PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 30/07

Methods for the acceleration of “non-local means” Noise reduction algorithm

Noam Shaham

Eduardo Sany Laber

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 30/07 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena December, 2007

Methods fot the acceleration of ”non-local means” noise reduction algorithm

Noam Shaham
Eduardo Sany Laber

shahamn@gmail.com, laber@inf.puc-rio.br

Abstract. “Non Local Means” is an innovative noise reduction algorithm for images presented by Buades and Morel in 2004. It performs remarkably better than older generation algorithms but has a performance penalty that prevents it from being used in mainstream consumer application. The objective of this work is to find ways of reducing the time-complexity of the algorithm and enabling its use in main stream image processing applications such as home photography or photo printing centers.

Keywords: image processing; noise reduction; non-local means.

Resumo. “Non-local means” é um novo algoritmo de redução de ruídos para imagens apresentado por Buades e Morel em 2004. Este algoritmo funciona consideravelmente melhor do que os algoritmos anteriores, mas sua lenta execução causada pela alta complexidade o impede de ser usado em aplicações comuns. O objetivo deste trabalho é investigar maneiras de reduzir o tempo de execução do algoritmo, possibilitando seu uso em aplicações comuns de processamento de imagem, tal como fotografia e centros de impressão.

Palavras-chave: processamento de imagens; redução de ruído; médias não locais.

In charge of publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Contents

List of abbreviations

ENNS	Equal average Nearest Neighbor Search
EENNS	Equal average Equal variance Nearest Neighbor Search
HVS	Human Visual System
IEENNS	Improved *EENNS*
MSE	Mean Square Error
NLM	Non Local Means
SD	Standard Deviation
SNR	Signal to Noise Ratio
VQ	Vector Quantization

List of tables

List of figures

1
Introduction

Image denoising has long been studied as one of the most fundamental issues of image processing. This is due to the fact that any natural image contains undesirable noise and its removal is essential in vast selection of applications ranging from home photography to medical imaging and from seismographic analysis to astronomical research. The problem of restoring the original image from noisy data remained mainly unsolved, although generations of algorithms are making gradual progress in improving the quality of noisy images. A remarkable review of both classic and state-of-the-art image denoising methods is available at [1].

Increased interest in image denoising comes from technology advances which bring the problem closer to each of us. These days we are in the middle of the digital photography revolution. Most cameras sold today are digital. New film camera models are no longer designed by Canon, Kodak, and Nikon. Ricoh, Sanyo and Kyocera have stopped producing film cameras. This marks the end of film cameras in home photography and the completion of the revolution in the image-capture end. The revolution continues by transforming the image reproduction end as well. Digital printing is gaining territory from traditional silver-halide photographic paper development. The ease of use of digital technologies, either commercial prints through web access or home digital prints, increase the share of those technologies over traditional photographic paper development. Whether inkjet stays the technology of choice, or clear the way to another digital technology, digital is here to stay for a while.

With the new technology comes a set of new challenges, some of them unique to the digital era, and some that were inherited. One of those challenges, with have a mixture of old and new, is overcoming noise distortions. As already stated, noise is an inherent product of any imaging process. The most common noise in photography is of statistical nature and comes mainly from two sources:

1. Photonic noise – statistics of number of photons entering the image detector.
2. Thermal noise – “heat” photons counted by the image detector as “image photons”.

In addition to the above noise sources, additional noise categories, which are amplified in the digital environment, are:

3. Fixed pattern noise – difference between detector cells or “hot pixels” that appear mostly in low luminance environment which require long exposure.
4. Banding noise – caused by inaccuracies in detector cell positions. Highly camera dependant.

Reducing this noise at the image detector proves to be a very expensive task. For example, cooling the image detector is a common solution in night vision systems, but can hardly be afforded in the home (and even professional) photography market. The sensitivity of the detector cost (and therefore the equipment cost) to noise reduction characteristics creates high motivation for maintaining a considerable level of noise to be added to the image by the detector, and for the usage of image processing techniques to remove as much as possible of this noise along the imaging path.

Noise problem is not unique to the digital era or to natural images. The emphasis on digital photography of natural images in this work comes from the writer’s special interest in that field and from the opportunity to optimize parameters to this specific environment.

1.1. The problem

Non Local Means (*NLM*) is an innovative noise reduction algorithm for images presented by Buades, Coll and Morel in 2004 in [1]. It is inspired by Efros and Leung [2] use of weighted Euclidean distance for “texture synthesis by example”, a well known publication in the texture synthesis area. The basic idea of the algorithm is very simple: the denoised value of a pixel is a weighted average of all pixels in the image which have “similar neighborhood” to that pixel. The weight of each similar pixel in the average is set according to the level of “similarity” between the two pixels.

In their work, Buades, Coll and Morell conducted a very thorough presentation and analysis of image denoising techniques, and compare them to the *NLM* algorithm presented at the same paper [1]. It is proved to perform much better than any other known image denoising algorithm. This was confirmed in other (independent) studies provoked by the *NLM* publications [5] [6] [7] and by further work by the original authors [3][4].

Although the algorithm performs remarkably better than older generation algorithms in restoring the original image, it is inferior in terms of execution run-time. This prevents *NLM* from being used in mainstream consumer application. Actually, the time-complexity of the algorithm is so high that it is impractical to use in a digital photography environment where color images of 3-5Mpixels are a standard.

The challenge is therefore in enabling the use of concepts from *NLM* in mainstream applications by accelerating it to acceptable run-time levels.

1.2. Obtained results

Acceleration by factor of 4 to 25 was achieved depending on the noise level and image properties, normally with no degradation in image quality or with degradation lower than 10% in the *MSE*. Most of the proposed methods were proven to contribute to the acceleration, but the proposed clustering approach was found to be much less efficient than expected and could not deliver the expected benefits.

1.3. Thesis organization

The work is organized in the following way. Chapter 2 explains the basic concepts of noise and performance evaluation. The *NLM* algorithm is explained in details in Chapter 3, which also establishes the mathematical terms used throughout this work and analyzes the time-complexity of the algorithm. Chapter 4 presents an approach for optimizing the *NLM* execution-time based on various techniques. Some of the techniques are based on previous work on *NLM* and from other fields of image processing and some are based of properties of natural images and of the human vision system. The results obtained from various experiments of methods from chapter 4 are presented in chapter 5, which also presents the final algorithm proposal. In chapter 6 we discuss related work done in the area of *NLM* acceleration. Chapter 7 contains the conclusions learned during this work.

1.3.1. About viewing images

Please note that the images presented in this document are much better viewed on a computer screen than on a printed paper. The loss of resolution and the addition of half-toning, dot-gain and other artifacts, may cause loss of fine image details which are demonstrated in those examples. This is especially true in this work since most of those artifacts mask the noise that is added to most of the images. The reader is also encouraged to use the “zoom in” option of the viewing software to see fine image details that may escape the eye when viewed at the normal reading resolution.

2
Basic concepts

2.1.
Noise

Any image $v = \{v(i) \mid i \in I\}$ can be described as a composition of two images – the original image $u(i)$ and the noise image $n(i)$. The reduction of the noise component in that composition increases the image quality. Most image de-noising algorithms depend on a filtering parameter *h* which usually is a function of the noise standard deviation. The image can be described as a combination of two components, the smoothed image component and the noise component:

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

D_h is the de-noising method, $n(D_h, v)$ is the noise "guessed" by the method, and ideally $D_h v$ is smoother than *v*. Figure 1 is an example of the image and its noisy and filtered components.

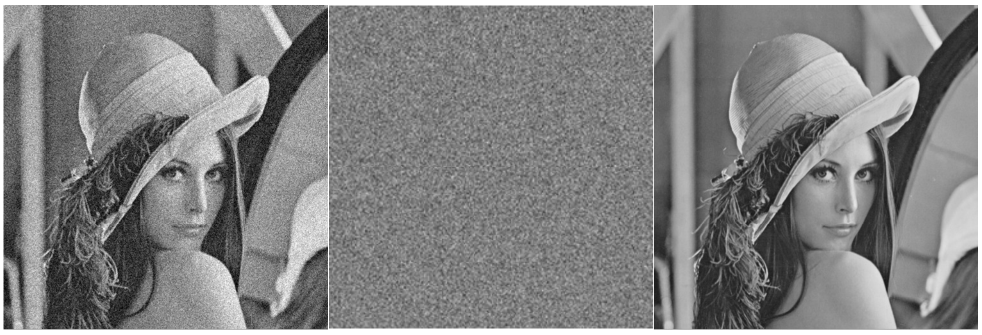


Figure 1 Image with additive white noise
left-Original image *v*; center-noise image *n(D$_h$,v)*; right-filtered image *D$_h$v*

A good measure for image quality in respective to noise is the *signal to noise ratio* (*SNR*) defined as the ratio between the signal and noise standard deviations.

$$SNR = \frac{\sigma(v)}{\sigma(n)} \text{ where } \sigma(v) = \sqrt{\frac{1}{|I|}\sum_{i \in I}(v(i) - \bar{v})^2}\ .$$

A good quality image has standard deviation of 60 or more (in images with pixel values of 0 to 255), and noise levels with standard deviation of up to 3 are usually unnoticeable by the average viewer. This makes $SNR \approx \frac{60}{3} = 20$ a good evaluator of image quality in terms of noise. In modeling noise, the noise values $n(i)$ and $n(j)$ of two different pixels are assumed to be independent random variables. This noise is called "white noise", a term used to describe noise with flat power spectral density. The use of the word "*white*" is compatible with the concept of white light which contains all the frequencies. Figure 2 has the original image on the left with standard deviation of about 50, the image in the middle has additive white noise with *SNR=20*, and the image on the right with *SNR=2*.

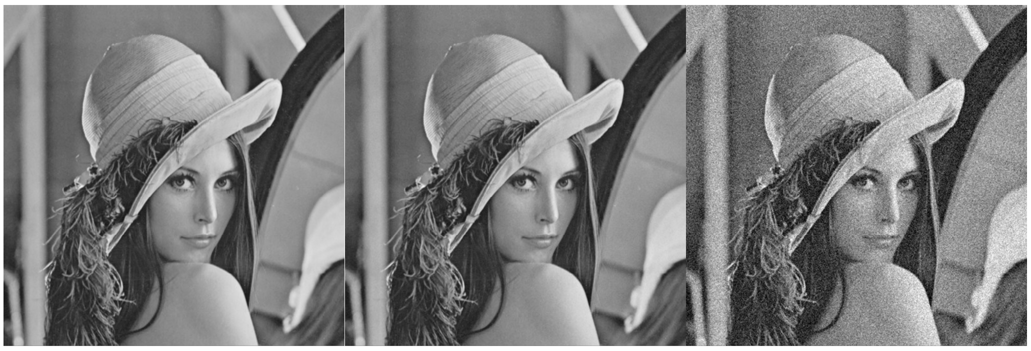


Figure 2 *SNR* as a quality measurement
left-original image; center-*SNR*=20; right-*SNR*=2

The noise in the middle image is unnoticeable as expected from *SNR=20*. In the right image, although distorted by heavy noise (*SNR=2*), all the original image details are noticeable. This unexpected fact hold the hope for finding successful denoising algorithms that can remove the noise but keep the original image details without distortion. The problem is that differentiating between noise and small image details is a difficult task. Denoising algorithms therefore tend to remove some image details together with the noise, causing distortions and artifacts.

2.2.
Performance evaluation

Our success criterion is to achieve a better price/performance ratio for the *NLM* algorithm. This means getting faster execution while minimizing perceived quality loss.

The execution acceleration can be measured easily comparing to the original algorithm execution time. Perceived image quality loss is a more problematic parameter to measure. The proposed measures for this parameter are:

1. Mathematical, based on ***mean square error*** (*MSE*).
2. Comparative images, based on the "***method noise***" presented by *NLM* writers in [1].

The method noise measure may be understood from the following example shown in Figure 3. The original image (top left) is added white noise with $\bullet_n$=10 (top right), and then filtered by Gaussian smoothing (middle left) and by *NLM* (middle right).

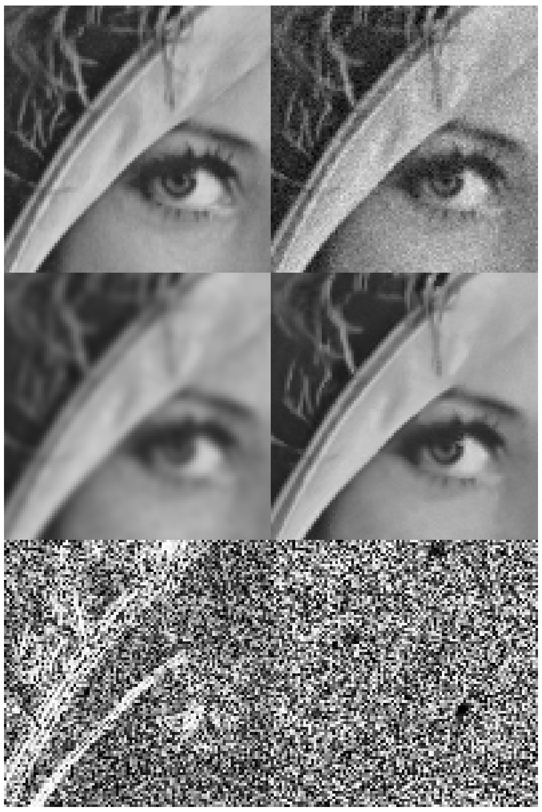


Figure 3 Method noise example

The absolute difference between the noisy image and the filtered image is the method noise image – bottom left for Gaussian smoothing method and bottom right for the *NLM* filter method. If only noise has been filtered out of the image, we expect the method noise to look like white noise, which is more or less the case for the *NLM* filter, but in the Gaussian filtering method noise image we can clearly see strong image features. This can only mean the filter has damaged the image details. This is also obvious from looking at the results of the two filters.

The Gaussian smoothing blurred the image completely by smoothing all the fast transitions in the image.

For high quality filtering, we require the method noise image to look as similar as possible to white noise image and to have as less as possible image structure in it. As can be understood, this measure is not quantified by a number. The only tool for evaluation is the human eye that can identify image structure in the presented method noise image.

2.3. Tools used for evaluation

Evaluation program with C-language library of *NLM* functions and *PERL* scripts was used to process the selected test images with the original and proposed algorithms. The scripts make use of the popular “ImageMagick®” (www.imagemagick.org) software suite for some image manipulation and analysis functions such as *MSE* calculation, histogram creation and image difference comparison.

3
Noise reduction

3.1.
NLM algorithm

As most noise reduction algorithms, the *NLM* algorithm uses averaging as a mean of getting rid of the random noise. The difference is that while most algorithms make use of the fact that close-by features of natural images tend to have similar values and therefore may be used to average, the *NLM* algorithm makes another assumption: natural images have repeating features and these may be found not only locally but globally. In order to remove the noise from pixel *p*, the algorithm looks for features similar to those surrounding *p* all over the image, and assigns a weight to each pixel according to the "similarity" of its neighborhood to the neighborhood of *p*. The filtering of *p* is therefore done by a weighted average of all the pixels in the image. Similarity can be explained using Figure 4:

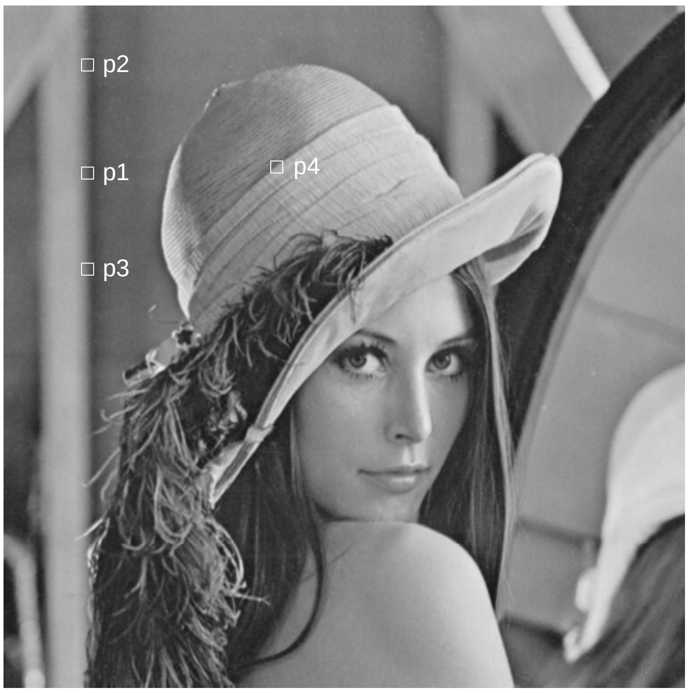


Figure 4 similarity

Similar pixel neighborhoods like those of *p₁*, *p₂* and *p₃* give high weight values $w(p_1, p_2)$ and $w(p_1, p_3)$ while very different neighborhoods like those of *p1* and *p4* produce low weight value $w(p_1, p_4)$. The values of pixels p_2 and p_3 will have much higher weight in the averaging of p_1 than the value of p_4.

Similarity is computed using ***weighted Euclidean distance*** of pixel neighborhoods. This metric was found effective by [2], and found to be consistent in a noisy environment since it increases the distance between two originally identical pixels by a constant. This will be demonstrated after the following definitions.

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

Equation 1. $$NL(v)(i) = \sum_{j \in I} w(i, j) v(j)$$

where naturally, a weight is between 0 and 1 and the sum of all weights is 1

$0 \le w(i, j) \le 1$ and $\sum_{j \in I} w(i, j) = 1$.

A neighborhood system on *I* is a family $N = \{N_i\}_{i \in I}$ of subsets of *I* such that for all $i \in I$:

1. $i \in N_i$
2. $j \in N_i \Rightarrow i \in N_j$

The subset N_i is called the ***neighborhood*** or the ***similarity window*** of i. Similarity windows may have different sizes and shapes, but for simplicity, a rectangular window is used. Similarity between two pixels i and j depends on the similarity of the intensity gray level vectors $v(N_i)$ and $v(N_j)$. In Figure 4, the neighborhoods of pixels p_1, p_2, and p_3 are the collection of all gray level values of pixels in the surrounding squares. See Figure 5 for example.

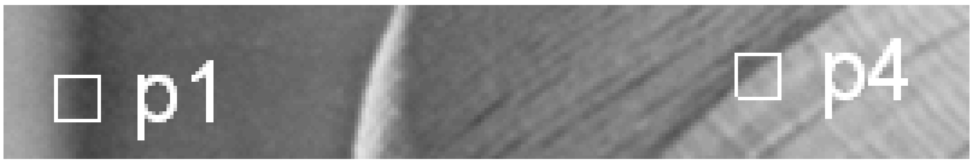


Figure 5 similarity neighborhood

Pixels with gray-level neighborhoods similar to $v(N_i)$, will have larger weights in the average of pixel i. Similarity is computed using a weighted version of the Euclidean distance. If ***Square Euclidean Distance*** (L^2) between two vectors

$$x = (x_1, x_2, \ldots, x_k),\ y = (y_1, y_2, \ldots, y_k)$$

is defined as:

Equation 2. $$d^2(x, y) = \sum_{j=1}^{k} (x_j - y_j)^2$$

the ***weighted square Euclidean distance*** (weighted L^2) is defined as:

Equation 3. $$d_a^2(x, y) = \sum_{j=1}^{k} a_j (x_j - y_j)^2 \quad and \quad a_j \geq 0$$

where the contribution of each axis *j* distance to the total distance is weighted by the coefficient a_j.

