A second order model for 3D-texture extraction

Maïtine Bergounioux, Minh Phuong Tran

A second order model for 3D-texture extraction

Maïtine Bergounioux and Minh Phuong Tran

Abstract In this paper we present the 3D-implementation of a second-order model for texture extraction that has been fully described in [3]. Numerical experimentation has been performed for 2D-images. We generalize the discrete model to the 3D case. In particular we describe the whole discretization process. In addition, we add an algorithmic modification that improves texture extraction using a modified Hessian matrix. We end with numerical examples arising in biomedical imaging

1 Introduction

In this paper we present the 3D- implementation of a second-order model for texture extraction that has been fully described in [3]. Numerical experimentation was performed for 2D-images. We generalize the discrete model to the 3D case. In particular we describe the complete discretization scheme. In addition, we add an algorithmic modification that improves texture extraction significantly using a modified Hessian matrix. This is also a generalization of the 2D-case (see Piffet [10, 11]). First, we recall the main definitions and present the generic second order variational model. Section 2 is devoted to the 3D-discretization and implementation. Then we present an "anisotropic" improvement of the algorithm which takes into account the (local) contours to compute the second-order derivative. We end with numerical examples arising in biomedical imaging, namely angiography MRI images.

Université d'Orléans, Laboratoire MAPMO - UMR 6628, Fédération Denis-Poisson, BP 6759, F-45067 Orléans Cedex 2, France e-mail: maitine.bergounioux@univ-orleans.fr

1.1 Bounded Variation Spaces of first and second order

Let $\Omega \subset \mathbb{R}^n$ $(n \geq 2)$ be an open bounded set. The space of functions of bounded variation, $BV(\Omega)$ is well known. We refer to [1, 2, 3] for example. We denote by $TV(u)$ the total variation of $u \in BV(\Omega)$:

$$TV(u) = \sup \left\{ \int_\Omega u \, \mathrm{div}\varphi \, \mathrm{d}x : \varphi \in \mathcal{C}_0^1(\Omega), \|\varphi\|_\infty \leq 1 \right\} \tag{1}$$

Following Demengel [7] and Piffet [10] we may define the space of functions of bounded second-order variation (or hessian bounded) as

$$BV^2(\Omega) := \{ u \in W^{1,1}(\Omega) \mid TV2(u) < +\infty \} .$$

Here the second-order total variation is defined as

$$TV2(u) := \sup \left\{ \int_\Omega \langle \nabla u, \mathrm{div}(\varphi) \rangle_{\mathbb{R}^n} \mid \varphi \in \mathcal{C}_c^2(\Omega, \mathbb{R}^{n\times n}), \, \|\varphi\|_\infty \leq 1 \right\} \tag{2}$$

where

$$\mathrm{div}(\varphi) = (\mathrm{div}(\varphi_1), \mathrm{div}(\varphi_2), \cdots, \mathrm{div}(\varphi_n)),$$

with

$$\forall i, \ \varphi_i = (\varphi_i^1, \varphi_i^2, \ldots, \varphi_i^n) \in \mathbb{R}^n \quad \text{and} \quad \mathrm{div}(\varphi_i) = \sum_{k=1}^{n} \frac{\partial \varphi_i^k}{\partial x_k}.$$

The space $BV^2(\Omega)$ endowed with the norm

$$\|u\|_{BV^2(\Omega)} = \|u\|_{W^{1,1}(\Omega)} + TV2(\Omega)$$

is a Banach space. Moreover, it has been proved in [10] that

$$BV^2(\Omega) = \left\{ u \in W^{1,1}(\Omega) \mid \forall i \in \{1, 2, \cdots, n\} : \frac{\partial u}{\partial x_i} \in BV(\Omega) \right\} .$$

1.2 The abstract second-order model

We recall the variational model described in [3]. We refer to this paper for a precise motivation of this second-order model. Let $\Omega \subset \mathbb{R}^n$ be an open bounded set (smooth enough, for example with Lipschitz boundary). We consider the following functional:

$$\begin{array}{c} F : BV^2(\Omega) \to \mathbb{R}^+ \\ (v) \mapsto F(v) \end{array}$$

$$F(v) = \frac{1}{2}\left\|u_d - v\right\|^2_{L^2(\Omega)} + \lambda TV2(v) + \delta \left\|v\right\|_{W^{1,1}(\Omega)}$$

where $u_d \in L^2(\Omega)$ and $\lambda, \delta \geq 0$ and we are looking for a solution to the optimization problem:

$$\inf_{v \in BV^2(\Omega)} F(v) \tag{3}$$

It has been proved in [3] that problem (3) has a unique solution for $\lambda > 0$ and $\delta > 0$. However, this result is still true for the discretized problem even with $\delta = 0$. Moreover, in [4] we prove that the existence result still holds true for the infinite dimensional problem if the function v satisfies $\dfrac{\partial v}{\partial n}_{|\partial\Omega} = 0$ and $\Omega = \prod_{i=1}^{n}]a_i, b_i[$ is a square subset of $\mathbb{R}^n$. In what follows, we investigate the finite-dimensional problem, so we assume that $\delta = 0$.

