Noise smoothing in the Fourier domain by a multi-directional diffusion^{*}

Ryuichi Ashino[†] Steven J. Desjardins^{‡ §} A. A. Kolyshkin[¶] Rémi Vaillancourt^{‡||}

CRM-2934

September 2003

∥remi@uottawa.ca

^{*}This research was partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research (B), 14340045(2002–2003), (C), 13640171(2001-2002), 15540170(2003), the Natural Sciences and Engineering Research Council of Canada and the Centre de recherches mathématiques of the Université de Montréal.

[†]Division of Mathematical Sciences, Osaka Kyoiku University, Kashiwara, Osaka 560-0043, Japan; ashino@cc.osaka-kyoiku.ac.jp [‡]Department of Mathematics and Statistics, University of Ottawa, Ottawa, ON, Canada K1N 6N5

 $[\]S$ desjards@mathstat.uottawa.ca

[¶]Department of Engineering Mathematics, Riga Technical University, Riga, Latvia LV 1048; akoliskins@rbi.lv

Abstract

Several noise removing methods are briefly surveyed. In the context of pseudodifferential operators, the diffusion equation is applied to noisy images in the Fourier domain to slightly smooth out the noise in many directions at once without oversmearing out edges and details.

Dedicated to Michihiro NAGASE on the occasion of his 60th birthday

To appear in

Boundary Field Problems and Computer Simulation of the Scientific Proceedings of RTU in the series: Computer Science.

Résumé.

On rappelle quelques méthodes pour le débruitage de l'image. Dans le cadre des opérateurs pseudodifférentiels, on applique l'équation de la chaleur unidimensionnelle selon plusieurs directions dans le domaine des fréquences pour débruiter l'image sans trop en endommager les arêtes et les détails.

1 Introduction

1.1 Noise and the Diffusion Equation

In today's world, the storage and transmission of audio and visual data have become of paramount importance. But the collecting, storage and transmission of digital data often introduce noise.

A greyscale image can be represented by the set: $\{(x_1, x_2, I(x_1, x_2))\}$ where (x_1, x_2) represents the pixel coordinates $(1 \le x_1 \le m, 1 \le x_2 \le n, \text{ for integers } m \text{ and } n)$ and $I(x_1, x_2)$ is the corresponding intensity value. Typically, $0 \le I(x_1, x_2) \le 255$ for integer $I(x_1, x_2)$ or $0 \le I(x_1, x_2) \le 1$ for real $I(x_1, x_2)$. Such an image can also be represented by the matrix [I(i, j)], where I(i, j) represents the intensity at pixel (i, j). For comparison, a colour picture could be represented by $\{(x_1, x_2, I_R(x_1, x_2), I_G(x_1, x_2), I_B(x_1, x_2))\}$, where $I_R(x_1, x_2), I_G(x_1, x_2)$ and $I_B(x_1, x_2)$ represent the red, green and blue intensities, respectively.

Additive noise, $N(x_1, x_2)$, distorts the intensity values: $I(x_1, x_2) + N(x_1, x_2)$. Random and Gaussian noises will be considered.

Human vision can often see the information in a noisy image though fine details may potentially be lost. If the original image and/or a detailed knowledge of the noise process that has distorted the image is available, it may be possible to remove the noise and restore the image. However, in real-world applications, the original clean image is unknown and the noise process may not be well-understood. Thus, any attempt to remove the noise must proceed with caution, lest more damage be done to the image.

One of the first problems encountered is how to identify the noise. Conceptually, the noise would correspond to transient features in the image. But this is unhelpful since the edges (contours of regions in the image) and fine details would also be of a transient nature and yet highly important. So, any procedure that aims to eliminate noise by eliminating transient features may further damage the image.

Many different techniques have been used to de-noise images. One common method is to use a filter matrix. Essentially, the image, as a matrix, is multiplied by another matrix, the filter. A simple example is an averaging filter, where each pixel's intensity is replaced by a (possibly weighted) average of its neighbours' intensities.

The heat, or diffusion, equation in two dimensions,

$$u_t = u_{x_1 x_1} + u_{x_2 x_2},\tag{1}$$

has been used in the reduction of noise, the rationale being that noise represents pertubations to the image. Application of the diffusion equation will smooth over the pertubations, although it may smooth edge data and fine details, further distorting the image.

