Noise Removal Using Smoothed Normals and Surface Fitting

Marius Lysaker
Stanley Osher
Xue-Cheng Tai

January 2003

CAM Report 03-03

Noise Removal Using Smoothed Normals and Surface Fitting

Marius Lysaker, Stanley Osher, Xue-Cheng Tai*

M. Lysaker and X-C. Tai are with the Department of Mathematics, University of Bergen, Johannes Brunsgate 12, N-5009 Bergen, Norway (e-mail: {mariusl,tai}@mi.uib.no). S. Osher is with the Department of Mathematics UCLA, California, USA (e-mail: sjo@math.ucla.edu).

The support from the Norwegian Research Council (project number 135302/320) and from IMAR under the contract #ICA1-CT-2000-70022 with the European Commission is greatly acknowledged.

Abstract

In this work, we use partial differential equation techniques to remove noise from digital images. The removal is done in two steps. We first use a total-variation (TV) filter to smooth the normal vectors of the level curves of the noise image. After this, we try to find a surface to fit the smoothed normal vectors. For each of these two stages, the problem is reduced to a nonlinear partial differential equation. Finite difference schemes are used to solve these equations. A broad range of numerical examples are given in the paper.

Keywords

Image denoising, nonlinear PDEs, normal processing.

I. Introduction

A digital image d can contain random noise η superimposed on the pixel intensity value by the formula

$$d_0(x,y) = d(x,y) + \eta(x,y). \tag{1}$$

We would like to recover the true image $d(x,y)$ from its noisy observation $d_0(x,y)$. Noise is recognized as fast oscillating signals and can therefore be removed by the process of low pass filtering or smoothing, unfortunately at the expense of some high-frequency information (i.e. edges). A central thrust in image processing techniques comes via partial differential equations (PDE). Among the different techniques proposed in the literature, let us mention [1], [2]. The TV-norm filter proposed in [1] gives a rigorous mathematical tool to introduce nonlinear diffusion filters and it has been used as a regularization method for many other applications where one needs to identify discontinuous functions. The TV-norm filter preserves edges but has the sometimes undesirable stair case effect, meaning that smooth functions are some times transformed into piecewise constants. To overcome this problem (but then maybe sacrifice the good property of TV-norm on the edges) many other nonlinear filters have been suggested in the literature and during the last few years higher order PDEs have been of special interest [3], [4], [5], [6]. The method we shall use in this paper is somehow related to the methods using higher order PDEs. We solve two second order nonlinear partial differential equations sequentially. If we can combine the two equations together we would need to solve a higher order nonlinear partial differential equation. To be more precise, our method is a two-step method. In the first step, we try to smooth the normal vectors of the levelset curves of the noise image using a TV-norm filter. After the normal vectors are smoothed, we try to find an image that fits the normal vectors and this image is taken as the recovered image for the noise image.

Our method is related to some techniques already in the literature. In Kenny and Langan [7] a method to restore images from modified flow field was proposed. In its simplest form this process takes a single image, modifies its gradient field and then constructs a new image from the modified field. They introduce an objective function based on a flow field to implicitly determine the edges of the image.

The idea we use in this paper is more inspired by the work [8]. In this work, they are trying to process three dimensional surfaces. The essential idea is to manipulate the normal vectors for a given three dimensional surface and then try to find a new surface that matches the processed normal vectors in a suitable way. In this work, we are trying to extend this idea to do image noise removal. Further we would like to mention that recently normal processing has also been used in mesh optimization [9].

Another closely related approach is the inpainting approach of [10]. In this work, they try to minimize an energy functional with respect to two variables: a vector field $\vec{n}$ which represents the direction of the level curves of $d(x,y)$ and the intensity value $d(x,y)$. Note that image denoising is quite different from inpainting. In [10] they clearly point out how shape recovery is achieved by first computing the vector field $\vec{n}$ approximately. We have also utilized this idea in our algorithm. We solve a minimization problem in the second step and the functional for the minimization problem in this step is identical to one of the terms in the inpainting functional [10]. Another closely related work is Vese and Osher [11], where an elegant way for solving

$$\inf_V F(V), \qquad F(V) = \int_\Omega \left| \nabla \left(\frac{V}{|V|} \right) \right|^p dx \tag{2}$$

for a vector valued function $V = (u,v) : \Omega \to R^2$ was proposed. The tools we shall use to smooth the normal vectors are closely related to problem (2). In our approach, we are essentially taking $V = \nabla d$ and p=1 in (2) and we add a fidelity term to balance the smoothing and the edge preserving.

In fact, the approach we propose here is also strongly motivated by the fourth order method proposed in [3]. The original TV-norm filter is to solve

$$\min_d F(d), \qquad F(d) = TV(d) := \int_\Omega |\nabla d| \, dx. \tag{3}$$

In order to overcome the staircase effects, it was suggested in [3] to replace the TV-norm by

$$F(d) = \int_\Omega (|d_{xx}| + |d_{yy}|) \, dx \text{ or } F(d) = \int_\Omega \sqrt{|d_{xx}|^2 + 2|d_{xy}|^2 + d_{yy}|^2} \, dx. \tag{4}$$

With these minimization functionals, we are trying to minimize the total variational norm of the gradient of d instead of d. Rather good numerical performance was obtained in [3]. We could go one step further. Instead of minimizing the TV-norm of ∇d, we could minimize the TV-norm of $\nabla d/|\nabla d|$, i.e.

$$F(d) = \int_\Omega \left|\nabla \frac{\nabla d}{|\nabla d|}\right| dx. \tag{5}$$

We know that $\nabla d/|\nabla d|$ is the unit normal vectors for the level curves of d. However, the equations obtained from the minimizer of the above functional is hard to solve numerically because of the restrictive time step needed. That is the reason that we propose in this work to split this into two steps, i.e. we first smooth the unit normal vectors and then find a surface to fit the obtained normal vectors.

This paper is organized in the following way. In Section II we introduce the two-step method we use to do noise removal. Details are given to show how we obtain and solve the associated nonlinear partial differential equations coming from the two-step method. Finite difference approximations and some implementation details are explained in Section III. Numerical results are given in Section IV. In the numerical experiments, we compare our method with some related algorithms in the literature. Finally, the Appendix is devoted to describing a transformation used in Section II.

II. Flow field smoothing and surface fitting

For a given image d, $\vec{n} = \nabla d/|\nabla d|$ is the unit normal vector of the level curves of d. For the noisy image d_0, we shall try to smooth the normal vectors $\vec{n}_0 = \nabla d_0/|\nabla d_0|$. Because the normal vectors can be discontinuous vector functions we use the TV-norm to do the smoothing. Similar to [1], [3], we also add a fidelity term to balance the smoothing and edge capturing. To be more precise, we solve the following minimization problem to get a smoothed flow field (i.e. normal vectors):

$$\min_{|\vec{n}|=1} \{\int_\Omega |\nabla \vec{n}|\, dx + \frac{\lambda}{2}\int_\Omega |\vec{n} - \vec{n}_0|^2\, dx\}. \tag{6}$$

If we have more information about the flow field, the value of λ could be updated dynamically. In our simulations, we just fixed the value of λ to be a positive constant. At the end of this section we give some advices on how to choose this constant.

Once the flow field is calculated, we try find an image d that will match the field. There are different ways to do this. In this work, we are trying to find a d that solves the following minimization problem:

$$\min_{\int_\Omega |d-d_0|^2 dx = \sigma^2} \int_\Omega \Big(|\nabla d| - \nabla d \cdot \vec{n}\Big)\, dx. \tag{7}$$

In the above, σ is the noise level in $L^2(\Omega)$ norm and we assume that we know it approximately. In case the noise level is not known, we just add a fidelity term to the minimization functional and drop the noise level constraint. Assume that α is the angle between ∇d and $\vec{n}$, it is clear that

$$\int_\Omega (|\nabla d| - \nabla d \cdot \vec{n})\, dx = \int_\Omega |\nabla d|(1 - cos(\alpha))\, dx. \tag{8}$$

This functional is always nonnegative. To minimize this functional, we need to minimize the angle between ∇d and $\vec{n}$.