1.3 Discretization of the 3D - problem

In [3] the problem has been discretized in the case of 2D images and numerical tests have been performed. Here we generalize this work to the 3D-case and extend the anisotropic correction of the algorithm of [11] . In the sequel, $n = 3$ and the image size is $N_1 \times N_2 \times N_3$. The generic component of u is $u_{i,j,k}$ and we denote similarly the continuous function (previous section) and the corresponding (discretized) tensor.
We denote $X = \mathbb{R}^{N_1 \times N_2 \times N_3}$ endowed with inner product and norm

$$\langle u, v \rangle_X = \sum_{\substack{1 \leq i \leq N_1 \\ 1 \leq j \leq N_2 \\ 1 \leq k \leq N_3}} u_{i,j,k} v_{i,j,k} \text{ and } \left\|u\right\|_X = \sqrt{\sum_{\substack{1 \leq i \leq N_1 \\ 1 \leq j \leq N_2 \\ 1 \leq k \leq N_3}} u^2_{i,j,k}}$$

and set $Y = X \times X \times X$.

(a) We first compute the discrete gradient $\nabla u \in Y$ of the image $u \in X$:

$$(\nabla u_{i,j,k}) = (\nabla u^1_{i,j,k}, \nabla u^2_{i,j,k}, \nabla u^3_{i,j,k})$$

where

$$\nabla u^1_{i,j,k} = \begin{cases} u_{i+1,j,k} - u_{i,j,k} & i < N_1 \\ 0 & i = N_1 \end{cases}$$

$$\nabla u^2_{i,j,k} = \begin{cases} u_{i,j+1,k} - u_{i,j,k} & j < N_2 \\ 0 & j = N_2 \end{cases}$$

$$\nabla u^3_{i,j,k} = \begin{cases} u_{i,j,k+1} - u_{i,j,k} & k < N_3 \\ 0 & k = N_3 \end{cases}$$

(b) Discretization of the term $TV2(v)$.
We have

$$\langle \nabla u, \mathrm{div}\phi \rangle = - \left\langle \phi, \nabla^2 u \right\rangle \ .$$

Then,

$$TV2(v) \simeq \sum_{\substack{1\leq i\leq N_1\\ 1\leq j\leq N_2\\ 1\leq k\leq N_3}} \|(Hv)_{i,j,k}\|_{\mathbb{R}^9}$$

where

$$\begin{aligned}(Hv)_{i,j,k} =&(Hv^{11}_{i,j,k}, Hv^{12}_{i,j,k}, Hv^{13}_{i,j,k}, Hv^{21,}_{i,j,k}\\ &Hv^{22}_{i,j,k}, Hv^{23}_{i,j,k}, Hv^{31}_{i,j,k}, Hv^{32}_{i,j,k}, Hv^{33}_{i,j,k}).\end{aligned}$$

For every $i = 1, ..., N_1$, $j = 1, ..., N_2$ and $k = 1, ..., N_3$, the computation of Hv gives

$$(Hv)^{11}_{i,j,k} = \begin{cases} v_{i+1,j,k} - v_{i,j,k} + v_{i-1,j,k} & 1 < i < N_1 \\ v_{i+1,j,k} - v_{i,j,k} & i = 1 \\ v_{i,j,k} - v_{i-1,j,k} & i = N_1 \end{cases}$$

$$(Hv)^{12}_{i,j,k} = \begin{cases} v_{i,j+1,k} - v_{i,j,k} - v_{i-1,j+1,k} + v_{i-1,j,k} & 1 < i \leq N_1 \\ & 1 \leq j < N_2 \\ 0 & j = N_2 \\ 0 & i = 1 \end{cases}$$

$$(Hv)^{13}_{i,j,k} = \begin{cases} v_{i,j,k+1} - v_{i,j,k} - v_{i-1,j,k+1} + v_{i-1,j,k} & 1 < i \leq N_1 \\ & 1 \leq k < N_3 \\ 0 & i = 1 \\ 0 & k = N_3 \end{cases}$$

$$(Hv)^{21}_{i,j,k} = \begin{cases} v_{i+1,j,k} - v_{i,j,k} - v_{i+1,j-1,k} + v_{i,j-1,k} & 1 \leq i < N_1 \\ & 1 < k \leq N_3 \\ 0 & i = N_1 \\ 0 & k = 1 \end{cases}$$

$$(Hv)^{22}_{i,j,k} = \begin{cases} v_{i,j+1,k} - v_{i,j,k} + v_{i,j-1,k} & 1 < j < N_2 \\ v_{i,j+1,k} - v_{i,j,k} & j = 1 \\ v_{i,j,k} - v_{i,j-1,k} & j = N_2 \end{cases}$$

$$(Hv)^{23}_{i,j,k} = \begin{cases} v_{i,j,k+1} - v_{i,j,k} - v_{i,j-1,k+1} + v_{i,j-1,k} & 1 < j \leq N \\ & 1 \leq k < N_3 \\ 0 & j = 1 \\ 0 & k = N_3 \end{cases}$$

$$(Hv)^{31}_{i,j,k} = \begin{cases} v_{i+1,j,k} - v_{i,j,k} - v_{i+1,j,k-1} + v_{i,j,k-1} & 1 < k \leq N_3 \\ & 1 \leq i < N_1 \\ 0 & k = 1 \\ 0 & i = N_1 \end{cases}$$

$$(Hv)^{32}_{i,j,k} = \begin{cases} v_{i,j+1,k} - v_{i,j,k} - v_{i+,j+1,k-1} + v_{i,j,k-1} & 1 \leq j < N \\ & 1 < k \leq N_3 \\ 0 & j = N_2 \\ 0 & k = 1 \end{cases}$$

$$(Hv)^{33}_{i,j,k} = \begin{cases} v_{i,j,k+1} - v_{i,j,k} + v_{i,j,k-1} & 1 < k < N_3 \\ v_{i,j,k+1} - v_{i,j,k} & k = 1 \\ v_{i,j,k} - v_{i,j,k-1} & k = N_3 \end{cases}$$