Several attempts have been made to correct this limitation and to try to enhance edges by running diffusion backwards in time in the vicinity of an edge. Perona and Malik [1] were the first to try such an approach. Their idea was to replace the diffusion equation with an anisotropic diffusion equation,

$$u_t = \nabla \cdot (g(|\nabla u|) \,\nabla u),$$

where $g(\cdot)$ is a non-negative, monotonically decreasing function with g(0) = 1. Diffusion is controlled by the function $g(\cdot)$. Along an edge or contour, the gradient is large in magnitude and normal to it. Diffusion is encouraged within regions where ∇u is small, but not across the boundaries of regions (edges). So, $g(\cdot)$ is larger within smoothness regions and smaller at edges. The goal is to smooth in directions parallel to the edge, but not perpendicularly to it to preserve the edge and to try to run the diffusion backwards perpendicularly to the edge to enhance it.

Other researchers, including Alvarez, Lions and Morel [2, 3, 4, 5, 6, 7], have furthered this work and corrected limitations in Perona and Malik's scheme. They discovered that Perona and Malik's scheme actually enhances and does not remove some types of noise and it is unstable, as the solutions to slightly different initial conditions may diverge. Also, Perona and Malik's scheme requires pre-filtering in the case of noisy images. They suggested some further extensions [7]:

$$u_t = \nabla \cdot (g(|\nabla G_{\sigma} * u|) \nabla u), \qquad G_{\sigma}(x) = C\sigma^{-1/2} \exp(-|x|^2/4\sigma),$$

where G_{σ} is a Gaussian with variance σ and * is convolution, namely,

$$f * g = \int_{-\infty}^{\infty} f(x) g(t - x) dx.$$

This model is like Perona and Malik's, with the function $g(\cdot)$ to control edge enhancement, but now there is a different argument that is a superior estimator. The need for pre-filtering noise is eliminated. A more improved scheme is:

$$u_t = g(|G * \nabla u|) |\nabla u| \nabla \cdot \frac{\nabla u}{|\nabla u|},$$

where G is a smoothing kernel (like a Gaussian) [5]. The term

$$|\nabla u|\nabla \cdot \frac{\nabla u}{|\nabla u|}$$

ensures that diffusion proceeds in directions orthogonal to ∇u . The term $g(|G * \nabla u|)$ controls the edge enhancement as in their previous scheme above. A set of axioms for image processing is formalized in [2, 3, 4].

Torkamani-Azar and Tait [8] have suggested

$$u_t = \nabla \cdot (g(\nabla[h * u]) \nabla u), \quad \text{where} \quad h(x_1, x_2) = \frac{\beta}{2} \exp(-\beta(|x_1| + |x_2|)),$$

and β is a constant. Their method was also developed to correct the limitations of Perona and Malik and to be simpler to implement when discretized. Better smoothing is achieved than in Perona and Malik [8]. Torkamani-Azar and Tait found that there was a trade-off between sharpening edges and removing noise when choosing the value of the constant β . Smaller β led to better noise removal, whereas larger β preserved edges better. It was thus desirable to run several iterations with small β for the first run and larger β for the rest to remove noise on the first pass and then enhance the edges after.

The results from these schemes are good. Noise is significantly reduced and edges are preserved or enhanced. To implement these schemes, the partial differential equations (PDE's) have to be discretized into difference equations. The intricacies in the above schemes stem from the desire to distinguish edges from noise and preserve or enhance the edges while diffusing the noise away. The above methods attempt to control the direction of diffusion using the gradient of the image and then diffuse a little in some areas, diffuse more in others and run the diffusion backwards in time in other regions of the image.

Wavelets have also proven useful in the smoothing of noise in digital images [9]. In [10] Fontaine and Basu use wavelets to solve the anisotropic diffusion equation for edge detection since wavelets can provide a better representation of singularities. An efficient scheme for solving the PDE is developed using the wavelet expansion of the image, resulting in improved performance over the standard techniques. In [11], Mallat describes operations of signal and image processing using wavelets for compression and denoising. In [12] a statistical model for images is used in denoising by means of an overcomplete wavelet representation. The coefficients are modelled statistically as a Gaussian scale mixture and a local Wiener estimator is used. Gaussian white noise is removed from the Barbara image very well. In [13], multiwavelet filter banks are used in image compression and denoising, with results that are superior to scalar wavelet techniques. Multiwavelets offer orthogonality, symmetry and short support. A new family of multiwavelets, called constrained pairs, is developed for the applications. The books by Strang and Nguyen [14] and Mallat [15] cover image denoising with wavelets.

Lina, Turcotte and Goulard [16] have used complex-valued symmetric Daubechies wavelets to denoise and enhance images. The complex representation of a real signal adds phase information that is exploited by projection onto convex spaces (POCS). The POCS algorithm reconstructs the image through the coherency of the phase information. The complex wavelet transform of a real image results in the presence of both a smoothing kernel and its Laplacian in the scaling function. This information is used to synthesize a new image from the coefficients. The technique can be applied to denoising, enhancement, restoration or estimation of images.