In the *NLM* algorithm, the weights $a_j,\ j = 1, \ldots, k$ are assigned using a two-dimensional Gaussian kernel. The kernel function is a two dimensional version of the normal distribution function and it is described by the following equation:

$$G(i, j) = \frac{1}{4\pi\sigma^2} e^{\frac{i^2 + j^2}{-4\sigma^2}}$$

The general form of the two dimensional Gaussian kernel is demonstrated on the right side of Figure 6, and a 5x5 kernel is plotted on the left. When applying the kernel to a neighborhood centered on pixel *p*, the center pixel (*p*) gets the heaviest weight and the other neighborhood pixels are weighted exponentially inversed to their distance from *p*.

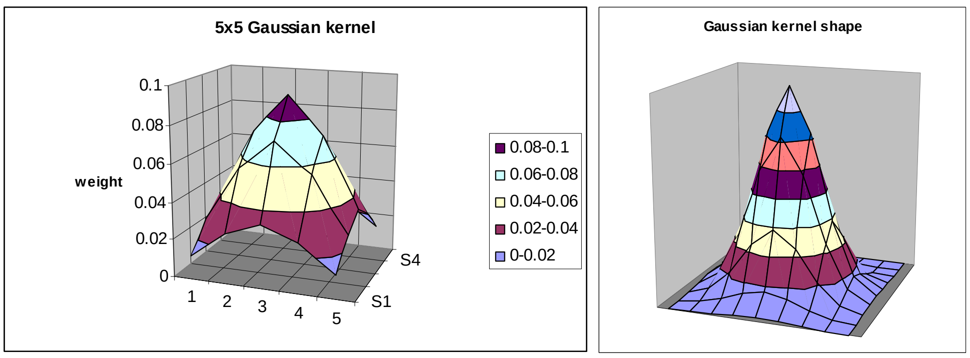


Figure 6 Gaussian kernel

In the above example, a 5x5 pixel neighborhood and Gaussian kernel are defined; d_a^2 is computed between two vectors of length 25. The 5x5 kernel is coefficient values are computed from the two dimensional Gaussian function with standard deviation $a: G_a(i,j) = \frac{1}{4\pi a^2} e^{\frac{i^2+j^2}{-4a^2}}$ for values of $i, j = -2,-1,0,1,2;$ The coefficients are normalized so that $\sum_{j=1}^{k} a_j = 1$ by $a_{i*5+j} = \frac{G_a(i,j)}{\sum_{i,j=(-2,\ldots,2)} G_a(i,j)}$

The corresponding R^{25} kernel vector in this case is described in Figure 7:

$\frac{1}{577}$ x

2	7	12	7	2
7	31	52	31	7
12	52	127	52	12
7	31	52	31	7
2	7	12	7	2

Figure 7 5x5 Gaussian kernel

Note the sum of all 25 vector components (after normalizing by $\frac{1}{577}$) is 1, and the weight of each pixel in the kernel is inversely proportional to its distance from the center.

After understanding the weighting policy, we can go back to the distance definition. Ni and Nj are similarity windows centered at pixels i and j correspondently and therefore

$v(Ni) = (v(Ni_1), v(Ni_2), \ldots, v(Ni_k))$ and $v(Nj) = (v(Nj_1), v(Nj_2), \ldots, v(Nj_k))$

are both intensity gray-level vectors with same cardinality of the Gaussian kernel. The weighted L^2 distance, d_a^2 may be written like this:

$$\| v(Ni) - v(Nj) \|_{2,a}^2 = \sum_{l=1}^{k} a_l \left(v(Ni_l) - v(Nj_l) \right)^2$$

Having defined the distance metric, we are ready to define the way weights are assigned to each pixel according to Equation 1. The weights associated with the distance are defined by:

Equation 4. $$w(i,j) = \frac{1}{Z(i)} e^{(-1)\frac{\|v(Ni)-v(Nj)\|_{2,a}^2}{h^2}},$$

where Z(i) is the normalized factor:

Equation 5. $$Z(i) = \sum_{j \in I} e^{(-1)\frac{\|v(Ni)-v(Nj)\|_{2,a}^2}{h^2}},$$

and the *decay factor h* controls the decay of the exponential function and therefore the decay of the weights as a function of the Euclidean distance. Note that *h* is a "filtering factor" in a way that it is responsible for setting the right weight assigned according to the distance computed. For small *h* values, a very small distance must be computed in order to have any contribution, and for high *h* values, even considerable distance may influence a pixel value. This means that if we choose *h* to be too high, we may distort the image, and if we choose *h* too low, we may not remove enough noise. Figure 8 demonstrates the fast decay of the weight function at low h values as a function of the distance. The upper curve, $h^2 = 20$, is slowly decaying with distance growth while the lower curve of $h^2 = 2$ falls sharply and produces very low weights for distances above $d_a^2 = 5$.

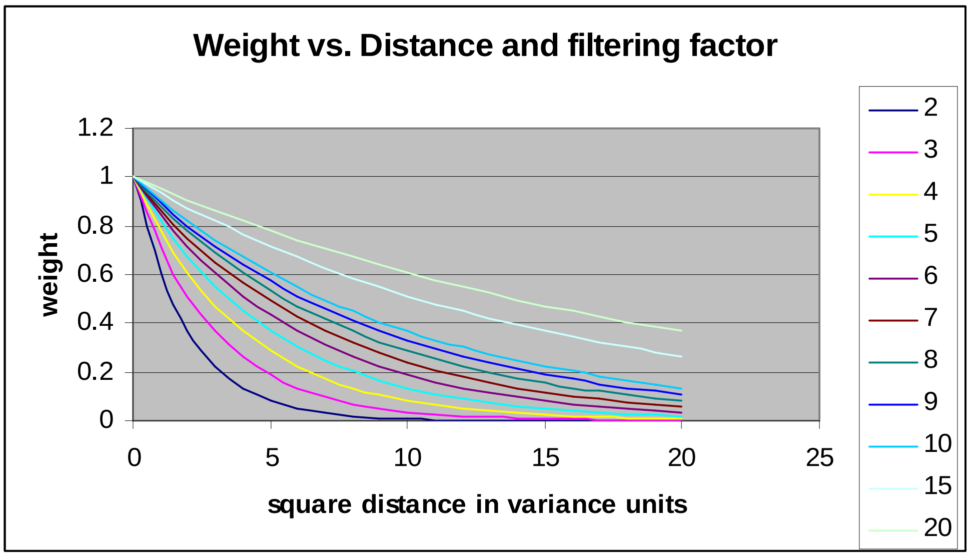


Figure 8 Weight as a function of the distance and decay factor

Figure 9 demonstrates the effect of the decay factor value on the filtered image. On the top left, a noisy image with $\bullet_n$=20. The three other images are *NLM* filtered with different decay factor values. The top right image is filtered with $h^2 = 200$, bottom left $h^2 = 100$ and bottom right with $h^2 = 2000$.

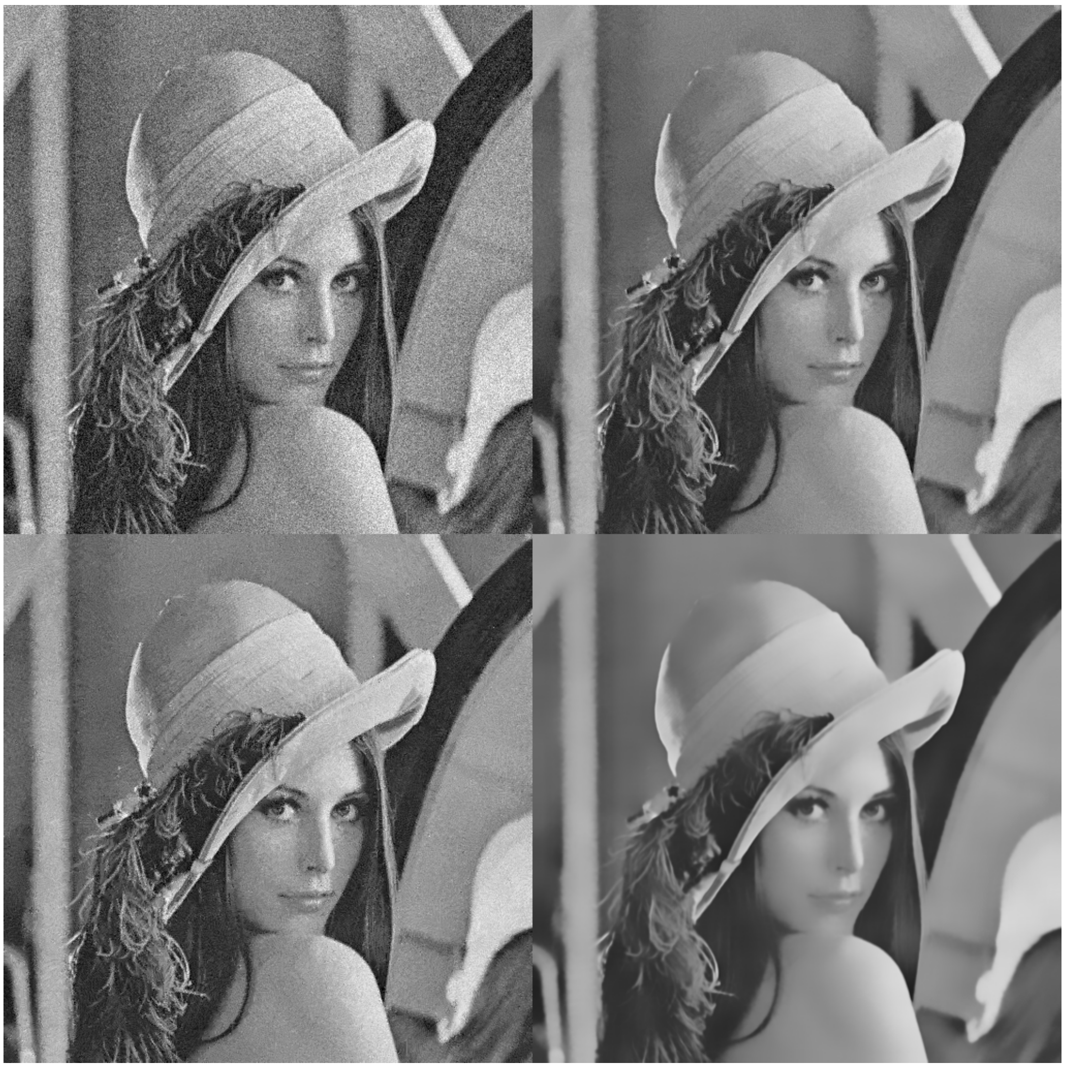


Figure 9 Effect of decay factor on the filtered image

The bottom left image is "under filtered" and a considerable level of noise still exists. The bottom right image is "over smoothed" and many of the image fine details have disappeared together with the noise. The top right image, filtered according to the original recommendations of $h^2 = 10\sigma_n$, presents a good tradeoff between detail loss and noise residues.

Note that for "white noise" with standard deviation σ_n and zero mean, the expected weighted L^2 distance value between two neighborhoods is given by:

$$E\left\|v(N_i)-v(N_j)\right\|_{2,a}^2 = \left\|u(N_i)-u(N_j)\right\|_{2,a}^2 + 2\sigma_n^2 .$$

The weighted L^2 distance of two noisy image neighborhoods has a fixed distance of $2\sigma_n^2$ from the distance of the same neighborhoods in the original image before the addition of white noise. If two originally similar neighborhoods are compared $\left(\left\|u(N_i)-u(N_j)\right\|=0\right)$, the noisy neighborhoods are expected to have a weighted L^2 distance of $2\sigma_n^2$. This shows that the weighted L^2 distance is actually consistent between the original and noisy image, adding twice the noise variance to the computed distance.

3.2. A bit on *NLM* for video

Now that *NLM* principle is understood, this is the time to mention that *NLM* is not specific to still images. *NLM* may operate in a similar way on frames of a film, using for averaging pixels from various other frames. This is logical since the probability of a series of consecutive frames to share a great amount of similar pixels is very high. Previous methods that were trying to use inter-frame averaging usually run into the “motion estimation” problem: the need to assess the relative movement between each two frames so that the filtered pixel could be tracked through the frames. *NLM* however, is free of motion estimation since the search for similar neighborhoods can be done on all the pixels in a given subset of frames, or in a search window fashion limiting the search in three dimensions instead of two. This algorithm has the same time complexity of the original algorithm multiplied by the number of frames that participate in averaging. The high complexity of this algorithm prevents it from being used in main stream video applications.

3.3. *NLM* time-complexity

The time-complexity of *NLM* can be written as $O\left(n^2 w\right)$ where n is the number of pixels in the image and w is the size of the neighborhood window. When considering use of *NLM* for a standard 3Mpixel color image, with three color planes and a small neighborhood window of size 25, the required number of operations reaches $3*(3*2^{20})^2*25=7.42*10^{14}$. This is clearly impractical to achieve in reasonable time. For this reason, *NLM* defines a *search window* in which the search for similar neighborhoods will be conducted. This relies on the assumption that in natural images close by features tend to be similar. This allows us to reduce the time-complexity to $O(nsw)$, when s is the size of the search window. Repeating the example above with search window of size 400, results in $3*(3*2^{20})*400*25=9.44*10^{10}$ operations. This is still in the neighborhood of 100Giga operations.

Note that by using the search window instead of searching similarity over the whole image, the modified algorithm is giving up on the possibility to filter similar image features, which are at a bigger distance than the search window size. In Figure 10, while filtering p_1, we can use the value of p_2 for filtering but not of p_3 since it is out of the search window range (marked as a white square around p_1).

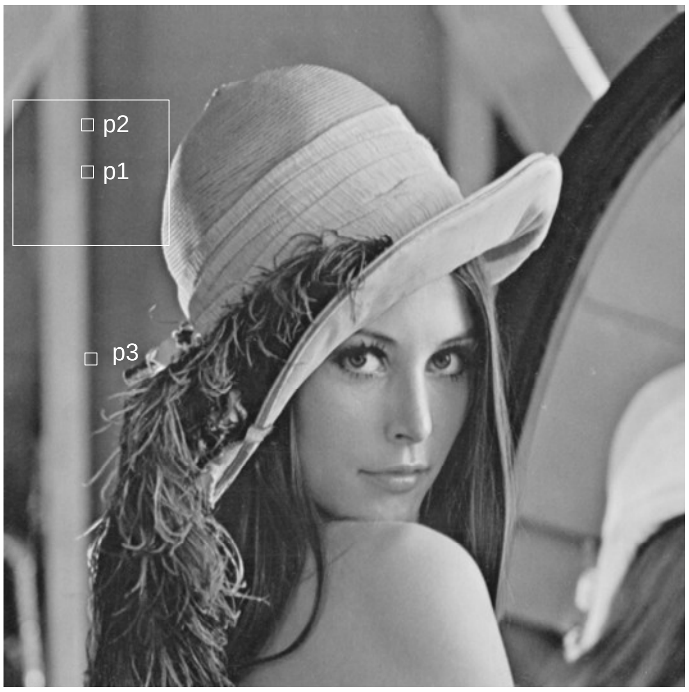


Figure 10 similarity

To conclude, even while sacrificing some of the original algorithm abilities, the modified algorithm does not succeed in reducing the time-complexity to an acceptable level.

Another straightforward way to reduce complexity is by reducing the similarity (neighborhood) window size. This may cause image distortions due to insufficient similarity information. The authors of [1] recommend a 9x9 or 7x7 similarity window for gray level images and 5x5 or even 3x3 similarity windows for low noise color images. We may deduce there isn’t much room for acceleration by sacrificing this important feature.

4
Optimizing *NLM*

This section details a proposal to optimize *NLM* using not only mathematical algorithmic approach, but also considering various image properties and characteristics of the human vision system (*HVS*) that were left out from the original *NLM* proposal. The following reviews the different approaches.

4.1.
Predicting Similarity Results

The high time-complexity of *NLM* comes from the need to perform the weighted L^2 distance computation on all pairs of pixels, or at least between each pixel and all the pixels in its search window. If we could have prior knowledge of comparisons that will provide a similarity so low that it's contribution to the weighted average of a pixel will be negligible, we could omit all computations of this pair of pixels. What we need is a simple test that can predict the results of the weighted L^2 distance computation.

A scheme similar to that is already in use in other image processing algorithms. This work tries to make use of the knowledge accumulated in those areas in the acceleration of *NLM*. Image coding by *vector quantization* (*VQ*) is dealing with the need to match similar pixel vectors, which can also be looked at as neighborhoods, and do it fast enough to enable video compression. The following is a review of some *VQ* acceleration methods and their adaptation to the *NLM* environment.

4.1.1.
Vector Quantization

VQ is an image compression scheme in which the image is coded by a limited number of vectors. This way we can have a compact description of the image based on a dictionary of vectors of pixels and an image pointing to the codebook instead of preserving the full image at original resolution. This is possible mainly because a natural image is build of features predominantly containing smooth gradients. Adjacent pixels of those features have similar values and quantizing those values to a codeword produces a very small error, which in most cases (depending on the quantization level) is invisible to the human eye. *VQ* is asymmetric in its nature. Compression is highly computational intensive mainly in building the codebook and partially in finding the best match for each vector. Decompression however, uses a single look up table access per vector and is therefore very fast. The following is a mathematical background review of *VQ* mapping of vectors and code-words:

We can define *VQ* as mapping *Q* of the *k*-dimensional Euclidean space R^k to a finite subset of codewords C of R^k. Thus: $Q : R^k \rightarrow C$ where $C = \{C_i : i = 1,2,...,N\}$ is the set of reproduction vectors called ***codebook*** of size *N*. Each *k*-dimensional image vector $x = (x_1, x_2,..., x_k)$ is compared with all codewords in the codebook according to the distortion measure of square Euclidean distance (L^2) as we have already defined in Equation 2, only this time the distance is measured between a pixel vector X and a codeword C_i

$$d^2(X, C_i) = \sum_{j=1}^{k} (x_i - c_{ij})^2 .$$

The *Q* mapping will match *X* with the best matching codeword C_{bm} that will satisfy the condition:

$$d^2(X, C_{bm}) = \min_{i=1,..,N} d^2(X, C_i).$$

Our interest in *VQ* comes from the second stage of compression – finding the best matching codeword for an input vector. As can be seen, *VQ* encoding requires a full search algorithm for finding the best matching codeword for each vector, while *VQ* decoding can be implemented as a simple lookup table. In order to speed up the encoding process, several algorithms have been proposed for the acceleration of the codeword matching process. All methods investigated in this work implement full search with a preliminary evaluation stage before starting the L^2 computations. The evaluation stage uses “selection parameters” that can rapidly predict the lower bounds of the L^2 computations. By evaluating the selection parameters, we are able to select appropriate codewords to be compared to each vector, and reject inappropriate codewords without performing the L^2 distance computations.

4.1.1.1. Equal-average Nearest Neighbor Search – *ENNS*

This method was proposed by Ra and Kim [8] and is common in *VQ* coding. It uses the fact that the square Euclidean distance between two vectors is usually in the neighborhood of the square means distance of these vectors. In simple words, if two vectors have a small distance then their means have similar values. The technique uses the average value of a vector as a selection parameter to reject code vectors not similar to the input vector and that way reduces similarity computation considerably.

When k is the dimension of the vectors, X is the input pixel vector and C is the collection of code vectors, we can write the following inequality from [8]:

Equation 6. $$d^2(X, C_i) \geq k(\overline{X} - \overline{C_i})^2$$

The L^2 distance will be greater or equal to the vector dimension multiplied by the average difference square. The search for the codeword will work by calculating the distance of the first codeword, setting it as a minimum and calculating L^2 distance only for those codewords that may provide a lower distance according to the average inequality.

Experimental results ([11], [12]) show reduction of 81-95% in number of computations due to the high rejection rate in the selection process. This leads to reducing the compression time by a factor of 4 to 15.

