An unbiased implementation of regularization mechanisms

Thierry Viéville

An unbiased implementation of regularization mechanisms

T. Viéville,**INRIA**, Sophia BP93 06902 Sophia France+33 4 92 38 76 88

Keywords: Regularization methods, Diffusion operator, Unbiased implementation.

1 Introduction

Perceptual processes, in computer or biological vision, require the computation of "maps" of quantitative values. The image itself is a "retinotopic map": for each pixel of the image there is a value corresponding to the image intensity at this location. This is a vectorial value for color images. A step further, in early-vision, the retinal image contrast is computed at each location, allowing to detect image edges related to boundaries between image areas. Such maps encode not only the contrast magnitude, but several other cues: contrast orientation related to edge orientation, shape curvature, binocular disparity related to the visual depth, color cues, temporal disparity between two consecutive images in relation with visual motion detection, etc.. There are such detectors in both artificial and biological visual systems (see e.g. [12] for a general introduction and e.g. [17] for an overview about biological vision). Such maps are not only parametrized by retinotopic locations, but also using 3D locations, or other parameters.

The Partial Differential Equation (PDE) approach

In computer vision, a relevant and efficient theory is available regarding the definition and computation of such maps of quantitative values (see e.g. [1] for a didactic introduction about the "axiomatization" of this part of computer vision). This formalism not only provides a clear basic of "what is to be done" (i.e. requirements in order to have coherent and

consistent definitions) but also of "how to do it" since the theory is effective in the sense that efficient implementations may be derived (see e.g. [2] for a recent treatise on this subject). The "what is to be done" level is usually formalized in terms of a criterion to minimize and the "how to do it" level is related to the, so called, Euler-Lagrange equations which allow to improve an initial guess of the solution and get closer to the optimal solution. At the technical level, as revisited in this paper, these quantities are efficiently computed using implementations of partial differential equations which define regularization processes allowing to obtain well-defined estimations of these quantities. This powerful methodology is also very general in the sense that a large variety of computational problems are solvable within this framework (see e.g. [9,8] for a review).

A step further, PDE calculations regularize their input, i.e. provide a well-defined and stable estimation even in the case of noisy or partially defined data. Thanks to this property, algorithms have performances closer to those of biological systems than other methods. The biological plausibility of such methods is out of the scope of this paper but is addressed elsewhere [30,4].

Implementing PDE on sampled data

This nice formalism has a real drawback: it is not obvious to implement because very specific numerical scheme must be chosen and approximations must be made, which biased the original method (see e.g. [2]). This is both a theoretical limitation (it is not clear if theoretical results derived in the continuous case still apply to the discrete approximation) and a practical one (some hacks have to be introduced while the method was originally supposed to get rid of such, not easy to learn, tricks).

As it will made explicit in the sequel, the difficult point is the anisotropic diffusion operator implementation. Concretely, such mechanisms must integrate the information in a bounded neighborhood so that the cooperation between these local iterative computations allow a global computation of the quantitative map. In other words, we must provide a "particle implementation" of such numerical computations. Such a method, used in fluid dynamics computation has been introduced by Leonard [19] followed by Raviat and Mas-Gallic [23] and developed by Degond and Mas-Gallic [7].

It is based on an integral approximation of the diffusion operator used in the regularization mechanisms.

The goal of this paper is to describe how computer vision partial differential equations can be implemented using such mechanisms.

What is the paper about

Following this track, in the next section, we are going to revisit the computer vision partial differential equation methodology in the case of vectorial maps.

We then are going to re-derive an improved version of the Degond and Mas-Gallic method, using bounded neighborhoods and considering an optimal implementation of such an integral operator.

We will further develop the fact that when used on sampled data such as image pixels or 3D data voxels, it provides an unbiased discrete implementation of such an operator.

We finally will illustrate the present development with an experiment of image denoising and another experiment of visual motion estimation.

Notations We write *vectors* and *matrices* in bold letters, matrices being written with capital letters and scalars in italic. The dual of a quantity $\mathbf{x}$ is represented as its transpose $\mathbf{x}^T$, the dot-product between vectors being written $\mathbf{x}^T\mathbf{y}$. We represent the components of a vector using superscripts, e.g.: $\mathbf{x} = (x^0, x^1, x^2)^T$.