1.4 Numerical computation of the solution of (3)

Let us consider $H^* : X^9 \to X$ defined as follows (H^* is the adjoint of operator H): for every $p = (p^{11}, p^{12}, p^{13}, p^{21}, p^{22}, p^{23}, p^{31}, p^{32}, p^{33}) \in X^9$,

$$(H^*p)_{i,j,k} = \sigma^{11}_{i,j,k} + \sigma^{12}_{i,j,k} + \sigma^{13}_{i,j,k} + \sigma^{21}_{i,j,k} + \sigma^{22}_{i,j,k} + \sigma^{23}_{i,j,k} + \sigma^{31}_{i,j,k} + \sigma^{32}_{i,j,k} + \sigma^{33}_{i,j,k}$$

where

$$\sigma^{11}_{i,j,k} = \begin{cases} p^{11}_{i+1,j,k} - 2p^{11}_{i,j,k} + p^{11}_{i-1,j,k} & 1 < i < N_1 \\ p^{11}_{i+1,j,k} - p^{11}_{i,j,k} & i = 1 \\ p^{11}_{i-1,j,k} - p^{11}_{i,j,k} & i = N_1 \end{cases}$$

$$\sigma^{22}_{i,j,k} = \begin{cases} p^{22}_{i,j+1,k} - 2p^{22}_{i,j,k} + p^{22}_{i,j-1,k} & 1 < j < N_2 \\ p^{22}_{i,j+1,k} - p^{22}_{i,j,k} & j = 1 \\ p^{22}_{i,j-1,k} - p^{22}_{i,j,k} & j = N_2 \end{cases}$$

$$\sigma^{33}_{i,j,k} = \begin{cases} p^{33}_{i,j,k+1} - 2p^{33}_{i,j,k} + p^{33}_{i,j,k-1} & 1 < k < N_3 \\ p^{33}_{i,j,k+1} - p^{33}_{i,j,k} & k = 1 \\ p^{33}_{i,j,k-1} - p^{33}_{i,j,k} & k = N_3 \end{cases}$$

$$\sigma^{12}_{i,j,k} = \begin{cases} p^{12}_{i+1,j,k} & i=1, j=1 \\ -p^{12}_{i+1,j-1,k} & i=1, j=N_2 \\ p^{12}_{i+1,j,k} - p^{12}_{i+1,j-1,k} & i=1, 1<j<N_2 \\ -p^{12}_{i,j,k} & i=N_1, j=1 \\ p^{12}_{i,j-1,k} & i=N_1, j=N_2 \\ p^{12}_{i,j-1,k} - p^{12}_{i,j,k} & i=N_1, 1<j<N_2 \\ p^{12}_{i+1,j,k} - p^{12}_{i,j,k} & 1<i<N_1, j=1 \\ p^{12}_{i,j-1,k} - p^{12}_{i+1,j-1,k} & 1<i<N_1, j=N_2 \\ p^{12}_{i,j-1,k} - p^{12}_{i,j,k} - p^{12}_{i+1,j-1,k} + p^{12}_{i+1,j,k} & 1<i<N_1, 1<j<N_2 \end{cases}$$

$$\sigma^{13}_{i,j,k} = \begin{cases} p^{13}_{i+1,j,k} & i=1, k=1 \\ -p^{13}_{i+1,j,k-1} & i=1, k=N_3 \\ p^{13}_{i+1,j,k} - p^{13}_{i+1,j,k-1} & i=1, 1<j<N_3 \\ -p^{13}_{i,j,k} & i=N_1, k=1 \\ p^{13}_{i,j,k-1} & i=N_1, k=N_3 \\ p^{13}_{i,j,k-1} - p^{13}_{i,j,k} & i=N_1, 1<k<N_3 \\ p^{13}_{i+1,j,k} - p^{13}_{i,j,k} & 1<i<N_1, k=1 \\ p^{13}_{i,j,k-1} - p^{13}_{i+1,j,k-1} & 1<i<N_1, k=N_3 \\ p^{13}_{i,j,k-1} - p^{13}_{i,j,k} - p^{13}_{i+1,j,k-1} + p^{13}_{i+1,j,k} & 1<i<N_1, 1<k<N_3 \end{cases}$$

$$\sigma^{21}_{i,j,k} = \begin{cases} p^{21}_{i,j+1,k} & j=1, i=1 \\ -p^{21}_{i-1,j+1,k} & j=1, i=N_1 \\ p^{21}_{i,j+1,k} - p^{21}_{i-1,j+1,k} & j=1, 1<i<N_1 \\ -p^{21}_{i,j,k} & j=N_2, i=1 \\ p^{21}_{i-1,j,k} & j=N_2, i=N_1 \\ p^{21}_{i-1,j,k} - p^{21}_{i,j,k} & j=N_2, 1<i<N_1 \\ p^{21}_{i,j+1,k} - p^{21}_{i,j,k} & 1<j<N_2, i=1 \\ p^{21}_{i-1,j,k} - p^{21}_{i-1,j+1,k} & 1<j<N_2, i=N_1 \\ p^{21}_{i-1,j,k} - p^{21}_{i,j,k} - p^{21}_{i-1,j+1,k} + p^{21}_{i,j+1,k} & 1<j<N_2, 1<i<N_1 \end{cases}$$