4.1.1.2. Equal-average Equal-variance Nearest Neighbor Search – *EENNS*

This method was proposed by Lee and Chen [9] and is also very common in *VQ* coding. Two vectors with the same mean can still have a big distance from each other. By taking into account the variance difference between two vectors as a secondary selection parameter, we can reject many of those codewords with similar mean and big distance. In conjunction with *ENNS* criterion, we use the following criterion from [9] to reject some of the candidates that passed through the first filter.

Equation 7. $d^2\left(X,\overline{C_i}\right) \geq k\left(V_x - V_{C_i}\right)^2$

where V_x^2 represent the variance of the vectors, defined as:

$$V_x^2 = \frac{1}{k}\sum_{j=1}^{k}\left(x_j - \overline{x}\right)^2$$

The L^2 distance will be greater or equal to the vector dimension multiplied by the standard deviation difference square. The search for the codeword will work by calculating the distance of the first codeword, setting it as a minimum and calculating L^2 distance only for those codewords that may provide a lower distance according to *ENNS* and *EENNS* inequalities.

Note that while all *ENNS* and *VQ* operations are relatively simple, *EENNS* requires also square root computation. Even so, the additional computation is worthwhile since higher rejection rate can be achieved and much smaller number of codewords needs to be computed for L^2 distance.

Experimental results [11], [12]) show reduction of 89-98% in number of computations due to the high rejection rate in the selection process. This leads to reducing the compression time by a factor of 6 to 25.

4.1.1.3. Improved Equal-average Equal-variance Nearest Neighbor Search – *IEENNS*

In addition to the two inequalities presented by *ENNS* and *EENNS*, a third one was presented by Baek, Jeon and Sung [10]:

Equation 8. $d^2\left(X,C_i\right) \geq k\left(\overline{X} - \overline{C_i}\right)^2 + k\left(V_x - V_{C_i}\right)^2$

Notations are similar to *ENNS* and *EENNS*, using average and standard deviation measures as selection parameters to rapidly compute the lower bound for the L^2 distance. This improvement uses the same information as *EENNS* without any additional memory, and manages to combine the two statistical measures of average and standard deviation presented in Equation 6 and Equation 7 into a single elegant inequality. The sum distance limit is of course higher than each of the distances set by previous methods and thus higher rejection rate is achieved.

Experimental results [11], [12]) show reduction of 92-98% in number of computations due to the high rejection rate in the selection process. This leads to reducing the compression time by a factor of 8 to 29.

4.1.1.4. Additional *VQ* methods and the noisy environment

Additional *VQ* acceleration techniques have been considered for the purpose of *NLM* acceleration. Among them the algorithms suggested by Pan Lu and Sun [11] and Wang and Tang [12]. Those algorithms are based on increasing the rejection rate by dividing the vector into sub-vectors and using the previously explained methods (*IENNS* etc.) on the sub-vector.

The use of these techniques in *NLM* acceleration may be further investigated but is left out of the scope of this work.

4.1.2. Use of *IEENNS* for *NLM* acceleration

The resemblance between *VQ* and *NLM* similarity measures makes *VQ* acceleration methods obvious candidates for *NLM* acceleration. It may be possible to reduce the number of weighted L^2 distance computations by rejecting vectors with low limit on the weight contribution to a specific pixel. The main differences between the *VQ* and *NLM* algorithms are:

1. *VQ* uses L^2 distance metrics while *NLM* uses ***weighted*** L^2 distance metrics.
2. *VQ* compares a vector to a set of code-words from a codebook, while *NLM* compares **all** combinations of two vectors within a set or sub-set.

In order to take advantage of the acceleration algorithm developed for *VQ* in the *NLM* environment, we need to develop new inequalities to match the weighted L^2 metrics, and new selection parameters for this metrics. Following is a mathematical adaptation of L^2 metrics used in *IEENNS* to the weighted L^2 metrics used in *NLM*.

4.1.2.1. Metrics Definitions

When $x = (x_1, x_2, ..., x_k)$ and $y = (y_1, y_2, ..., y_k)$ are vectors in R^k the L^2 distance is given by previously defined Equation 2. $d^2(x,y) = \sum_{j=1}^{k}(x_j - y_j)^2$.

The mean and variance are defined by Equation 9 and Equation 10:

Equation 9. $\bar{x} = \frac{1}{k}\sum_{j=1}^{k} x_j$

Equation 10. $V_x^2 = \frac{1}{k}\sum_{j=1}^{k}(x_j - \bar{x})^2$

and the ***weighted*** L^2 distance was already defined in Equation 3.

$$d_a^2(x,y) = \sum_{j=1}^{k} a_j (x_j - y_j)^2 \quad and \quad a_j \geq 0 .$$

Recall the *NLM* algorithm restricts the sum of the coefficients to be 1 ($\sum_{j=1}^{k} a_j = 1$), this restriction however, is not relevant for this metrics.

4.1.2.2. Adaptation of *IEENNS* to the weighted L^2 distance metrics

In this section we shall look for selection process parameters similar to the mean and standard deviation parameters used by *IEENNS*. Those parameters will provide us with fast evaluation of the prospects of the L^2 distance computation to provide us with any significant weight for the averaging process.

From the weighted L^2 distance definition (Equation 3) we can easily get back to the L^2 distance (Equation 2).

$$d_a^2(x,y) \equiv \sum_{j=1}^{k} a_j (x_j - y_j)^2 = \sum_{j=1}^{k}\left(\sqrt{a_j}\,x_j - \sqrt{a_j}\,y_j\right)^2 \equiv d^2(x_a, y_a)$$

by defining $x_a = \left(\sqrt{a_1}x_1, \sqrt{a_2}x_2, ..., \sqrt{a_k}x_k\right)$ and $y_a = \left(\sqrt{a_1}y_1, \sqrt{a_2}y_2, ..., \sqrt{a_k}y_k\right)$ as the "*root weighted vectors*".

We have found that the weighted L^2 distance between two vectors is equal to the ***non-weighted*** L^2 distance between the root weighted vectors. Now we can use the *VQ* acceleration inequalities for the weighted *L*^{2} metrics. All we need is to find the corresponding selection parameters (mean and standard deviation) definition for the root weighted vectors. In this space the ”*root weighted mean*”, according to Equation 9 will be:

Equation 11. $$\bar{x}_a = \frac{1}{k}\sum_{j=1}^{k} x_{aj} = \frac{1}{k}\sum_{j=1}^{k}\sqrt{a_j}\,x_j$$

and the *”root weighted variance”*, according to Equation 10, will be:

Equation 12. $$V_{x_a}^2 = \frac{1}{k}\sum_{j=1}^{k}\left(x_{aj} - \bar{x}_a\right)^2 = \frac{1}{k}\sum_{j=1}^{k}\left(\sqrt{a_j}\,x_j - \bar{x}_a\right)^2$$

and the *IEENNS* inequality (Equation 8) for the weighted metrics will be:

$$d_a^2(x,y) = d^2(x_a, y_a) \geq k(\bar{x}_a - \bar{y}_a)^2 + k\left(V_{x_a} - V_{y_a}\right)^2$$

therefore, the weighted *L*^{2} distance between two vectors cannot be smaller than:

Equation 13. $$d_{a\,\min}^2(x,y) = k(\bar{x}_a - \bar{y}_a)^2 + k\left(V_{x_a} - V_{y_a}\right)^2$$

From Equation 11, Equation 12 and Equation 13 we can write:

Equation 14. $$d_{a\,\min}^2(x,y) = (x' - y')^2 + \left(V_x' - V_y'\right)^2$$

defining the selection process parameters as:

Equation 15. $$x' = \frac{1}{\sqrt{k}}\sum_{j=1}^{k}\sqrt{a_j}\,x_j$$

Equation 16. $$V_x' = \sqrt{\sum_{j=1}^{k}\left(\sqrt{a_j}\,x_j - \frac{1}{\sqrt{k}}x'\right)^2}$$

The minimum weighted *L*^{2} distance is therefore given by the *L*^{2} distance between the two corresponding points in the (x', V_x') selection parameters space. The equations for the minimum weighted *L*^{2} distance and the selection parameters (Equation 14, Equation 15 and Equation 16) will accompany us through this work as the basis for the “Minimum distance filter” (4.1.2.3) and for the suggested “Clustering scheme” (4.2.1).

4.1.2.3. Minimum distance filter

We now have a metric that can be used to set a lower bound on the weighted L^2 distance of two neighborhoods. This of course sets an upper bound on the weight contribution of the pixels in their mutual filtering.

By calculating and saving the $(\bar{x}_a, V_{xa})$ parameters for each of the pixels in the image, we can determine the lower bound on the distance between the pixels. From Equation 6, we can find the maximum weight contribution expected from this comparison:

$$w_{\max}(x,y) = e^{(-1)\frac{d_{a\min}^2(x,y)}{h^2}}$$

Or in short

$$w_{\max} = e^{\frac{-d_{a\min}^2}{h^2}}$$

We can now set a threshold value for comparisons. If w_{TH} is the lowest acceptable weight we are willing to invest a weighted L^2 computation for, we shall compute weighted L^2 distance only if:

Equation 17. $$d_{a\min}^2 = (x' - y')^2 + (V_x' - V_y')^2 < -h^2 \ln(w_{TH})$$

Remember that the expected square distance between two identical non-noisy neighborhoods is $2\sigma_n^2$. Following that logic, we shall search for a limiter proportional to σ_n^2. We need to remember that the expected weighted L^2 distance may be much larger than the lower bound we have found with Equation 17, so setting a low enough threshold can eliminate most cases of non-contributing computations, without blocking computations that will yield in some contribution even if somewhat higher than the threshold we are setting.

4.1.3. Gradient filter

Let us first understand the meaning of the gradient function in the context of images. If $I(x,y)$ is a two dimensional image intensity level function, then the gradient of $I(x,y)$, marked $\nabla I(x,y)$, provides us with information about the steepness of the $I(x,y)$ function at any point (x,y). The direction of the steepest slope at each point of $I(x,y)$ is given by the gradient angle, while the gradient magnitude tells us how steep the slope is. The gradient is calculated using horizontal and vertical high-pass filter masks. For example, the horizontal and vertical kernels used for a 5x5 gradient computation are:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \end{bmatrix}.$$

The gradient magnitude is computed by the derivations in both horizontal and vertical directions using convolution operation with the above kernels. If the result of the convolutions are $V(x,y) = \dfrac{\partial I(x,y)}{\partial x}$ and $H(x,y) = \dfrac{\partial I(x,y)}{\partial y}$ then the gradient magnitude is

$$\left\| \nabla I(x,y) \right\| = \sqrt{H(x,y)^2 + V(x,y)^2}$$

and the orientation is given by:

$$\angle \nabla I = \arctan \frac{H(x,y)}{V(x,y)}$$

Figure 11 is an example of image gradient. The left is the original image, the center is a representation of the image gradient magnitude, and the right shows the image gradient orientation. Low image gradients are darker than the steep slopes, and image angle is quantized into four gray levels showing gradients pointing left, right, up and down. Note that flat areas have a very low (dark) gradient magnitude and an unstable orientation, while steep gradients are defined by white lines in the magnitude image and well defined orientation.

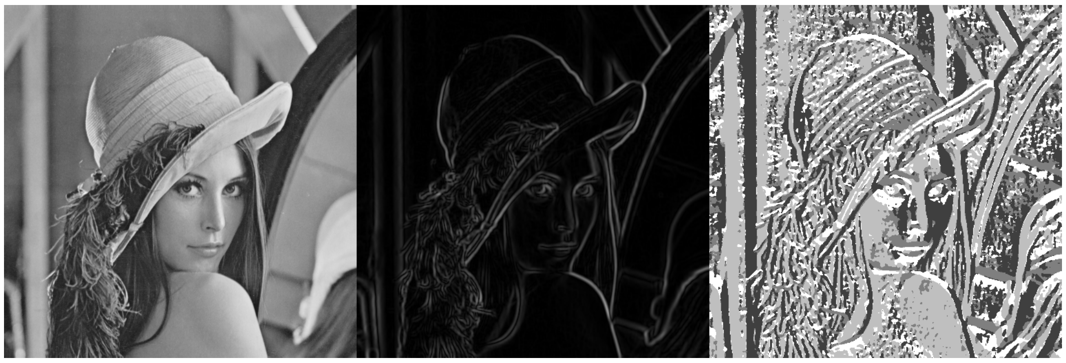


Figure 11 Image gradient example

Now that gradient is explained, let us understand how we can use the gradients of pixel neighborhoods to reject unnecessary weighted L^2 distance computations. Although no exact lower bound equation can be presented in this case (as in the case of the minimum distance filter), it is intuitive that two neighborhoods of significant gradient magnitudes are not likely to produce significant weight if the angle between their gradient orientations is too big. A filter based on this observation was used by Mahmoudi and Sapiro [5], which calculated the average gradient magnitude and average gradient orientation for each pixel neighborhood. When σ_∇ is a threshold on gradients magnitude and σ_θ is a threshold on the angle between gradients orientations, the actual filter they used to reject comparisons of pixels *i* and *j* is given by:

If $\|\nabla v(i)\| > \sigma_\nabla \;\; and \;\; \|\nabla v(j)\| > \sigma_\nabla \;\; and \;\; \theta(i,j) > \sigma_\theta$ then reject comparison between the neighborhoods of *i* and *j*.

The gradient filter used in this work uses local gradient values instead of the average values used in [5]. We have found the local gradient to be more relevant to the weighted L^2 distance result than the average gradient. This is a complementary filter to the minimum distance filter which allows filtering away weighted L^2 computations between neighborhoods of similar root weighted mean and variance, but are rotated at a certain angle and are not expected to produce any significant weight.

4.2. Clustering techniques

Another way to eliminate redundant computations is by clustering the pixel neighborhood into groups of already similar pixels. The clustering scheme is suggested as a preliminary step before computing similarities. Similarities will be computed only within a cluster of proximities, saving the vast majority of irrelevant computations. This is similar to the scheme suggested by Mahmoudi and Sapiro in [5]. They suggest clustering by the neighborhood mean and gradient and testing similarities inside and between adjacent blocks. The proposed method here will also use clustering into blocks, but will also provide exact limits to the expected contribution of a pixel within a block.

When using clustering we need to consider the following factors:

1. The clustering process must have lower time-complexity so it will not penalize the overall algorithm.
2. The clustering process should create an associated neighborhood for each pixel, in which the original *NLM* algorithm will operate.
3. The clustering neighborhood should have high number of similar pixels.

4.2.1. Clustering scheme

The clustering scheme uses classification of each pixel neighborhood according to parameters that can be used to predict the outcome of the weighted L^2 distance computation. The two chosen parameters are the selection process parameters defined in "4.1.2.2. Adaptation of *IEENNS* to the weighted L^2 distance metrics" section.

Figure 12 shows classification of "*lena*" noisy image neighborhoods according to the x' and V_x' parameters. The horizontal axis represents the root weighted mean (x') while the vertical represents root weighted standard deviation (V_x'). Each white point represents the location of one or more neighborhoods in the image. Each rectangle in the grid represents a bucket of neighborhood vectors. The diagonal line marks the locations of "flat zones", areas where the local intensity level variance (not weighted variance) within the neighborhood is low. Since "lena" is a natural image with many flat zones, we can see that most of the neighborhoods in the image are along this line, and the bucket density drops as the distance from the line grows.

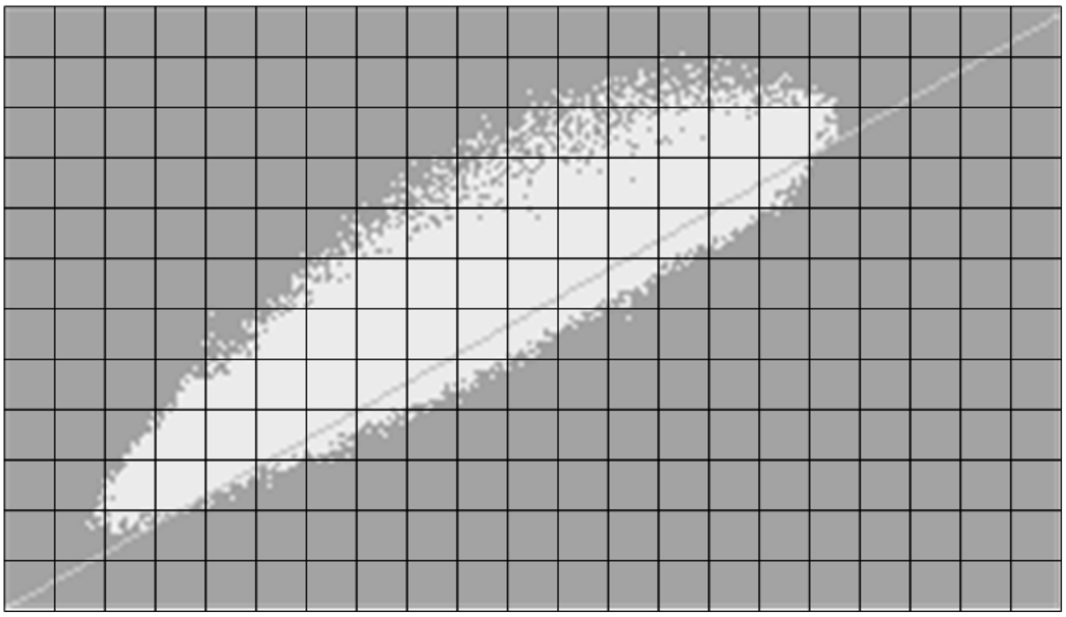


Figure 12 Classification by root weighted mean and root weighted standard deviation

We can now compute the lower bound of the weighted L^2 distance between **any** two pixel neighborhoods in the “lena” image by finding the L^2 distance between the two corresponding points in Figure 12. To do that effectively, we need to sort the image vectors into buckets. Each bucket containing only pixels with similar root weighted mean (x') and root weighted standard deviation (V'_x). The proposed organization is in the form of a two dimensional table pointing to linked lists of similar pixels. The entries to the table are the quantized x' and V'_x values. Given a list of pixels with same (x', V'_x) values, access to lists with pixels within a known distance bound can be done in $O(1)$ time-complexity. Populating the table can be done as a preliminary stage and the time-complexity of this stage is linear to the number of pixels in the image.

A search for similar neighborhoods can now be done in the clustering table instead of the original image. This reduces the original time-complexity from quadratic on the number of pixels in the image to quadratic on the number of pixels in related lists. This of course, might have worst case time-complexity equal to the original one (in case of a totally flat image for example), but in typical case of natural images the reduced complexity model is much more probable. Linear time-complexity can be forced by fixing a maximum size search window in the clustering table domain, the same way it was done in the *NLM* search window approach.

Note that searching in the clustering table domain brings us much closer to the original idea of searching for matching features in the whole image. This principle was sacrificed in the search window version of the original *NLM* to ensure acceptable time-complexity. Clustering may fit many high detailed image features into the same or near-by buckets, even if geographic distance between those features is bigger than the original search window size.

4.3. Properties of Natural Images

Natural images tend to have large flat zones. These are areas with small image gradient and small variance between neighboring pixels. In the proposed classification scheme, pixels in these areas are likely to produce similar x' and V'_x values. Since those are typically large zones, we expect the distribution of the classification table to be of much higher density in the flat zones than in the high contrast areas. This can be seen as both a problem and an opportunity. On one hand we have many pixels to participate in averaging a noisy area, on the other for faster execution we prefer smaller image sets.

As for the flat zones, a search for similar neighborhoods in a small geographical distance may yield in most cases better results than a search through the pre-classified table. In this case, the original *NLM* search window approach may really have an advantage. In the scarce high contrast areas, the clustering scheme should have an advantage over the geographical search. This is also a conclusion that can be obtained from [5], which reports an advantage to the original *NLM* algorithm in flat areas and an advantage to gradient/mean clustering in the high contrast areas. For this reason, the proposal is to keep the geographic search for the flat zones, and use cluster based search for the high contrast zones.