We shall use a Lagrange multiplier to deal with the noise constraint $\int_\Omega |d - d_0|^2 dx = \sigma^2$. The Lagrange functional is defined as

$$L(d,\mu) = \int_\Omega (|\nabla d| - \nabla d \cdot \vec{n})\, dx + \frac{\mu}{2}\Big(\int_\Omega |d - d_0|^2 dx - \sigma^2\Big). \tag{9}$$

To find a minimizer for (7), we need to find the saddle points for L. The optimality conditions for the saddle points are:

$$-\nabla \cdot \left(\frac{\nabla d}{|\nabla d|} - \vec{n}\right) + \mu(d - d_0) = 0 \text{ in } \Omega, \quad \left(\frac{\nabla d}{|\nabla d|} - \vec{n}\right) \cdot \nu = 0 \text{ on } \partial\Omega \tag{10}$$

and

$$\int_\Omega |d - d_0|^2 dx = \sigma^2. \tag{11}$$

In (10), ν is the outwards unit normal vector on the boundary $\partial\Omega$.

It is not easy to find the minimizer for (6). We shall use ideas from [11]. As $\vec{n}$ is a unit vector, we use polar coordinate to represent it, i.e. $\vec{n}=(u,v) = (\cos\theta, \sin\theta)$. It is known that

$$|\nabla\vec{n}| = |\nabla\theta|, \tag{12}$$

see Appendix for the details of calculations. Thereupon, (6) becomes

$$\min_\theta \{\int_\Omega |\nabla\theta| \, dx + \frac{\lambda}{2}\int_\Omega |\vec{n} - \vec{n}_0|^2 \, dx\} \tag{13}$$

The optimality condition for θ for the above problem is:

$$\nabla \cdot \frac{\nabla\theta}{|\nabla\theta|} - \lambda(\vec{n} - \vec{n}_0) \cdot \frac{\partial\vec{n}}{\partial\theta} = 0. \tag{14}$$

Thereafter, we introduce an artificial time variable t and note that θ is the steady state of the following equation:

$$\theta_t = \nabla \cdot \frac{\nabla\theta}{|\nabla\theta|} - \lambda(\vec{n} - \vec{n}_0) \cdot \frac{\partial\vec{n}}{\partial\theta}. \tag{15}$$

One could solve θ directly from the above equation. One of the troubles is that θ can be multivalued. As an alternative, we note that (c.f. [11])

$$\theta = \tan^{-1}(\frac{v}{u}), \;\; \nabla\theta = \frac{u\nabla v - v\nabla u}{u^2+v^2}, \text{ and } \frac{\partial\vec{n}}{\partial\theta} = (-\sin\theta, \cos\theta) = (-v, u)$$

and with the notation $\vec{n}_0=(n_1, n_2)$ it follows from (15) that

$$\begin{aligned} \frac{uv_t - vu_t}{u^2+v^2} &= \text{div}\Big(\frac{u\nabla v - v\nabla u}{|u\nabla v - v\nabla u|}\Big) - \lambda[-(u-n_1)v + (v-n_2)u] \\ &= \text{div}\Big(\frac{u\nabla v - v\nabla u}{|u\nabla v - v\nabla u|}\Big) - \lambda[-uv + vn_1 + vu - un_2]. \end{aligned} \tag{16}$$

We know that $u^2+v^2=1$ which gives us $2uu_t + 2vv_t = 0$, so (16) can be separated into

$$v_t = u\text{div}\Big(\frac{u\nabla v - v\nabla u}{|u\nabla v - v\nabla u|}\Big) - \lambda u[vn_1 - un_2] \tag{17}$$

and

$$u_t = -v\mathrm{div}\Big(\frac{u\nabla v - v\nabla u}{|u\nabla v - v\nabla u|}\Big) + \lambda v[vn_1 - un_2]. \tag{18}$$

To find the values of d and μ satisfying (10) and (11), we also introduce an artificial time variable t and solve the following equation to steady state:

$$d_t = \mathrm{div}\Big(\frac{\nabla d}{|\nabla d|} - \vec{n}\Big) - \mu(d - d_0)\ . \tag{19}$$

The value of μ also needs to be determined in such a way that the above equation has a steady state and the condition (11) is fullfilled at the steady state. We use the noise level constraint (11) and (10) to obtain such a formula for μ. Using the same idea as in [1], we multiply (10) by $d - d_0$ and integrate over Ω to get

$$\int_\Omega \nabla \cdot \left(\frac{\nabla d}{|\nabla d|} - \vec{n}\right)(d - d_0)\, dx = \mu \int_\Omega (d - d_0)^2\, dx = \mu\sigma^2\ . \tag{20}$$

Using Green's Theorem, we get the following formula for μ

$$\mu = -\frac{1}{\sigma^2}\int_\Omega \left(\frac{\nabla d}{|\nabla d|} - \vec{n}\right)\cdot \nabla(d - d_0)\, dx. \tag{21}$$

In the numerical simulations, an explicit finite difference scheme is used to calculate d. The value of μ at each time level is updated according (21), see (30). In order to update μ using the above formula, we need to know the noise level approximately. In case that the noise level is not known, we can just fix a constant value for μ by trial and error.

In our simulations, we do not have a dynamic updating formula for λ and this parameter was chosen by trial and error. By inspecting (17) and (18) we see that $0 \ll \lambda$ is not a good choice since the dominating term then are

$$\begin{aligned} v_t &\approx -\lambda u[vn_1 - un_2], \quad u \neq 0, \\ u_t &\approx +\lambda v[vn_1 - un_2], \quad v \neq 0. \end{aligned} \tag{22}$$

Solve to steady state will force $vn_1 - un_2 = 0 \Rightarrow v = n_2$ and $u = n_1$. In this way u and v will be as noisy as their observations and no progress is made. On the other hand if $0 < \lambda \ll 1$

$$\begin{aligned} v_t &\approx +udiv\Big(\frac{u\nabla v - v\nabla u}{|u\nabla v - v\nabla u|}\Big), \\ u_t &\approx -vdiv\Big(\frac{u\nabla v - v\nabla u}{|u\nabla v - v\nabla u|}\Big) \end{aligned} \tag{23}$$

would be the dominating part. Both equations (23) reach steady state when $\nabla u \approx 0$ and $\nabla v \approx 0$, meaning that u and v are very smooth. Depending on the noise level we found that $\lambda \in (1e^{-2}, 1e^{-1})$ was a good choicc.

III. Implementation

We discretize the system (17) and (18) by finite difference and for short we introduce

$$[\mathrm{div}]^n := \left[\mathrm{div}\Big(\frac{u\nabla v - v\nabla u}{|u\nabla v - v\nabla u|}\Big)\right]^n, \text{ and } [vu]^n := [vn_1 - un_2]^n.$$

Details of how to discretize $[\mathrm{div}]^n$ in space will follow the same scheme as for $\mathrm{div}\big(\frac{\nabla d}{|\nabla d|}\big)$ described in (29). The following semi-implicit scheme of [11] is used to solve u and v:

$$u^{n+1} = u^n + \frac{\Delta t}{2}(v^{n+1} + v^n)\Big(- [\mathrm{div}]^n + \lambda[vu]^n\Big), \tag{24}$$

$$v^{n+1} = v^n + \frac{\Delta t}{2}(u^{n+1} + u^n)\Big(+ [\mathrm{div}]^n - \lambda[vu]^n\Big). \tag{25}$$

To solve the previous algebraic system v^{n+1} is used in (24) to get

$$\begin{aligned} u^{n+1} &= u^n + \frac{\Delta t}{2}\Big(v^n + \frac{\Delta t}{2}(u^{n+1} + u^n)\Big([\mathrm{div}]^n - \lambda[vu]^n\Big) + v^n\Big)\Big(- [\mathrm{div}]^n + \lambda[vu]^n\Big) \\ &= u^n + v^n\Delta t\Big(- [\mathrm{div}]^n + \lambda[vu]^n\Big) - \Big(\frac{\Delta t}{2}\Big)^2\Big([\mathrm{div}]^n - \lambda[vu]^n\Big)^2(u^{n+1} + u^n). \end{aligned} \tag{26}$$