$$\sigma^{23}_{i,j,k} = \begin{cases} p^{23}_{i,j+1,k} & j=1,k=1 \\ -p^{23}_{i,j+1,k-1} & j=1,k=N_3 \\ p^{23}_{i,j+1,k} - p^{23}_{i,j+1,k-1} & j=1,1<k<N_3 \\ -p^{23}_{i,j,k} & j=N_2,k=1 \\ p^{23}_{i,j,k-1} & j=N_2,k=N_3 \\ p^{23}_{i,j,k-1} - p^{23}_{i,j,k} & j=N_2,1<k<N_3 \\ p^{23}_{i,j+1,k} - p^{23}_{i,j,k} & 1<j<N_2,k=1 \\ p^{23}_{i,j,k-1} - p^{23}_{i,j+1,k-1} & 1<j<N_2,k=N_3 \\ p^{23}_{i,j,k-1} - p^{23}_{i,j,k} - p^{23}_{i,j+1,k-1} + p^{23}_{i,j+1,k} & 1<j<N_2,1<k<N_3 \end{cases}$$

$$\sigma^{31}_{i,j,k} = \begin{cases} p^{31}_{i,j,k+1} & k=1,i=1 \\ -p^{31}_{i-1,j,k+1} & k=1,i=N_1 \\ p^{31}_{i,j,k+1} - p^{31}_{i-1,j,k+1} & k=1,1<i<N_1 \\ -p^{31}_{i,j,k} & k=N_3,i=1 \\ p^{31}_{i-1,j,k} & k=N_3,i=N_1 \\ p^{31}_{i-1,j,k} - p^{31}_{i,j,k} & k=N_3,1<i<N_1 \\ p^{31}_{i,j,k+1} - p^{31}_{i,j,k} & 1<k<N_3,i=1 \\ p^{31}_{i-1,j,k} - p^{31}_{i-1,j,k+1} & 1<k<N_3,i=N_1 \\ p^{31}_{i-1,j,k} - p^{31}_{i,j,k} - p^{31}_{i-1,j,k+1} + p^{31}_{i,j,k+1} & 1<k<N_3,1<i<N_1 \end{cases}$$

$$\sigma^{32}_{i,j,k} = \begin{cases} p^{32}_{i,j,k+1} & k=1,1=1 \\ -p^{32}_{i,j-1,k+1} & k=1,j=N_2 \\ p^{32}_{i,j,k+1} - p^{32}_{i,j-1,k+1} & k=1,1<j<N_2 \\ -p^{32}_{i,j,k} & k=N_3,j=1 \\ p^{32}_{i,j-1,k} & k=N_3,j=N_2 \\ p^{32}_{i,j-1,k} - p^{32}_{i,j,k} & k=N_3,1<j<N_2 \\ p^{32}_{i,j,k+1} - p^{32}_{i,j,k} & 1<k<N_3,j=1 \\ p^{32}_{i,j-1,k} - p^{32}_{i,j-1,k+1} & 1<k<N_3,j=N_2 \\ p^{32}_{i,j-1,k} - p^{32}_{i,j,k} - p^{32}_{i,j-1,k+1} + p^{32}_{i,j,k+1} & 1<k<N_3,1<j<N_2 \end{cases}$$

It is straightforward to prove that

Theorem 1. *The solution to problem (3) verifies:*

$$v = ud - P_{\lambda K}(u_d)$$

where $P_{\lambda K}$ *is the orthogonal projector operator on* λK *and*

$$K := \{H^* p \mid p \in X^9,\ \|p_{i,j,k}\|_{\mathbb{R}^9} \leq 1,\ 1 \leq i \leq N_1, 1 \leq j \leq N, 1 \leq k \leq N_3\}.$$

Proof. It is quite similar to the 2D-case proof. We refer to [3].

To compute $P_{\lambda K}(u_d)$ we have to solve the following problem:

$$\begin{cases} \min \|\lambda H^* p - u_d\|_X^2 \\ p \in X^9 \\ \|p_{i,j,k}\|_{\mathbb{R}^9}^2 \leq 1,\ 1 \leq i \leq N_1, 1 \leq j \leq N_2, 1 \leq k \leq N_3 \end{cases}$$

Following [6] and [3] we use the following algorithm to compute $P_{\lambda K}(u_d)$

Algorithm

Choose $\tau > 0$

1. Let $p^0 = 0, n = 0$.
2. Suppose p^n is known, we compute p^{n+1} as follows:

$$p^n_{i,j,k} = p^{n+1}_{i,j,k} + \tau \left[(H \left[H^* p - \frac{u_d}{\lambda} \right])_{i,j,k} + \left\| (H \left[H^* p^n - \frac{u_d}{\lambda} \right])_{i,j,k} \right\|_{R^9} p^{n+1}_{i,j,k} \right]$$

which implies:

$$p^{n+1}_{i,j,k} = \frac{p^n_{i,j,k} - \tau (H \left[H^* p^n - \frac{u_d}{\lambda} \right])_{i,j,k}}{1 + \tau \left\| (H \left[H^* p^n - \frac{u_d}{\lambda} \right])_{i,j,k} \right\|_{\mathbb{R}^9}}$$

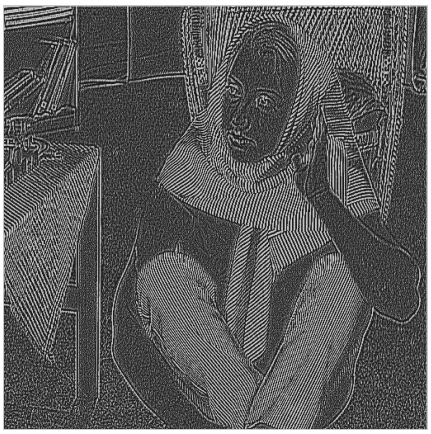
Theorem 2. *Let* $\tau \leq 1/8^3$*, then* $\lambda(H^* p^n)_n$ *converges to* $P_{\lambda K_2}(u_d)$ *as* $n \to \infty$.