There are other ways to make use of the reviewed properties, but in order to make use of these properties in the proposed algorithm; this information needs to be considered in relation to some of the human vision system characteristics. Combining the two issues with *NLM* parameters have a potential of improving both *NLM* execution time and perceived output quality.

4.4.
Human Vision System considerations

Two major factors dominate the quality perception of a natural image by the human eye: blur and noise. The two of them need to be taken into consideration in any attempt of improving image quality through image processing. In many cases the two factors are inter-related and improving one of them may degrade the other. For example, in many noise reduction techniques, the averaging of adjacent pixels is a great tool for noise reduction, but it is accompanied by a strong blurring effect that degrades image quality. On the other hand, a convolution high-pass filter that is meant to increase image sharpness (reduce blur), gives a great boost to any noise in the image.

Following is a review of issues concerning *HVS* sensitivity to noise and blur, that need to be attended, and may open up opportunities in approaching *NLM* acceleration.

4.4.1.
HVS noise sensitivity

The Human Vision System (*HVS*) sensitivity to noise is highly dependent on the background image activity. A smooth image is much more sensitive to noise than an image with many details. Winkler and Süsstrunk [13] showed linear relationship between the noise standard deviation and the minimal image standard deviation needed for the standard viewer to notice that noise. In simple words, the same level of noise which is noticeable on a flat image may be “masked” by the details of a high contrast image.

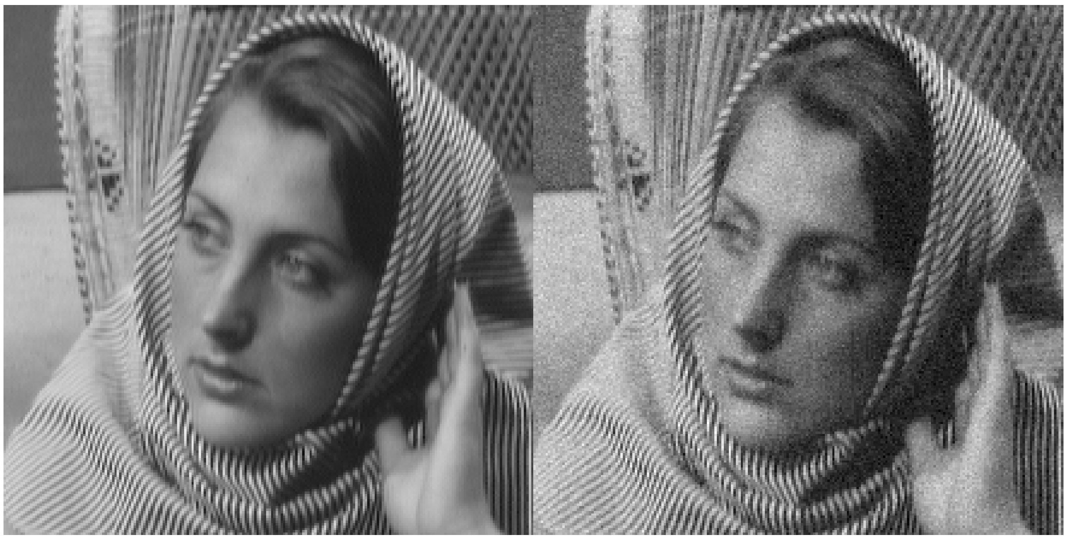


Figure 13 Noise in flat and high contrast areas. Left - original images, Right – added noise with $\sigma_n = 20$

In this example (Figure 13) our vision attention is drawn to the noisy face but the same level of noise that exist in the stripy head cover needs much more of our attention to be noticed.

The concept of “noticeable noise” is used in digital image watermarking, a technology to add visible or invisible data on top of images. An example of digital watermarking is given in Figure 14.

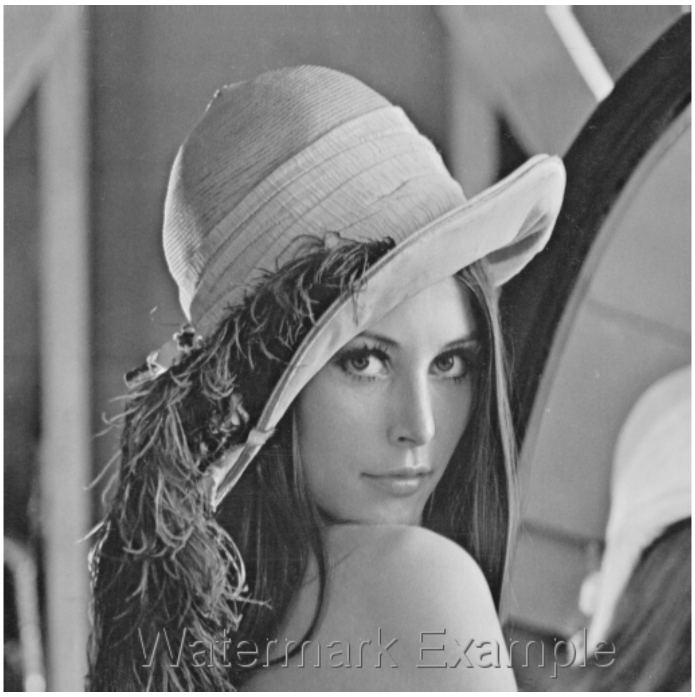


Figure 14 Digital watermarking

Figure 14 shows an example of a visible watermark. In this kind of watermarks it is essential to ensure their visibility on changing background. Other watermarks are invisible. In those applications it is important to keep the watermark image “masked” by the image.

In watermarking it is common to use a parameter called the *Noise Visibility Function* (*NVF*) which uses Gaussian model to estimate how much texture there is on any area of the image. This function, suggested by Voloshynovskiy et al. in [14] is defined

$$NVF(i) = \frac{1}{1+\theta\sigma_x^2(i)}$$

where $\sigma_x^2(i)$ is the local variance of the image around pixel *i*, and θ is a tuning parameter defined as $\theta = \frac{D}{\sigma_{x\max}^2}$ where $\sigma_{x\max}^2$ is the maximum image variance $\sigma_{x\max}^2 = \max_{i\in I}\sigma_x^2(i)$ and *D* is an experimental parameter between 50 and 100. For high contrast image sections, which are characterized by high variance, the *NVF* is close to 0, while for flat zones, *NVF* is close to 1.

4.4.2. *HVS* blur sensitivity

HVS contrast sensitivity depends on the spatial frequency. Figure 15, by Izumi Ohzawa [15] is an image of lines alternating at spatial frequency increasing from left to right, and contrast increasing from top to bottom.

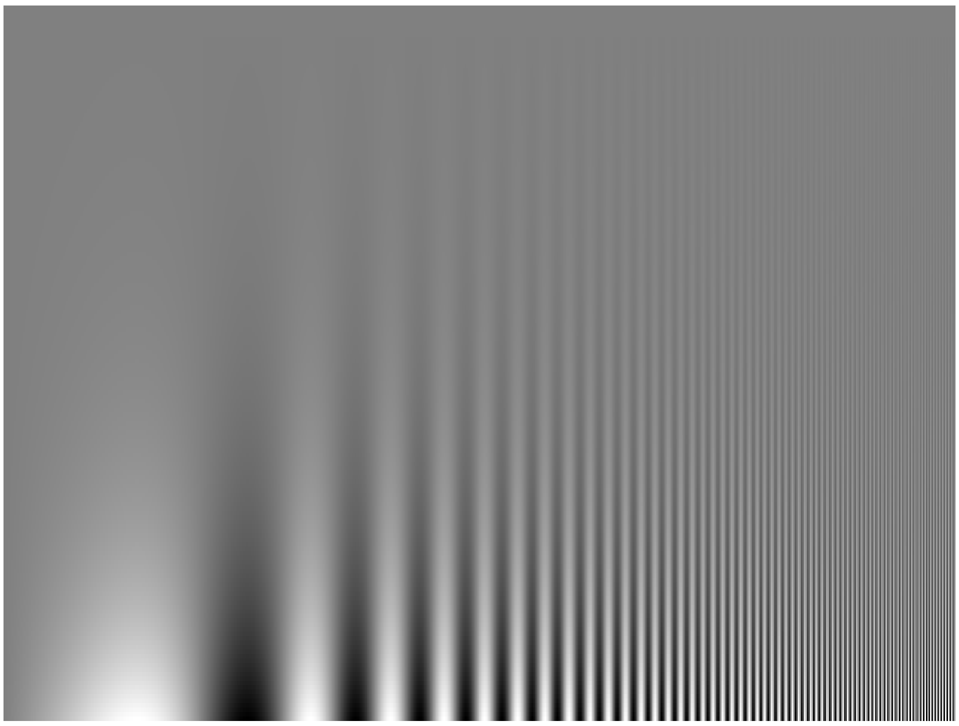


Figure 15 *HVS* contrast sensitivity

It can be seen that lines are easier to see through the whole height of the image in the middle section. This brought Campbell and Robson [16] to present the contrast sensitivity function which may change from viewer to viewer but has the general form showed in Figure 16.

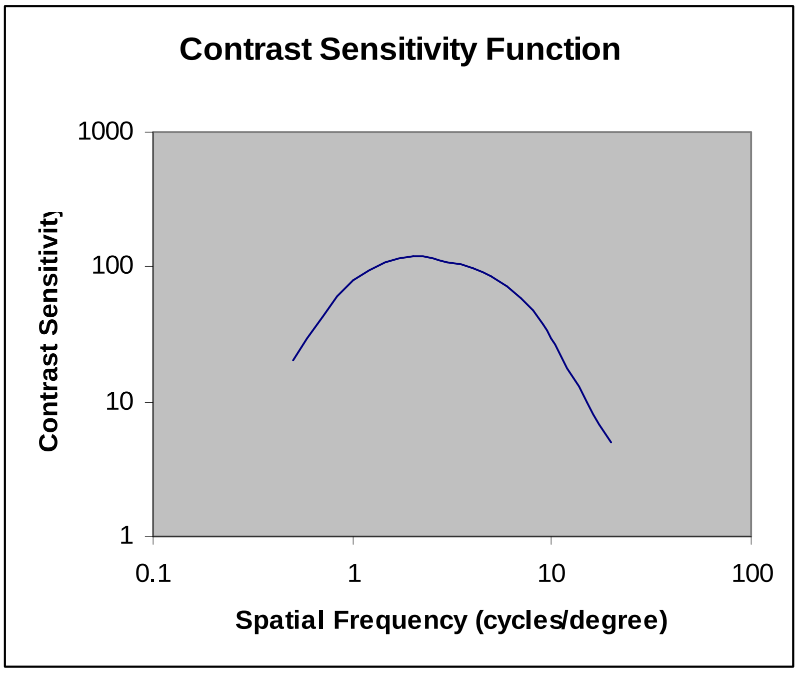


Figure 16 Contrast Sensitivity Function

A side effect of average pixels in the process of noise reduction is decrease of contrast. As seen from the contrast sensitivity function, at certain spatial frequencies this can cause the loss of details.

Johnson and Fairchild [17] conducted a large scale psycho-physical experiment to study *HVS* perception of sharpness. Sharpness is part of the subjective image quality criteria. They have found that image resolution is the main factor in sharpness perception. An interesting finding is that low levels of additive noise increase image sharpness perception.

It is important that during the noise filtering process, the image sharpness perception will not be damaged too much. It is also important to keep small image details intact. This is actually one of the main reasons for *NLM*, the ability to filter similar pixels makes sure averaging keeps the values close to the original, but using a high filtering factor on high resolution areas may degrade the sharpness perception.

4.4.3. A few words about color

When dealing with color images, all above discussion about noise and blur sensitivity still holds, but with varying levels in different image channels depending on the color scheme used. The luminance channel of *CIELab* space is much more sensitive to noise and blur than the chromatic channels. The Green channel of *RGB* is more sensitive than Red and Blue channels; The Yellow channel of *CMYK* is the less sensitive than the Black Magenta and Cyan channels.

With that information in mind, it is clear that if operating *NLM* on single color dimension we need to take into account the specific sensitivity of each color channel. Possible optimizations in color images filtering may be achieved by adopting techniques from other color processing algorithms such as JPEG compression. Analyzing similarity in the luminance channel for example and applying results to all other channels may be the way to go.

This work will not try to achieve additional advances in the area of color, based on the assumption that what is achieved for gray images should function just as well, if not better in color images.

4.5. Tuning *NLM* parameters

Some of the suggestions made in this work for improving *NLM* are related to controlling *NLM* parameters. The first one is the decay factor, already defined by *NLM*. The others parameters are introduced in the following sub-sections.

4.5.1. Decay factor

The important role of the *decay factor* (*h*) can now be better understood after reviewing the *HVS* sensitivity to noise and sharpness. As seen in section 3.1, the weights are assigned according to Equation 4 which can be written as:

$$w(i,j) = \frac{1}{Z(i)} e^{(-1)\frac{d_a^2(N_i,N_j)}{h^2}}$$

For a fixed weighted L^2 distance between two neighborhoods, h^2 will determine the weight of pixel *j* in averaging of pixel *i*. A high h^2 value will cause high filtering level through high weights in the average, and low h^2 value will better preserve image details through low weights in the averages. A good tradeoff should be achieved to have just the right filtering level. A lower level will not remove enough noise and a higher level will blur details away. The authors of [1] recommend on using a filtering factor of the order of the noise standard deviation. In their example, a value of $10\sigma_n$ is used. The relationship between noise level and filtering level is obvious, but there is no compensation for detail loss caused by high filtering level. The effect of the *decay factor* is summarized in Table 1.

h	Noise reduction	Image blur
Low	bad	good
High	good	bad

Table 1 *decay factor* influence

Table 2 summarizes the sensitivity of the *HVS* to noise and to blur in flat areas and high contrast areas.

HVS sensitivity	Flat area	High contrast area
Noise	high	low
Blur	low	high

Table 2 *HVS* sensitivity to noise and blur

The obvious conclusion is that by adapting the decay factor to each pixel's contrast level, we can optimize *NLM* quality. The first proposal of this work in relation to *NLM* parameters is to use a variable filtering factor instead of the fixed one. The variable filtering factor is linear to the standard deviation of the noise as suggested in [1], but should also be a function of the contrast of the neighborhood of the filtered pixel. In general: $h_i^2 = \sigma_n^2 f\left(\sigma_{Ni}^2\right)$ when σ_n^2 is the noise variance and σ_{Ni}^2 is the neighborhood variance of the filtered pixel.

At this point it is essential to understand that calculating the variance of the noisy image, produces the sum of the original image variance and the noise variance:

$$\sigma_i^2(v) = \sigma_i^2(u) + \sigma_n^2$$

This should be taken into account and adjusted before using σ_{Ni}^2 in any calculation.

4.5.2. Weight objective

As explained, when clustering pixels into blocks, we expect the blocks containing flat neighborhoods to be highly populated while high contrast neighborhood blocks to be poorly populated.

On one hand, this is exactly what we need in terms of noise reduction, a high number of averaging pixels for flat area. On the other hand, since the algorithm complexity is linear to the number of pixels in a proximity block, the execution of each of the pixels in a block could take long time if it will not be limited somehow.

The proposed mechanism for stopping the filtering algorithm is to filter each pixel to a point that a certain weight is achieved. Going back again to *HVS* noise sensitivity, this level should also be a function of the filtered pixel neighborhood contrast $w_{THi} = f\left(\sigma_{Ni}^{2}\right)$.

Low frequency areas will be filtered to a high weight objective with a low decay factor allowing for higher filtering level in an area sensitive to noise and less sensitive to detail loss. High frequency areas will be filtered with low weight objective and a high decay factor.

4.5.2.1.
Use of weight objective in the image domain

The original *NLM* algorithm limits the search for similar pixel neighborhoods by a fixed search window. There is no importance to the order of the search since all pixels within the search window will contribute their weights to the averaging. If we are using the weight objective as a limiter instead of, or in addition to the search window, then we are hoping not to cover all the pixels in the window in our search and the search order should start with those pixels which are more probable of being similar to the filtered pixel. The suggestion is to conduct the search in an increasing radius around the central pixel. This way we look for similar pixels first where the change of finding them is higher.

				10				
	9	8	7	6	7	8	9	
	8	5	4	3	4	5	8	
	7	4	2	1	2	4	7	
	6	3	1	A	1	3	6	B
	7	4	2	1	2	4	7	
	8	5	4	3	4	5	8	
	9	8	7	6	7	8	9	

Figure 17 increasing radius search

The search starts at the center pixel (point A) and advances through the pixels by their distance order from the center pixel, spiraling around it until the accumulated weight reaches the weight objective (point B). The original search window may be used as a worst case bound in case the desired weight objective is not reached.

4.5.3. Minimum weight contribution

When searching for similar pixels in the clustered domain, a third parameter is used for computing the valid blocks to participate in the filtering of a given pixel. For such a pixel we already know the decay factor, so in order to decide the distance of the blocks considered in the filtering, we should asses the maximal weight contribution of pixels in those groups. If blocks are divided using quantized root weighted mean distance of d_m, and quantized root weighted standard deviation of d_v then by *IEENNS*, the minimal distance of a block n_m root weighted means and n_v root weighted standard deviation away is given by:

$$d_{\max}^2 = k\left(\left((n_m - 1)d_m\right)^2 + \left((n_v - 1)d_v\right)^2\right)$$

and the maximum weight that can be contributed is

$$w(i, j) = \frac{1}{Z(i)} e^{(-1)\frac{d_{\max}^2}{h_i^2}}$$

Since $Z(i)$ is only the normalization constant, a very small value of $e^{-\frac{d_{\max}^2}{h_i^2}}$ is considered insignificant to the filtering process. The minimum weight contribution $w_{\min}$ will allow the computation of the search radius between adjacent blocks

$$r^2 = d_{\max}^2 = -h_i^2 \ln(w_{\min})$$

4.5.4. *Weighted L^2 distance* of flat areas

It is possible that the *IEENNS* lower bound measure can replace the exact weighted L^2 distance computation as a “similarity measure” in very flat areas without any visible artifacts. If so, significant acceleration may be achieved.

4.6. Symmetry

When calculating the weighted L^2 distance between two pixel neighborhoods, we may rotate the compared neighborhood around the center pixel in an angle that minimizes the weighted L^2 distance between the two neighborhoods. This is not done in the original *NLM* algorithm or in [5]. This way a higher “hit-rate” may be achieved in comparisons and the size of the search window may be reduced.

Finding the right angle to rotate may be done by computing the local gradient of the neighborhood at the central pixel $\nabla v(i)$. and computing its orientation. This should be done only for neighborhoods with high gradient magnitude, otherwise the gradient may be the product of noise only and therefore unreliable.

4.7. Additional methods

In a pseudo-code description of the *NLM* algorithm [18] posted by Buades, of the original authors, reveals a detail that escaped the original *NLM* publications. According to the original algorithm, each pixel contributes its own “self-weight” to its filtering. The self-weight always produces $w(x,x)=1$, and may be much higher than weights contributed by other pixels, thus reducing the filtering level of that pixel leaving it with a considerable amount of noise. The pseudo-code prevents “over-weighting” by lowering the self-weight to a level similar to other pixel weights. More accurately, instead of assigning $w(x,x)=1$, the weight of each pixel in its own filtering will be the maximum weight value assigned by any of the other participating pixels:

$$w(x,x) = \max w(x,y) \mid y \neq x, y \in search\_window(x) .$$

This should be helpful in smoothing rare pixels that cannot find similar enough pixels to be filtered with. Although this is not part of our proposition, we feel the need to test its influence on the results since no discussion of this feature can be found in the original *NLM* publications.