For vector of integer indices $\alpha = (\alpha_1 \cdots \alpha_n) \in N^n$ we write:

$$|\alpha| = \alpha_1 + \cdots + \alpha_n \text{ and } \alpha! = \alpha_1! \cdots \alpha_n!$$

so that we can write concisely:

$$\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \text{ and } \partial_{\mathbf{x}}^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}}$$

while the Taylor expansion of a function $h : \mathbb{R}^n \to \mathbb{R}$ becomes:

$$h(\mathbf{x}) = h(\mathbf{x}_0) + \sum_{|\alpha|=1}^{r} \frac{\partial^\alpha h}{\alpha!} (\mathbf{x} - \mathbf{x}_0)^\alpha + R^r h$$

where the remainder $R^\epsilon h$ may be written using an integral form:

$$R^r h = \sum_{|\alpha|=r+1} \frac{r+1}{\alpha!} \int_{[0,1]} (\mathbf{x} - \mathbf{x}_0)^\alpha (1-u)^r \, \partial^\alpha h(\mathbf{x}_0 + u(\mathbf{x} - \mathbf{x}_0)) \, du$$

We also use the Sobolev function space $W^{s,\infty}(\mathbb{R}^n)$ provided with the norm:

$$||f||_{s,\infty} = \sup_{0\leq k\leq s}\left[\sup\ \mathrm{ess}_{|\alpha|=k,\,\mathbf{x}\in\mathbb{R}^n}|\partial^\alpha f(\mathbf{x})|\right]$$

where $\sup\ \mathrm{ess}_{\mathbf{x}} u(\mathbf{x})$ is the smallest constant, if any, for which $u(\mathbf{x})$ is bounded except in a negligible set of measure zero. Generally speaking, the set of admissible functions F used here is a dense linear subset of a Hilbert space H, the scalar product of which is denoted by $(\cdot,\cdot)_H$. This has to be mentioned here but in the sequel this aspect is only a formal point.

2 Revisiting anisotropic diffusion operators.

Let us briefly revisit the computer vision partial differential equation methodology in the case of Euclidean vector maps (how it can be generalized to non-Euclidean computation maps and how a general class of nonlinear diffusion operators are implemented within the present frameworks is discussed elsewhere [30]). The goal of this section is to introduce this approach and point out to where the key problem.

Here, as required in some situations (e.g. [1,8,14]), we consider functions whose values are not scalars but vectors.

2.0.0.1 Defining a computational map function from a criterion. Let us consider a vector map:

$$\mathbf{h} : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}^m.$$

from a bounded open set Ω of $\mathbb{R}^n$ into $\mathbb{R}^m$. $\mathbf{h}$ belongs to a set F of admissible functions. Here, $\bar{\mathbf{h}}(\mathbf{x})$ is a "reference" function related to an input or measure $\mathbf{m}(\mathbf{x})$, as shown in Fig. 1.

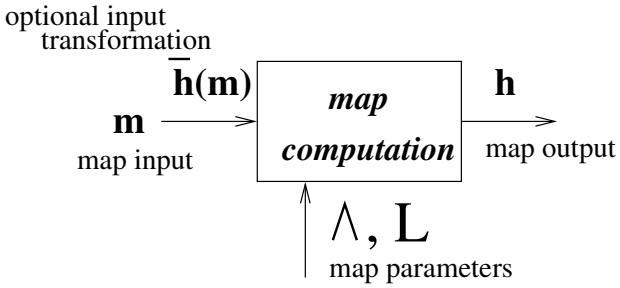


Fig. 1. Input/output scheme of a regularization process, see text for details.

In order to define this map, we consider the following optimization problem (following e.g. in [8]):

$$\mathbf{h}^* = arg\ min_{\mathbf{h}} L(\mathbf{h}) \quad \text{with} \quad L(\mathbf{h}) = \frac{1}{2} \int \underbrace{||\mathbf{h} - \bar{\mathbf{h}}||^2_{\mathbf{\Lambda}}}_{\text{input}} + \underbrace{\varphi_{\mathbf{L}}(D\mathbf{h})}_{\text{regularization}} \qquad (1)$$

where $D(\mathbf{h})$ is the first order derivative of the function $\mathbf{h}$, i.e. its $m \times n$ Jacobian matrix, $\varphi_{\mathbf{L}}$ is a positive definite quadratic form which can be written in terms of the gradients the components of $\mathbf{h}$ as

$$\varphi_{\mathbf{L}}(D\mathbf{h}) = D\mathbf{h}\, \mathbf{L}\, D\mathbf{h}^T = \sum_{i,j} (Dh^i)^T \mathbf{L}_{ij} Dh^j,$$

and

$$||\mathbf{h} - \bar{\mathbf{h}}||^2_{\mathbf{\Lambda}} = (\mathbf{h} - \bar{\mathbf{h}})^T \mathbf{\Lambda} (\mathbf{h} - \bar{\mathbf{h}}).$$

