. As a result, we only need to compute weights of neighbourhoods for $j > i$. Finally, we normalize the accumulated contribution matrix via element-wise division by the weight normalization matrix, in order to obtain the estimated image. These concepts can also be applied to the *vector*-based NLMeans filter: here the products $w(i,j)b(k)Y_{i+k}$ and $w(i,j)b(k)Y_{j+k}$ are added to the accumulated contribution matrix at positions $j+k$ and $i+k$ respectively, for $k = -K, ..., K$. At the same positions, the products $w(i,j)b(k)$ are added to weight normalization matrix.

5.2. Fast computation of the weight functions based on the Euclidean distance

For the robust functions in Table 1, $w(i,j)$ is a function of the Euclidean distance $\|\mathbf{y}_i - \mathbf{y}_j\|$. When considering a constant position difference Δi (i.e. $j = i + \Delta i$), the function

$$\beta(i) = \|\mathbf{y}_i - \mathbf{y}_{i+\Delta i}\|^2$$

can be practically implemented using a moving average filter applied to the squared difference between the signals

Type	$g(\mathbf{r})$	$\rho(\mathbf{r})$
"BLUE"	$= \begin{cases} 1/\left\|\mathbf{r}\right\|^2 & \left\|\mathbf{r}\right\| > h \\ 1/h^2 & \left\|\mathbf{r}\right\| \leq h \end{cases}$	$= \begin{cases} \frac{1}{2} + \log\left\|\mathbf{r}\right\| & \left\|\mathbf{r}\right\| > h \\ \frac{\left\|\mathbf{r}\right\|^2}{2h^2} & \left\|\mathbf{r}\right\| \leq h \end{cases}$
Cauchy	$= 1/\left(1 + \left\|\mathbf{r}\right\|^2/h^2\right)$	$= \frac{1}{2}h^2 \log\left(h^2 + \left\|\mathbf{r}\right\|^2\right)$
Tukey	$= \begin{cases} (1 - \frac{\left\|\mathbf{r}\right\|^2}{h^2})^2 & \left\|\mathbf{r}\right\| \leq h \\ 0 & \left\|\mathbf{r}\right\| > h \end{cases}$	$= \begin{cases} \frac{h^2}{6}\left[1 - \left(1 - \frac{\left\|\mathbf{r}\right\|^2}{h^2}\right)^3\right] & \left\|\mathbf{r}\right\| \leq h \\ \frac{h^2}{6} & \left\|\mathbf{r}\right\| > h \end{cases}$
Andrews	$= \begin{cases} \frac{\sin(\pi\left\|\mathbf{r}\right\|/h)}{\pi\left\|\mathbf{r}\right\|/h} & \left\|\mathbf{r}\right\| \leq h \\ 0 & \left\|\mathbf{r}\right\| > h \end{cases}$	$= \begin{cases} \frac{h^2}{\pi^2}\left(1 - \cos\left(\pi\left\|\mathbf{r}\right\|/h\right)\right) & \left\|\mathbf{r}\right\| \leq h \\ \frac{h^2}{\pi^2} & \left\|\mathbf{r}\right\| > h \end{cases}$
Leclerc	$= \exp\left(-\frac{\left\|\mathbf{r}\right\|^2}{2h^2}\right)$	$= h^2 - h^2 \exp\left(-\frac{\left\|\mathbf{r}\right\|^2}{2h^2}\right)$
Bisquare	$= \begin{cases} \left(1 - \frac{\left\|\mathbf{r}\right\|^2}{h^2}\right)^2 & \left\|\mathbf{r}\right\| \leq h \\ 0 & \left\|\mathbf{r}\right\| > h \end{cases}$	$= \begin{cases} \frac{h^2}{6}\left(\frac{\left\|\mathbf{r}\right\|}{h} - 1\right)^3\left(\frac{\left\|\mathbf{r}\right\|}{h} + 1\right)^3 & \left\|\mathbf{r}\right\| \leq h \\ 0 & \left\|\mathbf{r}\right\| > h \end{cases}$
Modified Bisquare	$= \begin{cases} \left(1 - \frac{\left\|\mathbf{r}\right\|^2}{h^2}\right)^8 & \left\|\mathbf{r}\right\| \leq h \\ 0 & \left\|\mathbf{r}\right\| > h \end{cases}$	$= \begin{cases} \frac{h^2}{18}\left(\frac{\left\|\mathbf{r}\right\|}{h} - 1\right)^9\left(\frac{\left\|\mathbf{r}\right\|}{h} + 1\right)^9 & \left\|\mathbf{r}\right\| \leq h \\ 0 & \left\|\mathbf{r}\right\| > h \end{cases}$

Table 1. Overview of multivariate robust M-functions for extending the NLMeans algorithm, with h a smoothing parameter.