4.8. Summary of proposed methods

So far this work has the following propositions with potential to improve *NLM* and accelerate its execution:

1. Pre-classification of pixels neighborhoods according to their root weighted mean and root weighted standard deviation to reject pixels with high weighted L^2 distance from the filtered pixel.
2. Adaptive *decay factor* as a function of local image variance.
3. Use of *weight objective* in addition the *search window* as stop conditions to pixel filtering.
4. Rotation of compared neighborhoods to a point where their gradients are in the same direction to increase “hit-rate” of similar pixels.
5. Cluster image according the classification parameters and conduct a search for similarities in the clustered dimension.
6. Conduct similarity searches of pixels residing in flat areas in the image plane, and of pixels residing in high contrast areas in the clustered dimension table.

Note that except for the clustering proposition (5,6), all other propositions may be used with the original *NLM* and may bring some benefit both in image quality and acceleration as will be demonstrated in the “Experimental results” section. In some cases the method cannot be tested to full extent with the original *NLM*. For example, using a very low decay factor for high variance neighborhoods is expected to fail with the original *NLM* since the probability of finding similar neighborhoods in the limited search window is small. The detachment of the proposed methods from the clustering method, enables testing of the various methods independently as modified options on the original *NLM* algorithm.

4.9. Proposed algorithm

First, the image is scanned and each pixel’s neighborhood is preprocessed to extract the following parameters:

1. Local variance

2. Local root weighted mean
3. Local root weighted standard deviation
4. Local gradient

This is the pixel classification process which is demonstrated visually in the following section (4.10). The image is then segmented into blocks defined by the local root weighted mean and local root weighted standard deviation. Each pixel is added to a linked list holding pixels with the same quantized weighted mean and variance. The table is then processed list by list, pixel by pixel in the following way:

1. The filtered pixel *i* is rotated so that it’s gradient is pointing north.
2. The variance σ_i^2 of i is computed and from it the three filtering parameters:
    a. Decay factor $h_i^2 = f(\sigma_i^2)$
    b. Weight objective $w_{TH}(i) = f(\sigma_i^2)$
    c. Max distance $d_{\max}^2 = f(h_i^2)$
3. According to local image variance, a decision is made if search for similar neighborhoods will be conducted in the image plane or the clustering table plane.
4. While no stop condition is reached, we keep looking for filtering pixels in the neighborhoods of the root weighted mean and root weighted standard deviation. If *IEENNS* condition $d_{\max}^2 \geq (i' - j')^2 + (V_i' - V_j')^2$ is met for any pixel *j* then the actual distance $d_a^2(i, j)$ is computed while rotating pixel *j* with its gradient facing north, and the filtered pixel weight w_i and value $u(i)$ are incremented according to the comparison’s result.
5. After stop condition was met, pixel *i* is updated with the filtered value $u(i)$.

4.10. Pixel classification

The pixel classification scheme calculates a series of parameters for each of the pixels in the image that are later used by the algorithm. The calculated parameters are:

1. Root weighted mean
2. Root weighted standard deviation
3. Mean
4. Variance
5. Gradient magnitude
6. Gradient orientation (angle)

The following is an example of image pixel classification. The original image is “lena” and the added white noise has standard deviation of $\sigma_n = 20$. The kernel used for filtering and for classification in this case is a 7x7 Gaussian kernel. Each parameter is presented as an image. In all images, white areas represent high values while dark areas represent low values. The images, left to right and top to bottom are:

1. Original image.
2. Noisy image ($\sigma_n = 20$).
3. Root weighted mean image.
4. Root weighted standard deviation image.
5. Variance image, after removing the noise variance, and clipping all levels above 255 to white.
6. Mean image.
7. Gradient magnitude image. The dark areas represent low gradient magnitude and white areas for steep gradients.
8. Gradient orientation image. For demonstration, there are four gray levels in this image, referring to the gradient direction up, right, down or left.

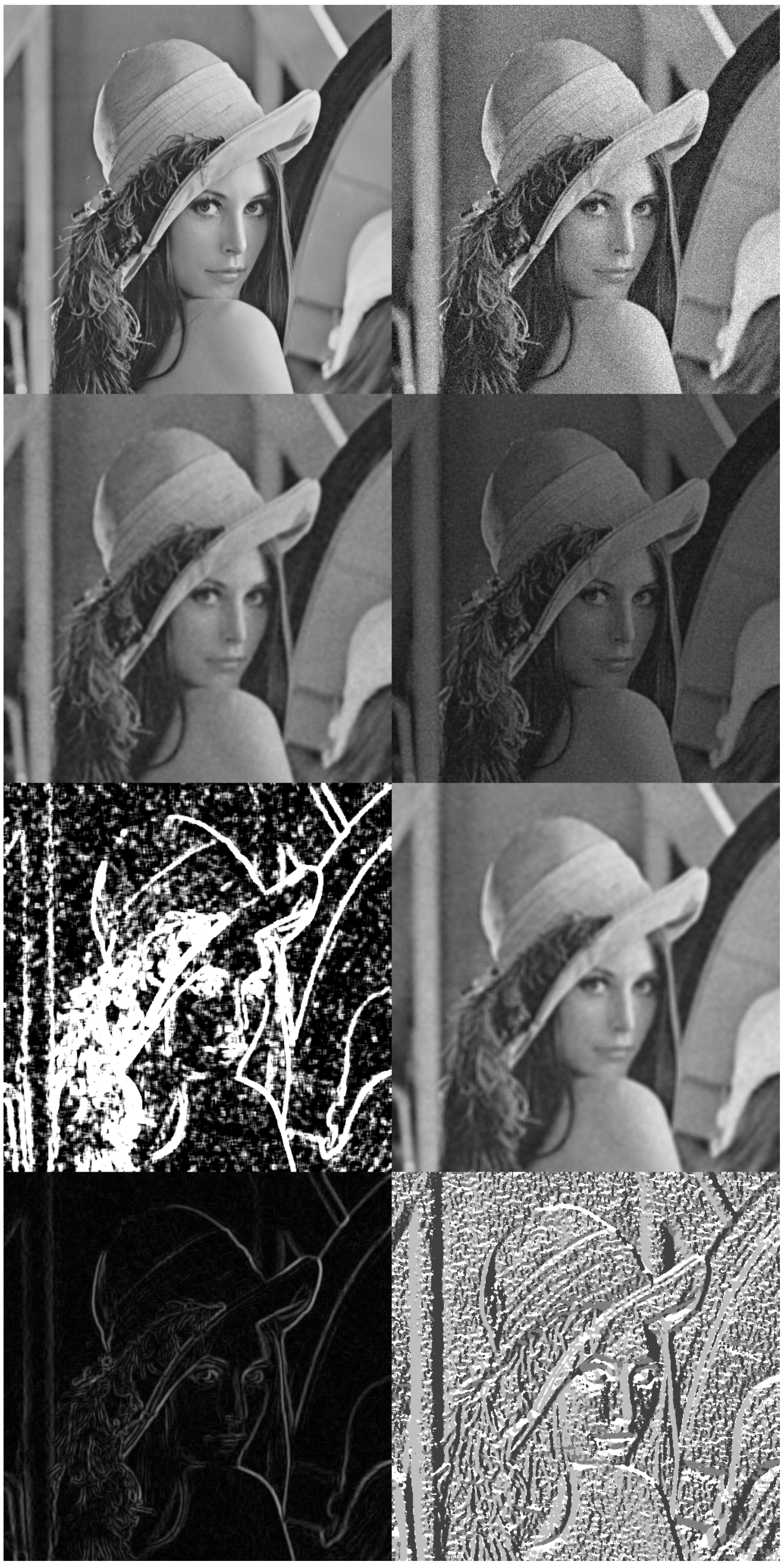


Figure 18 Pixel Classification

4.11. Calibrating Expectations

There is an inherent limit to the acceleration that can be achieved by the proposed algorithm. Since it uses the same similarity measure as the original algorithm, we should expect the same high number of computations per pixel comparison. The improvement should come from lowering the number of pixel comparisons and not from accelerating each comparison. The results should be compared to the “search window” approach and not to the “full image search” algorithm since the latter is impractical.

The original set-up of *NLM* experiments [3] will be used as a reference for comparison. That is, a search window of $21*21$ pixels, a similarity square of $7*7$ pixels and decay factor of $h^2 = 10\sigma_n$. In this setup, the original search window algorithm performs $21*21-1=440$ comparisons per pixel.

In the proposed algorithm, the number of comparisons depends on the characteristics of the filtered pixel. For the pixels that reside in flat zones, a minimum number of comparisons should take place in order to average each pixel. We cannot expect this number to be better than other “simple” filtering algorithms. If we take as a reference Gaussian smoothing with 5x5 kernel for example, we should expect each pixel to be filtered with at least 25 pixels. Even if the search for those pixels is easier in flat zones, the best acceleration that can be achieved is about x20.

For the active zones, the number of comparisons may be much smaller than the original algorithm since the expected number of similar pixels is lower than the search window size, and the number of required pixels for active zones is generally lower than in flat zones taking into account the difficulty of the *HVS* to detect variations in those areas. This shows that the proposed algorithm is expected to perform better on highly detailed images, than on relatively flat images. Since the majority of natural image pixels tend to be in the flat zones, our expectation for acceleration drops sharply. Estimating about 50% pixels in flat zones, an acceleration of x30-40 seems to be the upper limit of what can be achieved for natural images with this proposal.

5
Experimental results

The experiments were conducted on a series of test images that were taken from the “USC-SIPI image database” [19] and the “Ohio-state university signal analysis and machine perception laboratory” [20] and can be seen in Figure 19. For reference, the image names, left to right and top to bottom, are: “barb”, “boat”, “bricks”, “f16”, “goldhill”, “herringbone”, “lena”, “mandrill” and “slope”.

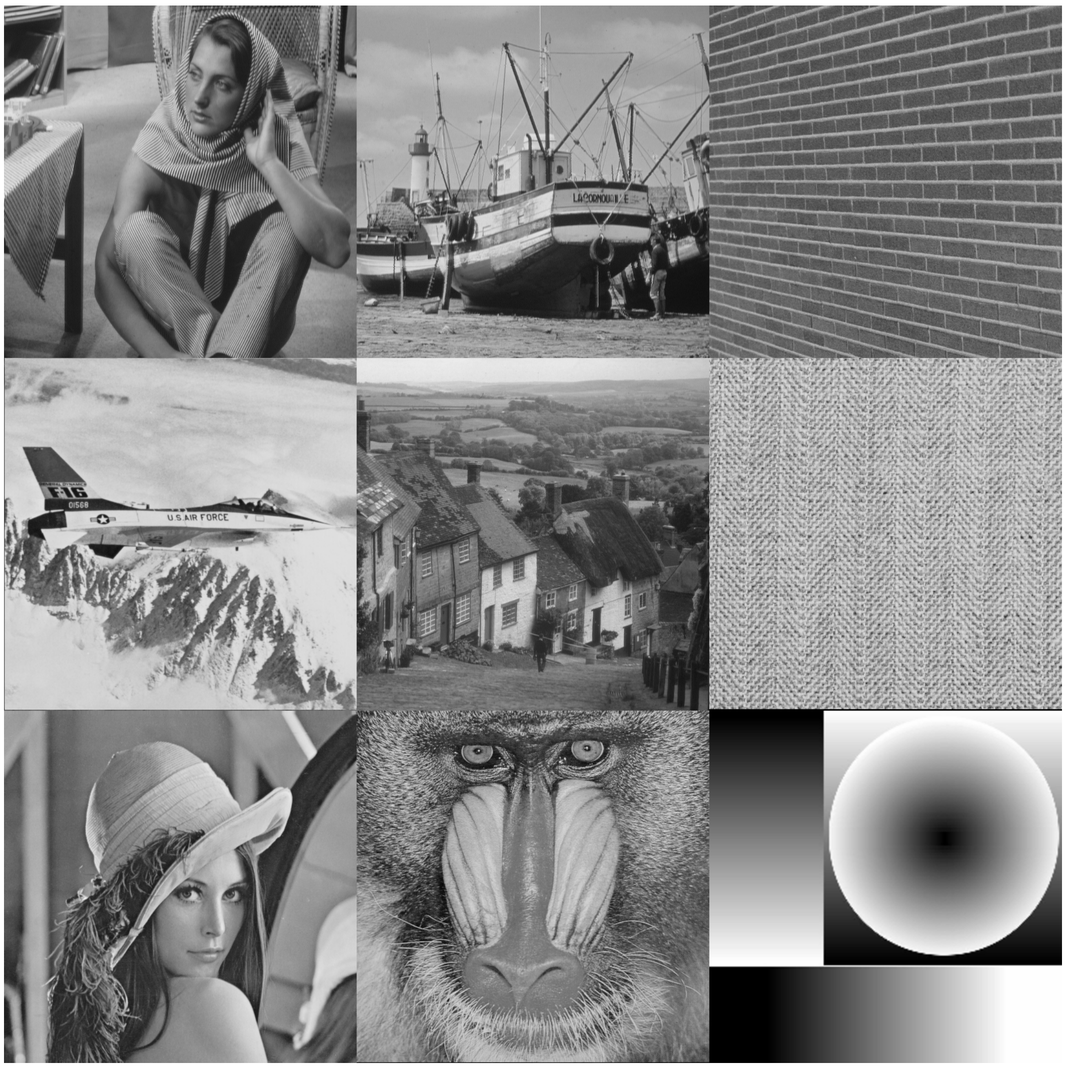


Figure 19 Test images

The images include six natural images with mixtures of flat zones, high contrast areas, different levels of detail density and some periodic image features, two images (bricks and herringbone), which are aimed at testing periodic features at different cycles, and the synthetic “slope” image to test behavior on slow changing image vignettes.

To facilitate testing and the understanding of obtained results, suggested improvement methods were tested independently when possible. The following independent tests were conducted on the series of the above pre-selected images.

1. Preventing pixel overweighting as described in section “4.7”.
2. Minimum distance filtering as described in section “4.1.2.3”. Rejecting computations of weighted L^2 distance for pixels that according to the minimum distance test are expected to provide negligible weights for the averaging operation.
3. Gradient orientation filtering as described in section “4.1.3”. Rejecting computations of weighted L^2 distance for pixels that according to the minimum angle test are expected to provide negligible weights for the averaging operation.
4. Neighborhood rotation to closest gradient orientation as described in section “4.6” in attempt to increase the “hit-rate” of the compared neighborhoods.
5. Replacement of the *search window* with *weight objective* conducting a spiral search in the image plane as described in sections “4.5.2 and “4.5.2.1”.
6. Using an adaptive *decay factor* as a function of local image variance in addition to the *weight objective* method as described in section “4.5.1”.
7. Clustering pixel neighborhood vectors by pre-classification of pixels neighborhoods according to their root weighted mean and root weighted standard deviation, and conducting the similarity searches in the clustered domain as described in section “4.2”.
8. Alternative similarity measure for flat zones as described in section 4.5.4”. Replacing the weighted L^2 distance measure with another similarity measure once previous filters did not reject the comparison.

After conducting the experiments described above, we have combined all successful methods into a single final algorithm. This algorithm has been tested in two stages:

9. Final algorithm – combining all successful methods from previous experiments.
10. Additional experiments with final algorithm while pushing the limits. This includes maximizing rejection of comparisons, and minimizing the filtering of active areas.

In all above experiments there are two main factors which are measured:

1. Quality
    a. Always in terms of *MSE* compared to original algorithms *MSE*.
    b. In some cases the method noise measure was also used.
2. Acceleration.

The acceleration is measured as the number of weighted L^2 distance operations in the original algorithm divided by the number of weighted L^2 distance operations in the tested algorithm. This measure is very close to actual time measurements but reduces the importance of specific implementation that may not be optimized. Section “5.1” gives a more detailed explanation about result calculation.

5.1. Understanding the results

Performance measurement of both acceleration and quality are key issues to understand when evaluating the experimental results presented in this chapter. The following sections explain the way the measurements are taken, and the reasons behind it.

5.1.1. Measuring acceleration

In order to reduce the effects of software implementation on the acceleration results, there are no real run-time measurements in this section. Execution may be affected more by the actual implementation than by the algorithm. To be able to work freely, with high accuracy and as close as possible to the original algorithm, most of the mathematical functions are using double accuracy floating point variables and the original mathematical functions such as *exp*() and *arctan*() with no attempts to optimize code to fixed point mathematics and approximated functions. This kind of optimization should be done after the algorithmic results are obtained.

Acceleration is measured as the ratio between the number of weighted L^2 distance operation performed by the original algorithm and by the accelerated algorithm. This is the same criterion use by Pan in [11] and by Wang in [12] to evaluate the performance of *VQ* acceleration techniques and we have found it to be the best isolator of the algorithmic approach from implementation details.

5.1.2. Measuring quality

In most cases the quality measure used is *MSE*. The *MSE* is calculated on the whole image using “ImageMagick®” compare function, which computes *MSE* as if each pixel is of 16bit intensity level.

When $u = \{u(i) : i \in I\}$is the original image, $v = \{v(i) : i \in I\}$ is the noisy image, each pixel *u(i)* is of 16bit $0 \le u(i) < 2^{16}$ and *N* is the number of pixel in the image:

$$MSE(u,v) = \frac{1}{N}\sum_{i \in I}\left(u(i) - v(i)\right)^2$$

In cases where visual artifacts are produced, even though *MSE* may be in the desired range, the artifacts are described and a reference image is attached.

5.2. Experiments

Next sections describe the individual experiments done at this stage.

5.2.1. Prevent Pixel Overweighting experiment

This method, proposed in [18] cannot help with acceleration, but may improve image quality. The image below shows the quality improvement of this method. It is noticeable that some singular pixels without enough pixels to average with, (zoom-in on the area between the eye and the eyebrow) remains noisy in the original method, and smoothed when limiting the weight of the original pixel to be the highest weight of all other pixels in the search window. Comparison between the *MSE* of the two filtered images also gives an advantage to the smoother image (1.462x10^6 with overweight prevention vs. 1.532x10^6 with the original algorithm).