The unknown u^{n+1} is collected on the right hand side and the following explicit formula is obtained for (18)

$$u_{i,j}^{n+1} = \frac{u_{i,j}^n - v_{i,j}^n\Delta t\Big([\mathrm{div}]_{i,j}^n - \lambda[vu]_{i,j}^n\Big) - u_{i,j}^n\Big(\frac{\Delta t}{2}\Big)^2\Big([\mathrm{div}]_{i,j}^n - \lambda[vu]_{i,j}^n\Big)^2}{1 + \Big(\frac{\Delta t}{2}\Big)^2\Big([\mathrm{div}]_{i,j}^n - \lambda[vu]_{i,j}^n\Big)^2}, \tag{27}$$

and (17) is approximated by the same technique

$$v_{i,j}^{n+1} = \frac{v_{i,j}^n + u_{i,j}^n\Delta t\Big([\mathrm{div}]_{i,j}^n - \lambda[vu]_{i,j}^n\Big) - v_{i,j}^n\Big(\frac{\Delta t}{2}\Big)^2\Big([\mathrm{div}]_{i,j}^n - \lambda[vu]_{i,j}^n\Big)^2}{1 + \Big(\frac{\Delta t}{2}\Big)^2\Big([\mathrm{div}]_{i,j}^n - \lambda[vu]_{i,j}^n\Big)^2}. \tag{28}$$

¿From (27) and (28), it is easy to calculate that

$$|u^{n+1}|^2 + |v^{n+1}|^2 = |u^n|^2 + |v^n|^2, \ \forall n.$$

As soon as steady state is reached for (27) and (28) we fix (u^{n+1}, v^{n+1}) to (u,v). Let us use the notation $\Delta^x_\mp d_{i,j} = \mp(d_{i\mp1,j} - d_{i,j})$ and $\Delta^y_\mp d_{i,j} = \mp(d_{i,j\mp1} - d_{i,j})$ for backward and forward difference at a given pixel (i,j). The numerical approximation to (19) and (21) is

$$\begin{aligned} d^{n+1}_{i,j} = d^n_{i,j} + \frac{\Delta t}{h}\Big[\Delta^x_-\Big(&\frac{\Delta^x_+ d^n_{i,j}}{[(\Delta^x_+ d^n_{i,j})^2 + (m(\Delta^y_+ d^n_{i,j}, \Delta^y_- d^n_{i,j}))^2]^{\frac{1}{2}}} - \frac{u_{i,j}}{[u^2_{i,j} + v^2_{i,j}]^{\frac{1}{2}}}\Big) \\ &+\Delta^y_-\Big(\frac{\Delta^y_+ d^n_{i,j}}{[(\Delta^y_+ d^n_{i,j})^2 + (m(\Delta^x_+ d^n_{i,j}, \Delta^x_- d^n_{i,j}))^2]^{\frac{1}{2}}} - \frac{v_{i,j}}{[u^2_{i,j} + v^2_{i,j}]^{\frac{1}{2}}}\Big)\Big] \\ &-\Delta t \mu^n (d^n_{i,j} - d^0_{i,j}) \; . \end{aligned} \tag{29}$$

Here we have used the notation

$$m(a,b) = \text{minmod}(a,b) = \Big(\frac{\text{sign } a + \text{sign } b}{2}\Big) \min(|a|, |b|)$$

and μ is defined discretely via

$$\begin{aligned} \mu^n = -\frac{h}{\sigma^2}\sum_{i,j}\Big(&\frac{\Delta^x_+ d^n_{i,j}}{[(\Delta^x_+ d^n_{i,j})^2 + (m(\Delta^y_+ d^n_{i,j}, \Delta^y_- d^n_{i,j}))^2]^{\frac{1}{2}}} - \frac{u_{i,j}}{[u^2_{i,j} + v^2_{i,j}]^{\frac{1}{2}}}\Big)\Delta^x_+(d^n_{i,j} - d^0_{i,j}) \\ &+ \Big(\frac{\Delta^y_+ d^n_{i,j}}{[(\Delta^y_+ d^n_{i,j})^2 + (m(\Delta^x_+ d^n_{i,j}, \Delta^x_- d^n_{i,j}))^2]^{\frac{1}{2}}} - \frac{v_{i,j}}{[u^2_{i,j} + v^2_{i,j}]^{\frac{1}{2}}}\Big)\Delta^y_+(d^n_{i,j} - d^0_{i,j}). \end{aligned} \tag{30}$$

The evaluation of $[\text{div}]^n$ is done similarly as shown in (29). The minmod$(\cdot,\cdot)$ action is applied to the terms $uv_x - vu_x$ and $uv_y - vu_y$.

If we hold $u = v = 0$ during the iterations between (29) and (30), then the updating of (29) and (30) is exactly the original TV-smoothing algorithm of [1].

To evaluate $\nabla d_0/|\nabla d_0|$ we need that $|\nabla d_0| \neq 0$. In order to overcome this problem in the simulations, we use the standard trick to replace $\nabla d_0/|\nabla d_0|$ by $\nabla d_0/\sqrt{|\nabla d_0|^2 + \epsilon}$ where ϵ is a small number. Normally, we choose ϵ to be very small (for example $\epsilon = 10^{-9}$) and the value of ϵ does not seem to influence the results much. The same trick is used for the calculation of $[\text{div}]^n$.

To summarize, our noise removal approach is done in the following two steps:

- Choose a positive λ and take $\nabla d_0/|\nabla d_0|$ to be the initial values for (u,v), solve (27) and (28) to steady state.
- Take d_0 as the initial value for d and let (u,v) be the value of the smoothed normals. Solve (29) and (30) to steady state.

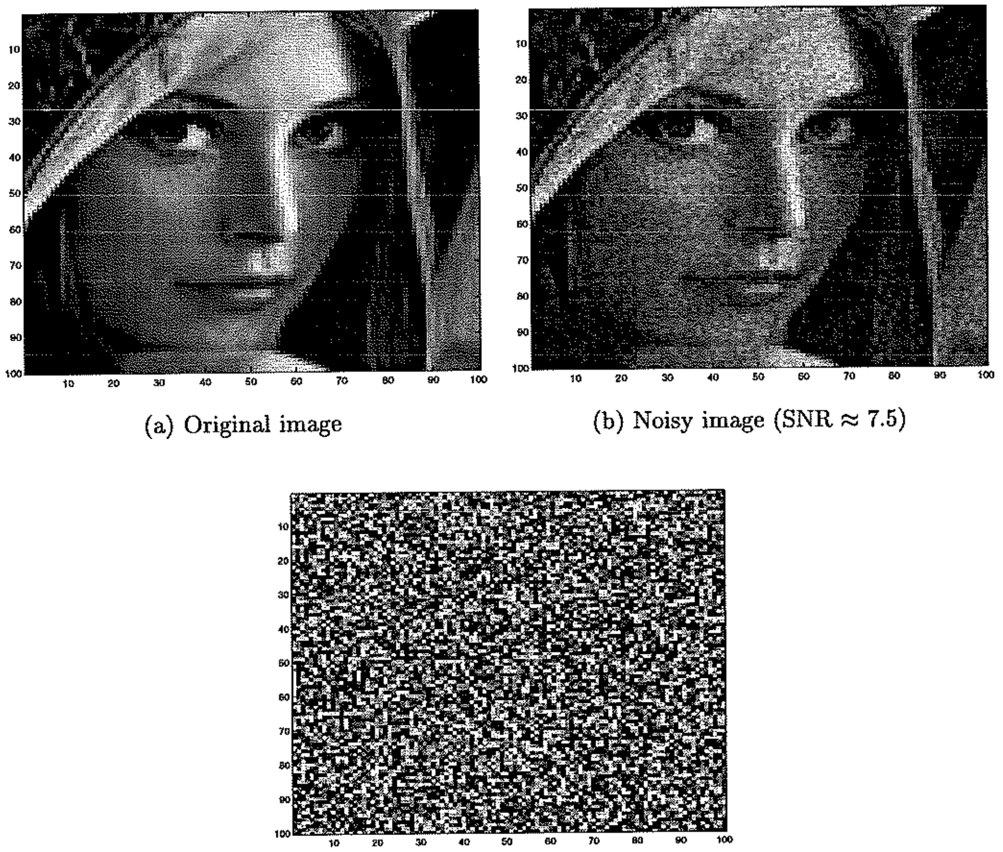
IV. Numerical results

In this part we present some of the results obtained with our system of coupled equations. First of all we wish to compare our result with two related methods [1], [3]. Both [1], [3] have their strengths and weakness and it will be shown that our new method does an overall better job than both of them. The classical TV model from [1] is known to give good results for almost all kinds of images. The chief criticism is that smooth regions are transformed into piecewise constant regions. On the other hand, the TV method works almost perfectly for block images. To avoid the staircase effect Lysaker-Lundervold-Tai [3] suggested a fourth order PDE. By construction, this approach allows linear changes in the intensity value and is therefore well suited for processing images with smooth transitions like a human face. From a theoretical point of view [3] will not fulfill the good property of the TV method on the edges. For the evaluations, we have used images like Fig.1, Fig.4, Fig.6 and some others to show the robustness, strength and weakness for the different algorithms concerned here.

