Recursivity and PDE's in image processing *

L.Alvarez[1], R.Deriche[2] and F.Santana[1]

[1]Departamento de Informática y Sistemas
Universidad de Las Palmas de Gran Canaria
Campus de Tafira, 35017 Las Palmas, SPAIN

[2]INRIA Sophia-Antipolis
2004 Route de Lucioles
06902 Sophia-Antipolis
France

Abstract

Recursive filtering structures reduce drastically the computational effort required for different tasks in image processing. These operations are done with a fixed number of operations per output point independently of the size of the neighborhood considered. In this paper we show as implicit numerical implementations of some partial differential equations (PDE's) provide algorithms that can be interpreted in terms of recursive filters. We show, in particular, that a classical second order recursive filter introduced by Deriche, is, in fact, a particular implementation of the heat equation. Using the well-known Neumann boundary condition for the heat equation, we propose some new implementation of the filter. We extend this linear filter to a nonlinear recursive smoothing filter, following the general idea of slowing the diffusion in the region where the gradient is high. We present some comparison results with the classical Perona-Malik model.

Keywords: Recursivity, Image Smoothing, Partial Differential Equations, Multiscale Analysis, Image Processing, Denoising.

1 INTRODUCTION

In the fields of image processing and computer vision, a large amount of research has focused on the use of multi-resolution techniques. The main idea is to convolve the image

*Research supported in part by the Spanish Proyect PIB95-1225 and Acciones Integradas Proyect HF1998 0098-PICASSO 99050.

with operators of increasing width, sorting out the relevant changes at each resolution and combining them into a representation that can be used effectively by later processes. The multiply sized convolutions masks required makes all these approaches computationally very time consuming. Recursive filtering structures reduce drastically the computational effort because the number of operations per output point is independent of the size of the convolution mask. Because of the excellent computational performance of the recursive filtering, it has been used intensively in the computer vision community. In particular, very fast and accurate approximations of the gaussian filter convolution have been proposed by Deriche [7] using recursive filtering

Implicit numerical implementation of some partial differential equations (PDE's) can be interpreted in terms of recursive schemes. In this paper we show the connection between the classical recursive smoothing filter introduced by Deriche in [6],[8], given by the convolution mask :

$$S_\alpha(n) = k(\alpha \mid n \mid +1)e^{-\alpha|n|} \tag{1}$$

and the numerical implementation of the heat equation. We point out the difference between these approaches and we analyze the influence of the boundary condition in the numerical schemes. The classical Neumann boundary condition for the heat equation provides some new implementations of the filter which preserve better the mass of the original signal

We extend the linear filter (1), to a nonlinear recursive smoothing filter following the general idea of slowing the diffusion in the places where the gradient of the signal is high. This is a well-known idea introduced in computer vision by Perona and Malik in [11] and extended to other models for a number of authors (see [1], [13], or [9])

The paper is organized as follows: In section 2 we show the connection between the heat equation and filter (1). In section 3 we present the nonlinear extension of the linear filter and we present some comparison results with the Perona-Malik model. In section 4 we analyze the case of $2-D$ signals, and finally in section 5 we present some conclusions.

2 Recursive filters and the Heat Equation.

The convolution mask given by (1) is the integral of the optimal edge detector according to Canny's criteria [5] for infinite extent filters. k is chosen in such a way that

$$\sum_{n=-\infty}^{n=\infty} S(n) = 1$$

We can readily show that this requires:

$$k = \frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$$

We can interpret α as a scale parameter. Multiscale analysis of images is currently a very important research domain in computer vision, see for instance [1], [12], or [10]

Let us consider that x_n is an input signal and $y(n)$ is the output signal after convolution with the above mask. As it was showed in [8] a second order recursive realization of the smoothing filter $S(n)$ may be derived by the following causal and anticausal sequences:

$$y_n^1 = k\left(x_n + e^{-\alpha}(\alpha-1)x_{n-1}\right) + 2e^{-\alpha}y_{n-1}^1 - e^{2\alpha}y_{n-2}^1 \tag{2}$$

$$y_n^2 = k\left(e^{-\alpha}(\alpha+1)x_{n+1} - e^{-2\alpha}x_{n+2}\right) + 2e^{-\alpha}y_{n+1}^2 - e^{-2\alpha}y_{n+2}^2$$

$$y_n = y_n^1 + y_n^2. \tag{3}$$

The support of the mask associated to filter $S(n)$ is infinity. However, the above recursive implementation of such filter requires only 8 multiplications and 7 additions per output element. This fact point out the computational advantages of recursive realization of filters.

In figure 1 we present an example of a numerical realization of this recursive scheme for some values of α. We use as input signal the grey level values of a line of the image of the figure 2.