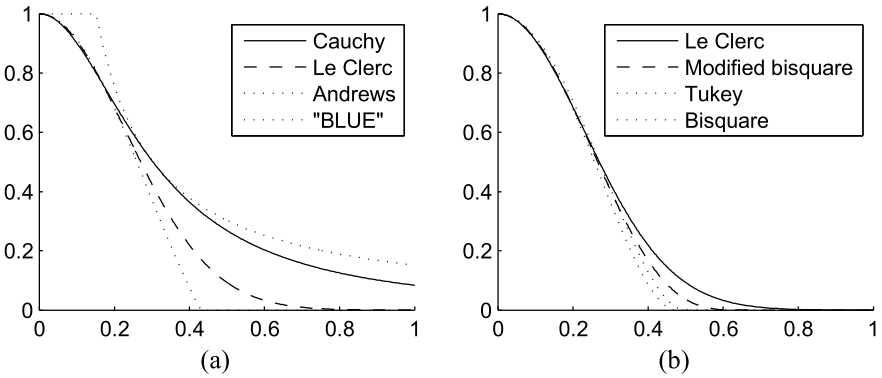


Figure 2. Comparison of the weighting functions $g(r)$ for commonly used robust functions. Here, $r = \|\mathbf{y}_i - \mathbf{y}_j\|$ is the Euclidean distances between two vectors $\mathbf{y}_i$ and $\mathbf{y}_j$

Y_i and $Y_{i+\Delta i}$, requiring only 2 pixel accesses instead of $2K+1$ (in 1-d). For images, we use the straightforward 2-d extension of the moving averaging filter. Together with the weight symmetry, this brings the complexity down to approximately $\mathcal{O}(2N^2)$, yielding a speed up of a factor $(2K+1)^2/2$. When considering square neighbourhoods of size 11×11, $K = 5$ and the overall speedup is a factor $121/2$ compared to our previous NLMeans filter in [16].

5.3. Limited search window

The limited search window strategy assigns zero weights to neighbourhoods that are too far away from each other, i.e. $w(i,j) = 0$ if $|i-j| > N_w$. This technique may decrease the strength of the NLMeans, especially for images with many repetitive structures, but for many real images, the denoising performance is not significantly affected. This stems from the fact that many *similar* blocks can be found in the *local* neighbourhood. In this paper, we use a limited search window of 31×31 pixels, in order to keep the computational cost of the algorithm low. We remark again that in combination with the iterative approach from Section 4.2, the *effective* search window is extended by $N_w - 1$ in each dimension, hence when running at least $\lceil N/(N_w - 1) \rceil$ iterations, the complete image will be searched.

5.4. Pseudo-code

For illustrative purposes, the pseudo-code of the pixel-based 1-d implementation is shown in Figure 4. In the pseudo-code, the boundary handling is omitted for clarity, as well as the initialization (warm-up) operations of the moving average filter. The algorithm uses two loops[2]: the first loop iterates on Δi in a limited search window, the second loop runs over the whole signal. The similarities corresponding to zero displacements ($\Delta i = 0$) are treated differently, because the weights are easier to compute in this case. It can be seen that the three acceleration techniques proposed in this Section, can be elegantly and efficiently combined in this algorithm. We further remark that for the *vector*-based implementation (see Section 2), additional changes are required: a different weight accumulation matrix needs to be used for each position in the local neighbourhood.

```
% outer loop (limited search window)
weight_cum = zeros(1,N);
weight_norm = zeros(1,N);

for di=1:Nw
   % some initialization of the
   % moving average operation
   ...
   % inner loop
   for j=1:N
      i=j+di;
      % moving average (sum) based computation
      % of the Euclidean distance
      eucl_dist=eucl_dist+(y(i+K)-y(j+K))^2;
      eucl_dist=eucl_dist-(y(i-K-1)-y(j-K-1))^2;

      % weight computation
      weight=g(eucl_dist);

      % weight accumulation
      weight_cum(i) = weight_cum(i)+weight*y(j);
      weight_norm(i) = weight_norm(i)+weight;
      % symmetry
      weight_cum(j) = weight_cum(j)+weight*y(i);
      weight_norm(j) = weight_norm(j)+weight;
   end;
end;

di=0; % zero displacement
for i=1:N
    weight=1;
    weight_cum(i)=weight_cum(i)+weight*y(i);
    weight_norm(i)=weight_norm(i)+weight;
end;

% final estimate
x_hat = weight_cum ./ weight_norm;
```