Naoki Saito [17] uses a minimum description length (MDL) principle to achieve simultaneous additive white Gaussian noise suppression and signal compression. Because of the MDL criterion, this algorithm does not require the user to specify any parameter or threshold values. The Coifman shift-denoise-average algorithm is applied with the MDL based algorithm to reduce noise and Gibbs-like phenomena around edges so that the residual error becomes closer to white Gaussian noise.

A non-linear algorithm based on singular value decomposition block processing has been proposed by Devčić and Lončarić [18] for filtering noise in images. By filtering the singular values and the left and right singular vectors, a gain of the order of 3.6 dB (decibels) is obtained for the Lena image which had been degraded by Gaussian noise to a signal-to-noise ratio of 10 dB, which is $10 \log_{10}(SNR)$ with SNR defined by (2).

In this paper a simple multi-directional diffusion method is proposed to remove noise with PDE's while preserving edges and details by diffusing in all directions by a small amount in the Fourier domain, thereby reducing the distortion caused by the noise and yet not damaging the image details too much at the same time [19, 20, 21].

The quality of a restored image is measure in decibels by the signal-to-noise ratio (SNR) [8]:

$$SNR = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} u(i,j)^2}{\sum_{i=1}^{m} \sum_{j=1}^{n} [u(i,j) - U(i,j)]^2} = \frac{\|u\|_F^2}{\|u - U\|_F^2},$$
(2)

where [u(i, j)] and [U(i, j)] represent the original and noisy images, respectively, as $m \times n$ matrices and $\|\cdot\|_F$ is the Frobenius matrix norm. Ideally, if noise were perfectly removed from a noisy image, the result would be u = U and

 $SNR = \infty$. In general, higher SNR values signify a better result, though visual observation is the true measurement especially at low bit per pixel.

1.2 Fourier Transforms

The continuous Fourier transform $\hat{f}(\xi)$ of a function f(x) defined over \mathbb{R}^2 and the inverse Fourier transform of $\hat{f}(\xi)$ is

$$\widehat{f}(\xi) = \mathcal{F}\{f(x)\} = \int e^{-i\xi \cdot x} f(x) \, dx, \qquad f(x) = \mathcal{F}^{-1}\{\widehat{f}(\xi)\} = \frac{1}{(2\pi)^2} \int e^{ix \cdot \xi} \widehat{f}(\xi) \, d\xi. \tag{3}$$

A fundamental result that is used in this paper is concerned with the Fourier transform of parallel straight lines and parallel straight segments. The continuous Fourier transform of a line impulse distribution in \mathbb{R}^2 is a line impulse distribution at a right angle with respect to the original line impulse distribution. It is enough to state this result for a line impulse along the horizontal x_1 -axis.

Proposition 1. Let

$$f(x_1, x_2) = 1_{x_1} \otimes \delta(x_2)$$

be a line impulse distribution along the x_1 -axis. Then the Fourier transform

$$f(\xi_1,\xi_2) = 2\pi\delta(\xi_1) \otimes 1_{\xi_2}$$

is a line impulse distribution along the ξ_2 -axis. The Fourier transforms of parallel line impulse distributions differ by the phase of their elements.

The discrete Fourier transform (DFT) $X(k_1, k_2)$ of a sequence $x(n_1, n_2)$ and the inverse Fourier transform of $X(k_1, k_2)$ are

$$X(k_1, k_2) = \sum_{n_2=1}^{N} \sum_{n_1=1}^{N} x(n_1, n_2) e^{-2\pi i (k_1 - 1)(n_1 - 1)/N} e^{-2\pi i (k_2 - 1)(n_2 - 1)/N}$$
(4)

and

$$x(n_1, n_2) = \frac{1}{N^2} \sum_{k_2=1}^{N} \sum_{k_1=1}^{N} X(k_1, k_2) e^{-2\pi i (k_1 - 1)(n_1 - 1)/N} e^{-2\pi i (k_2 - 1)(n_2 - 1)/N}.$$
(5)

A result similar to Proposition 1 holds for the discrete Fourier transform of a line impulse. Hereafter, the MATLAB convention of a colon, that is, 1 : N means 1, 2, ..., N, will be used for convenience.

Proposition 2. Let

$$x(n_1, n_2) = \begin{cases} 1, & n_1 = 1, & n_2 = 1 : N, \\ 0, & otherwise. \end{cases}$$

be a line impulse along the first row of an $N \times N$ matrix. Then the discrete Fourier transform

$$X(k_1, k_2) = \begin{cases} 1, & k_1 = 1 : N, & k_2 = 1, \\ 0, & otherwise. \end{cases}$$

is a line impulse along the first column of the matrix. The Fourier transforms of parallel line impulses differ by the phase of their elements.