Here $\mathbf{\Lambda}$ and $\mathbf{L}$ are not constants but depend on the map location $\mathbf{x}$. They also may "slowly" evolve with time (i.e. be adapted after the optimum has been reached to attain a better optimum).

In words, the *specification* of this map of values corresponds to an "objective" or a "criterion" to attain. This criterion is built from two terms:
(i) the input term, related to the data input

(i.e. looking for a solution as compatible as possible with the input) and

(ii) the regularization term, related to the a priory information

(i.e. looking for a solution with plausible properties: here which variation is minimal).

Regarding the term related to the input :

- The function $\mathbf{\Lambda} : \mathbb{R}^n \rightarrow S_m^+ \in W^{s,\infty}(\mathbb{R}^n)$, where S_m^+ is the set of square symmetric positive semi-definite matrices of size m, defines a so called, *measurement information metric*.

  This metric allows to represent:
  (i) the *precision of the input function*: the higher this precision in a given direction, the higher the value of $\mathbf{\Lambda}$ in this direction (in a statistical framework, $\mathbf{\Lambda}$ corresponds to the inverse of a covariance matrix)

(ii) *partial observations*: if the input function $\bar{\mathbf{h}}(\mathbf{x})$ is only defined in some directions, it corresponds to a matrix $\mathbf{\Lambda}$ definite only in these directions (for instance, if $\bar{\mathbf{h}}(\mathbf{x})$ is only defined in the direction $\mathbf{u}$ at a given location we write $\mathbf{\Lambda} = k\,\mathbf{u}\,\mathbf{u}^T$ for some k),
(iii) *missing data*: if the input function $\bar{\mathbf{h}}(\mathbf{x})$ is not defined for some $\mathbf{x}$, we simply have to state $\mathbf{\Lambda} = 0$ at this location; more generally it represents:
(iv) *linear relations between some measures* $\mathbf{m}$ *and the parameter estimation* $\mathbf{h}$, say $\mathbf{M}\,\mathbf{h} = \mathbf{m}$, which is obviously equivalent to require $||\mathbf{h} - \bar{\mathbf{h}}||^2_{\mathbf{\Lambda}} = 0$ with $\mathbf{\Lambda} = \mathbf{M}^T\,\mathbf{M}$ and $\bar{\mathbf{h}} = \mathbf{M}^T\,\mathbf{m}$ (see e.g. [33] for a development).

On the other hand, regarding the term related to "regularization":

- The functions $\mathbf{L}^{ij} : \mathbb{R}^n \to S_n \in W^{s,\infty}(\mathbb{R}^n)$, where S_n is the set of square symmetric matrices define a so called, *diffusion tensor* $\mathbf{L}$, which is :
  - symmetric i.e. $\mathbf{L}^{ij} = \mathbf{L}^{ji}$ and
  - "positive" i.e. so that $\forall \mathbf{M} \in \mathbb{R}^{n\times n}, \mathbf{M}^T\,\mathbf{L}\,\mathbf{M} = \sum_{ijkl} \mathbf{M}^k_i\,\mathbf{L}^{ij}_{kl}\,\mathbf{M}^l_j \geq 0$
  in order the previous definition to be coherent. It defines the *profile* of this regularization.

  Thanks to the *regularization* term $||D\mathbf{h}||^2_{\mathbf{L}}$, since the variation of $\mathbf{h}$ is minimized, a "smoothed" but also well-defined version of $\mathbf{h}$ is obtained [25].

  (a) When the problem is ill-posed, i.e. if there are many (and usually numerically unstable) solutions, the key idea is to choose the solution which variations are minimized: this defines a unique solution but also a well-defined solution.

  (b) When the input function is partially or approximately defined at some points, as discussed previously, the value at such a point is defined using information "around" which diffuses (as discussed now) from well-defined values to undefined or ill-defined values.