Proof. Once again the proof is quite technical but similar to the 2D-case proof ([3]).

2 Introducing anisotropy

L. Piffet [10, 8, 11] has observed (in the 2D-case) that cancelling one or more coefficients of the Hessian matrix permits to get rid of the contours along the corresponding direction.

(a) Original image (Barbara)



(b) Texture part without anisotropic strategy

(c) Texture part without horizontal and vertical contours

Fig. 1 Effects of anisotropic improvement strategy

We give a 2D-example in Figure 1 : here the coefficients $(Hv)^{1,1}$ and $(Hv)^{2,2} = 0$ have been globally set to 0. We can see that horizontal and vertical contours are not involved in the texture part any longer. This method has been improved since there were two major inconveniences :

- First, the same transform is performed at every pixel, so that the image is globally treated. All the vertical and horizontal are removed;
- Second, the transform is depended on the chosen (fixed) cartesian axis and it is not possible to remove contours that are not horizontal, vertical or diagonal.

Therefore, the Hessian matrix is now locally computed at every pixel. First , a rotation is performed so that the gradient direction is the new y-axis (or x-axis). The corresponding Hessian matrix is computed and suitable coefficients are canceled. Then the inverse rotation is performed. For more details on can refer to [10, 11].

We compute the (local) 3D- Hessian matrix at a voxel (i, j, k) using this technique. We have to perform two rotations r_α and r_β to compute an modified hessian matrix H'. More precisely, we perform a change of variables (with the rotations) to compute the Hessian matrix and the adjoint matrix as in the previous section: the local axis (with the gradient vector as z-axis) are considered instead of the original fixed cartesian axis. Then, we may cancel the Hessian matrix terms corresponding to the gradient direction (for example), to get rid of the corresponding contour (if it is significant) in the extracted texture. Finally we go back to the original axis with the inverse rotations. Let us detail the process :

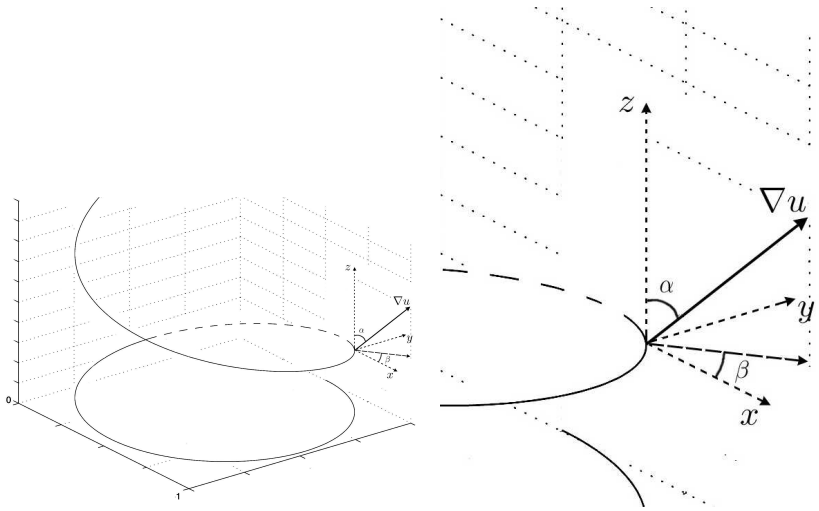


Fig. 2 Definition of local axis and angles α and β

The angles α and β are defined at point $X_o = (x_o, y_o, z_o)$ as follows : α is the (azimuthal) angle between the gradient $\nabla u(x_o, y_o, z_o)$ and the z-axis . β is the angle between the orthogonal projection of

$$\nabla u(x_o, y_o, z_o) := \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} (x_o, y_o, z_o)$$

(on the xOy plane) and the x -axis. Note that we can perform this transformation with axis Ox or Oy instead of Oz . Let us define the two rotations :

r_α and r_β which matrices are :

$$R_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \text{ and } R_\beta = \begin{pmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix} ,$$

with

$$\alpha = \text{atan} \left(\frac{u_z}{\sqrt{u_x^2 + u_y^2}} \right)(X_o), \quad \beta = \text{atan} \left(\frac{u_y}{u_x} \right)(X_o) .$$

The change of variables from the fixed basis to the local one is given par

$$\tilde{X} = R_\beta R_\alpha X, \quad \text{with } X = (x, y, z) \in \mathbb{R}^3 .$$

Moreover

$$X = (R_\beta R_\alpha)^{-1} \tilde{X} = R_\alpha^{-1} R_\beta^{-1} \tilde{X} = R_{-\alpha} R_{-\beta} \tilde{X} .$$