1.3 Pseudodifferential Operators

Given a function f(x) and a symbol $g(x,\xi)$, defined over \mathbb{R}^2 and $\mathbb{R}^2 \times \mathbb{R}^2$, respectively, the classical pseudodifferential operator G is defined by the formula

$$Gf(x) = \frac{1}{(2\pi)^2} \int e^{ix \cdot \xi} g(x,\xi) \widehat{f}(\xi) \, d\xi, \tag{6}$$

or

$$Gf(x) = \frac{1}{(2\pi)^2} \iint e^{i(x-y)\cdot\xi} g(x,\xi)f(y) \, dy \, d\xi$$

If we define the kernel

$$k(x,y) = \frac{1}{(2\pi)^2} \int e^{iy \cdot \xi} g(x,\xi) \, d\xi,$$

then G has the integral operator representation

$$Gf(x) = \int k(x, x - y)f(y) \, dy.$$

Friedrichs [22], p. 15, has introduced the so-called cokernel

$$\gamma(\chi,\xi) = \int e^{-ix\cdot\chi}g(x,\xi)\,dx$$

to define

$$Gf(x) = \frac{1}{(2\pi)^4} \iint e^{ix \cdot \chi} \gamma(\chi - \xi, \xi) \widehat{f}(\xi) \, d\xi \, d\chi.$$

Finally, harmonic analysts [23], p. 304–305, consider the operator G as a time-frequency operator with weight, or spreading function,

$$\sigma(\chi, y) = \int e^{-ix \cdot \chi} k(x, y) \, dx.$$

Thus

$$Gf(x) = \frac{1}{(2\pi)^2} \iint \sigma(\chi, x - y) e^{ix \cdot \chi} f(y) \, dy \, d\chi$$
$$= \frac{1}{(2\pi)^2} \iint \sigma(\chi, u) e^{ix \cdot \chi} f(x - u) \, du \, d\chi$$
$$= \frac{1}{(2\pi)^2} \iint \sigma(\chi, u) (M_{\chi} T_u f)(x) \, du \, d\chi,$$

where the translation operator T and the modulation operator M are defined by the formulae

$$T_y f(x) = f(x - y), \qquad M_\omega f(x) = e^{ix \cdot \omega} f(x).$$

In the context of this paper, the representation (6) is the most convenient. Taking the Fourier transform of the diffusion equation (1) for $u(x_1, x_2, t)$, one has

$$\widehat{u}_t = -\xi \cdot \xi \,\widehat{u},$$

which admits the solution, in the frequency domain,

$$\widehat{u}(\xi,t) = \widehat{u}(\xi,0) e^{-\xi \cdot \xi t}$$

or, in the time domain,

$$u(x,t) = \frac{1}{(2\pi)^2} \int e^{ix\cdot\xi} e^{-\xi\cdot\xi\,t} \widehat{u}(\xi,0) \,d\xi.$$

$$\tag{7}$$

This is a pseudodifferential operator representation of the solution to (1), with symbol $\exp(-\xi \cdot \xi t^2)$, independent of x. Since the heat equation is a hypoelliptic equation, to give a meaning to this pseudodifferential operator over $L^2(\mathbb{R}^2)$ when the equation has variable coefficients, one needs to appeal to the Calderón-Vaillancourt theorem [24]. However, in this paper, the discretized versions of these pseudodifferential operators (6) and (7) have meaning over a finite matrix even if its symbol depends also on x. The discretization of more general pseudodifferential operators may be done by means of pseudodifference operators [25, 26].

The product filter, introduced in Section 2, applies the diffusion equation in the form (7) in many directions simultaneously to a matrix representing a noisy image.

2 The Product Filter

The product matrix filter uses a one-dimensional version of equation (1) to diffuse minimally, but in many directions. In other words, it is a blind diffusion process that proceeds without any restrictions with regards to position in the image or the nature of the point (be it a noisy point or an edge point), under the belief that the diffusion is strong enough to smooth some of the noise, but not so strong as to ruin image details.