Regularization as a diffusion mechanism. Here[⋆] (see e.g. [30]), (1) is minimal only if:

$$\frac{\partial L}{\partial \mathbf{h}} = \mathbf{\Lambda}(\mathbf{h} - \bar{\mathbf{h}}) - \Delta_{\mathbf{L}}\mathbf{h} = 0 \tag{2}$$

with a so-called diffusion term:

$$[\Delta_{\mathbf{L}}\mathbf{h}]_i = \mathrm{div}(\sum_{j=1}^{m} \mathbf{L}_{ij} D h^j) \tag{3}$$

This equation simply states that, at an extremum, the criterion is locally "flat" i.e. its 1st order variation vanishes.

The term of diffusion $\Delta_{\mathbf{L}}\mathbf{h}$ allows to "propagate" some information about $\mathbf{h}$ from one point $\mathbf{x}$ to another, because it depends on the variation of $\mathbf{h}$ at $\mathbf{x}$ i.e. on what happens "around" $\mathbf{x}$.

Clearly, without the term of diffusion, the solution is $\mathbf{h} = \bar{\mathbf{h}}$.

Obviously, at the implementation level, the key problem is the calculation of $\Delta_{\mathbf{L}}\mathbf{h}$.

In order to solve (2) which corresponds to the *gradient* of the criterion (1) a 1st order *gradient descent* is usually proposed[⋆⋆] . Given an initial estimate

[⋆] Our criterion L being regular, its first variation (also called its Gâteaux derivative) at $\mathbf{h} \in F$ in the direction $\mathbf{k} \in H$ is defined by (see, e.g., [6])

$$\delta_{\mathbf{k}} L(\mathbf{h}) = \lim_{\varepsilon \to 0} \frac{L(\mathbf{h} + \varepsilon \mathbf{k}) - L(\mathbf{h})}{\varepsilon}$$

If the mapping $\mathbf{k} \to \delta_{\mathbf{k}} L(\mathbf{h})$ is linear and continuous, the Riesz representation theorem [11] guarantees the existence of a unique vector, denoted by $\nabla_H L(\mathbf{h})$, and called the gradient of L, which satisfies the equality

$$\delta_{\mathbf{k}} L(\mathbf{h}) = (\nabla_H L(\mathbf{h}),\, \mathbf{k})_H$$

for every $\mathbf{k} \in H$. The gradient depends on the choice of the scalar product $(\cdot, \cdot)_H$ though, a fact which explains our notation. If a minimizer $\mathbf{h}^*$ of L exists, then the set of equations $\delta_{\mathbf{k}} L(\mathbf{h}) = 0$ must hold for every $\mathbf{k} \in H$, which is equivalent to $\nabla_H L(\mathbf{h}) = 0$. These equations are called the Euler-Lagrange equations associated with the energy functional L.

[⋆⋆]**About 1st order scheme convergence**. The partial differentiation rule is written $\frac{\partial L}{\partial t} = \nabla_H L \frac{\partial \mathbf{h}}{\partial t}$. Considering $\frac{\partial \mathbf{h}}{\partial t} = -\nabla_H L$ yields $\frac{\partial L}{\partial t} = -||\nabla_{\mathbf{h}} L||^2 < 0$ so that the criterion is strictly decreasing. Since $L \geq 0$ it is also bounded and thus converges towards a minimal value. At this local or global minimum $||\nabla_{\mathbf{h}} L|| = 0$ so that L is stationary. In fact, the gradient magnitude smoothly decreases with time. In practice, convergence is detected as soon as the gradient is below a given numerical threshold, which always occurs in finite time.

$\mathbf{h}_0 \in F$, a time-dependent differentiable function, also noted $\mathbf{h}$, from the interval $[0, +\infty[$ into H is computed as the solution of the following initial value problem:

$$\begin{cases} \frac{d\mathbf{h}}{dt} = -\frac{\partial L}{\partial \mathbf{h}} = \mathbf{\Lambda}(\bar{\mathbf{h}} - \mathbf{h}) + \Delta_{\mathbf{L}}\mathbf{h} \\ \mathbf{h}(0)(.) = \mathbf{h}_0(\cdot) \end{cases} \tag{4}$$

It leads to a local minimum of the criterion, when iteratively computed using, e.g. an Euler rule of the form: $\mathbf{h}_{t+1} = \mathbf{h}_t + \Delta t \, \frac{d\mathbf{h}}{dt}$. Here, since the related quadratic criterion is convex, calculating the *local* partial differential equation at each point leads to the *global* minimization of the criterion (1).