In the sequel, we set $\tilde{u}(\tilde{X}) := u(X)$ and $R_{\alpha,\beta} \stackrel{def}{:=} R_{-\alpha} R_{-\beta}$ and we compute the first and second order derivative of $\tilde{u}$:

$$\nabla \tilde{u} = \begin{pmatrix} \frac{\partial \tilde{u}}{\partial \tilde{x}} \\ \frac{\partial \tilde{u}}{\partial \tilde{y}} \\ \frac{\partial \tilde{u}}{\partial \tilde{z}} \end{pmatrix} \text{ and } \tilde{H} := \begin{pmatrix} \frac{\partial^2 \tilde{u}}{\partial \tilde{x}^2} & \frac{\partial^2 \tilde{u}}{\partial \tilde{x} \partial \tilde{y}} & \frac{\partial^2 \tilde{u}}{\partial \tilde{x} \partial \tilde{z}} \\ \frac{\partial^2 \tilde{u}}{\partial \tilde{x} \partial \tilde{y}} & \frac{\partial^2 \tilde{u}}{\partial \tilde{y}^2} & \frac{\partial^2 \tilde{u}}{\partial \tilde{y} \partial \tilde{z}} \\ \frac{\partial^2 \tilde{u}}{\partial \tilde{x} \partial \tilde{z}} & \frac{\partial^2 \tilde{u}}{\partial \tilde{y} \partial \tilde{z}} & \frac{\partial^2 \tilde{u}}{\partial \tilde{z}^2} \end{pmatrix} .$$

A short computation gives

$$\frac{\partial \tilde{u}}{\partial \tilde{x}} = \frac{\partial u}{\partial x}\frac{\partial \tilde{x}}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial \tilde{y}}{\partial x} + \frac{\partial u}{\partial z}\frac{\partial \tilde{z}}{\partial x} = \nabla u \cdot \frac{\partial \tilde{X}}{\partial x} = \nabla u \cdot R(:, 1) ,$$

where $\cdot$ denotes the $\mathbb{R}^3$ scalar product and $R(:, 1)$ is the first column of R. Finally, we get

$$\nabla \tilde{u} = R_{\alpha,\beta} \nabla u . \tag{4}$$

Now we compute $\tilde{H}$; we set $\tilde{v} = \dfrac{\partial \tilde{u}}{\partial \tilde{x}}$ and estimate $\nabla \tilde{v}$ as above : this will be the first column of $\tilde{H}$.

$$\nabla \tilde{v} = R_{\alpha,\beta} \nabla v = R_{\alpha,\beta} \begin{pmatrix} \frac{\partial^2 u}{\partial x^2} \\ \frac{\partial^2 u}{\partial y \partial x} \\ \frac{\partial^2 u}{\partial z \partial x} \end{pmatrix} .$$

Finally

$$\tilde{H} = R_{\alpha,\beta} H \ . \tag{5}$$

As already mentioned, the idea is to cancel some terms of the Hessian matrix to get rid of (or to keep) the contours. However, without performing the rotations, there would be only few possible directions, for example vertical, horizontal and diagonal in the 2D-case so that many contours are not considered. Performing the change of variables allows to identify the gradient direction (that is the contour direction if the gradient is large enough) with the z-axis and then cancel corresponding terms of the matrix $\tilde{H}$. Of course, we have to get back to the original situation. Let us denote by $\mathcal{L}$ the (linear) transformation that assigns 0 to some coefficients of $\tilde{H}$ (this is a projection). The whole process is described by

$$H \to \tilde{H} = R_{-\alpha} R_{-\beta} H \to \mathcal{L}(\tilde{H}) := \tilde{H}' \to [R_{\alpha,\beta}]^{-1} \mathcal{L}(\tilde{H}) = R_\beta R_\alpha \mathcal{L}(\tilde{H}) \ ,$$

that is

$$H \to \left[R_\beta R_\alpha \mathcal{L} R_{-\alpha} R_{-\beta} \right] H \ . \tag{6}$$

So, algorithm p.8 is modified as follows

Algorithm

Choose $\tau > 0, \mu > 0$ and compute ∇u. Use a threshold process to identify the contours $\left(\|\nabla u\| \geq \mu\right)$. Set I_μ the set of voxels corresponding to these "significant contours".

1. Let $p^0 = 0, n = 0$.
   For voxels in I_μ, modify H with the following rule

$$H \to \tilde{H} = R_{-\alpha} R_{-\beta} H \to \mathcal{L}(\tilde{H}) = [\mathcal{L} R_{-\alpha} R_{-\beta}] H := H'$$

   and compute $(H')^*$
2. Same as before p.8 with H' instead of H.

3 Numerical examples

Numerical experimentation has been done in the context of biomedical imaging. We consider a stack of 50 MRI images of the vessel network of brain mice.[1] The challenge is to identify the network to get structural informations. Using 2D segmentation and interpolation methods is not possible, since the slices are not exploitable (see Figure 3.)

[1] We thank J.C. Belœil, S. Même and F. Szeremeta, from CBM Laboratory in Orléans, for the use of these images, `http://cbm.cnrs-orleans.fr/`.