To apply the diffusion equation in a specific direction, a change of variable is required. Diffusion in the x_1 -direction is governed by $u_t = u_{x_1x_1}$ and in the x_2 -direction by $u_t = u_{x_2x_2}$. To diffuse in the direction of a line that makes an angle of θ with the x_1 -axis, the governing equation is

$$u_t = \cos^2 \theta \, u_{x_1 x_1} + 2 \sin \theta \cos \theta \, u_{x_1 x_2} + \sin^2 \theta \, u_{x_2 x_2}. \tag{8}$$

The Fourier transform of (8) is

$$\widehat{u}_t = -\xi_1^2 \cos^2 \theta \, \widehat{u} - 2 \, \xi_1 \, \xi_2 \, \sin \theta \cos \theta \, \widehat{u} - \xi_2^2 \sin^2 \theta \, \widehat{u}$$
$$= -[\xi_1 \cos \theta + \xi_2 \sin \theta]^2 \, \widehat{u}.$$

This equation is easily solved in the Fourier domain:

$$\widehat{u}(\xi_1, \xi_2, t) = \widehat{u}(\xi_1, \xi_2, 0) \exp(-[\xi_1 \cos \theta + \xi_2 \sin \theta]^2 t).$$
(9)

Applications of (9) to an image in the Fourier domain require multiplications, as compared, say, to finite differences in x-space.

Now, if diffusion is applied in many directions at once, specified by angles θ_k , (8) and (9) become

$$u_t = \sum_k [\cos^2 \theta_k \, u_{x_1 x_1} + 2 \sin \theta_k \cos \theta_k \, u_{x_1 x_2} + \sin^2 \theta_k \, u_{x_2 x_2}],$$

and

$$\widehat{u}_t = -\left(\sum_k [\xi_1 \cos \theta_k + \xi_2 \sin \theta_k]^2\right) \widehat{u},\tag{10}$$

respectively. Equation (10) has solution

$$\widehat{u}(\xi_1, \xi_2, t) = \widehat{u}(\xi_1, \xi_2, 0) \exp\left(-\sum_k [\xi_1 \cos \theta_k + \xi_2 \sin \theta_k]^2 t\right).$$
(11)

So, if the initial condition, $\hat{u}(\xi_1, \xi_2, 0)$, is taken to be the Fourier transform of a noisy image,

 $\widehat{u}(\xi_1,\xi_2,0) = \mathcal{F}\{I(x_1,x_2) + N(x_1,x_2)\},\$

then (11) reduces to matrix multiplication with an appropriately chosen value of t, as will be explained below.

The Fourier transform of an image, which is represented by a matrix, say of size $m \times n$, will also be an image of the same dimensions in MATLAB. The origin of the $\xi_1\xi_2$ coordinate system in the Fourier domain will be at the centre of the image matrix,

$$((m+1)/2, (n+1)/2),$$

with the ξ_1 -axis running downwards and the ξ_2 -axis to the right (this corresponds to the matrix coordinate system of MATLAB). And so, the (ξ_1, ξ_2) coordinates of matrix element (i, j) are

$$(\xi_1,\xi_2) = (i - (m+1)/2, j - (n+1)/2).$$

The algorithm to apply (11) proceeds in the following manner. The number of directions, p, is chosen defining the p angles $\{\theta_k \mid 0 \le k \le p-1\}$, where

$$\theta_0 = 0, \ \theta_1 = \pi/p, \ \theta_2 = 2\pi/p, \ \dots, \ \theta_{p-1} = (p-1)\pi/p$$

For each θ_k , a matrix with (i, j)th elements

$$[(i - (m+1)/2)\cos\theta_k + (j - (n+1)/2)\sin\theta_k]^2$$

is generated. These matrices are summed to produce a matrix with elements

$$\sum_{k=0}^{p-1} [(i - (m+1)/2)\cos\theta_k + (j - (n+1)/2)\sin\theta_k]^2.$$

This matrix is multiplied by the chosen value of t (typically of the order of 10^{-4} , as will be explained below), normalized by dividing by p, and negated. The resulting matrix is exponentiated elementwise to produce the matrix with elements

$$\exp\left(-\sum_{k=0}^{p-1} [(i-(m+1)/2)\cos\theta_k + (j-(n+1)/2)\sin\theta_k]^2 t/p\right),\tag{12}$$

which is the product filter matrix for the algorithm.

The fast Fourier transform (FFT) (using MATLAB's fft2 function) of the noisy image is multiplied elementwise by (12). MATLAB's fftshift is used to move the DC to the centre of the matrix. The inverse Fourier transform (MATLAB's ifft2) is calculated to produce the smoothed image. MATLAB's fftshift is used to move the DC to the centre of the matrix. **Lemma 1.** Filtering along orthogonal directions, θ and $\theta + \pi/2$, amounts to filtering in all directions.