A step ahead, a recurrent equation, allowing the iterative numerical computation of the function $\mathbf{h}$ values for a mesh of samples is derived and (see e.g. [8]) is usually of the form:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \Upsilon \left[\mathbf{\Lambda}(\bar{\mathbf{h}} - \mathbf{h}) + \Delta_{\mathbf{L}}\mathbf{h}\right] \tag{5}$$

where the regular matrix Υ, with $||\Upsilon|| > 0$ allows to control[⋆⋆⋆] the iteration convergence. Obviously, with $\Upsilon = \Delta t Id$ if it is the Euler rule, while improved forms could be used (as discussed in the sequel).

However, better than these technical properties, a key point is that we automatically obtain a general non-linear filter from the previous specifications. Let us discuss this aspect now.

Relation with linear filtering. The present mechanism is easily related to a linear filtering operator in the case where $\mathbf{\Lambda}$ and $\mathbf{L}$ are constant. In this particular case, (2) is now a linear differential equation with constant coefficients.

[⋆⋆⋆] As noticed e.g. in [18], if $||\Upsilon|| > 0$ fixed points of this iterative equation are solution of the Euler equation, while if $||\Upsilon|| \to 0$ the series is contracting so that the convergence is guaranteed. In the worst case the convergence may be obtained towards a sub-optimal value.
Furthermore, the matrix Υ is related to the fact that, at each step, we compute the new value of $\mathbf{h}$ form the previous value of $\mathbf{h}$ *solving a "simple" linear system only.* This includes Gauss-Seidel or Jacobi iteration methods and seems to be a general scheme.

If we consider the Fourier transform $F[\mathbf{h}(\mathbf{x})] = \mathbf{h}(\mathbf{w})$ of the quantities in (2), using the explicit form given in (3), it is straightforward to obtain in the Fourier domain:

$$\mathbf{h}(\mathbf{w}) = \mathbf{G}_{\mathbf{\Lambda},\mathbf{L}}(\mathbf{w})\,\bar{\mathbf{h}}(\mathbf{w}) \text{ with } \mathbf{G}_{\mathbf{\Lambda},\mathbf{L}}(\mathbf{w}) = [\mathbf{\Lambda}_{kl} + \sum_{ij} \mathbf{L}^{ij}_{kl}\, w_i\, w_j]^{-1}\,\mathbf{\Lambda} \qquad (6)$$

so that we finally obtain the convolution: $\mathbf{h}(\mathbf{x}) = \mathbf{G}_{\mathbf{\Lambda},\mathbf{L}}(\mathbf{x}) * \bar{\mathbf{h}}(\mathbf{x})$.

From (6), it is visible that the corresponding linear filter is defined by the $n^2 + m^2\, n\,(n+1)/2$ coefficients of $\mathbf{\Lambda}$ and $\mathbf{L}$ (the former being a general $n \times n$ matrix, the latter being a symmetric tensor). From obvious linear algebra in the general case, $\mathbf{G}_{\mathbf{\Lambda},\mathbf{L}}(\mathbf{w})$ is a rational fraction with a denominator of degree less than $2\,n$ and a numerator of degree less than $2\,(n-1)$. This corresponds to rather general filters. Although not all linear filters can be defined from such a formula, it appears that general filters can easily be approximated.

A step ahead, as shown in [27], if $\mathbf{L}$ is constant and if $\mathbf{\Lambda}$ is negligible, $G_{\mathbf{\Lambda},\mathbf{L}}$ corresponds, at a given time t of the diffusion process defined in (5) to an oriented Gaussian kernel:

$$G_{\mathbf{L},t}(\mathbf{x}) = \frac{1}{\sqrt{(2\,\pi)^n\, 4\,t\,|\mathbf{L}|}}\, e^{-\frac{\mathbf{x}^T\,\mathbf{L}^{-1}\,\mathbf{x}}{4\,t}}$$

However, in both cases, defining the problem from (1) is more informative than defining a filter, since we formally define what is the "objective". Furthermore, implementing it using (5) is much more efficient than explicitly computing a convolution at each point.