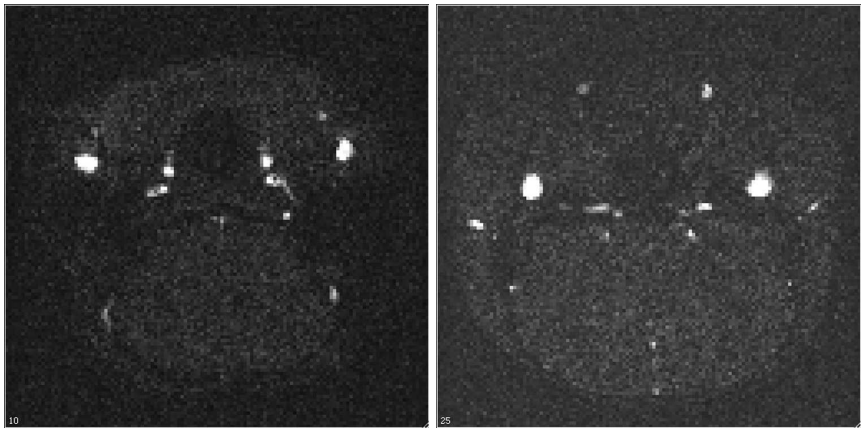
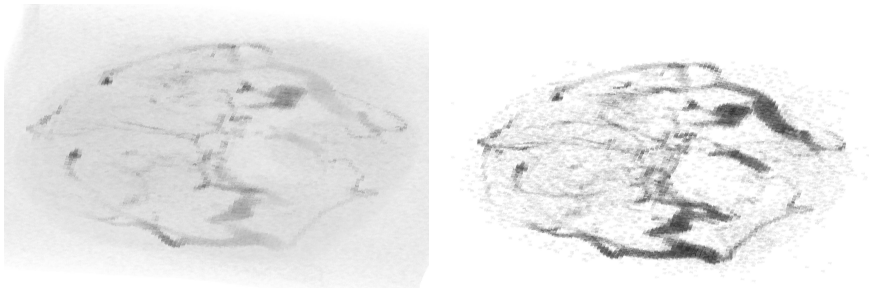


Fig. 3 2D slices example (slices 10 an 25)

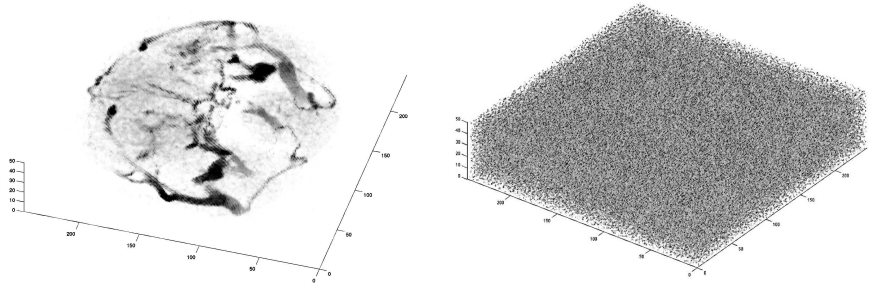
Therefore we have to deal with the complete 3D information. We consider that noise and very small vessels effect is texture. Extracting texture gives the remainder part, the so-called "cartoon" (smooth part). We expect that the contours are kept in the cartoon part which in the cleaned image in some sense. Then classical segmentation methods (as threshold for example) can be used. The following results have been obtained without any anisotropic strategy. Indeed, computational time is large and we still have to improve the speed of algorithm. However, we present a comparison between the two methods with and without anisotropy strategy. The results show that the anisotropy technique is quite efficient and we have good hope to keep the whole contour information contour in the cartoon part.

We have tested many values for λ and the maximum number of iterations. We present some results to show the influence of λ (images have been contrasted). We shall speed up the method in the future using (for example) Nesterov algorithms as in [12] .



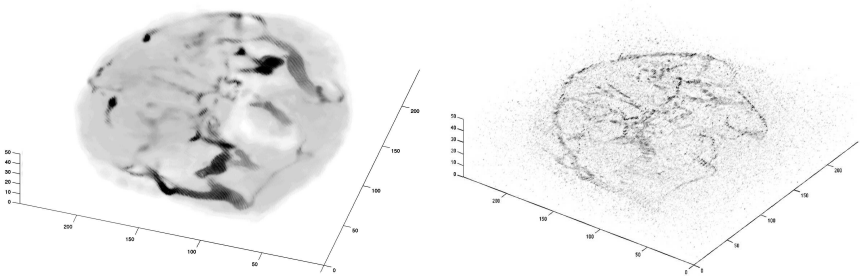
(a) Original Image (b) Image with treshold at grey value = 210

Fig. 4 3D angiography image



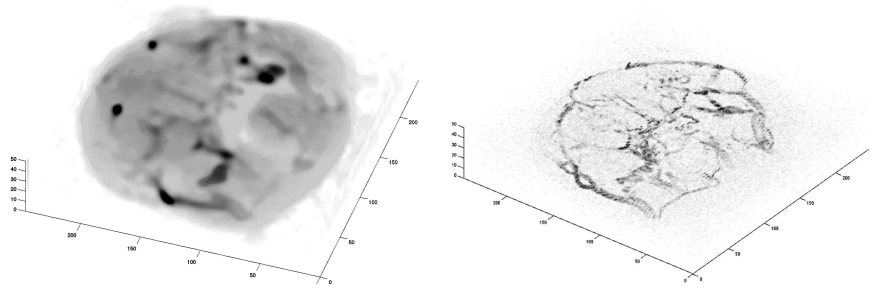
(a) Cartoon (b) Texture

Fig. 5 No anistropy strategy : $\lambda = 1$ and 5 000 iterations - The choice of small λ allows to denoise the image quite efficiently : here the texture is the noise and the cartoon the denoised image