Example 1: In this first test the well known Lena image is exposed with noise. ¿From the original and noisy image we calculate the original and noisy normal vectors. We use the noisy observations as input to our algorithm, and the smoothed normals are given in Fig.2(c). For better visualization, we have only plotted a small portion of the flow field for the images in Fig.2. In this specific test we fix $\lambda = 0.1$.

The processed unit vectors point in the wrong directions some places in Fig.2(c) but it is for sure a big improvement compared with Fig.2(b) where the normals more or less have a random orientation. Together with (19) and (21) we use the smoothed normals to restore a new image for the noisy one. The result is compared with the TV method (i.e. $u = 0$ and $v = 0$) and the fourth order smoother [3]. We also visualize the difference between the input and output image. In an ideal case the difference image should only contain noise as given in Fig.1(c).

¿From the restored images in Fig. 3 it is clear that much of the noise is suppressed. As expected, the TV algorithm transforms smooth regions into piecewise constant regions (see Lena's cheek). By evaluating the image of the difference between the smoothed image and the noisy image, it is obvious that neither the fourth order smoother or the TV method


(a) Original image

(b) Noisy image (SNR ≈ 7.5)

(c) Difference image

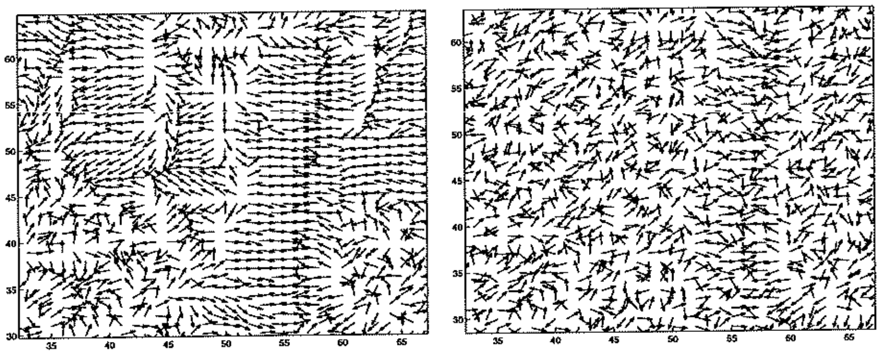
Fig. 1. Image used for evaluation.

are as good as our new method.

Example 2: A blocky image which should favor the TV method is used in this second test. Note that some of the object is as small as 2x2 pixels and other as narrow as 1x10 pixels. It is a nontrivial case to smooth out noise and simultaneously maintain all edges for an image like this.

Since the noise level is increased we fix $\lambda = 0.03$ during the normal processing. With this kind of synthesized images $\nabla d = 0$ almost everywhere in Ω. Due to this we do not show unit normal vectors but just evaluate the final result. The restored image obtained with our method is also compared with the two methods mentioned above.

We see that all methods are able to suppress much noise but some ghosts are visible in the image restored by the fourth order scheme (around the narrow object of 1x10 pixels).


(a) Original normals (b) Noisy normals

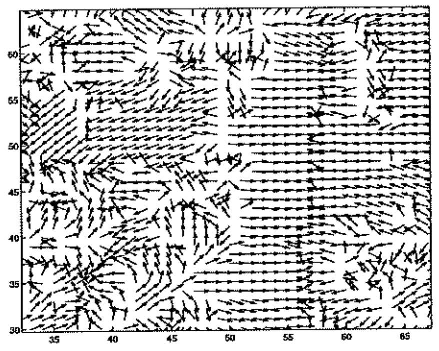
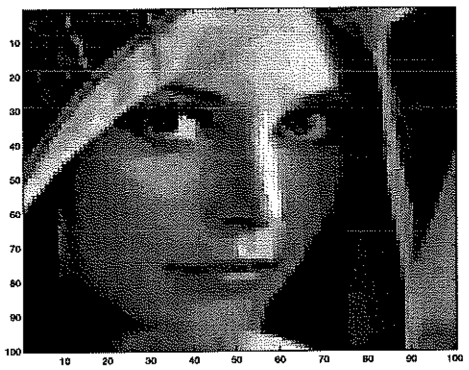

(c) Processed normals

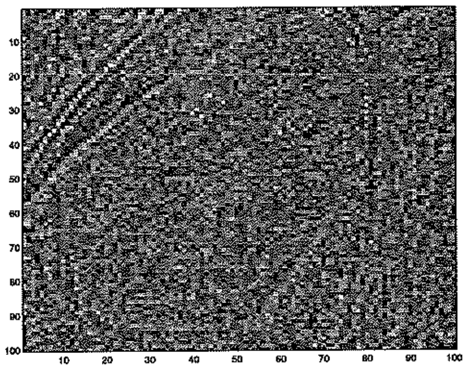
Fig. 2. Plot for the normal vectors. For better visualization, we have only plotted the normals for a small portion of the images.

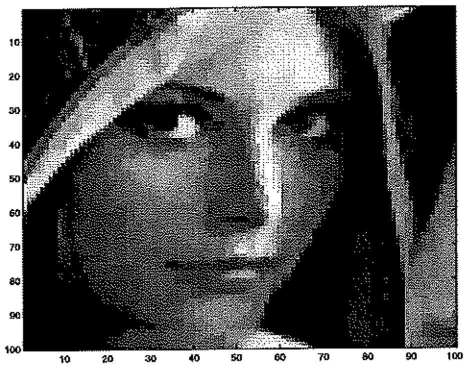
This is not surprising since the fourth order scheme does not allow discontinuous jumps in the same way as the other two methods do. From Fig.5(f) it is clear that the fourth order scheme filters out some edges, and in some sense the same is also observed in Fig.5(d).

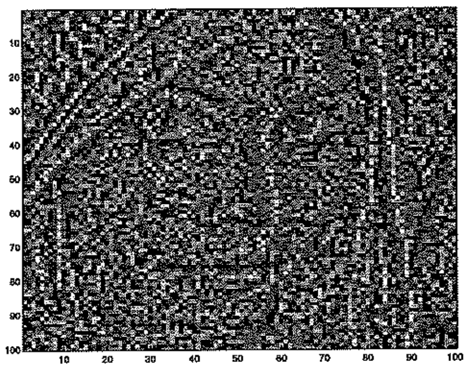
Example 3: Here we try our new method on an image composed of 4 different Brodatz textures as given in Fig.6(a).

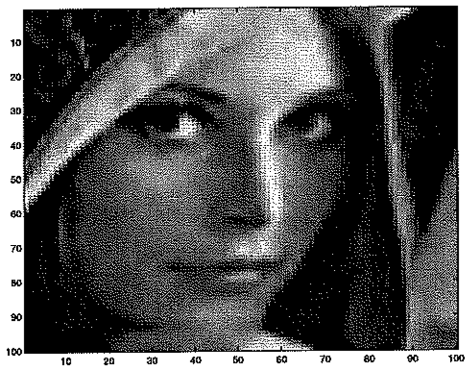
Processing texture images is generally a hard task since fine texture details often are filtered out. The texture in the lower left and upper right part of Fig.6(a) mostly contains high frequency information (i.e. very oscillating intensity values) and can easily be mixed up with noise. From Fig.7(b) we see that this is the part of the image with the poorest result, but the main quality is maintained. Both Fig.7(d) and Fig.7(f) reveal oversmoothing for some of the fine structures. Methods like [4] may be better suited to deal with texture images, but this example shows the robustness of our new algorithm.