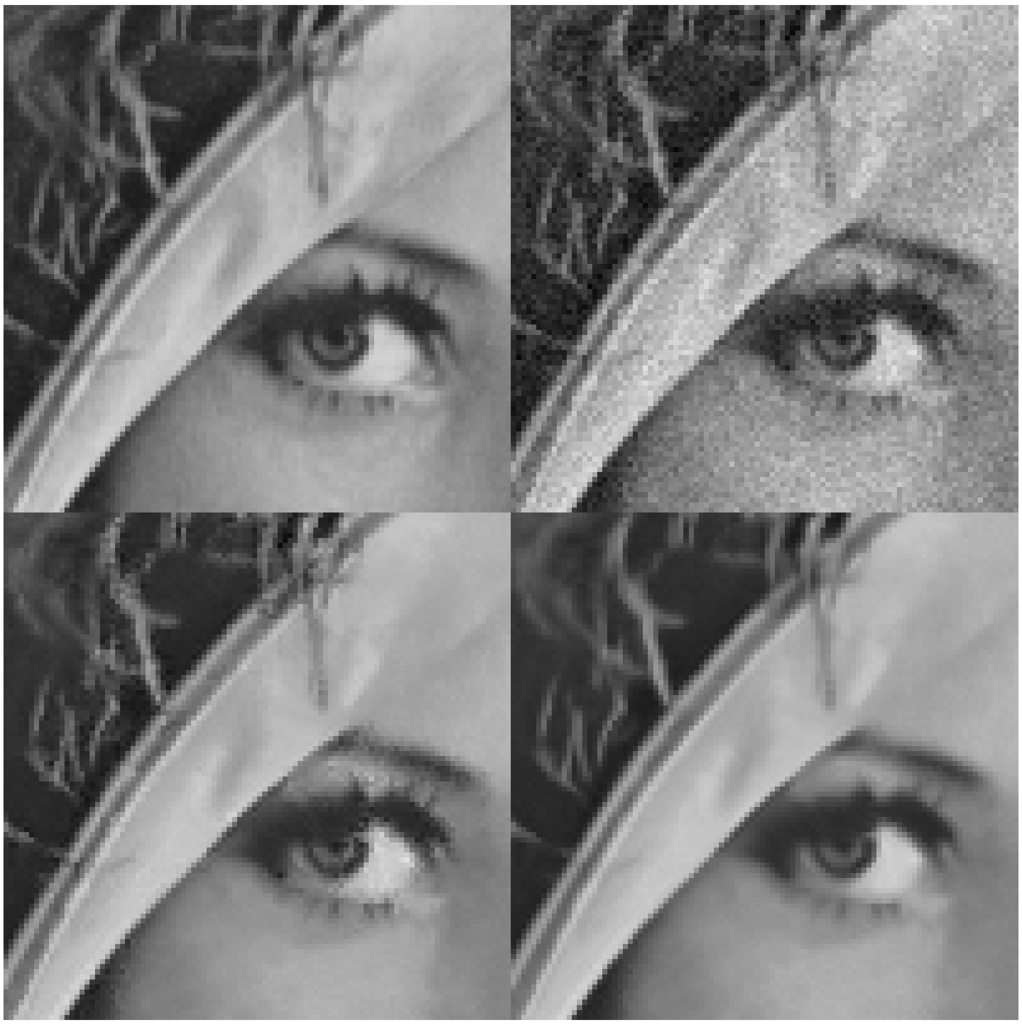


Figure 20 Preventing pixel overweighting; top left to bottom right: original, noisy, *NLM*-filtered, *NLM*-filtered when limiting pixel overweight

Actually, this method seems to have been in use in the original *NLM* algorithm, since the results demonstrated by the original authors seems to be as smooth as the bottom-right image. This method has been tested on the whole image test suite and was found to consistently improve image quality. For this reason, in all comparisons of image quality between the original algorithm and any proposed acceleration algorithm, the overweight prevention method is used in both original and proposed algorithms.

5.2.2. Minimum distance filtering test

This method is based on the minimum distance formula from section “4.1.2.3”.

Equation 14. $d^2_{a\,\min}(x,y) = (x' - y')^2 + (V'_x - V'_y)^2$

The minimum distance is computed from the classification images, and the weighted L^2 distance is computed only if the minimum distance is lower than a predefined threshold. The test uses standard experiment setup:

- Images
  - “f16”, “bricks”, “herringbone_weave1”, “lena”, “mandrill”, “barb” and “slope”.
- Noise standard deviation
  - 5, 10, 15, 20, 25, 30
- Search window size 21x21
- Decay Factor: $h^2 = 10\sigma_n$

The images were filtered using the following test:

If $(x' - y')^2 + (V'_x - V'_y)^2 \leq t * \sigma_n^2$ then $d^2_{xy} = \|v(x) - v(y)\|^2_{2,a}$

Else $d^2_{xy} = 0$

t ranges from 0.1 to 1000.

Figure 21 contains the quality results of this experiment presenting *MSE* as a function of the *t* coefficient for all images and noise levels.

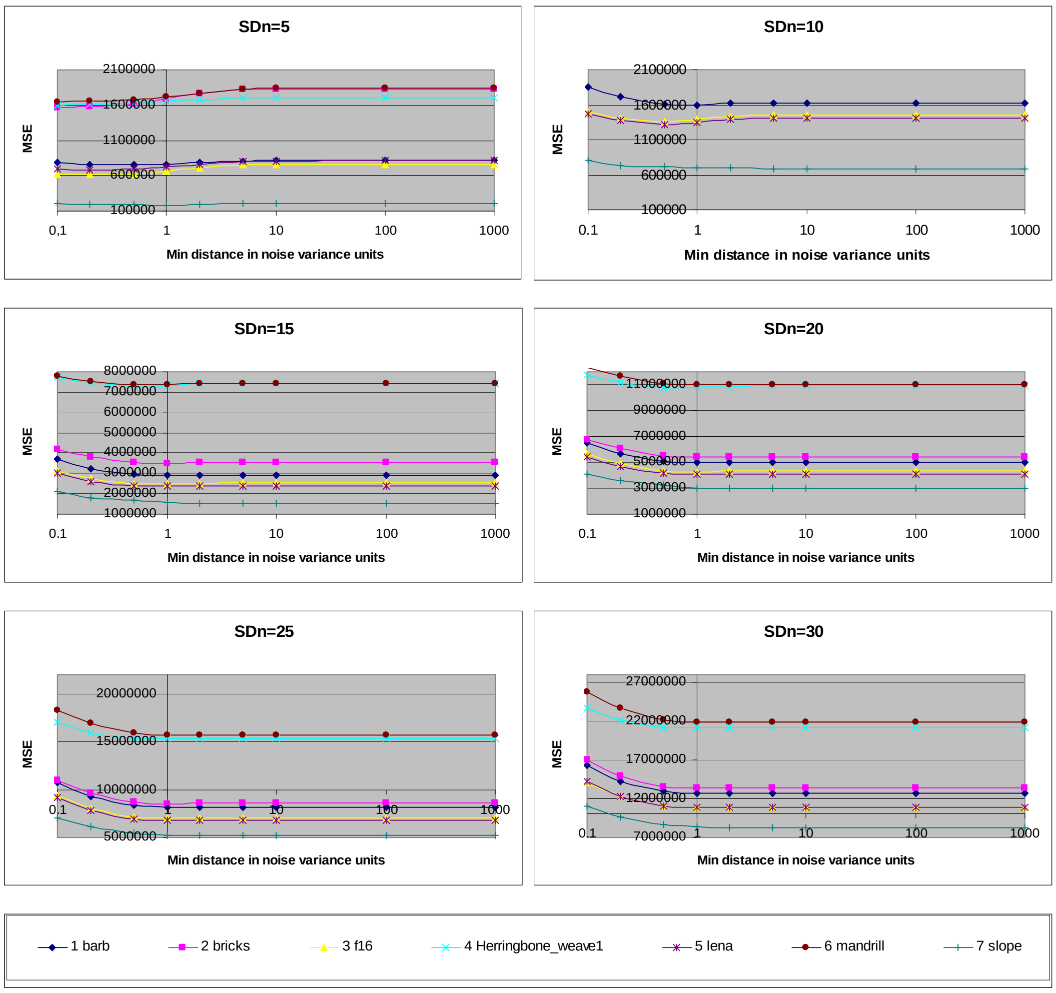
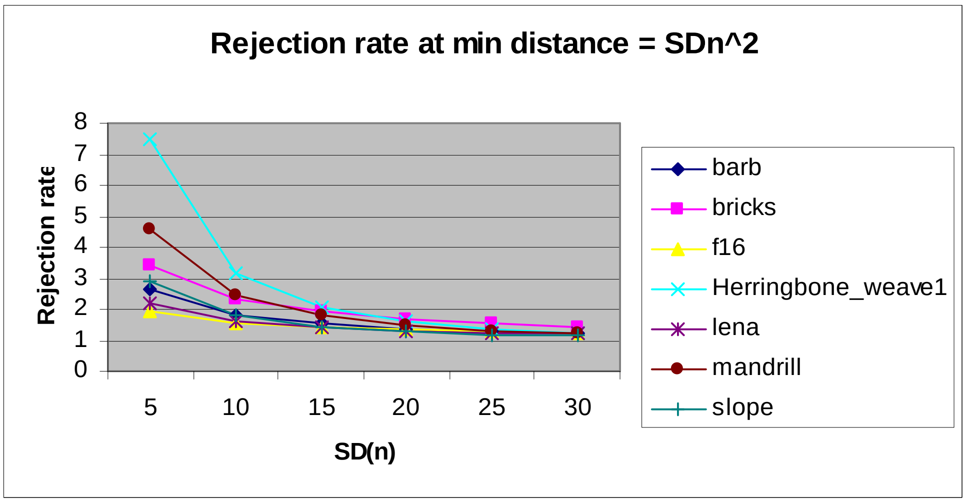
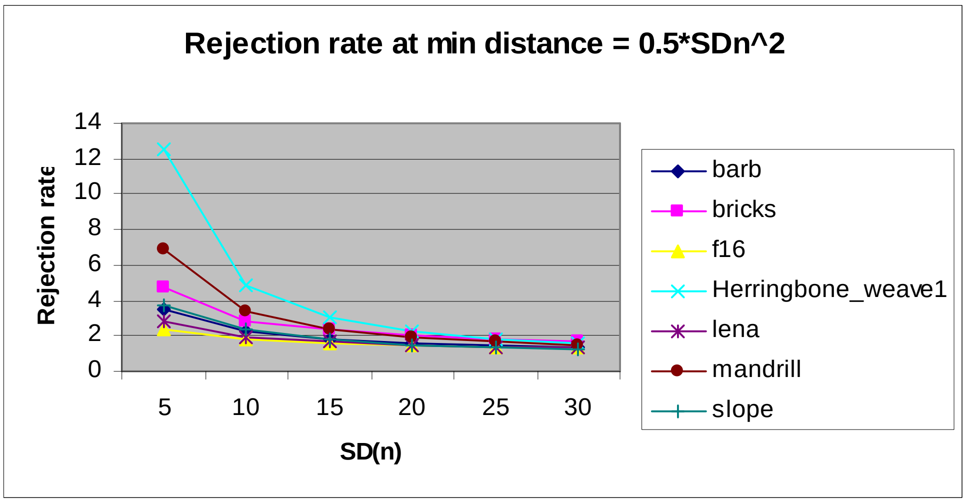


Figure 21 Minimum distance filter results *MSE=f(t)*

We can see that the filter at *t*=1 we already have a stable *MSE*. A lower value of t=0.5 causes only a small increase in the *MSE*. An interesting fact is that at the low noise values (• $_{n}$=5,10), a strong filter actually improves the *MSE*.

Figure 22 contains the rejection rate results at two chosen points, rejecting at σ_n^2 and at $0.5\sigma_n^2$. This is the actual acceleration achieved by the filter for all images and noise levels.





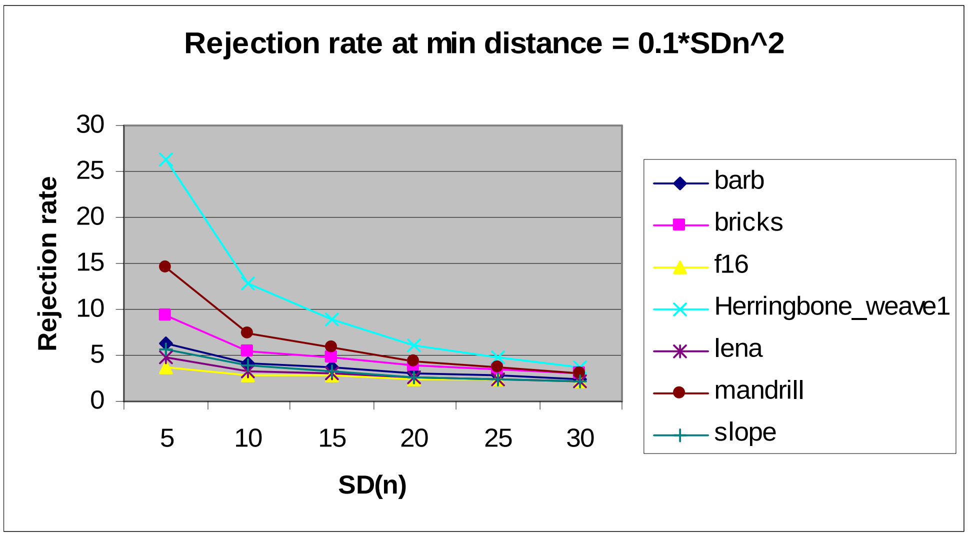


Figure 22 Minimum distance filter results, rejection rate, top t=1, middle t=0.5; bottom t=0.1;

It can be seen that the efficiency of this method drops sharply proportionally to the noise level. At the high noise levels the achieved acceleration is negligible (rejection rate below 50%) while in the low noise levels images acceleration is by factor of 2-12 (at minimum square distance of $\bullet_{n}^{2}$) depending on the nature of the image. High detailed images such as Herringbone and Mandrill have more to gain from this filter.

In the very low noise images, we see that filtering to minimum square distance of $0.1\bullet_{n}^{2}$ both improves *MSE* and causes acceleration of 4-26. This may be very useful for photography applications where the expected noise is at those levels.

5.2.3. Rotation of compared neighborhood experiment

Rotation has proven to be insignificant in a search window approach. It is difficult to find similar shapes in a different orientation in a very small window. It has a small effect on reducing the *MSE* in the high frequency areas and in searches in the clustered domain, but even this contribution is of no significant value. In our selection of test images, which we believe to be a good representation of the real world, the best position for distance computation is at the original orientation.

5.2.4. Adaptive Weight objective

Replacing the search window with an adaptive weight objective was tested independently of other methods and yielded very good results for the high contrast areas. Trying to force the weight objective down in the flat area to achieve small number of weighted L^2 distance computations resulted in “stains” artifact as can be seen in the bottom-left image of Figure 24. In the attached example a “lena” image with noise of $\bullet_{n}$=10 was filtered using the following weight objective graph:

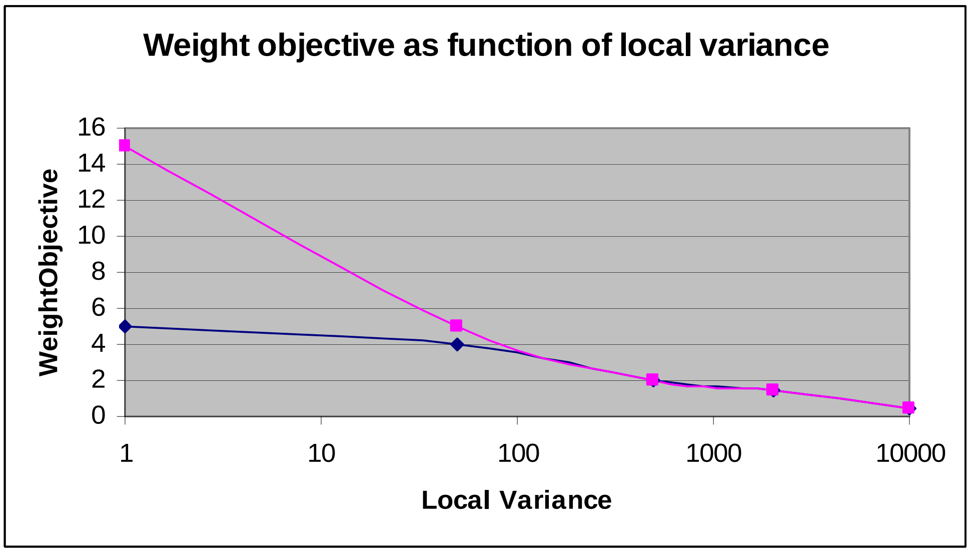


Figure 23 Weight objective as function of local variance

As can be seen, the two graphs differ only in the flat areas. The quality and acceleration results are summarized in Table 3:

Image	*MSE*	Acceleration
Original	$1.41x10^6$	1
Low weight objective	$1.44x10^6$	4.6
High weight objective	$1.39x10^6$	3.1

Table 3 Weight objective example results

The highest acceleration was achieved of course with the lowest weight objective. There is a slight degradation in *MSE* but there are strong “stains” artifacts that are unpleasant to the viewer’s eye[1]. At the higher weight objective, the *MSE* actually improved (insignificantly) while some acceleration has been achieved (x3).

[1] In order to see the stain better, use a digital version of this work and zoom in (200%) on the bottom left image comparing it to the other images. The stains may be masked by the print process in a printed version.

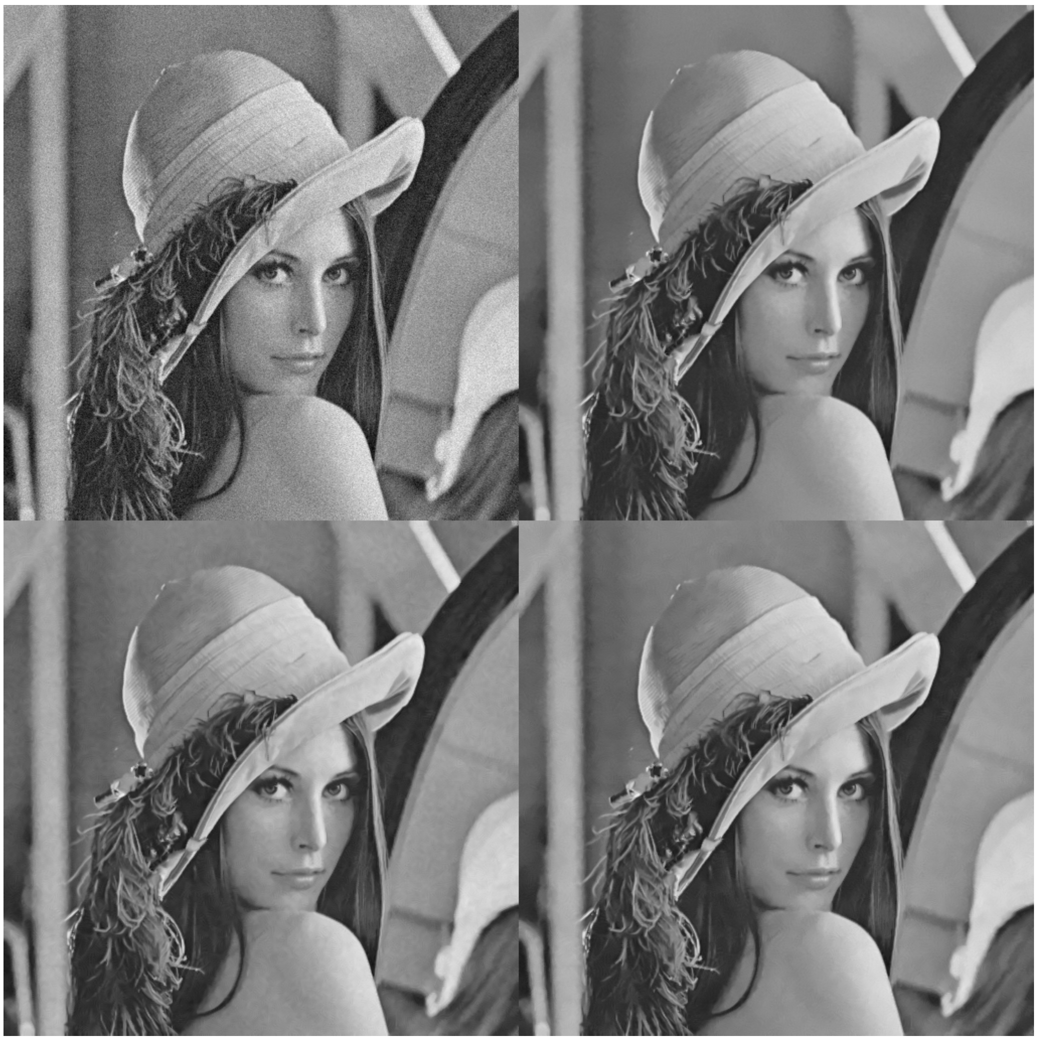


Figure 24 Weight objective example

In Figure 24 there is an example of the weight objective value effect. The original noisy ($\bullet_n$=10) image is at the top left, and the top right image is restored by the original algorithm. The bottom images are restored with the weight objectives from Figure 23, the left with the lower value starting at 5, and the right with the higher value starting at 15. The stains artifacts that can be seen at the bottom left image are common to Gaussian smoothing, and are the result of averaging locally with too few pixels. This means that flat areas need to be filtered using a high number of pixels. Since flat areas are common in natural images, this may be a problem and limit the acceleration. A solution may be found if those areas can be filtered using a different similarity measure that does not require weighted L^2 distance computation.