A step ahead, if $\mathbf{\Lambda}$ and $\mathbf{L}$ are not constant, the present framework allows to define complex input/output relationships between $\bar{\mathbf{h}}$ and $\mathbf{h}$ which are more general than what is obtained by a linear filter, introducing some "coupling" between each local convolution. This is also called bilateral filtering [2].

Introducing nonlinear diffusion operators. Let us now review how nonlinear regularization profiles influence this diffusion of information. More pre-

cisely, we consider for this section:

$$\mathbf{h}^* = arg\ min_{\mathbf{h}} \int \frac{1}{2}||\mathbf{h}-\bar{\mathbf{h}}||^2_{\mathbf{\Lambda}} + \Phi\left(\varphi_{\mathbf{L}}(D\mathbf{h})\right) \Rightarrow \mathbf{\Lambda}\left(\mathbf{h}-\bar{\mathbf{h}}\right) - \Delta_{\Phi,\mathbf{L}}\mathbf{h} = 0 \quad (7)$$

This corresponds to a large number of regularization mechanisms (see e.g. [26] for a recent review) and a natural (and in fact general) requirement for such approaches [1,9] is to:
(α) obtain isotropic diffusion for small gradient magnitude because we want to maximize the propagation of information in uniform (thus information less) parts of the parametric space and
(β) cancel the propagation of information for high gradient magnitudes in the direction of the gradient in order to preserve large (thus significant) variations of the input function.

In other words, we want to eliminate small, thus likely noisy, variations of the parametric map, but preserve large variations. This is the case, e.g. of $\Phi(u) = \sqrt{u}$ and several other profiles, as reviewed in [2].

Now the main question is: do we **need** to add such a non-linear profile to our specifications? Or could the anisotropic diffusion tensor $\mathbf{L}$ make the job?

To answer this question let us consider $\mathbf{L} = Id$. Using standard differential calculus (given in appendix A) we obtain: $\Delta_{\Phi,Id} = \Delta_{Id,\mathbf{L}'}$ with

$$\mathbf{L}'(D\mathbf{h}) = 2\,\Phi'\left(||D\mathbf{h}||^2\right)\ Id + 4\,\Phi''\left(||D\mathbf{h}||^2\right) Dh^i\,(Dh^j)^T \quad (8)$$

Here $\mathbf{L}'$ corresponds too a non constant operator which depends on $D\mathbf{h}$ itself:
(i) the term related to Φ' corresponds to an isotropic diffusion and
(ii) the term related to Φ'' corresponds to an anisotropic diffusion process in the direction η of the gradient,
exactly as required.

But the key point is that we do not need to complicate the present framework, it is thus useless to consider nonlinear profiles $\Phi()$ at the implementation level: anisotropic linear diffusion (using variable diffusion) is sufficient,

while, thanks to this derivation, we also obtain a suitable form of the tensor $\mathbf{L}$, defined from (8).

When implementing such non-constant diffusion, following, e.g. [20], we consider the initial or previous estimation $D\widehat{\mathbf{h}}$ of $D\mathbf{h}$: either the input function $\bar{\mathbf{h}}$ or the estimation obtained by previous steps of the iterative process. In other words, let us *feedback* the previous estimation $\widehat{\mathbf{h}}$ into $\mathbf{L}(D\widehat{\mathbf{h}})$ to obtain an improved estimate.

As reviewed in [1,2], when considering a nonlinear profile, at the implementation level, a similar *linear approximation* is always used. The convergence is guaranteed by the introduction of a smoothing in the feedback, yielding a variation slow enough to be stable.

To conclude with non-linear diffusion operator, let us briefly note (see [31] for details) that the same mechanism is generalizable to the computation of harmonic maps on Riemannian manifold. We will not further develop this point here, but simply want to point out such a perspective of the present approach: generalization from $\mathbb{R}^n$ to nonlinear manifolds is also to be considered because several computer vision parameters (e.g rotations, non-linear features, etc..) are not simple linear sub-space of an Euclidean space but are intrinsically manifolds.

3 Optimal implementation of a diffusion operator

3.1 Integral approximation of a differential operator

Considering the iterative equation (5) corresponding to the implementation of the regularization mechanisms discussed here, the key problem is the implementation of the diffusion operator $\Delta_{\mathbf{L}}$ (the other terms $\mathbf{\Lambda}\,\mathbf{h}_t - \mathbf{\Lambda}\,\bar{\mathbf{h}}_t$ corresponds to a simple punctual linear operation).