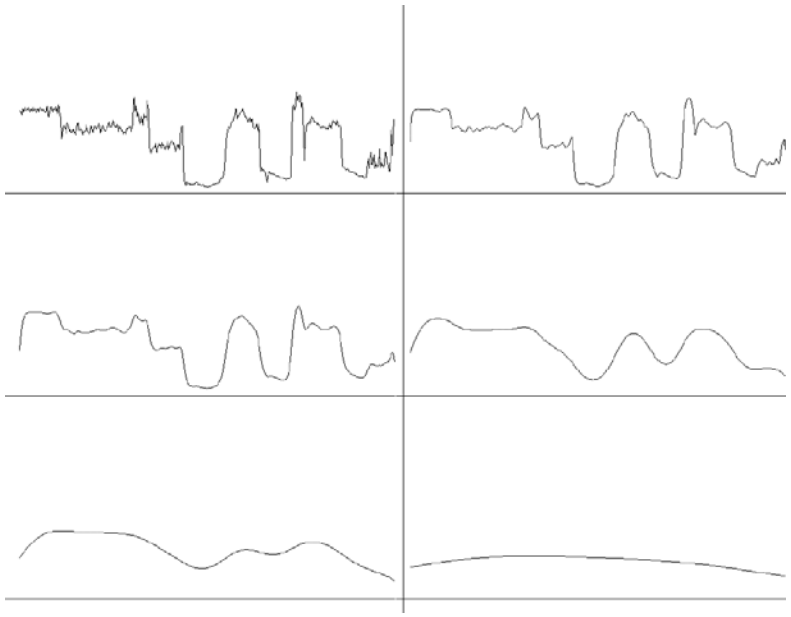

Figure 1: We present from left to right and from top to down, an original input signal and the convolution with the filter (2) with $\alpha = 1, 0.5, 0.1, 0.05, 0.01$

In practical cases, in order to apply the filter to a finite size signal, some assumptions have to be imposed about the behavior of the signal in the boundary. In the experience presented in figure 1, we extend by 0 the original signal beyond its boundary, as it was proposed by Deriche in [8]. We notice, as such boundary condition can introduce artificial

Figure 2: A $512x512$ input real image where we have emphasized the line 200.

edges near of the boundary. In order to avoid such artifacts we can use as boundary condition:

$$\begin{aligned} y_n &= x_1 \ for \ n < 1 \\ y_n &= x_N \ for \ n > N \end{aligned} \tag{4}$$

in this case we can show easily that it provides the next modification in the original recursive scheme:

$$\begin{aligned} y_1^1 &= x_1 \frac{k\left(1+e^{-\alpha}(\alpha-1)\right)}{1-2e^{-\alpha}+e^{-2\alpha}} \\ y_2^1 &= kx_2 + ke^{-\alpha}(\alpha-1)x_1 + \left(2e^{-\alpha}-e^{-2\alpha}\right)y_1^1 \\ y_N^2 &= x_N \frac{k\left(e^{-\alpha}(\alpha+1)-e^{-2\alpha}\right)}{1-2e^{-\alpha}+ke^{-2\alpha}} \\ y_{N-1}^2 &= k\left(e^{-\alpha}(\alpha+1)-e^{-2\alpha}\right)x_N + \left(2e^{-\alpha}-ke^{-2\alpha}\right)y_N^2 \end{aligned}$$

The transfer function of the filter (1) is given by the function

$$\widehat{S}_\alpha(w) = \frac{\left(-1+e^{-\alpha}\right)^2}{2\alpha e^{-\alpha}+1-e^{-2\alpha}} \frac{\left(\left(\alpha e^{-3\alpha}+\alpha e^{-\alpha}-e^{-\alpha}+e^{-3\alpha}\right)\left(e^{iw}+e^{-iw}\right)-4\alpha e^{-2\alpha}-e^{-4\alpha}+1\right)}{\left(1+e^{-2\alpha}-e^{-\alpha}(e^{iw}+e^{-iw})\right)^2}$$

the study of the transfer function is very important in our analysis because it provides the way of represents the filter in terms of numerical implementations of the heat equation.

We consider, next, the heat equation in dimension 1 given by the partial differential equation:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Let u_i^n an discrete approximation of $u(i\Delta x, n\Delta t)$, we are going to use 2 well-known discretization of the heat equation. The explicit scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{5}$$

we can interpret the computation of u^{n+1} as the convolution u^n with a kernel $E_\lambda(n)$ which has as transfer function:

$$\widehat{E_\lambda}(w) = (1 - 2\lambda) + \lambda(e^{iw} + e^{-iw}) \tag{6}$$

where $\lambda = \frac{\Delta t}{\Delta x}$

We will use also the implicit scheme:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} \tag{7}$$

a classical way to compute u^{n+1} from u^n for a signal of size N is to impose the Neumann boundary condition:

$$\begin{aligned} u_i^{n+1}(0) &= u_i^{n+1}(1) \\ u_i^{n+1}(N) &= u_i^{n+1}(N+1) \end{aligned} \tag{8}$$

in order to obtain $u^{n+1}i$, we need to solve a tridiagonal system. We use a LU decomposition (so called Thomas Algorithm) of the tridiagonal matrix in the following way:

$$\underbrace{\begin{pmatrix} a_1 & b_1 & 0 & 0 & 0 \\ c_1 & a_2 & b_2 & 0 & 0 \\ 0 & c_2 & . & . & 0 \\ 0 & 0 & . & . & b_{N-1} \\ 0 & 0 & 0 & c_{N-1} & a_N \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ l_1 & 1 & 0 & 0 & 0 \\ 0 & l_2 & . & . & 0 \\ 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & l_{N-1} & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} d_1 & u_1 & 0 & 0 & 0 \\ 0 & d_2 & u_2 & 0 & 0 \\ 0 & 0 & . & . & 0 \\ 0 & 0 & . & . & u_{N-1} \\ 0 & 0 & 0 & 0 & d_N \end{pmatrix}}_{U} =$$

where:

$$\begin{aligned} d_1 &= a_1 \\ l_j &= \frac{c_j}{d_j} \qquad j = 1, ..., N-1 \\ d_{j+1} &= a_{j+1} - l_j b_j \qquad j = 1, ..., N-1 \\ u_j &= b_j \qquad j = 1, ..., N-1 \end{aligned}$$