Figure 4. Pseudo-code for the improved NLMeans algorithm in 1-d (*pixel*-based implementation)

6. RESULTS AND DISCUSSION

To assess the improvement gained by using the modified Bisquare cost function over the Leclerc cost function (see Table 1), we set up a denoising experiment with 8 images and 9 noise levels. All images are corrupted with artificially generated white Gaussian noise with varying noise levels, and subsequently the non-iterative NLMeans algorithm is applied to them (including the post-processing filter). In Figure 5, we report the PSNR improvement by using the modified Bisquare cost function. We note that on average, the improvement is approximately 0.1dB. However the gain depends both on the image as on the noise level, and is maximal for 1) images with many structures or strong edges, such as *Barbara* and *house* and 2) for large noise levels (e.g. $\sigma \approx 60$). Because of the vast improvement for most images and/or noise levels, we will further use the Bisquare cost function.

The results for the proposed NLMeans filter are generated using a neighbourhood size of 11×11 (i.e. $K = 5$), a search window of 31×31 (i.e. $N_w = 15$). The Bisquare cost function is used and the h-parameter of the robust function is selected experimentally as: $h = 2.1\sigma$. The post-processing filter uses a subsampling factor 8 in the x and y direction for estimating the local covariance matrix (see Section 4.1). To further save computation time and to improve the numerical stability, the post-processing is done on neighbourhoods of size 5×5, selected from the center of the 11×11 neighbourhood (hence the aggregation weighting function $b(k) = \mathrm{I}(|k| \leq 2)/5$ is used, with $\mathrm{I}(\cdot)$ the indicator function). In Table 2, the denoising performance of the proposed method for white noise (NLMeans-W) is compared to BLS-GSM [5] (a state-of-the-art wavelet denoising technique) and BM-3D (a state-of-the-art non-local denoising technique) [13]. We give

[2] For images, four loops are needed (two outer loops and two inner loops for the x and y-directions).

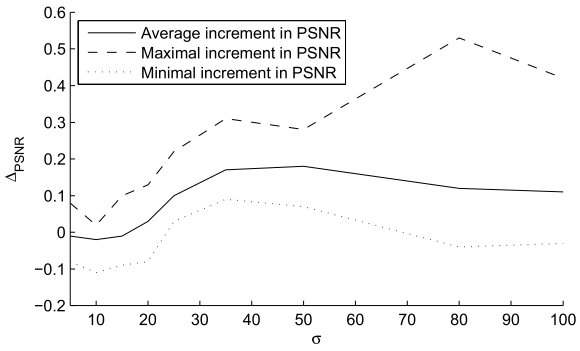


Figure 5. Comparison of using the Leclerc cost function versus the modified Bisquare cost function, for a denoising experiment on a set of 8 images (*Barbara*, *Lena*, *boats*, *couple*, *hill*, *man*, *peppers*, *house*). The values are the average (over the test set), maximal and minimal gain in PSNR by using the modified Bisquare cost function compared to the Leclerc cost function, for different noise levels σ.

results for both the non-iterative NLMeans (i.e. only one iteration) and the iterative NLMeans filter as presented in Section 4.2, with 25 iterations. The iterative NLMeans filter brings an additional improvement compared to the non-iterative NLMeans. This is because the similarity weights are refined iteratively, eventually resulting in a cleaner image with more details. The iterative NLMeans algorithm clearly outperforms the BLS-GSM filter and seems competitive to the BM-3D method.