(a) Our new method


(b) Difference image


(c) TV method


(d) Difference image


(e) Fourth order method

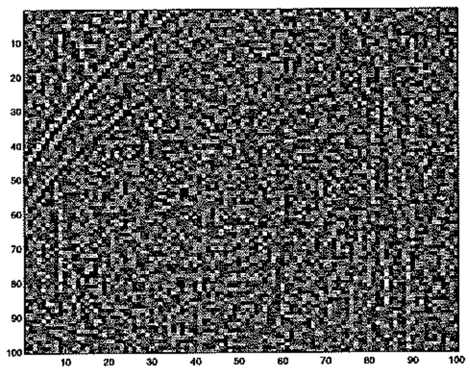
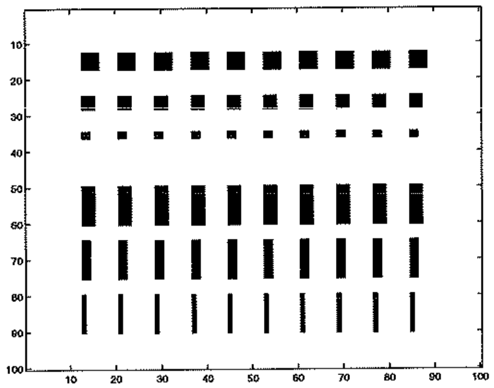
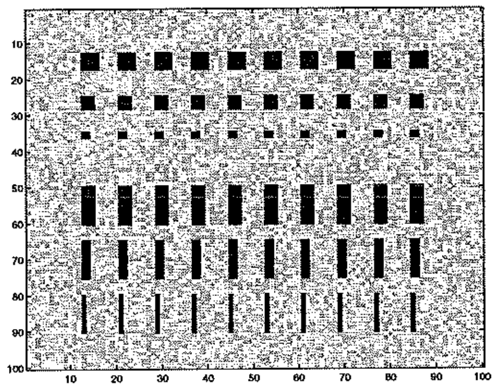

(f) Difference image

Fig. 3. Results for the Lena image.


(a) Original image


(b) Noisy image (SNR $\approx$ 8)

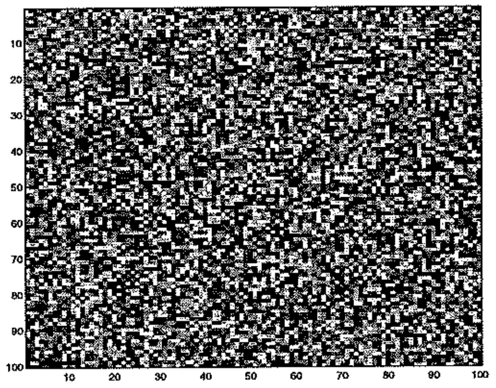

(c) Difference image

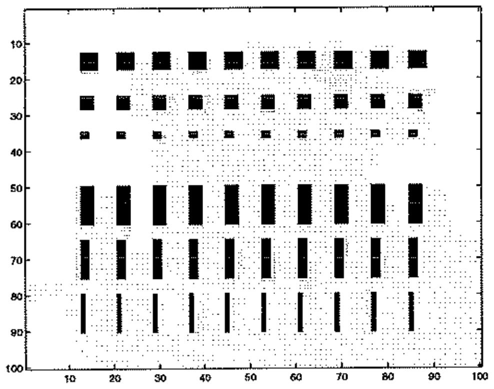
Fig. 4. Images used for evaluation.

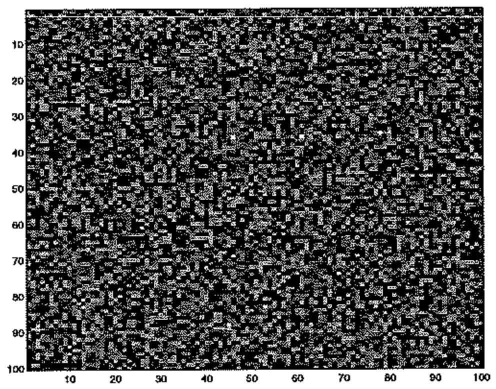
We will end this section by showing some more results from texture images, MR images, landscape images and a satellite image. The rest of this paper will only deal with results obtained with our new method.

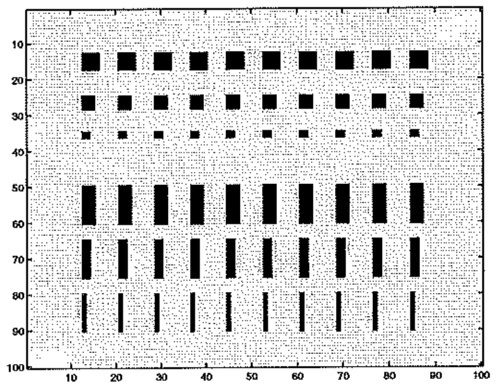
Our next example uses an image containing both a human face and some textures. The challenge with this image is to maintain both texture details and smooth transitions in the human face during processing.

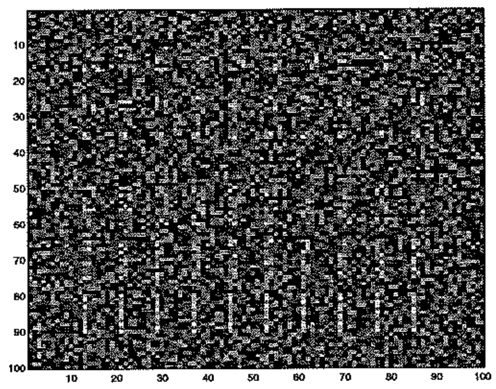
The background and human feature like a hand, shoulder and face is restored in a proper way, but the difference image tells us that textures on the scarf is smoothed to much, see Fig. 8.

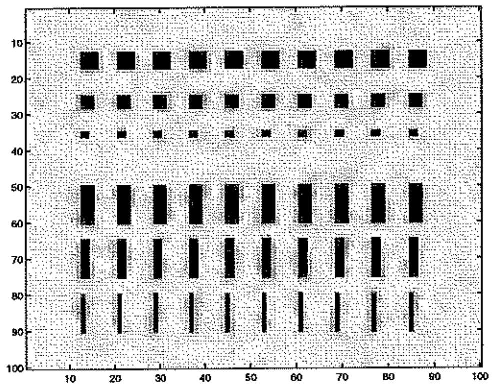
An image of leafs on a tree is evaluated next, see Fig. 9. Restoration algorithms may have troubles with this kind of images due to the lack of connection in the image. Observe


(a) Our new method


(b) Difference image


(c) TV method


(d) Difference image


(e) Fourth order method

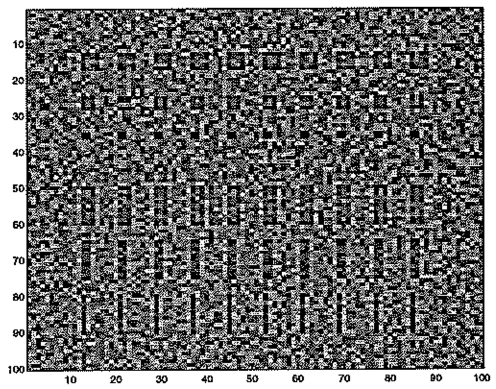
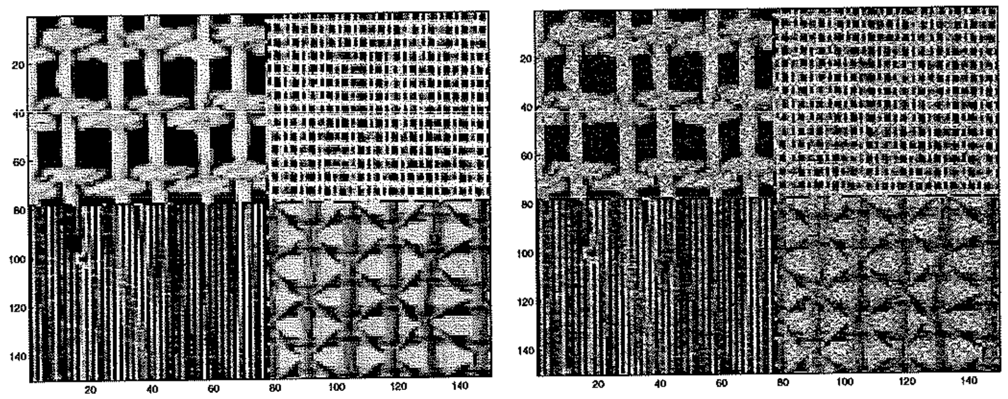

(f) Difference image

Fig. 5. Comparisons for a blocky image.



(a) Original image (b) Noisy image SNR$\approx$ 4.8

Fig. 6. The Brodatz image for evaluation.

that the leaves are not smeared together and this is an important quality. The only negative remark is that the wire (going left from the top of the pole) disappeared in some places. The same effect was observed for the TV method and the fourth order smoother. The result is not reported here but both of them reconstruct an image which is almost as good as the one given in Fig.9(c).

In our next example a MR image is exposed with noise, see Fig. 10. We use an image of a human brain, zoomed so it is possible to see all the fine features in the tissues. We see that the restoration algorithm is able to maintain all important information in the image, and in the same time filters out noise. This means that the intensity value is more equal inside each tissue region after processing.