Once the LU decomposition is obtained, we solve the tridiagonal system in the following way: First, we introduce $\{z_j\}_{j=1,...,N}$ defined by:

$$\begin{aligned} z_1 &= u_1^n \\ z_j &= u_j^n - l_{j-1} z_{j-1} \qquad j = 2, ..., N \end{aligned}$$

and finally, we obtain the output signal u_j^{n+1} by the following algorithm:

$$\begin{aligned} u_N^{n+1} &= \frac{z_N}{d_N} \\ u_j^{n+1} &= \frac{z_j - u_j u_{j+1}^{n+1}}{d_j} \qquad j = N-1, ..., 1 \end{aligned}$$

we notice that, in fact, the resolution of the tridiagonal system using the Thomas algorithm is a particular case of recursive filter structure. Because of its very nice properties, Thomas algorithm have been used before for other authors (see for instance [14] and [15]) in order to implement some nonlinear diffusion filtering models in computer vision.

The transfer function of filter $I_\lambda(n)$ is given by

$$\widehat{I_\lambda}(w) = \frac{1}{(1+2\lambda) - \lambda(e^{iw} + e^{-iw})} \tag{9}$$

Another way to implement $I_\lambda(n)$ is to use the decomposition of $\widehat{I_\lambda}(w)$ proposed by Alvarez-Mazorra in [3]:

$$\widehat{I_\lambda}(w) = \frac{\lambda}{\upsilon} \frac{1}{(1 - \upsilon e^{iw})} \frac{1}{(1 - \upsilon e^{-iw})} \tag{10}$$

where

$$\upsilon = \frac{1 + 2\lambda - \sqrt{1 + 4\lambda}}{2\lambda} \tag{11}$$

using this decomposition we can compute u_j^{n+1} in the following way:

$$\begin{aligned} 1. \qquad & u_1^{n+\frac{1}{3}} = \frac{u_1^n}{1-\upsilon} \\ 2. \qquad & u_j^{n+\frac{1}{3}} = u_j^n + \upsilon u_{j-1}^{n+\frac{1}{3}} \ for \ j = 2, .., N \\ 3. \qquad & u_N^{n+\frac{2}{3}} = \frac{u_N^{n+\frac{1}{3}}}{1-\upsilon} \\ 4. \qquad & u_j^{n+\frac{2}{3}} = u_j^{n+\frac{1}{3}} + \upsilon u_{j+1}^{n+\frac{2}{3}} \ for \ j = N-1, .., 1 \\ 3. \qquad & u_j^{n+1} = \frac{\upsilon}{\lambda} u_j^{n+\frac{2}{3}} for \ j = 1, .., N \end{aligned} \tag{12}$$

In the next lemma we show the way we can compute $S_\alpha(n)$ as a particular implementation of the heat equation

Lemma 1

$$\widehat{S}_{\lambda}(w) = \left(\widehat{I}_{\lambda_1(\alpha)}(w)\right)^2 \widehat{E}_{\lambda_2(\alpha)}(w) \tag{13}$$

where

$$\lambda_1(\alpha) = \frac{e^{-\alpha}}{(1-e^{-\alpha})^2}$$
$$\lambda_2(\alpha) = \frac{(\alpha e^{-3\alpha} + \alpha e^{-\alpha} - e^{-\alpha} + e^{-3\alpha})}{(2\alpha e^{-\alpha} + 1 - e^{-2\alpha})(-1+e^{-\alpha})^2}$$

Proof. A straightforward computation. ■

Remark. We can show easily that for $\alpha > 0$ we have that $\lambda_1(\alpha) > 0$ and $0 < \lambda_2(\alpha) < \frac{1}{6}$ and therefore the numerical implementation of the heat equation provided by the choice of $\lambda_1(\alpha)$ and $\lambda_2(\alpha)$ are stable.

Formally, when we apply the filter (2) on a infinite support signal, we obtain the same result using directly the convolution with the filter $S_\alpha(n)$ or the combination of $E_{\lambda_2(\alpha)}(n)$ and $I_{\lambda_1(\alpha)}(n)$ given in the above lemma. However, in practice, when we deal with finite support signal, the boundary condition that we impose can have an influence in the output signal. We are going to compare 3 different schemes:

1. $(S1)$ The original recursive scheme with the boundary condition (4).
2. $(S2)$ The implementation using the decomposition of the above lemma, and the Thomas algorithm to compute the implicit scheme $I_{\lambda_1(\alpha)}(n)$.
3. $(S3)$ The implementation using the decomposition of the above lemma, and the Alvarez-Mazorra algorithm to compute the implicit scheme $I_{\lambda_1(\alpha)}(n)$

From the point of view of the computational cost of the different algorithms, $(S1)$ requires 15 basic operations per pixel, $(S2)$ requires 17 basic operations per pixels and $(S3)$ requires 13 basic operations per pixel. Although $(S2)$ is a bit slower than the other methods, it is the only one with satisfies the property of mass preservation, that is:

$$\sum_{n=1}^{n=N} y_n = \sum_{n=1}^{n=N} x_n \tag{14}$$

In figure 3 we present the convolution of the original input signal with the model filter $S_\alpha(n)$ using schemes $(S1)$, $(S2)$ and $(S3)$ for different values of α.