The visual performance of these methods is compared in Figure 6-Figure 9. Here, the difference is astonishing: the NLMeans filter reconstructs smoother images with remarkably less artifacts than the other methods. Some details are better reconstructed, such as the fine stripes in the hat of Lena and the feathers, even though the PSNR of the NLMeans filter is slightly lower compared to BM-3D. Further investigation revealed that the inferior PSNR is caused by the reduced contrast (or oversmoothing) of fine structures, that are usually visually hardly noticeable (for example, the thick stripes in the hat of Lena located above the feathers in Fig.6f).

We also compared the denoising performance of the NLMeans extension to correlated noise (NLMeans-C) to recent techniques for correlated noise: BLS-GSM [5] and our recently proposed MP-GSM [25]. The NLMeans filter brings a significant improvement both visually as in PSNR: there are almost no ringing artifacts and obviously no wavelet artifacts. We also compare to NLMeans-W (that is not adapted to the correlated noise), just to illustrate that the NLMeans filter, that uses weighting functions based on the Euclidean distance, does not perform well in the presence of correlated noise. The computation time for the non-iterative NLMeans filter is approximately 20s. on a Pentium IV processor and for a 512×512 grayscale image. The iterative filter takes approximately 8 minutes, when 25 iterations are used.

	Noise standard deviation				
	10	25	35	50	100
	LENA				
BLS-GSM [5]	35.59	31.58	30.05	28.45	25.49
BM-3D [13]	**35.89**	**32.04**	**30.52**	**28.79**	25.49
NLMeans-W (non-it)	35.55	31.70	30.03	28.26	24.72
NLMeans-W (25 it)	35.53	31.74	30.40	28.64	**25.73**
	BARBARA				
BLS-GSM [5]	34.51	29.30	27.44	25.58	22.82
BM-3D [13]	**35.38**	**30.93**	**29.13**	**27.25**	**23.53**
NLMeans-W (non-it)	35.01	30.56	28.62	26.57	22.95
NLMeans-W (25 it)	34.91	30.59	28.82	26.99	23.40

Table 2. PSNR [dB] results for white noise

7. CONCLUSION

By exploiting the repetitivity of structures in an image, a significant gain in denoising performance is obtained compared to purely local methods. We have shown that the NLMeans algorithm is the first iteration of the Jacobi optimization algorithm for robustly estimating the noise-free image, based on the Leclerc loss function. Next, we have proposed several improvements to the NLMeans filter that affect both the visual quality and the computation time. Our proposed method now compares favourably to state-of-the-art non-local denoising techniques in PSNR and even offers a superior visual quality: the denoised images contain less artifacts without sacrificing sharpness. The proposed improvements can also be applied to other image processing tasks such as intra-frame super-resolution, non-local demosaicing and deinterlacing.

8. REFERENCES

[1] L. Rudin and S. Osher, "Total variation based image restoration with free local constraints," in *Proc. of IEEE International Conference on Image Processing (ICIP)*, vol. 1, Nov. 1994, pp. 31–35.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings International conference on computer vision*, 1998, pp. 839–846.

[3] D. L. Donoho, "De-Noising by Soft-Thresholding," *IEEE Trans. Inform. Theory*, vol. 41, pp. 613–627, May 1995.

[4] L. Şendur and I. Selesnick, "Bivariate shrinkage with local variance estimation," *IEEE Signal Processing Letters*, vol. 9, pp. 438–441, 2002.

[5] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Transactions on image processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[6] J. Portilla, "Image restoration using Gaussian Scale Mixtures in Overcomplete Oriented Pyramids (a review)," in *Wavelets XI, in SPIE's International Symposium on Optical Science and Technology, SPIE's 50th Annual Meeting, Proc. of the SPIE*, vol. 5914, San Diego, CA, aug 2005, pp. 468–482.

[7] A. Pižurica and W. Philips, "Estimating the probability of the presence of a signal of interest in multiresolution single- and multiband image denoising," *IEEE Transactions on image processing*, vol. 15, no. 3, pp. 654–665, 2006.

[8] F. Luisier, T. Blu, and M. Unser, "A New SURE Approach to Image Denoising: Interscale Orthonormal Wavelet Thresholding," *IEEE Trans. Image Processing*, vol. 16, no. 3, pp. 593–606, Mar. 2007.