Proof. Since

$$\cos(\theta + \pi/2) = -\sin\theta, \qquad \sin(\theta + \pi/2) = \cos\theta,$$

we have

$$\begin{split} \left[\xi_{1}\cos\theta + \xi_{2}\sin\theta\right]^{2} + \left[\xi_{1}\cos(\theta + \pi/2) + \xi_{2}\sin(\theta + \pi/2)\right]^{2} \\ &= \left[\xi_{1}\cos\theta + \xi_{2}\sin\theta\right]^{2} + \left[-\xi_{1}\sin\theta + \xi_{2}\cos\theta\right]^{2} \\ &= \xi_{1}^{2} + \xi_{2}^{2}. \end{split}$$

Thus the inverse Fourier transform of

$$\widehat{u}_t = \left(\xi_1^2 + \xi_2^2\right)\widehat{u}$$

is

$$u_t = u_{x_1x_1} + u_{x_2x_2}. \quad \bot$$

Let p = 2 and $\theta_1 = \theta_0 + \phi$ in (12). If $\phi = 0$, the level lines of function (12) are parallel lines orthogonal to the line with slope θ_0 . As ϕ increases, the level curves are ellipses with major axes orthogonal to the line with slope $\theta_1 + \phi/2$. If $\phi = \pi/2$, the level curves are circles.

3 Properties of the Product Filter

A quantitative study of the effect of the value of the t parameter in (11) proceeded in the following manner [19]. Two images, a constant and Barbara, were subjected to both random and Gaussian noises and the product filter was applied with several values of t and several values of p not all reported here. The SNR ratio of the processed image to the SNR of the noisy image was calculated. Since Barbara is a 256×256 image with mean intensity 118.8125, the constant image was taken to be 256×256 with all entries equal to 120 for a fair comparison.

Figure 1 presents the results for random noise of the form $50 \operatorname{rand}(m, n)$ and p = 256. In MATLAB, rand(m,n) generates a matrix of size $m \times n$ where all values are random numbers between 0 and 1 with a uniform distribution (*i.e.* all values are equally likely). Noise of the form $50 \operatorname{rand}(m, n)$ has a mean of 25 and a standard deviation of approximately 14.5. The SNR ratio increases for the constant image because more smoothing produces a smoother image. For Barbara, the SNR ratio increases to a peak at approximately $t = 10^{-4}$ and then decreases. So, numerically, the optimal t value is near 10^{-4} .

These calculations were repeated for Gaussian noise of the form $25 \operatorname{randn}(m, n)$. In MATLAB, randn(m,n) produces a matrix of size $m \times n$, whose elements are normally distributed with mean 0 and variance 1. If $\operatorname{randn}(m,n)$ is multiplied by a constant C, the result is a matrix with elements with mean 0 and standard deviation C (verified numerically). So, $25 \operatorname{randn}(m,n)$ produces Gaussian white noise with mean 0 and standard deviation 25. In the case of Gaussian noise, the SNR ratio for the constant image increased rapidly with t. For Barbara, the same basic shape that was seen with the uniformly distributed random noise was seen with the Gaussian noise — the SNR ratio increased to a maximum at $t = 2 \times 10^{-4}$ and then decreased. Numerically, t = 0.0002 was seen to be the optimum here.

A visual study of the value of the t parameter was conducted by varying t over 28 values from 0.00001 to 0.01 and applying the product filter with p = 256 directions to the Barbara, Caneraman, Moon and Saturn images with noise of the form $50 \operatorname{rand}(m, n)$. The SNR values are reported in Table 1.

In all cases, it was seen that as t increased, there was more smoothing of the images. For the smaller values of t, the filter does not do anything visibly noticable. For the larger values of t, there is significant smoothing and loss of detail. The optimal numerical value was found to be around $t = 10^{-4}$. This visual study indicates that the optimum range for all four images is t = 0.0001 to t = 0.0005. In this range, there is some smoothing of the noise without a significant loss of detail. Careful visual inspection of all four images in this range determined that a value of t = 0.0003 seems to be optimal. In this case, optimal means the best compromise between noise reduction, detail preservation and differences for the four images.

The times required to do standard calculations with the product filter have been measured. The 256×256 Barbara image and the 512×512 Full Barbara image had $50 \operatorname{rand}(m, n)$ noise added to them and were processed with t = 0.0003 and p = 256. The times required for the various steps in the algorithm's calculations were measured using MATLAB's tic and toc commands and are reported in Table 2.