We notice as in the practical cases, when we deal with big size signal (512 in this case) the difference between the 3 procedures are located near of the boundary, and when α is small. In other cases the results are similar, obviously because the theoretical convolution filter is the same for the 3 procedures. We notice also, as none of this procedures produces artifacts near of the boundary of the signal as it was the case if we extend by 0 the input signal beyond the boundary.

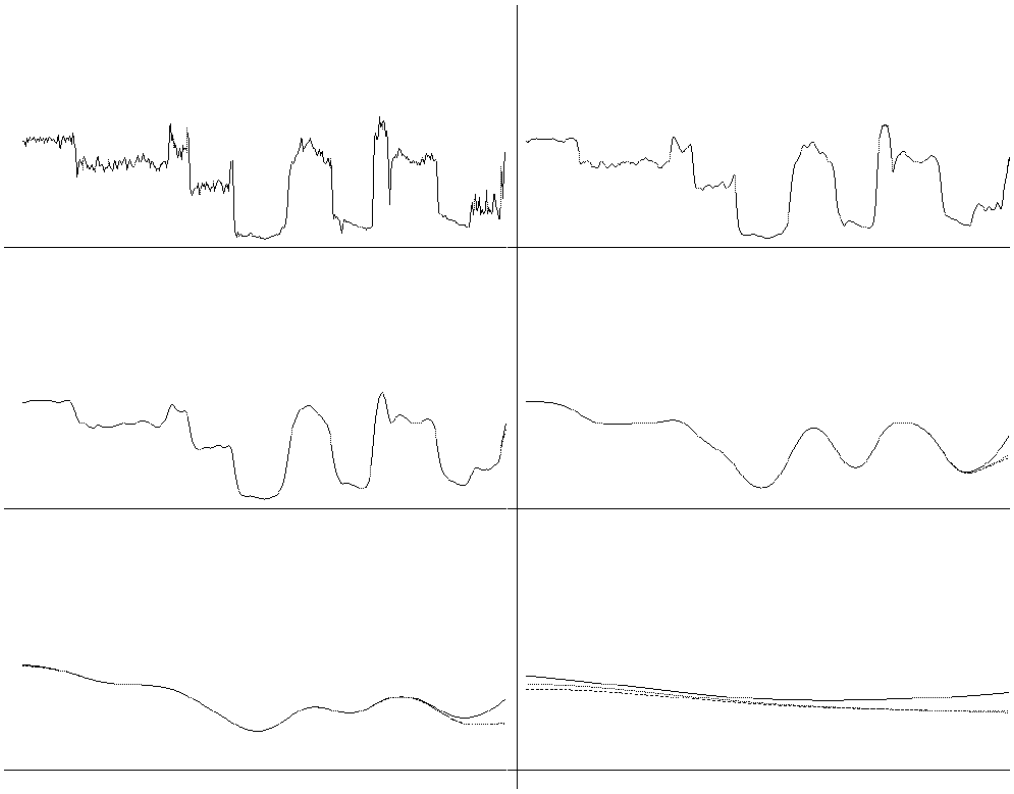

Figure 3: We present from left to right and from top to down, an original input signal and the convolution with the filter (2) with $\alpha = 1, 0.5, 0.1, 0.05, 0.01$ using the schemes $(S1)$: (continuous line), $(S2)$: (dotted line) and $(S3)$ (discontinuous line)

3 Nonlinear partial equation and recursivity.

Perona-Malik equation

$$\frac{\partial u}{\partial t} = div\left(g(\|\nabla u\|)\nabla u\right)$$

is a well-known nonlinear smoothing model (see [11]). $g(.)$ is a positive nonincreasing function which slows diffusion in the region where the norm of the gradient is high. It allows to preserve edges in the original signal. A classical choice for $g(.)$ is:

$$g(s) = e^{-\left(\frac{s}{K}\right)^2}$$

If we translate this idea of preserving edges to our original linear filter $S_\alpha(n)$, we can choose the constant α different for each input signal pixel. In fact, we take $\alpha = \alpha_n$ depending on the value of the derivative of the input signal.

$$\alpha_n = h(\mid \frac{\partial x}{\partial t}(t_n) \mid)$$

where $h(.)$ is a nondecreasing function with $h(0) = \varepsilon > 0$. In a point x_n where the derivative is big (x_n is an edge point), α_n is big and then the diffusion of the filter is lowered. In a point x_n where the derivative is small (x_n is inside a flat zone in the signal), α_n is small and then the diffusion of the filter is big. For simplicity, we are going to choose $h(.)$ linear, So we take

$$h(s) = \varepsilon + Ms$$

$h(.)$ depends on two parameters, $\varepsilon, M > 0$ In fact, the parameter what really matters is M which represents the slope of $h(.)$, ε is introduced in order to avoid the singularity when $\alpha = 0$ (ε is fixed to a small number, $\varepsilon = 0.001$, in our numerical experiences).