[9] J. A. Guerrero-Colon, L. Mancera, and J. Portilla, "Image restoration using space-variant gaussian scale mixtures in overcomplete pyramids," *IEEE Trans. Image Proc.*, vol. 17, no. 1, pp. 27–41, 2008.

[10] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *SIAM interdisciplinary journal: Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 290–530, 2005.

[11] A. Buades, B. Coll., and J. Morel, "A non local algorithm for image denoising," in *Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2005, pp. 60–65.

[12] M. Elad, "On the Origin of the Bilateral Filter and Ways to Improve it," *IEEE Transactions on Image Processing*, vol. 11, no. 10, pp. 1141–1151, 2002.

[13] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[14] N. Azzabou, N. Paragias, and G. F., "Image Denoising Based on Adapted Dictionary Computation," in *Proc. of IEEE International Conference on Image Processing (ICIP)*, San Antonio, Texas, USA, Sept. 2007, pp. 109–112.

[15] C. Kervrann, J. Boulanger, and P. Coupé, "Bayesian Non-Local Means Filter, Image Redundancy and Adaptive Dictionaries for Noise Removal," in *Proc. Int. Conf. on Scale Space and Variational Methods in Computer Visions (SSVM'07)*, Ischia, Italy, 2007, pp. 520–532.

[16] A. Dauwe, B. Goossens, H. Luong, and W. Philips, "A Fast Non-Local Image Denoising Algorithm," in *Proc. SPIE Electronic Imaging*, vol. 6812, San José, USA, Jan 2008.

[17] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," *IEEE Trans. Image Processing*, vol. 15, no. 10, pp. 2866–2878, 2006.

[18] T. Brox and D. Cremers, "Iterated Nonlocal Means for Texture Restoration," in *Proc. Int. Conf. on Scale Space and Variational Methods in Computer Visions (SSVM'07)*, vol. 4485. Ischia, Italy: Springer, LNCS, 2007.

[19] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Processing Letters*, vol. 12, no. 12, pp. 839–842, Dec. 2005.

[20] J. Wang, Y. Guo, Y. Ying, Y. Liu, and Q. Peng, "Fast non-local algorithm for image denoising," in *Proc. of IEEE International Conference on Image Processing (ICIP)*, 2006, pp. 1429–1432.

[21] B. R. C. and M. Vehvilainen, "Fast nonlocal means for image denoising," in *Proc. SPIE Digital Photography III*, R. A. Martin, J. M. DiCarlo, and N. Sampat, Eds., vol. 6502, no. 1. SPIE, 2007.

[22] B. Goossens, A. Pižurica, and W. Philips, "Wavelet domain image denoising for non-stationary and signal-dependent noise," in *IEEE International Conference on Image Processing (ICIP)*, Atlanta, GA, USA, oct 2006, pp. 1425–1428.

[23] D. D. Muresan and T. W. Parks, "Adaptive Principal Components and Image Denoising," in *Proc. Int. Conf. on Image Processing (ICIP)*, 2003.

[24] N. A. Campbell, H. P. Lopuhaä, and P. J. Rousseeuw, "On the calculation of a robust s-estimator of a covariance matrix." *Stat Med*, vol. 17, no. 23, pp. 2685–2695, Dec 1998.

[25] B. Goossens, A. Pižurica, and W. Philips, "Image Denoising Using Mixtures of Projected Gaussian Scale Mixtures," *IEEE Transactions on Image Processing*, 2008, (submitted).

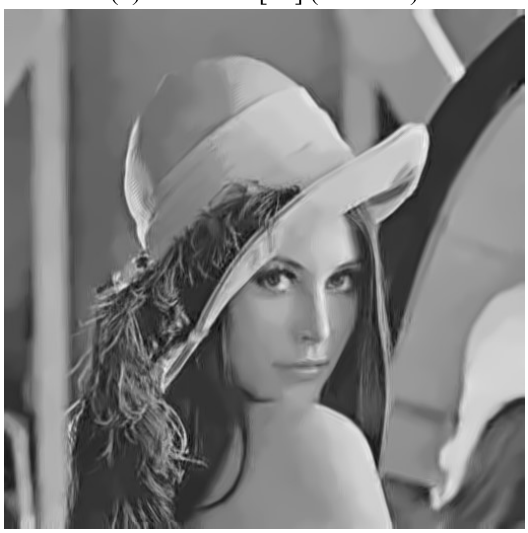
(a) Original

(b) Noisy (20.19dB)