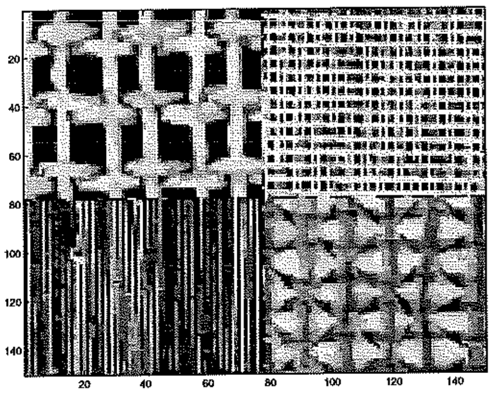
The final example concerns a satellite image. The recovered image coincides with the true one almost everywhere (c.f. Fig. 11).

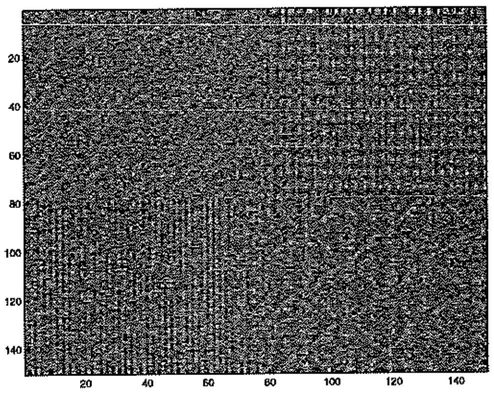
APPENDIX

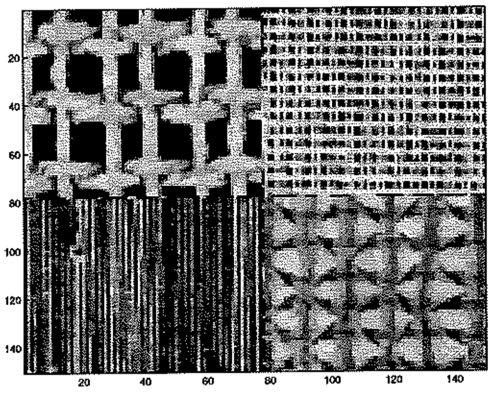
We give the details for the calculations for (12). Given $\theta = \theta(x_1, x_2)$.

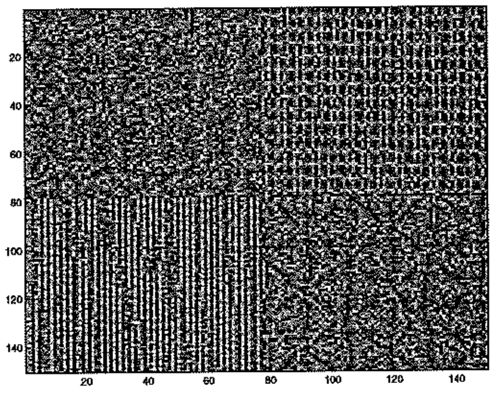
$$\nabla\theta = \Big(\frac{\partial\theta}{\partial x_1}, \frac{\partial\theta}{\partial x_2}\Big), \quad \text{and} \quad |\nabla\theta| = \sqrt{\Big(\frac{\partial\theta}{\partial x_1}\Big)^2 + \Big(\frac{\partial\theta}{\partial x_2}\Big)^2}. \tag{31}$$

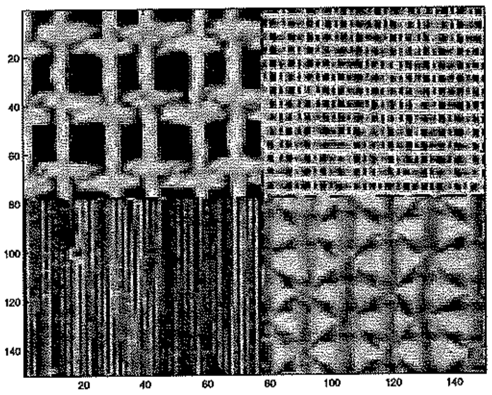
The gradient matrix and its norm of the vector-valued function $\mathbf{T}$ are respectively


(a) Our result


(b) Difference image


(c) TV result


(d) Difference image


(e) Fourth order result

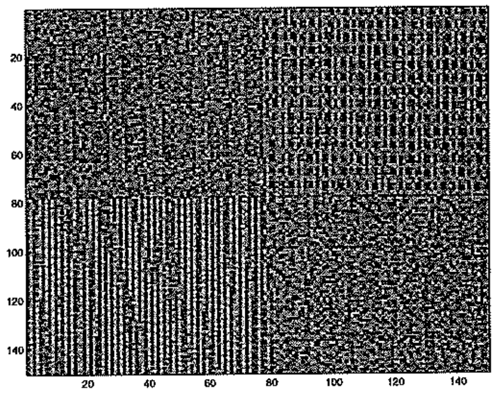

(f) Difference image

Fig. 7. Results for the Brodatz image.

defined by:

$$\nabla \mathbf{T} = \begin{pmatrix} \nabla T_1 \\ \nabla T_2 \end{pmatrix} = \begin{pmatrix} \frac{\partial T_1}{\partial x_1} & \frac{\partial T_1}{\partial x_2} \\ \frac{\partial T_2}{\partial x_1} & \frac{\partial T_2}{\partial x_2} \end{pmatrix}, \quad |\nabla \mathbf{T}| = \sqrt{\sum_{\mathrm{i}=1}^{2} \sum_{\mathrm{j}=1}^{2} \left(\frac{\partial \mathrm{T_i}}{\partial \mathrm{x_j}} \right)^2} \tag{32}$$

Let $\vec{n} = (\cos\theta, \sin\theta)$, it is easy to see that

$$\begin{aligned} |\nabla \vec{n}| &= \sqrt{\Big(-\sin\theta \frac{\partial \theta}{\partial x_1} \Big)^2 + \Big(-\sin\theta \frac{\partial \theta}{\partial x_2} \Big)^2 + \Big(\cos\theta \frac{\partial \theta}{\partial x_1} \Big)^2 + \Big(\cos\theta \frac{\partial \theta}{\partial x_2} \Big)^2} \\ &= \sqrt{\Big(\frac{\partial \theta}{\partial x_1} \Big)^2 (\sin^2\theta + \cos^2\theta) + \Big(\frac{\partial \theta}{\partial x_2} \Big)^2 (\sin^2\theta + \cos^2\theta)} \\ &= \sqrt{\Big(\frac{\partial \theta}{\partial x_1} \Big)^2 + \Big(\frac{\partial \theta}{\partial x_2} \Big)^2} = |\nabla \theta|. \end{aligned} \tag{33}$$

References

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

[2] A. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[3] A. Lundervold M. Lysaker and X. C. Tai, "Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images is space and time," *UCLA CAM Report*, Appllied Mathematics, no. 02-44, Available at http://www.math.ucla.edu/applied/cam/index.html, 2002.

[4] S. Osher, A. Sole, and L. Vese, "Image decomposition and restoration using total variation minimization and the H^{-1} norm," *UCLA CAM Report*, Applied Mathematics, no. 02-57, Available at http://www.math.ucla.edu/applied/cam/index.html, 2002.

[5] T. Chan, A. Marquina, and P. Mulet, "High-order total variation-based image restoration," *SIAM Journal on Scientific Computing*, vol. 22, no. 2, pp. 503–516 (electronic), 2000.

[6] Y.-L. You and M. Kaveh, "Fourth-order partial differential equation for noise removal," *IEEE Transactions on Image Processing*, vol. 9, no. 10, pp. 1723–1730, 2000.

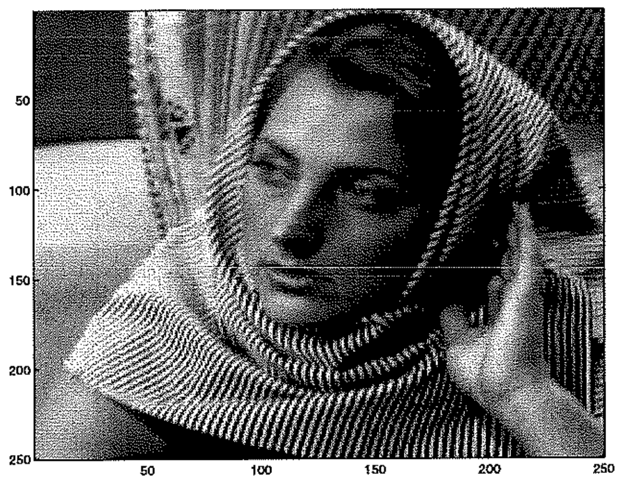
[7] C. Kenney and J. Lengan, "A new image processing primative: Reconstruction images from modified flow fields," *preprint*, pp. 1–10, 2002.