In order to introduce the nonlinearity given by function $h(s)$ in the linear recursive filter $S_\alpha(n)$ first we notice that it is very simple to include the nonlinearity in the scheme $(S2)$, it is enough to take $\lambda_1(\alpha_n)$ and $\lambda_2(\alpha_n)$ in the numerical implementation with the same boundary condition. However schemes $(S1)$ and $(S3)$ do not work properly when we introduce a nonconstant α. Indeed, the recursive procedures (2) and (12) work properly if α is constant, if α is not constant we can check easily that $(S1)$ and $(S3)$ produces unpredictable results which do not correspond to an smoothing filter.

The next lemma shows that in fact, the scheme $(S2)$ including the nonlinearity is consistent with a nonlinear partial differential equation.

Lemma 2 *Let $\{x_n\}_{n=1}^N$ be an input signal. We define*

$$\alpha_n = h(\left|\frac{\partial x}{\partial t}(t_n)\right|) = M\left|\frac{\partial x}{\partial t}(t_n)\right| + \varepsilon$$

the output signal $\{y_n\}_{n=1}^N$ is obtained in 3 steps

$$y_n^{\frac{1}{3}} = (1 - 2\lambda_2(\alpha_n))x_n + \lambda_2(\alpha_n)\left(x_{n+1} + x_{n-1}\right) \tag{15}$$

$$\begin{aligned}(1 + 2\lambda_1(\alpha_n))y_n^{\frac{2}{3}} - \lambda_1(\alpha_n)(y_{n+1}^{\frac{2}{3}} + y_{n-1}^{\frac{2}{3}}) &= y_n^{\frac{1}{3}} \\ (1 + 2\lambda_1(\alpha_n))y_n - \lambda_1(\alpha_n)(y_{n+1} + y_{n-1}) &= y_n^{\frac{2}{3}}\end{aligned} \tag{16}$$

where

$$\begin{aligned}\lambda_1(s) &= \frac{e^{-s}}{\left(1 - e^{-s}\right)^2} \\ \lambda_2(s) &= \frac{\left(se^{-3s} + se^{-s} - e^{-s} + e^{-3s}\right)}{\left(2se^{-s} + 1 - e^{-2s}\right)\left(-1 + e^{s}\right)^2}\end{aligned}$$

then the filtering procedure (15) is consistent with the nonlinear PDE:

$$\frac{\partial u}{\partial t} = \lambda_2(h(|\frac{\partial u}{\partial x}|))\frac{\partial^2 u}{\partial x^2} \tag{17}$$

and the filtering procedure (16) is consistent with the nonlinear PDE:

$$\frac{\partial u}{\partial t} = \lambda_1(h(|\frac{\partial u}{\partial x}|))\frac{\partial^2 u}{\partial x^2} \tag{18}$$

moreover the filtering is L^∞ stable, that is:

$$\max_{n=1,..,N}\{|y_n|\} \leq \max_{n=1,..,N}\{|x_n|\}$$

Proof. A straightforward computation shows as (15) is an explicit discretization (17), and (16) is an implicit discretization (18). To establish the L^∞ stability it is enough to show that

$$0 < \lambda_2(s) < \frac{1}{2} \quad \forall s > 0$$
$$0 < \lambda_1(s) \quad \forall s > 0$$

we can state easily that $\lambda_2(s)$ and $\lambda_1(s)$ are positive and nondecreasing for $s > 0$. Moreover:

$$Lim_{s\to 0}\frac{(se^{-3s} + se^{-s} - e^{-s} + e^{-3s})}{(2se^{-s} + 1 - e^{-2s})(-1 + e^s)^2} = \frac{1}{6} < \frac{1}{2}$$

■

Remark: we notice that in fact, this nonlinear filter is close to the Perona-Malik model using the functions $g(s) = \lambda_2(h(s))$ and $g(s) = \lambda_1(h(s))$. On the other hand we have the interesting property that when α is near a constant, the nonlinear filter approach the optimal filter $S_\alpha(n)$.

Next, we present some numerical comparison results with the Perona-Malik model. To approximate the solution of the Perona-Malik equation we will use the classical implicit numerical scheme:

$$\frac{u_i^{n+1} - u_i^n}{\triangle t} = \frac{g_{i+1}^n + g_i^n}{2}\frac{u_{i+1}^{n+1} - u_i^{n+1}}{(\triangle x)^2} - \frac{g_{i-1}^n + g_i^n}{2}\frac{u_i^{n+1} - u_{i-1}^{n+1}}{(\triangle x)^2}$$

where

$$g_i^n = e^{-\left(\frac{u_{i+1}^n - u_{i-1}^n}{2\triangle x K}\right)^2}$$

the computation of u_i^{n+1} from u_i^n yields to a tridiagonal linear system of equations that can be solved using the Thomas algorithm again. In figure 4 we present some comparison results between the nonlinear scheme (15)-(16) and the Perona-Malik model. we notice that there is not an exact equivalence between the parameters of the nonlinear filter (15)-(16) and the Perona-Malik model. So we can not expect the same results. We point out that the filter (15)-(16) is much more diffusive in the region where the gradient is small. That is because $\lambda_1(s)$ goes to ∞ when $s \to 0$.