(c) SV-GSM [9] (31.59dB)

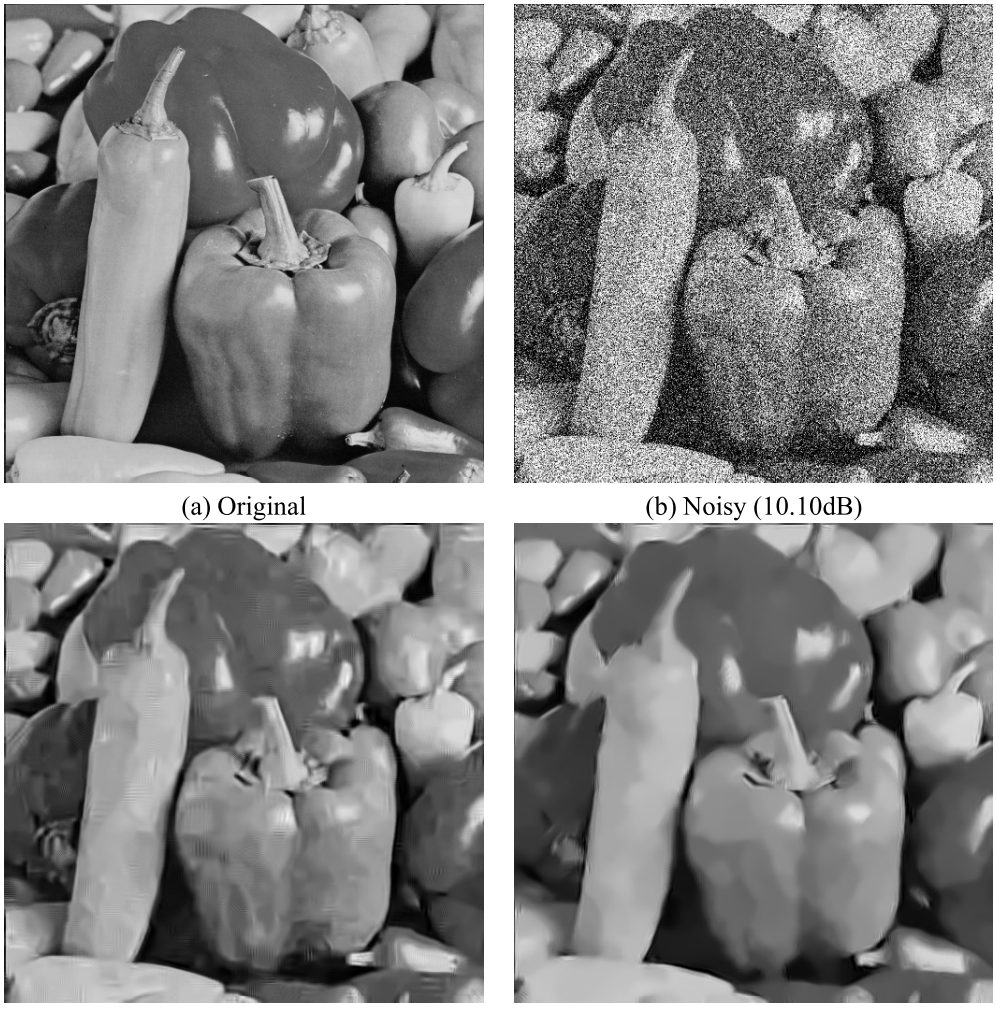
(d) MP-GSM [25] (31.81dB)

(e) BM-3D [13] (32.06dB)



(f) NLMeans-W (31.74dB)