[8] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher, "Geometric surface processing via normal maps," *UCLA CAM Reoprt*, Applied Mathematics, no. 02-3, Available at http://www.math.ucla.edu/applied/cam/index.html, 2002.

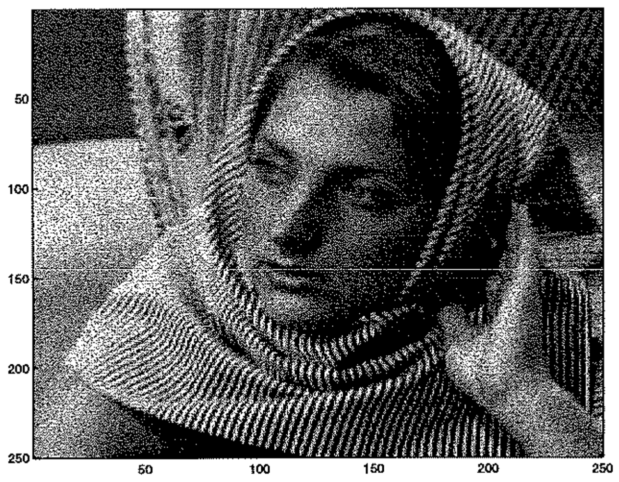
[9] Y Ohtake and A. G. Belyaev, "Dual-primal mesh optimization for polygonized implicit surfaces with sharpe features," *ACM Solid Modeling 2002*, Saarbrcken, Germany, June, 17-21, pp. 171–178, 2002.

[10] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Trans. Image Processing*, , no. 10, pp. 1200–1211, 2000.

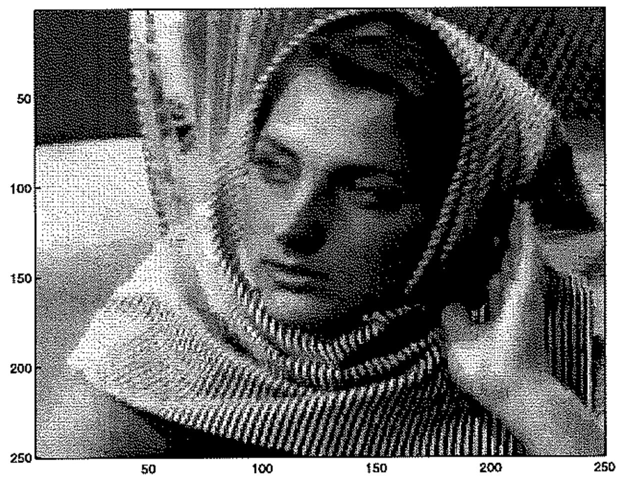
[11] L. A. Vese and S. Osher, "Numerical methods for p-harmonic flows and applications to image processing," *SINUM*, vol. 40, no.61, pp. 2085–2104, 2002.



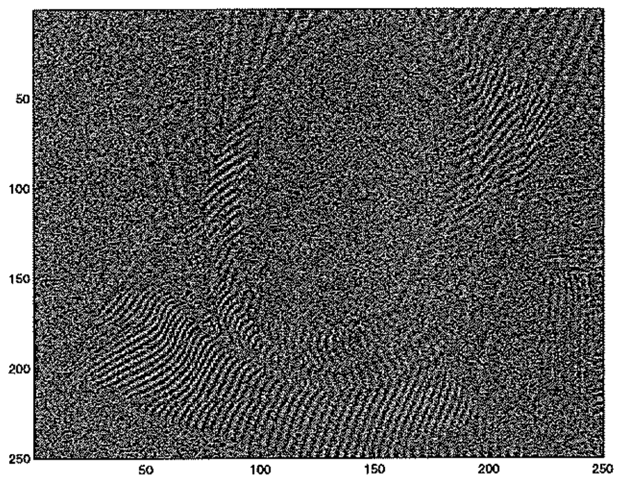
(a) Original image



(b) Noisy image SNR$\approx$ 11

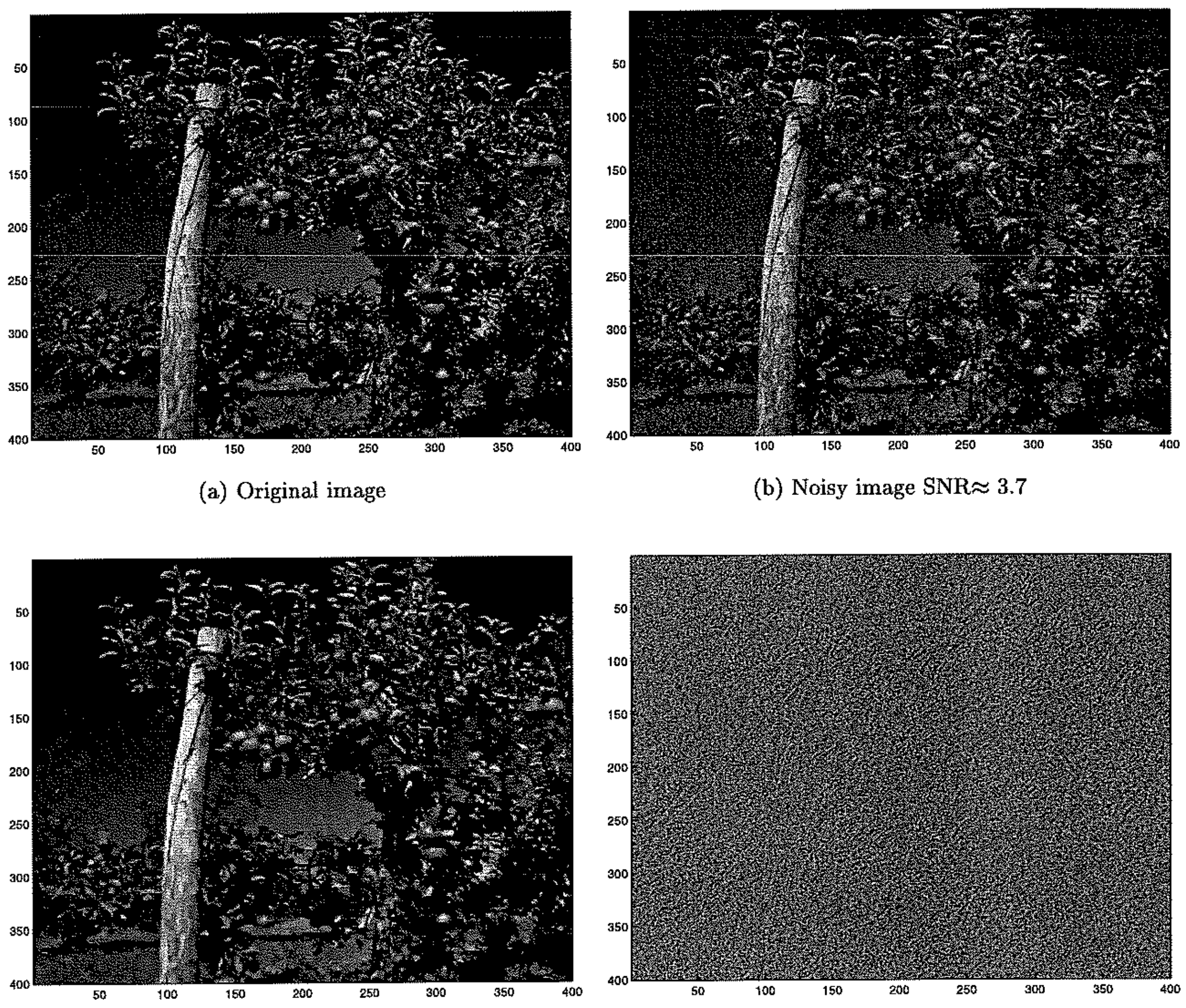


(c) Our result



(d) Difference image

Fig. 8. An image with a human face and some textures.