The times required to apply MATLAB's built-in filters, averaging and median, to these sizes of images are also given. The total times for the product filter calculations are approximately 20.2 s and 78.7 s, respectively, on a Sun



Figure 1: A graph of the SNR ratio of the processed image to the SNR of the noisy image with p = 256 directions and random noise, $50 \operatorname{rand}(m, n)$, for the constant image (\circ) and Barbara (*).

paramete	er t Barbara	Camerama	n Moon	Saturn
(noisy im	age) 19.8893	21.4869	10.6686	9.4123
0.0000	5 21.4260	23.5620	12.9007	11.1157
0.0000	6 21.5138	23.8086	13.0589	11.2771
0.0000	7 21.5515	24.0124	13.1801	11.4080
0.0000	8 21.5475	24.1786	13.2746	11.5152
0.0000	9 21.5092	24.3118	13.3492	11.6039
0.0001	l 21.4430	24.4163	13.4087	11.6779
0.0002	2 20.1207	24.5233	13.6321	12.0245
0.0003	3 18.7483	23.9878	13.6379	12.1179
0.0004	17.6938	23.3696	13.5928	12.1418
0.0005	5 16.9139	22.7933	13.5315	12.1385

Table 1: SNR for the product filter algorithm applied to the four images in the visual investigation of the parameter t. The largest value of SNR for each image is indicated in boldface.

Ultra 5, running at 360 MHz, with 256 MB RAM. It can be seen from Table 2 that the bulk of the calculation time required is to generate the filter matrix. So, if the appropriate size filter matrix already exists, the calculations is quick — approximately 5 s and 9 s, respectively (which includes display time). So, real-time implementation could be feasible in some applications.

4 Comparing the Product Filter

The performance of the product filter algorithm in denoising was evaluated and tested against other techniques, including MATLAB's built-in filter averaging.

Figures 2 to 5 present sample results of the product filter applied to two images, with different noises. Figures

Table 2: Times required for steps in the product filter algorithm applied to the Barbara	a and Full Barbara	ı images
with $50 \operatorname{rand}(m, n)$ noise, $t = 0.0003$ and $p = 256$. The times are quoted in seconds.	The times quoted	for the
MATLAB filters are the times required to apply the filters to the images.		

calculation step	256×256 image	512×512 image
load image	0.0264	0.5004
display image	1.2044	1.3370
add noise to image	0.0299	0.1729
display noisy image	1.6728	2.6414
take FFT of noisy image	0.1522	0.6605
generate filter matrix	15.3350	69.1373
apply filter	0.0783	0.4719
take IFFT	0.1800	0.7051
display processed image	1.4559	2.7622
calculate and display SNR	0.0772	0.2980
MATLAB's averaging filter	0.2799	0.4047
MatlaB's median filter	0.5512	0.6180

2 and 3 show results obtained with the Barbara image and random noise of the form $50 \operatorname{rand}(m, n)$. Figure 2 shows the original and noisy images. Figure 3 shows the results of the product filter and MATLAB's **averaging** filter (which replaces each pixel's intensity with the average over a 3×3 neighbourhood) applied to the noisy image. (SNR values are given in the captions.) It can be seen, on comparison of the images in Figure 3, that the product filter and MATLAB do a comparable job of noise smoothing and the product filter seems to preserve details a bit better. Figures 4 and 5 show results obtained with the Moon image and Gaussian noise of the form $25 \operatorname{rand}(m, n)$. Figure 4 shows the original and noisy images. Figure 5 shows the results of the product filter and MATLAB. (SNR values are given in the captions.) Again, the product filter seems to do better with detail preservation and, in this case, seems to be slightly superior to MATLAB in noise removal.



Figure 2: Left: the original Barbara image. Right: the Barbara image with random noise of intensity 50 added, SNR = 19.8893.

The product filter has been found to be comparable in denoising ability to MATLAB's **averaging** filter and inferior to the wavelet techniques [19]. However, the product filter does preserve image details reasonably well and is relatively simple and quick to implement.



Figure 3: Left: the results of the product filter applied (t = 0.0003, p = 256) to the noisy image from Figure 2, SNR = 18.7483. Right: the results of MATLAB's averaging filter, SNR = 15.6399.