(a) Cartoon (b) Texture

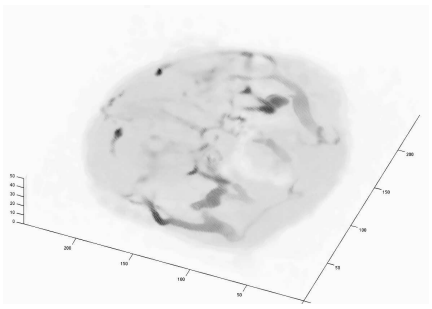
Fig. 6 No anistropy strategy :$\lambda = 10$ and 5 000 iterations



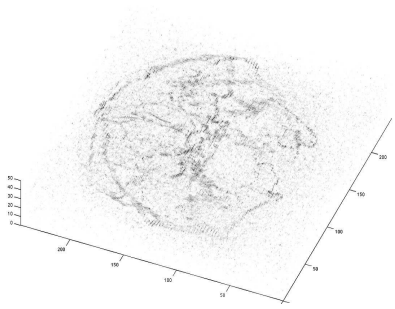
(a) Cartoon (b) Texture

Fig. 7 No anistropy strategy : $\lambda = 50$ and 10 000 iterations -The contours and the vessel network are recovered in the texture.

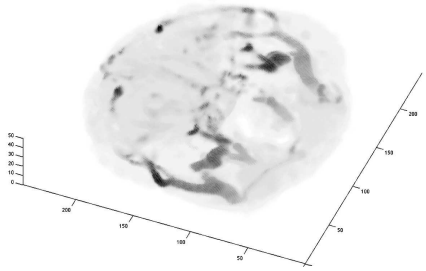
We have tested the algorithm with and without anisotropy strategy. We give below results for $\lambda = 10$ and 5000 iterations. As th 3D cartoon and texture pictures are not easy to compare we give pictures of the difference as well.



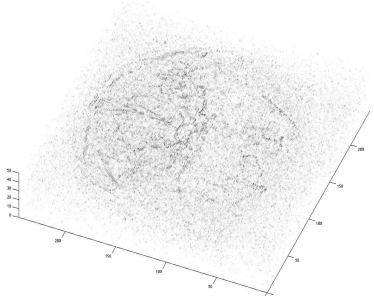
(a) Cartoon without anisotropy strategy



(b) Texture without anisotropy strategy : contours are involved in the texture

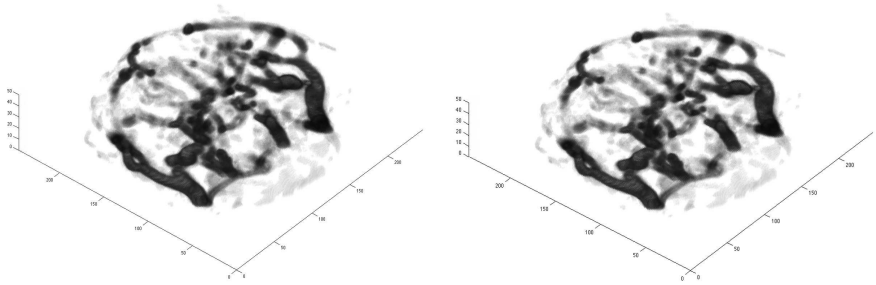


(c) Cartoon with anisotropy strategy



(d) Texture with anisotropy strategy : contours are not involved in the texture any longer

Fig. 8 Comparison between the two strategies for $\lambda = 10$ and 5 000 iterations



(a) Difference betwwen cartoons (b) Difference betwwen textures

Fig. 9 Absolute value of the difference for $\lambda = 10$ and 5 000 iterations : it is precisely the vessel-network which is alternatively included in the cartoon (when no anisotropy strategy is performed) or in the texture in the other case

References

1. Ambrosio L., Fusco N. , Pallara D. (2000) Functions of bounded variation and free discontinuity problems, Oxford mathematical monographs, Oxford University Press
2. Attouch H., Buttazzo G., Michaille G. (2006) Variational analysis in Sobolev and BV spaces : applications to PDEs and optimization, MPS-SIAM series on optimization
3. Bergounioux M., Piffet L. (2011) A second-order model for image denoising , Set-Valued Analysis and Variational Analysis, to appear, DOI : 10.1007/s11228-010-0156-6 `http://hal.archives-ouvertes.fr/hal-00440872/fr/`
4. Bergounioux M. (2010) On Poincaré-Wirtinger inequalities in spaces of functions of bounded variation, `http://hal.archives-ouvertes.fr/hal-00515451/fr/`
5. Bredies K., Kunisch K. , Pock T. (2009) Total Generalized Variation, preprint
6. Chambolle, A. (2004) An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Vision, **20**, 89–97
7. Demengel F. (1984) Fonctions à hessien borné, Annales de l'institut Fourier, **34**, no 2, 155–190
8. Echegut, R., Piffet, L. (2010) A variational model for image texture identification, Recent Advances in Optimization and its Applications in Engineering, Diehl, M.; Glineur, F.; Jarlebring, E.; Michiels, W. (Eds.), Springer
9. L.C. Evans, R. Gariepy, Measure theory and fine properties of functions, CRC Press, 1992
10. L. Piffet, Modèles variationnels du second ordre pour l'extraction de textures 2D, PhD Thesis, Orléans, 2010
11. L. Piffet, XXX, this book, 2011.
12. Weiss P., Blanc-Fraud L., Aubert, G. (2009) Efficient schemes for total variation minimization under constraints in image processing. SIAM journal on Scientific Computing, **31**, no 3, 2047–2080.