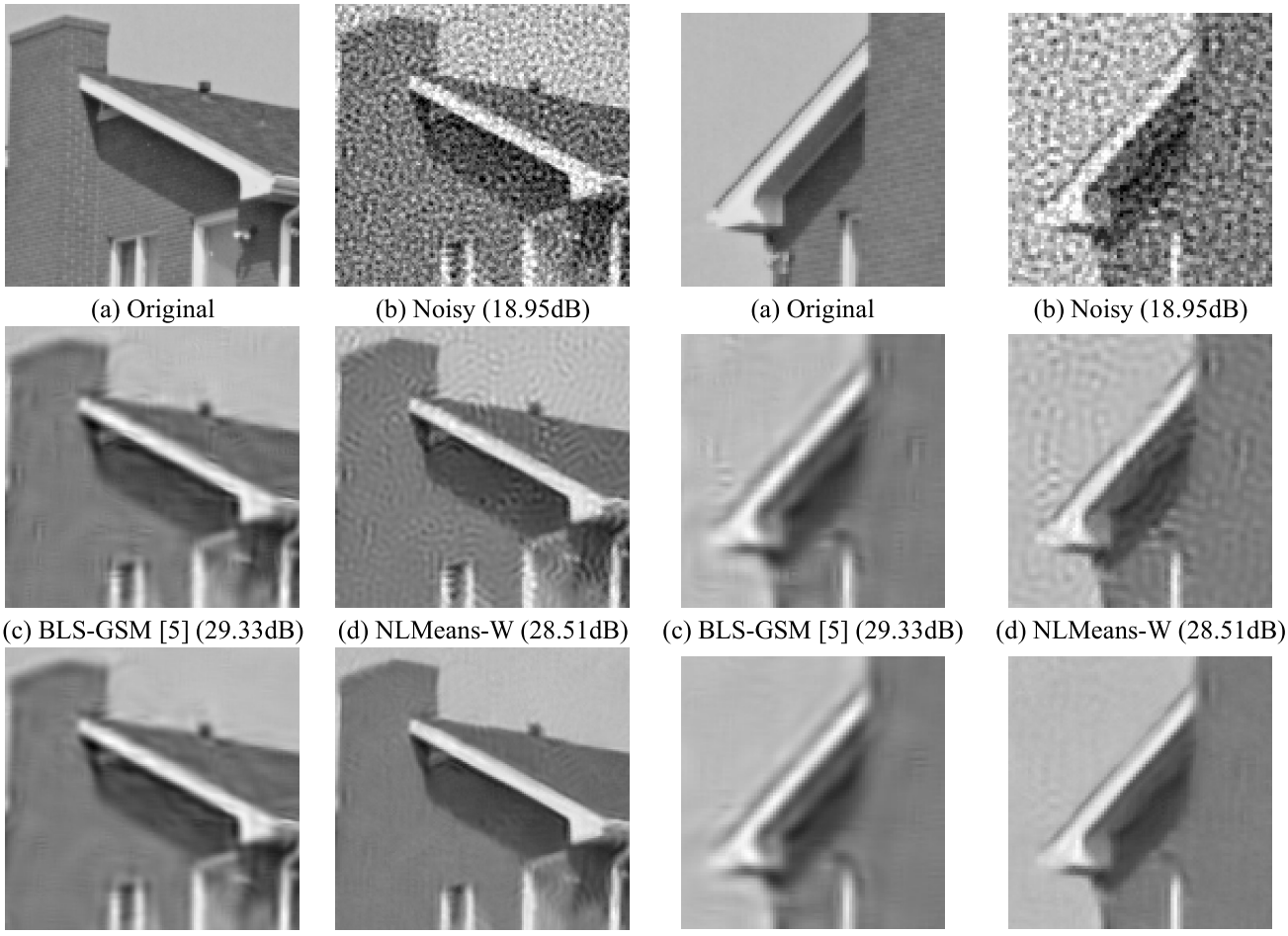
Figure 6. Visual results for the Lena image corrupted with *white* noise ($\sigma = 25$)



(a) Original (b) Noisy (10.10dB)

(c) BM-3D [13] (26.47dB) (d) NLMeans-W (26.59dB)

Figure 7. Visual results for the peppers image corrupted with *white* noise ($\sigma = 80$)



(a) Original (b) Noisy (18.95dB) (a) Original (b) Noisy (18.95dB)

(c) BLS-GSM [5] (29.33dB) (d) NLMeans-W (28.51dB) (c) BLS-GSM [5] (29.33dB) (d) NLMeans-W (28.51dB)

(e) MP-GSM [25] (30.02dB) (f) NLMeans-C (30.74dB) (e) MP-GSM [25] (30.02dB) (f) NLMeans-C (30.74dB)

Figure 8. Visual results for *correlated* noise: two different crop-outs of the *house* image. PSNR values are between parentheses.