References

- P. Perona and J. Malik, Scale-space and edge detection using anisotropic diffusion, IEEE Trans. Pattern Anal. Mach. Intel., 12(7) (July 1990) 629–639.
- [2] L. Alvarez, F. Guichard, P.-L. Lions and J.-M. Morel, Axiomes et équations fondamentales du traitement d'images (Analyse multiéchelle et E.D.P.), C.R. Acad. Sci. Paris, **315** série I (1992) 135–138.
- [3] L. Alvarez, F. Guichard, P.-L. Lions and J.-M. Morel, Axiomatisation et nouveaux opérateurs de la morphologie mathématique, C.R. Acad. Sci. Paris, **315** série I (1992) 265–268.
- [4] L. Alvarez, F. Guichard, P.-L. Lions and J.-M. Morel, Axioms and fundamental equations of image processing, Arch. Rational Mech. Anal., 123 (1993) 199–257.
- [5] L. Alvarez, P.-L. Lions and J.-M. Morel, Image selective smoothing and edge detection by nonlinear diffusion II, SIAM J. Numer. Anal., 29(3) (1992) 845–866.
- [6] L. Alvarez and L. Mazorra, Signal and image restoration using shock-filters and anisotropic diffusion, SIAM J. Numer. Anal., 31(2) (1994) 590–605.
- [7] F. Catté, P.-L. Lions, J.-M. Morel and T. Coll, Image selective smoothing and edge detection by nonlinear diffusion, SIAM J. Numer. Anal., 29(1) (1992) 182–193.
- [8] F. Torkamani-Azar and K. E. Tait, Image recovery using the anisotropic diffusion equation, IEEE Trans. Image Processing, 5(11) (Nov. 1996) 1573–1578.
- [9] R. Ashino, C. Heil, M. Nagase and R. Vaillancourt, Microlocal filtering with multiwavelets, Computers Math. Applic., 41(1-2) (2001) 111–133.
- [10] F. L. Fontaine and S. Basu, Wavelet-based solution to anisotropic diffusion equation for edge detection, Intl. J. Imaging Systems and Technology, 9(5 Special Issue SI) (1998) 356–368.
- [11] S. Mallat, Applied mathematics meets signal processing, Doc. Math. J. DMV Extra Volume ICM I (1998) 319–338.
- [12] J. Portilla, V. Strela, M. J. Wainwright and E. P. Simoncelli, Adaptive Wiener denoising using a Gaussian scale mixture model in the wavelet domain, Proc. 8th IEEE Int. Conf. Image Processing, Thessaloniki, Greece, Oct. 2001.
- [13] V. Strela, P. N. Heller, G. Strang, P. Topiwala and C. Heil, The application of multiwavelet filterbanks to image processing, IEEE Trans. Image Processing, 8(4) (April 1999) 548–563.



Figure 4: Left: the original Moon image. Right: the Moon image with Gaussian noise of mean 0 and standard deviation 25, SNR = 17.9926.

- [14] G. Strang and T. Nguyen, Wavelets and filter banks, rev. ed., Wellesly-Cambridge Press, Wellesley MA 02181, 1997.
- [15] S. Mallat, A wavelet tour of signal processing, 2nd edition, Academic Press, San Diego, CA, 1999.
- [16] J.-M. Lina, P. Turcotte and B. Goulard, Complex dyadic multiresolution analyses, in Advances in Imaging and Electron Physics, 109, 163–197, 1999.
- [17] N. Saito, Local feature extraction and its applications using a library of bases, In R. Coifman, ed. Topics in Analysis and its Applications. Selected Theses. World Scientific, Singapore, pp. 261-451, 2000.
- [18] L' Devčić and S. Lončarić, Non-linear image noise filtering algorithm based on SVD block processing, preprint.
- [19] S. J. Desjardins, Image Analysis in Fourier Space, Ph.D. Thesis, University of Ottawa, May 2002.
- [20] S. J. Desjardins and R. Vaillancourt, Image de-noising by a multi-directional diffusion equation, Math. Rep. Acad. Sci. Canada, 24(2) 2002, 77–84.
- [21] R. Ashino, S.J. Desjardins, C. Heil, M. Nagase et R. Vaillancourt, Microlocal analysis, smooth frames and denoising in Fourier space, J. of Asian Information-Science-Life, 1(2) (2002) 153-160.
- [22] K. O. Friedrichs, Pseudo-differential operators. An introduction, Notes prepared with the assistance of R. Vaillancourt, revised edition, April 1970, Courant Institute of Mathematical Sciences, New York University, New York, 1970
- [23] K. Gröchenig, Foundations of time-frequency analysis, Birkhäuser, Boston, 2001.
- [24] A. P. Calderón and R. Vaillancourt, On a class of bounded pseudo-differential operators, Proc. Nat. Acad. Sci. USA 69 (1972) 1185–1187.
- [25] R. Vaillancourt, A Strong form of Yamaguti and Nogi's stability theorem for Friedrichs' scheme, Publ. RIMS, Kyoto, 5 (1969) 113–117.
- [26] R. Vaillancourt, On the stability of Friedrichs' scheme and the modified Lax-Wendoff scheme, Math. Comput. 24 (1970) 767–770.



Figure 5: Left: the results of the product filter applied (t = 0.0003, p = 256) to the noisy image from Figure 4, SNR = 225.9920. Right: the results of MATLAB's averaging filter, SNR = 141.5333.