5.2.5. Adaptive decay factor

Experiments with adaptive decay factor were done in addition to the weight objective method. It seems that it is easier to change the weight objective parameters than to control the decay factor and get the same results. The main problem is again the stains in the flat areas which should be resolved using a different similarity measure if any acceleration is to be achieved in those areas.

5.2.6. Clustering

Experimenting with clustering according to the root weighted mean and root weighted standard deviation revealed major deficiencies in our assumptions. The fact that two pixels reside in the same bucket, and a priori have chance of having 100% similar neighborhoods, does not mean that their neighborhoods are similar enough to produce any significant weight for averaging. Experiments show that the search for similar pixels, even in cases of extremely active images, is better conducted in the image domain in the proximity of the filtered image, than in the clustered domain.

This does not mean that another clustering scheme, based on other classifiers, cannot produce good proximity of similar pixels. The example given by Mahmoudi and Sapiro in [5] gave significant acceleration for highly active small images. However, the results were obtained on relatively small images and seem to be significantly inferior in quality to the original algorithm (although probably superior in quality to other noise reduction algorithms). We also suspect the efficiency of this approach drops as the size of the image increases, and the pixels that are in the same bucket are much distanced from one another in the original image. More on this subject can be found in section 6, where we discuss related work done in this area.

If buckets cannot be proven to be filled with similar pixels, a large image in the neighborhood of 3Mpixels may create a search group much larger than the search window proposed in the original algorithm, thus decelerating the execution instead of accelerating it.

The best results are achieved by the clustering scheme for pixels that have a singular neighborhood in their close environment. Those pixels find their match more easily in the clustered domain than in the image domain limited search window. This advantage however loses its edge when taking into account the properties of natural images and the *HVS*:

1. There are very rare cases where a singular pixel can be found in a natural image.
2. When such a pixel is found in a natural image it is always in an active zone where the human eye is more concentrated on the activity than on the actual gray level. A noise at this point is hardly visible.
3. Those pixels will be processed fast enough in the proposed algorithm even without the clustering scheme since:
   a. The distance limit test will disqualify many of the adjacent pixels.
   b. The high local variance will drive the weight objective down, leading to a shorter search.
   c. We do not expect to find too many pixels of that kind.

The conclusion we draw from experimenting with clustering is that in the case of natural images, nature was able to do a much better job in clustering similar pixels in a radius around the filtered pixel, than any of our clustering schemes. Therefore, the clustering scheme will not be used in the final algorithm.

5.2.7. Gradient orientation filtering

Gradient orientation was used by Mahmoudi and Sapiro [5] as a secondary clustering dimension. Since our previous experiments with clustering and with neighborhood rotation have failed, we shall try to use the gradient as a secondary filter in addition to the minimum distance filter. This should be effective to reject comparisons between filters that have a high enough weight expectancy by the minimum distance filter, but eventually achieve a negligible weight due to their different gradient orientations.

The filter was set to reject all $\pi/6$ comparisons of neighborhoods with gradient of magnitude greater than 5 and gradient orientation greater than $\pi/6$. The filter resulted in acceleration of 2.5-6.5 with some small degradation in *MSE* in some images, up to 7% in low noise levels and up to 20% in high noise levels. A visual difference between original algorithm and this method is almost invisible. Even at the images with the highest *MSE* degradation, the difference between the result of this method and the original method is hardly visible. Figure 25 is an example of filtering a noisy image ($\bullet_n$=20) with the original algorithm and with the two filters applied to reject unnecessary computations. The modified algorithm yields a *MSE* 18% higher than the original, and acceleration of 3.5 times faster. The differences, if any, are hardly noticeable.

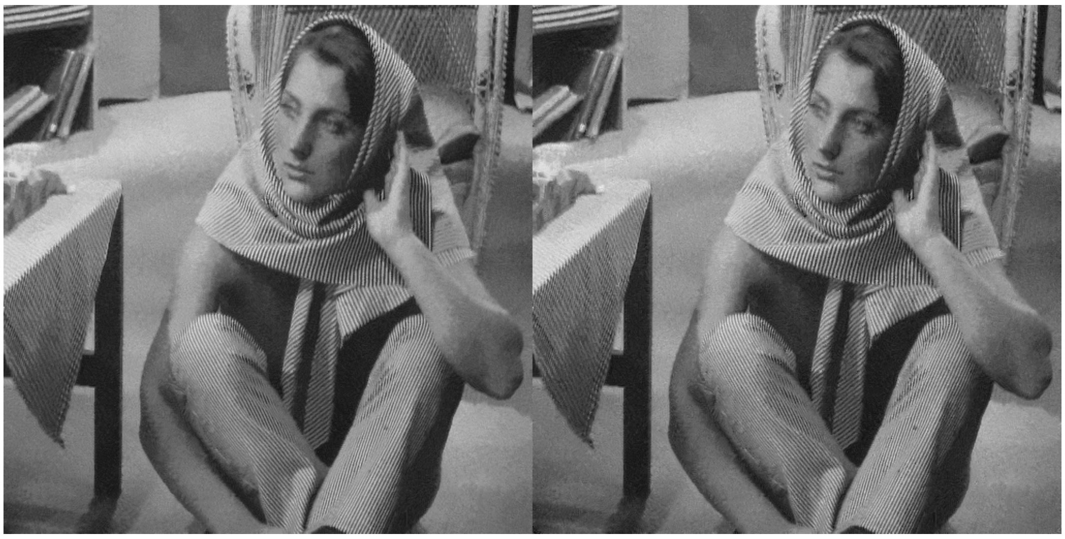


Figure 25 Gradient filter in addition to minimum distance filter. Left: image filtered by original algorithm; right: the two filters applied

5.2.8.
An alternative similarity measure for flat zones

As learned from previous experiments, flat zones require a high number of averaging pixels to avoid the “stains” artifact demonstrated in Figure 24. If a simplified similarity measure is found, then we can skip all weighted L^2 distance computations and accelerate the execution of the algorithm. Actually, if such a measure can be found for all image areas then the acceleration problem is solved but since no such global measure has been found yet, we still have a hope to find such a measure for the relatively simple flat zones.

The idea is that once we have passed the previous filters of minimum distance and gradient orientation, we should verify if we are in a flat zone by testing the local variance and gradient of the filtered pixel. If those values are found to be below a predefined threshold then the weight contribution of the compared pixel is a constant. Otherwise, weighted L^2 distance is computed.

If this measure works, it should accelerate the execution while maintaining a high number of averaging pixels for each of the pixels in the flat zone, thus avoiding the stains artifact.

This method is tested as part of the complete final algorithm.

5.3.
Final algorithm – putting it all together

The final algorithm to be tested contains all methods that were proven to contribute to acceleration without sacrificing too much image quality. The algorithm will be based on the adaptive weight objective as a stop condition to pixel averaging, the minimum distance and gradient filters for rejecting non-weight contributing comparisons, and on special treatment for the expected large flat zones.

5.3.1.
The algorithm

For each pixel extract the local variance and compute the weight objective and decay factor to be used. Start looking for averaging pixels in a search window with the following stop conditions:

1. Don’t stop before processing a small predefined window around the pixel – “minimum search window”.
2. Stop at search window boundaries.
3. Stop when reached the weight objective.

Filter only pixels that pass the two filters:

1. Minimum distance filter.
2. Gradient orientation filter.

If the filtered pixel is in a flat zone, any pixel that passes the filtering process contributes a constant (small) weight. Pixels in an active zone continue to be filtered according to the original weighted L^2 distance measure.

5.3.2. Experimenting

The following experiment setup was used:

1. Minimum search window of all immediate neighbors (8).
2. Stop condition – 21x21 search window or Weight objective.
3. Variance dependent parameters were setup according to Figure 26.

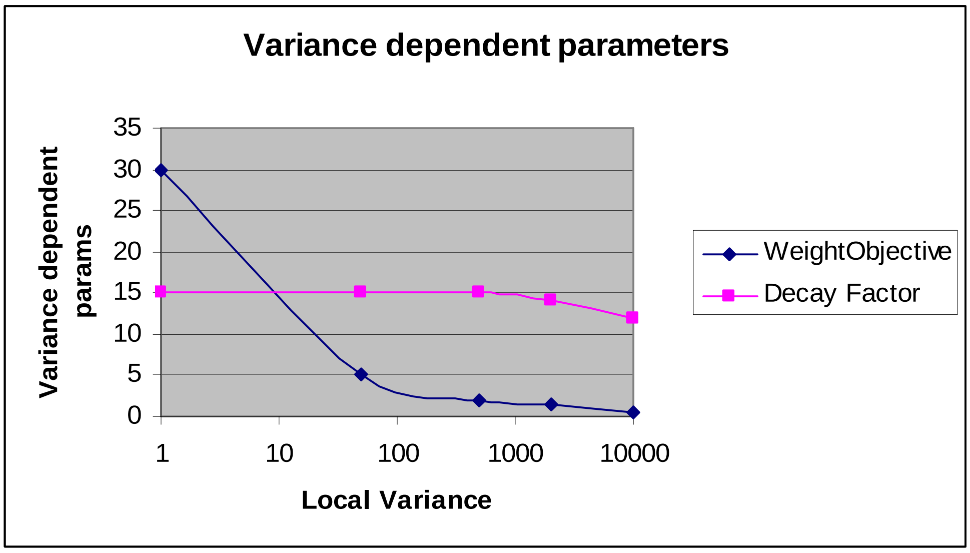


Figure 26 Variance dependent parameters for final algorithm test

4. The minimum distance filter threshold was set to $0.5\bullet{}_n^2$.
5. The gradient angle filter was set up to minimum magnitude of 6 and angle difference of $\pi/6$.
6. The Flat zones threshold was fixed for variance lower than 80 and gradient magnitude lower than 3. The constant used for flat zone weighting was 0.3.

Table 4 summarizes the comparison between the original search window algorithm in the original setup (search window of 21x21 and $h^2=10\bullet{}_n$) and the modified algorithm in the above setup. The ***MSE ratio*** column is the ratio between the *MSE* of the modified and original algorithms.

Noise *SD*	Image	*MSE* ratio	Acceleration
5	lena	99%	22.5
10	lena	98%	12.3
15	lena	94%	7.7
20	lena	87%	5.6
25	lena	82%	4.7
30	lena	79%	4.4
5	boat	98%	25.3
10	boat	98%	13.4
15	boat	96%	8.2
20	boat	90%	6.0
25	boat	85%	5.1
30	boat	81%	4.7
5	f16	94%	25.8
10	f16	96%	13.1
15	f16	92%	8.0
20	f16	86%	5.8
25	f16	81%	5.0
30	f16	77%	4.7
5	goldhill	100%	20.1
10	goldhill	101%	13.1
15	goldhill	99%	8.3
20	goldhill	94%	6.0
25	goldhill	88%	5.0
30	goldhill	84%	4.6
5	barb	107%	16.2
10	barb	108%	10.7
15	barb	102%	7.4
20	barb	95%	5.5
25	barb	89%	4.7
30	barb	85%	4.3
5	mandrill	109%	19.1
10	mandrill	105%	11.9
15	mandrill	101%	8.2
20	mandrill	96%	6.3
25	mandrill	91%	5.4
30	mandrill	87%	4.9

Table 4 Comparison of final and original algorithm results

The first examples are “lena”, “boat” and “f16” images with mostly low frequency areas. Algorithm performs on all images with constant improvement in *MSE* and acceleration between 4.4 in high noise images and 25.8 in low noise images.

The Barb and Mandrill images, with high portion of high frequency areas, perform better in the high noise areas and have a slight (up to 9%) increase in *MSE* comparing to the original algorithm and acceleration range of 4.3 to 19.1.

Goldhill, with a mixture of both high and low frequency areas is kept within the original *MSE* with acceleration of 4.6 to 20.1.

In Figure 27 there are four examples of both original and final algorithm. The Barb and Goldhill images were restored from $\bullet_n=10$, “lena” from $\bullet_n=15$ and Mandrill from $\bullet_n=20$. In Figure 28, we are using the method noise as a quality measure between the original and accelerated algorithm. Notice the image structure appears stronger in the original algorithm than in the accelerated one. The difference though is insignificant.

Comparing artifacts between the original and accelerated algorithm, we can notice higher noise residues in some flat areas, and stronger “shock” artifact in smooth vignettes resulting in quantization of the filtered intensity levels in the accelerated algorithm. On the other hand, we can find better defined small details in the accelerated algorithm.

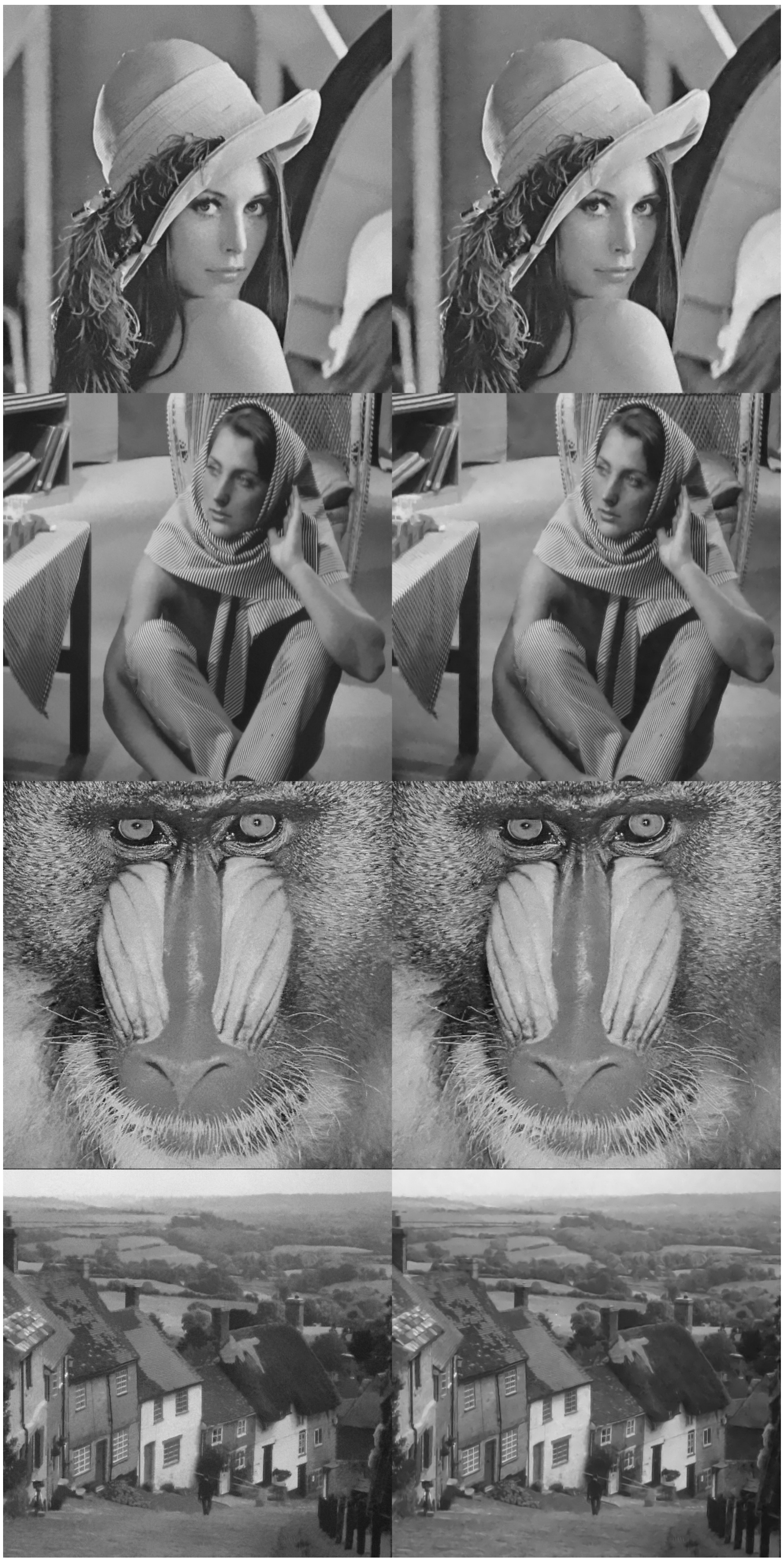


Figure 27 Original (left) and final (right) algorithms result examples

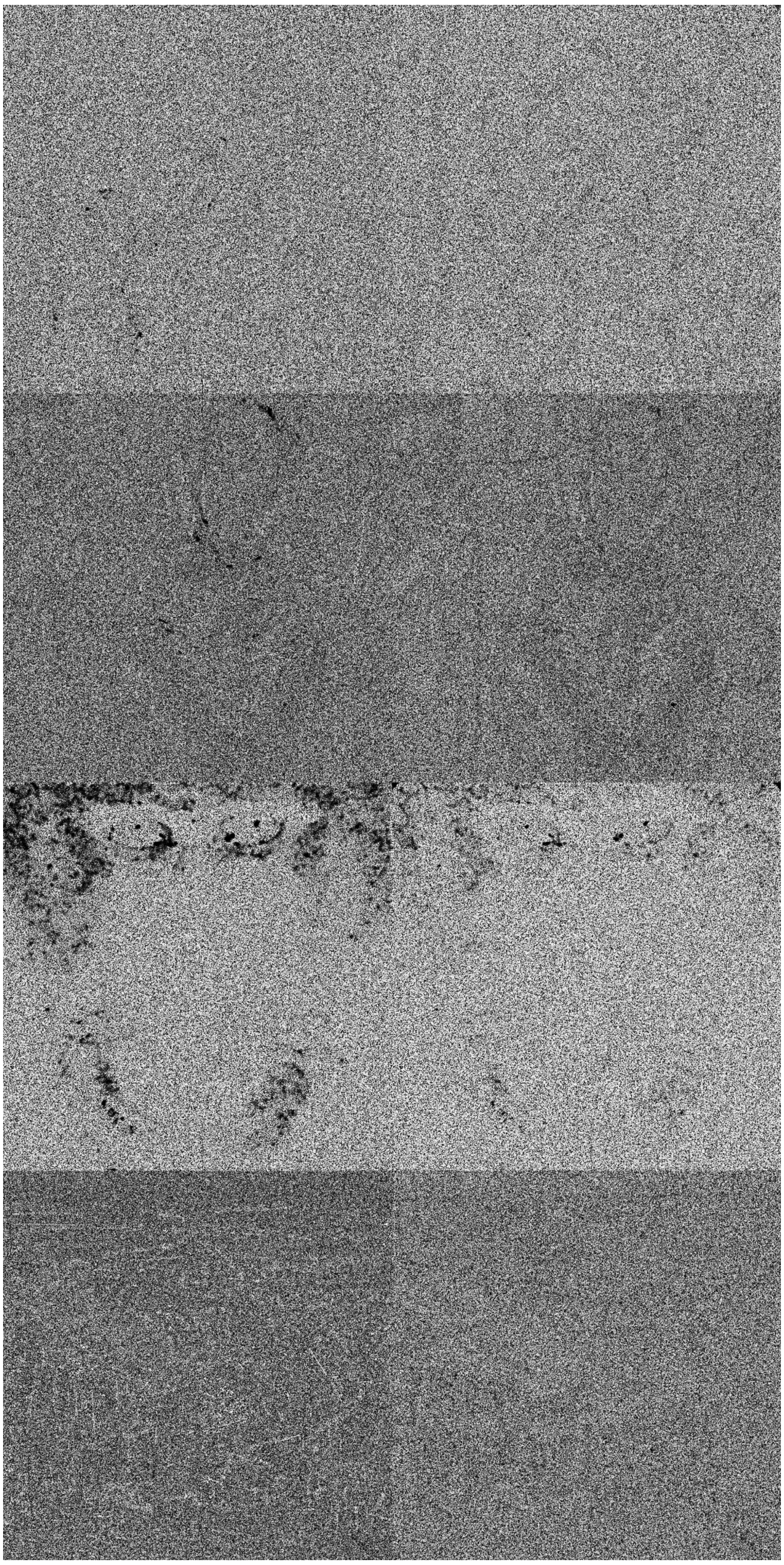


Figure 28 Original (left) and final (right) algorithms method-noise results

5.3.2.1. Pushing the limits

Additional tests in the low noise area only (•$_{n}$=5-15), pushing the filtering to higher levels (minimum distance at 0.1 •$_{n}^{2}$ and gradient orientation to •/10) resulted in accelerations of 11-33 and *MSE* degradation of less than 10% with hardly visible artifacts. Attempts to accelerate the original algorithm in the low noise images by using a smaller search window are bounded by a factor of 3-4 since the very small windows start producing the “stains artifact” in flat areas.