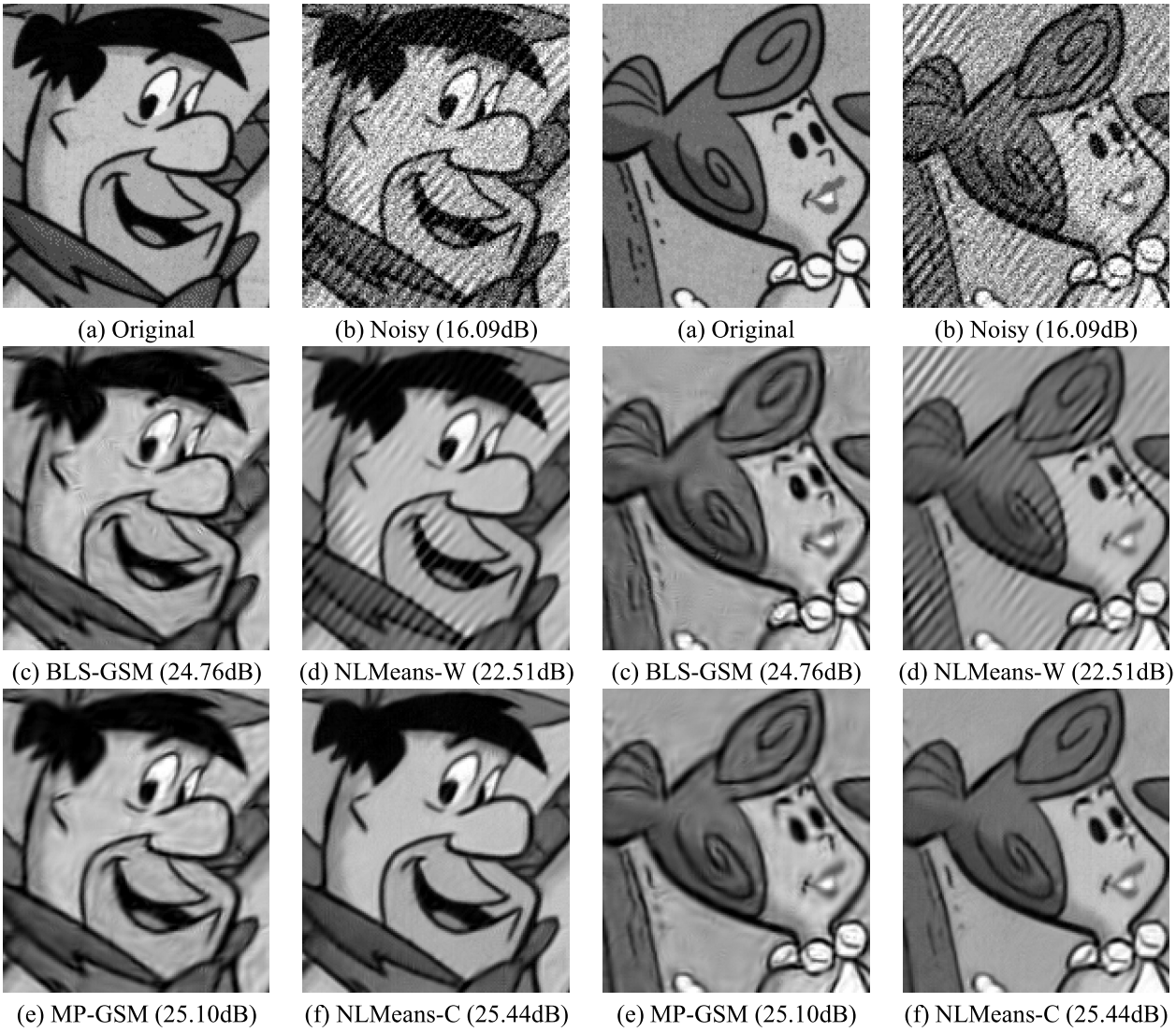


Figure 9. Visual results for *correlated* noise: two different crop-outs of the *flinstones* image. PSNR values are between parentheses.