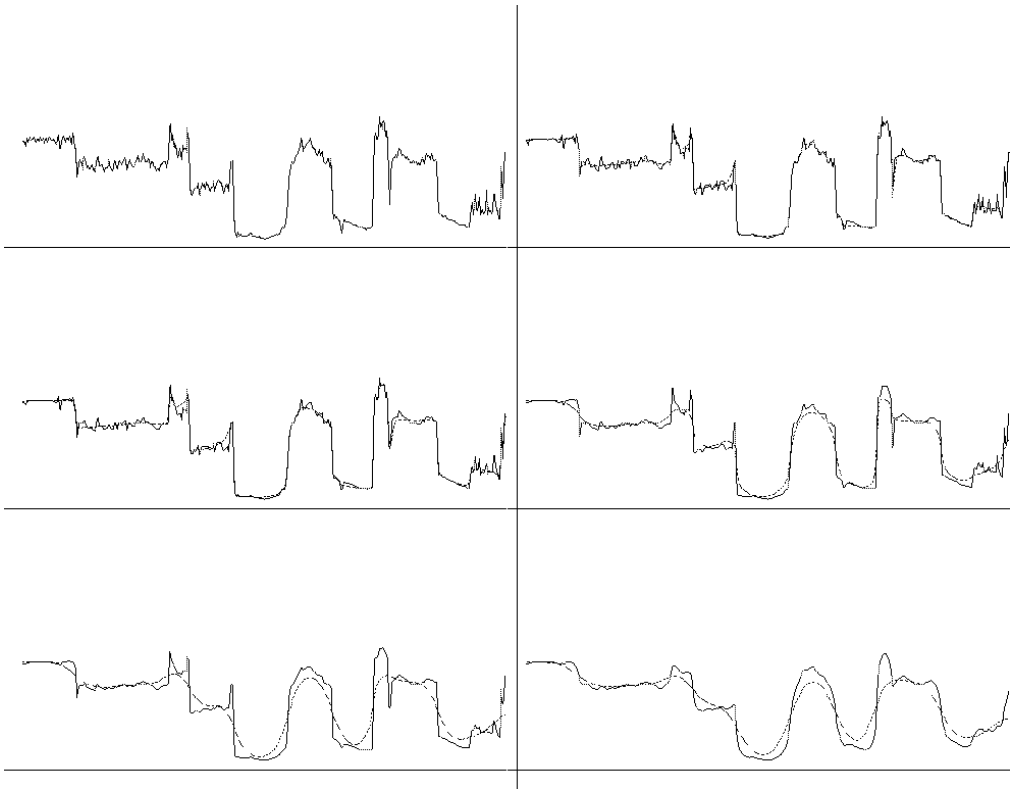

Figure 4: We present from left to right and from top to down, an original input signal and the results of nonlinear filter (15)-(16) (continuous line) with $M = 1, 0.5, 0.1, 0.05, 0.01$ and the results using the Perona-Malik algorithm (discontinuous line) with $\frac{\Delta t}{\Delta x^2} = 10$ and $K = 1, 2, 3, 7, 10$.

4 Extension to 2 dimensional signals

We can extend easily the filters to 2 dimensional signals by composing the procedures defined in the linear and nonlinear case on two orthogonal directions in the image, that is.

$$I_{output} = S^x \left(S^y \left(I_{input} \right) \right)$$

where S^x represents the filter in the x-direction and S^y represents the filter in the y-direction.

in figure 5 we present a numerical experience in $2-D$ where we use as input signal the image given in figure 2 where we have added an exponential white noise with standard deviation equal to 15.

In the nonlinear case we define α_{ij} for each pixel of the image as:

$$\alpha_{ij} = h(\|\nabla I\|_{ij})$$

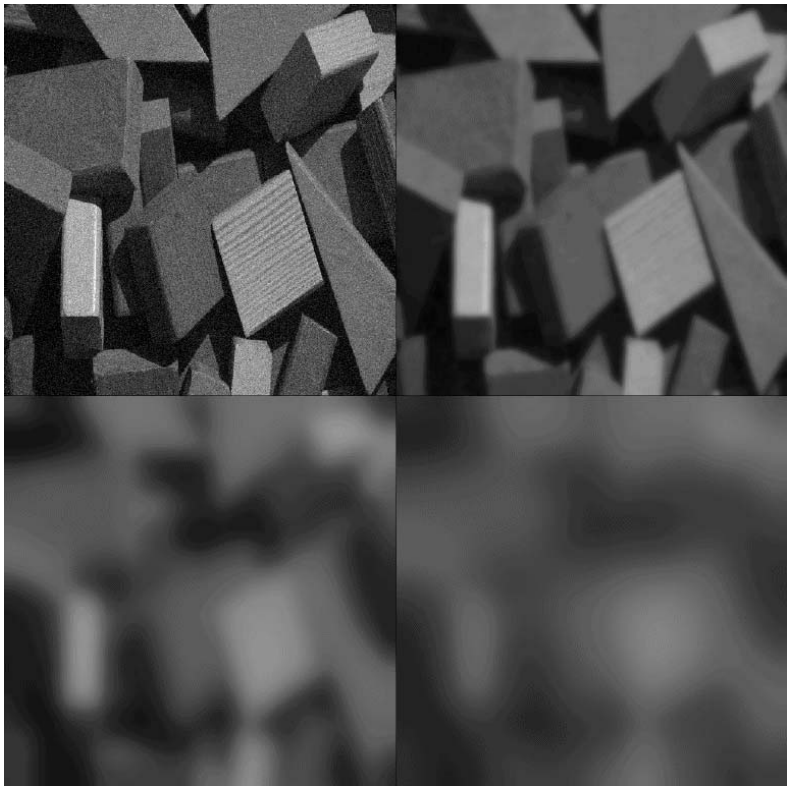

Figure 5: From left to right and from up to down, we present the original image and the convolution using the linear scheme $S2$ with $\alpha = 0.5, 0.1, 0.05$

we use the same value of α_{ij} when composing the $1-D$ procedure in 2 orthogonal directions. In this way, we minimize the influence of the order in we apply the filter en each direction. If we want a completely invariant scheme under 90^o rotations, we can define the filter as the average:

$$I_{output} = \frac{S^x\left(S^y\left(I_{input}\right)\right) + S^y\left(S^x\left(I_{input}\right)\right)}{2}$$

another way to obtain nonlinear filters invariant under 90^o rotations is to use additive schemes as it was introduced in [14].

Next, in figures 6, 7 we present some numerical experiences in $2-D$ using the nonlinear scheme (15)-(16) and the Perona-Malik model

5 CONCLUSIONS

In this paper, we have analyzed the powerful of the recursive filtering structure, in terms of computational costs. We have studied in details the classical filter given by:

$$S(n) = k(\alpha \mid n \mid +1)e^{-\alpha|n|}$$

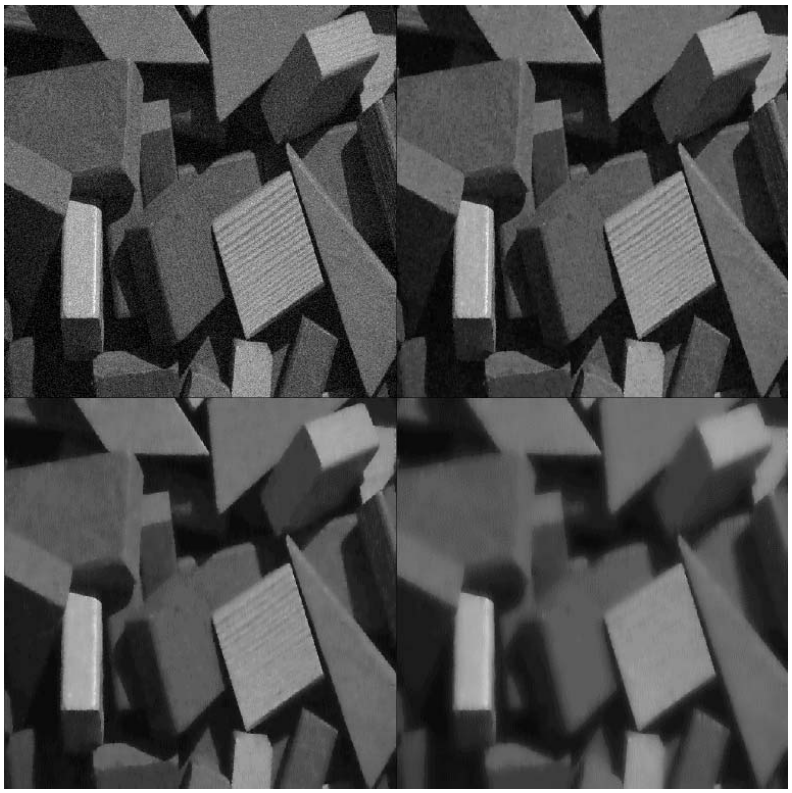

Figure 6: From left to right and from up to down, we present the original image and the filtered images using the nonlinear scheme (15)-(16) with $M = 0.25, 0.1, 0.05$.

We have showed as classical recursive implementations of such filter can be interpreted as a particular numerical implementation of the heat partial differential equation. Using the boundary condition analysis of PDE's, we provide new numerical implementations of the filters which do not create artifacts near of the boundary of the signal. We compare 3 different implementations following some criteria as the computational cost, the mass conservation, etc..

In order to preserve edges in the signal, we introduce a nonlinearity in the smoothing filter following the general idea of slowing the diffusion when the gradient of the signal is high. We show the consistence and stability of this new nonlinear filter. We present some comparison results with the classical Perona-Malik model. We present the extension of the filters in $2-D$ in a separable way. Concerning the nonlinear filter introduced, it represents an alternative to the classical nonlinear smoothing filters which enjoy of some nice properties as the stability and the simplicity and velocity of the recursive schemes.