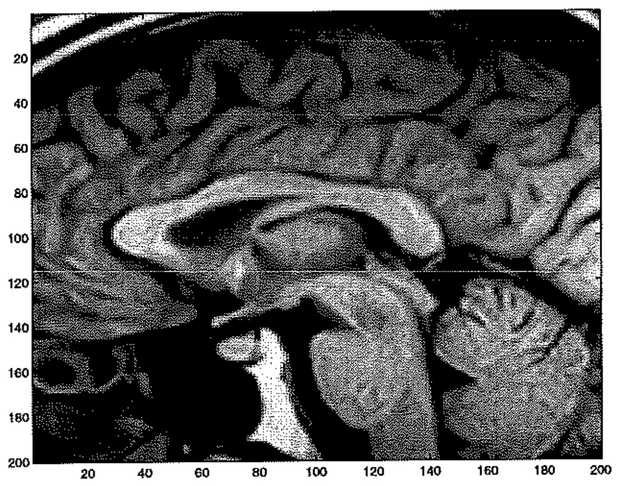
(a) Original image

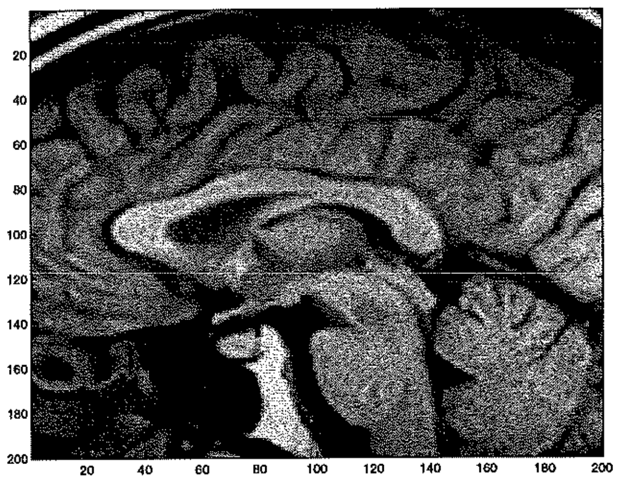
(b) Noisy image SNR$\approx$ 3.7

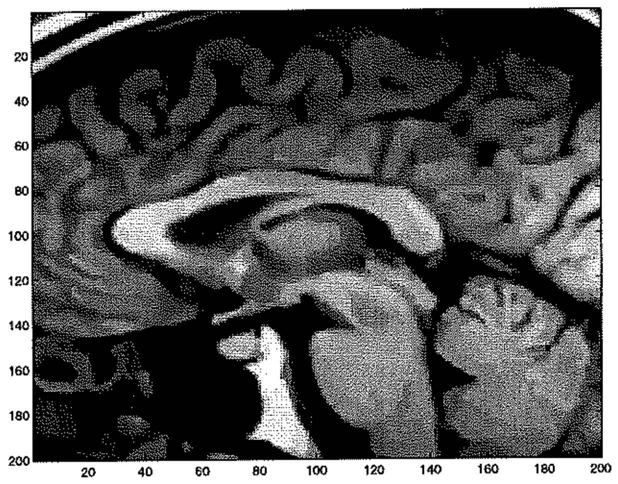
(c) Our result

(d) Difference image

Fig. 9. An image with a tree and leaves.


(a) Original image


(b) Noisy image SNR$\approx$ 10


(c) Our result

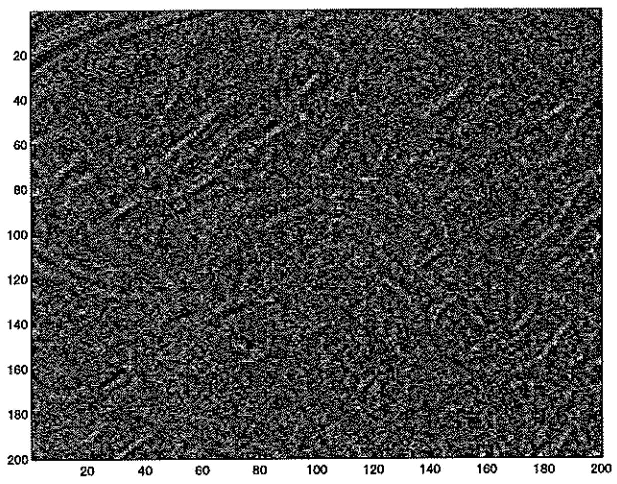
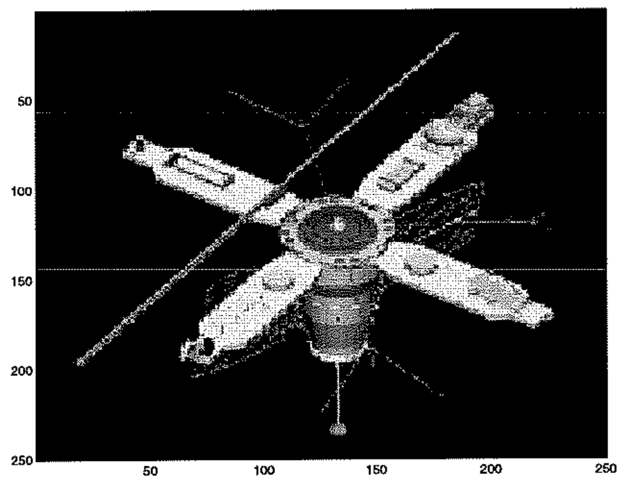
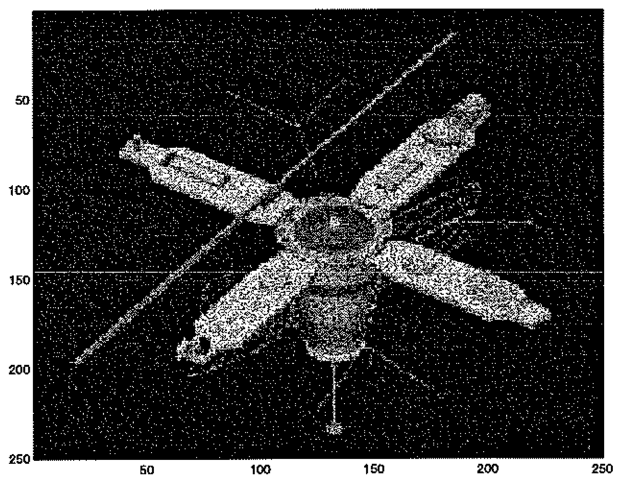

(d) Difference image

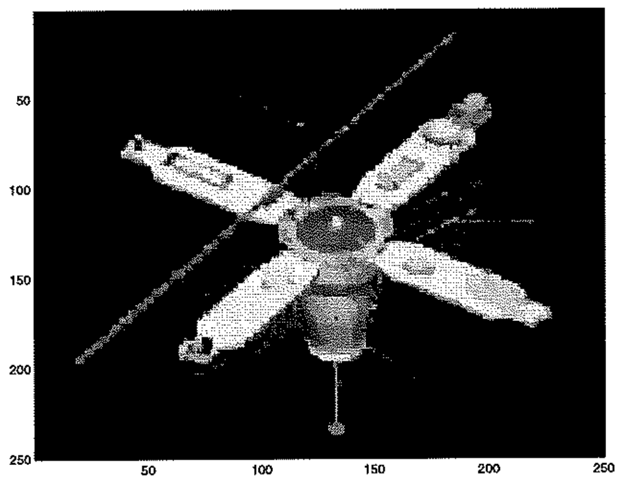
Fig. 10. A MR brain image.



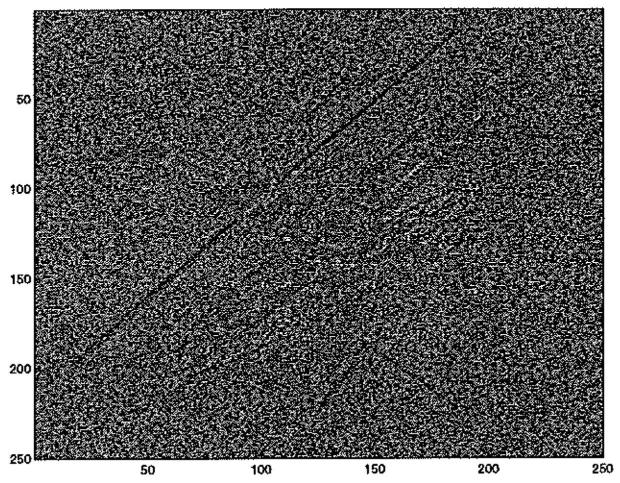
(a) Original image



(b) Noisy image SNR$\approx$ 1



(c) Our result



(d) Difference image

Fig. 11. Results for a satellite image.