When we are trying to push the limits even more and filter the high contrast areas with very few operations we may achieve good visible results according to the *HVS* model even though the *MSE* in this case may increase significantly, even beyond the performance of other inferior noise filters. Figure 29 gives an example of minimal filtering in the high contrast areas.

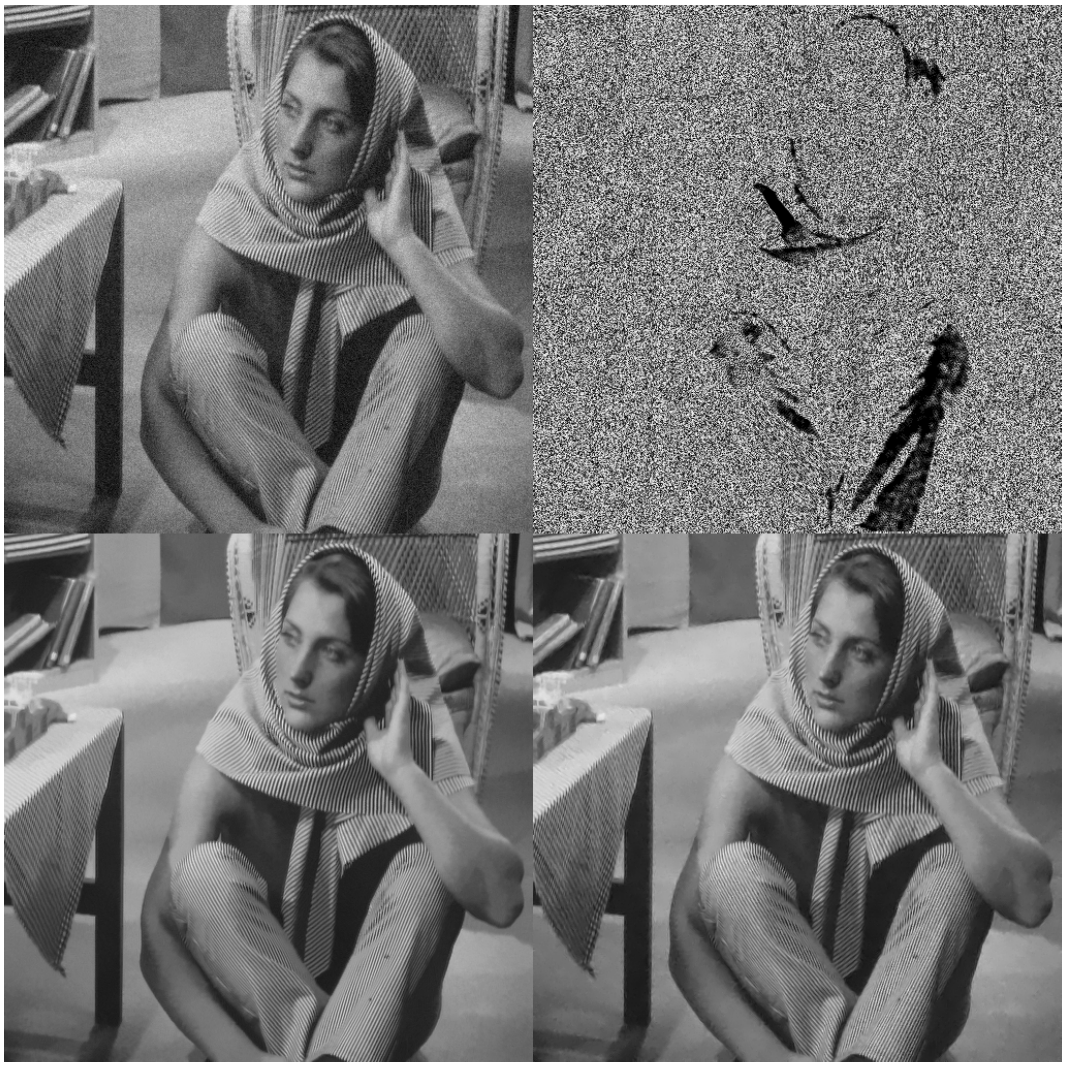


Figure 29 Minimal filtering of high contrast areas

The top left image is the original noisy (•$_n$=10) image. The bottom images are filtered by the original algorithm (left) and accelerated algorithm with only 8 pixels participating in the filtering of pixels in high frequency areas (above •$_{image}$=80). Noise residues may be seen in the bottom left image especially along the long structures (zoom in on the table's leg for example - Figure 30). The method noise image (top right) shows significant image structures that are left with high noise residues (black areas). The *MSE* of this image grew by 32%, yet the achieved acceleration of x23 may be worth the degradation since for the *HVS*, the degradation is not that significant.

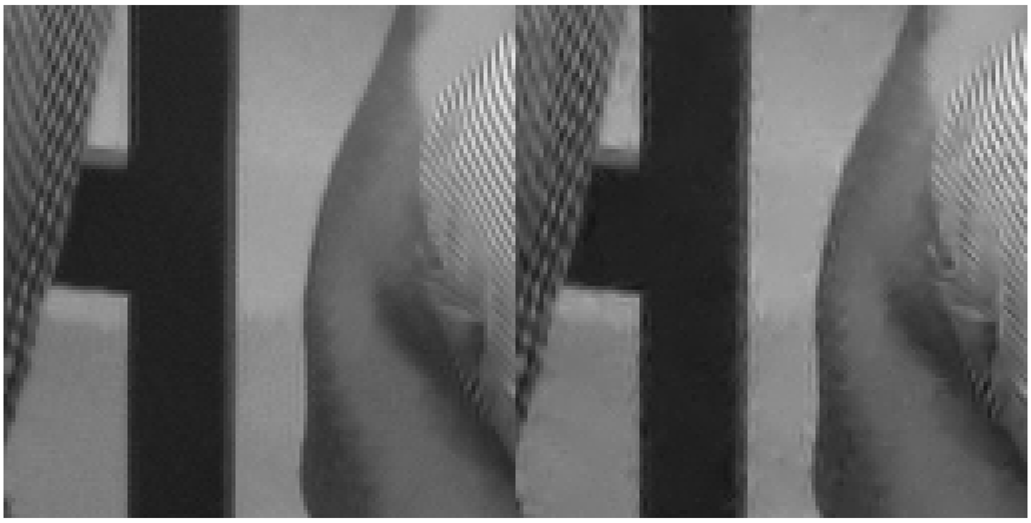


Figure 30 Minimal filtering of high contrast areas – zoom in

5.3.2.2. Color examples

Figure 31 demonstrates the effect of the algorithm on color images. In those examples we have the following setup, with noisy image on the left and accelerated algorithm results on the right:

- "lena", •$_n$=10 accelerated by 16
- "f16", •$_n$=15 accelerated by 8.3
- "mandrill", •$_n$=20 accelerated by 11.7.

In this demonstration, there is equal noise on all channels and the accelerated algorithm is performed on individual channels without trying to achieve additional acceleration based on any other color conversion technique.

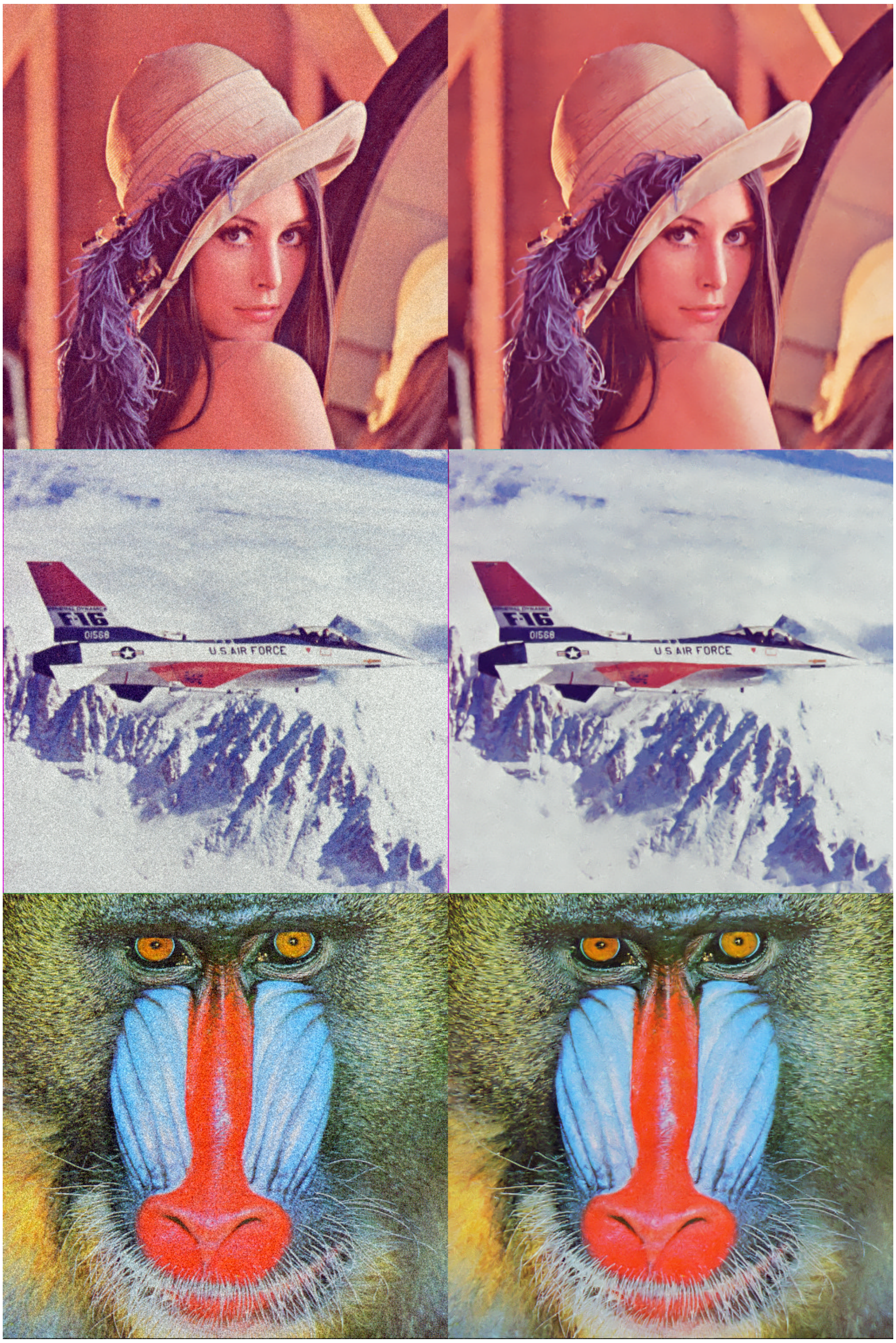


Figure 31 Accelerated algorithm on color images

6
Related work

We have found two previously published works with practical suggestions aimed at accelerating *NLM* execution. The first one, already referred to in previous sections of this work, is the work done by Mahmoudi and Sapiro [5]. The second one is a Ph. D. Thesis by Buades on *NLM* [21] which includes a short section on acceleration of the algorithm.

Our work, that started before [5] was published, gained a certain amount of optimism with the publication of [5] since it seemed to confirm the potential of clustering in *NLM* acceleration. We also considered the classifiers we adopted from *IEENNS* to be at least as suitable for the weighted L^2 environment as the mean classifier used in [5]. The results of our clustering scheme however, were quite disappointing. While we believe the results obtained by [5] are an improvement over other noise reduction algorithm, we did observe a significant degradation in quality between the accelerated and the original algorithm. There are no *MSE* comparisons in [5] between original and accelerated algorithms and our observation is based on visual inspection of the examples provided in [5] (see figures 1 and 3 in [5]). We also believe that it is hard to achieve same results for bigger images (in the neighborhood of 3Mpixels or more) without using additional techniques to limit the number of pixels in each cluster. The gradient filter suggested in [5] has proven to be a strong tool in rejecting unnecessary distance computations.

As for the thesis work, there are two suggestions for *NLM* acceleration and none of them is developed into experimental phase. Both suggestions modify the original algorithm in ways that may degrade quality, but there is no information of that kind in the cited work. The first suggestion based on sub-sampling the image data and has a potential of accelerating the algorithm by x16. The second one uses L^2 distance instead of the weighted L^2 distance and defines an *NLM* operator for vectors. From some experiments we made replacing the weighted L^2 distance with non-weighted L^2 distance we got difficulties in selecting a decay factor such that will eliminate noise without over smoothing the image. It seems that the image either stays noisy, or is totally over-smoothed. This shows that the weighted L^2 distance is in fact a more robust similarity criterion than the non-weighted distance.

We are not aware of additional studies in this area, but as we have seen the interest *NLM* provoked, and taking into account it was published in 2004, it is probable that we will see additional publications related to the subject in the near future.

.

7 Conclusions

During this work we have investigated some methods for the acceleration of *NLM* algorithm. We have approached the problem with the objective of finding a better price-performance ratio for the algorithm concentrating in the natural images environment; price, in terms of quality loss, and performance in terms of acceleration gain. The right price-performance tradeoff point is difficult to define especially in the area of image quality where mathematical indicators don't always reflect the subjective appreciation of the viewer. For this reason, we have tried to keep the results within a close distance to the original algorithm in terms of *MSE* and method noise, and to measure acceleration only in the areas where image quality degradation is minimal.

Since our attempts to achieve tight quality results in comparison to the original algorithm came in the way of our suggested clustering scheme, we gave up clustering, but kept the classifiers to be used as filters in the selection process that chooses which comparisons will be computed. Giving up clustering means admitting that nature's clustering scheme is more powerful than the one we have suggested. Remembering that the clustering approach was intended to grasp the "real spirit" of the original algorithm of filtering similar features all over the image, we understand now that we actually narrowed the algorithm to perform locally. As is the case for the search window *NLM* algorithm, the algorithm took the form of a "local neighborhood filter" with the weighted L^2 distance as a similarity measure. We feel that at least in the case of natural images, this is the real powerful attribute of the algorithm and not its ability to perform globally on the image.

We have demonstrated the power of the selection process, and we have shown that we can treat different image areas with different levels of filtering with minimal loss of image quality. We have also found out that our methods are less effective in high noise levels, but can provide significant acceleration for the low and mid noise levels. This should benefit the digital photography applications targeted at the introduction for this work, but the achieved acceleration may not be sufficient to be used in mainstream applications.

We feel that a key area for the acceleration of natural images is in the flat areas. Those areas are important since usually most of the pixels reside in such areas, and our *HVS* is more sensitive to noise in flat areas than in active areas. On top of that, we know that a high number of pixels need to be used for the filtering of those areas to avoid artifacts. Most noise reduction algorithms have relatively good results in the flat areas while encountering difficulties in the active areas. For these reasons we feel there is room for additional work that will try to combine other known filtering methods for flat areas, with *NLM* for active areas.

References

[1] A. Buades, B. Coll, and J Morel. “On image denoising methods.” Technical Report 2004-15, CMLA, 2004. (www.cmla.ens-cachan.fr/Cmla/Publications/2004/CMLA2004-15.pdf)

[2] A. Efros and T. Leung. Texture synthesis by non parametric sampling. In Proc. Int. Conf. Computer Vision, volume 2, pages 1033-1038, 1999.

[3] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In Computer Vision and Pattern Recognition (CVPR'05), pages 60.65, 2005.

[4] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. Multiscale Modeling & Simulation (SIAM interdisciplinary journal), 4(2):490.530, 2005.

[5] M. Mahmoudi and G. Sapiro. Fast image and video denoising via nonlocal means of similar neighborhoods. Signal Processing Letters, 12(12):839.842, 2005.

[6] http://www.cs.wisc.edu/~evanswes/cs766.html

[7] S. Yoshizawa, A. Belyaev and H.P. Seidel. “Smoothing by example: Mesh Denoising by Averaging with Similarity-based Weights”. International Conference on Shape Modeling and Applications (SMI), pp. 38-44, June 14-16, 2006 at Matsushima, JAPAN. (http://www.mpii.mpg.de/~shin/smi06ybs.pdf)

[8] .W. Ra and J.K. Kim. A fast mean-distance-ordered partial codebook search algorithm for image vector quantization. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 40(9):576 579, September 1993. (*ENNS*)

[9] C. H. Lee and L. H. Chen, "Fast closest codeword search algorithm for vector quantization," IEE Proc. Vision Image and Signal Processing, vol. 141, no. 3, pp. 143--148, 1994. (*EENNS*)

[10] S. J. Baek, B. K. Jeon, and K. M. Sung, "A fast encoding algorithm for vector quantization," IEEE Signal Processing Letters, vol. 4, no. 2, pp. 325--327, 1997.(*IEENNS*)

[11] Pan, J.-S., Z.-M. Lu and S.-H. Sun, An efficient encoding algorithm for vector quantization based on subvector technique, IEEE Signal Processing Letters, vol. 4, pp325-327, Feb 1997 (*PAN*)

[12] Chou-Chen Wang and Chi-Wei Tung, "A fast encoding algorithm for vector quantization," IEICE Electronics Express (ELEX), vol.2, no.17, pp.458-464, 2005.09 (*IPAN*)

[13] S. Winkler and S. Süsstrunk, Visibility of noise in natural images, Proc. IS&T/SPIE Electronic Imaging 2004: Human Vision and Electronic Imaging IX, Vol. 5292, pp. 121-129, 2004.

[14] S. Voloshynovskiy, A. Herrigel, N. Baumgartner, and T. Pun, "A stochastic approach to content adaptive digital image watermarking," in Proc. Int. Workshop on Information Hiding, Vol. LNCS 1768 of Lecture Notes in Computer Science, pp. 212–236, Springer Verlag, Dresden, Germany (1999).

[15] http://ohzawa-lab.bpe.es.osaka-u.ac.jp/ohzawa-lab/izumi/CSF/A_JG_RobsonCSFchart.html

[16] Campbell FW & Robson JG (1968). Application of Fourier analysis to the visibility of gratings. Journal of Physiology, 197: 551-566.

[17] G. M. Johnson and M. D. Fairchild, "Sharpness Rules," IS&T/SID 8th Color Imaging Conference, Scottsdale, 24-30 (2000).

[18] http://dmi.uib.es/~tomeucoll/toni/nlmeanscode.html

[19] http://sipi.usc.edu/database/

[20] http://sampl.ece.ohio-state.edu/database.htm

[21] “Image and film denoising by non-local means”, Ph. D. Thesis submitted by Antoni Buades, Universitat De Les Illes Baleares. (dmi.uib.es/~tomeucoll/toni/publicacions/tesi.pdf)