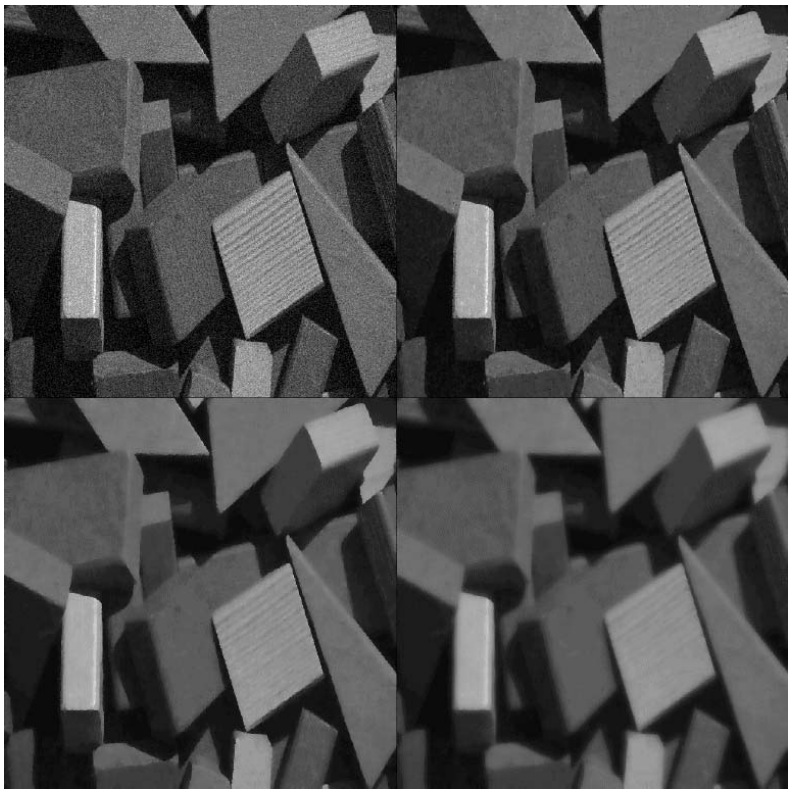

Figure 7: From left to right and from up to down, we present the original image and the filtered images using the Perona-Malik model with $\frac{\Delta t}{\Delta x^2} = 10$ and $K = 10, 20, 50$.

References

[1] L. Alvarez, F. Guichard,P.L. Lions and J.M. Morel, " Axioms and fundamental equations of image processing," *Arch. for Rat. Mechanics* Vol. 123 (3): 199-257, 1993

[2] L. Alvarez, P.L. Lions and J.M. Morel, " Image selective smoothing and edge detection by nonlinear diffusion (II)," *SIAM J. on Numerical Analysis* Vol. 29(3): 845-866, 1992

[3] L. Alvarez, and L. Mazorra, " Signal and image restoration using shock filter and anisotropic diffusion," *SIAM J. on Numerical Analysis* Vol. 31(2): 590-605, 1994

[4] F.Catté, P.L.Lions, J.M.Morel and T.Coll, " Image Selective Smoothing and Edge Detection by Nonlinear Diffusion" *SIAM J. on Numerical Analysis*, 29(1): 182-193, 1992

[5] J.F. Canny, " Finding edges and lines in images," *Technical Report 720, MIT, Artificial Intelligence*, Cambridge, Massachussets, 1983

[6] R. Deriche, " Using Canny's criteria to derive a recursively implemented optimal edge detector ," *The international J. of Computer Vision* , Vol. 1(2): 167-187, 1987.

[7] R. Deriche, " Recursively Implementing the Gaussian and its Derivatives" *In Proc. Second International Conference on Image Processing* , pp. 263-267, Singapore, 1992.

[8] R. Deriche, " Fast algorithms for low level vision," *IEEE transactions on Pattern Analysis and Machine Intelligence* , Vol. 12(1): 78-87, 1990.

[9] P. Kornprobst, R. Deriche, and G. Aubert. "Image sequence restoration: A PDE based coupled method for image restoration and motion segmentation" *In Proceedings of the 5th European Conference on Computer Vision*, volume II, pages 548-562, Fribourg, Germany, June,1998

[10] T. Lindeberg, " Scale Space theory in computer vision ," *Kluwer Academic Publishers*, 1994.

[11] P.Perona and J.Malik, " Scale-space and edge detection using anisotropic diffusion," *IEEE transactions on Pattern Analysis and Machine Intelligence* , Vol. 12: 429-439, 1990.

[12] B. Romeny (Ed.) " Anisotropic diffusion in image processing" *Teubner Verlag, Stuttgart,* 1998.

[13] J. Weickert " Efficient and reliable schemes for nonlinear diffusion filtering" *I*

[14] J. Weickert, B. Romeny and M. Viergever, " Efficient and reliable schemes for nonlinear diffusion filtering" *IEEE transactions on Pattern Analysis and Machine Intelligence*, Vol. 7(3): 398-410, 1998

[15] J. Weickert, K. Zuiderveld, B. Romeny and W. Niessen, " Parallel implementation of AOS schemes: A fast way of nonlinear diffusion filtering" *Proc. 1997 IEEE International Conference on Image Processing (ICIP-97, Santa Barbara)*, Vol. 3: 396-399, 1997