In practice, it is not possible to implement a differential operator such as this 1st and 2nd order diffusion operator without choosing an approximation and a numerical scheme, etc.. with the risk of loosing what was well-defined in the continuous case.

Why is it not possible in practice to implement the "punctual" operator ? Because what is given, in the real world, is a set of "samples". More precisely, we have to consider a set of "measures", defined as integral values of the continuous function over the measurement sensor receptive field. Furthermore, since numerical values contains uncertainties, it is a reasonable choice to "average" several values in order to smooth these uncertainties.

At the implementation level, what is in fact done ? A differential operator is implemented using a derivative filter, i.e. performing a convolution with some suitable kernel. In other words a differential operator is implemented as an integral operator. Let us make this basic fact explicit: this will guide us towards a very general, simple and nice solution.

Integral form of the diffusion operators. With the Dirac distribution $\delta(\mathbf{x})$ (see [24] for an introduction) we may formally[⋆⋆⋆⋆] write:

$$[\Delta_{\mathbf{L}}\mathbf{h}]_i = \sum_j \sigma_{ij} * h^j = \int_{\mathbb{R}^n} \sum_j \sigma_{ij}(\mathbf{x}-\mathbf{y})\, h^j(\mathbf{y})\, d\mathbf{y}$$

with:

$$\sigma_{ij} = \sum_k M_{ij}^k \frac{\partial \delta}{\partial x^k} + \sum_{k,l} L_{ij}^{kl} \frac{\partial^2 \delta}{\partial x^k\, \partial x^l} \text{ where } M_{ij}^k = [\mathrm{div}(\mathbf{L}_{ij})]^k$$

where $*$ represents the convolution.

In words, there is a direct mathematical and canonical link between a differential and an integral operator.

It verifies the conservation property:

$$\int_{\mathbb{R}^n} \Delta_{\mathbf{L}}\mathbf{h}(\mathbf{x})\, d\mathbf{x} = 0 \tag{9}$$

[⋆⋆⋆⋆] Let us expand the right-hand side of (3) (using the relation $\mathrm{div}(\mathbf{A}\mathbf{b}) = (\mathrm{div}\mathbf{A})\cdot\mathbf{b} + \mathrm{Tr}(\mathbf{A}\, D\mathbf{b})$, where $\mathbf{A}$ is a matrix and $\mathbf{b}$ a vector):

$$\sum_{j=1}^m \mathrm{div}(\mathbf{L}_{ij} Dh^j) = \sum_{j=1}^m \mathrm{div}(\mathbf{L}_{ij}) \cdot Dh^j + \mathrm{Tr}(\mathbf{L}_{ij}\mathbf{H}_j)$$

with:

$$Dh^j = D\delta * h^j \;\; \mathbf{H}_j = \mathbf{H}_\delta * h^j\,,$$

The divergence of a square matrix $\mathbf{A}$ of size n is the vector of size n whose ith component is the divergence of the ith row vector of $\mathbf{A}$. From this follows

$$\mathrm{div}(\mathbf{L}_{ij}) \cdot Dh^j = \mathrm{div}(\mathbf{L}_{ij}) \cdot (D\delta * h^j) = (\mathrm{div}(\mathbf{L}_{ij}) \cdot D\delta) * h^j,$$

and

$$\mathrm{Tr}(\mathbf{L}_{ij}\mathbf{H}_j) = \mathrm{Tr}(\mathbf{L}_{ij}\mathbf{H}_\delta * h^j) = \mathrm{Tr}(\mathbf{L}_{ij}\mathbf{H}_\delta) * h^j.$$

Note that $\mathrm{div}(\mathbf{L}_{ij}) \cdot D\delta$ and $\mathrm{Tr}(\mathbf{L}_{ij}\mathbf{H}_\delta)$ are first and second order differential operators:

$$\mathrm{div}(\mathbf{L}_{ij}) \cdot D\delta = \sum_k [\mathrm{div}(\mathbf{L}_{ij})]^k \frac{\partial \delta}{\partial x^k},$$

and

$$\mathrm{Tr}(\mathbf{L}_{ij}\mathbf{H}_\delta) = \sum_{k,l} L_{ij}^{kl} \frac{\partial^2 \delta}{\partial x^k\, \partial x^l}$$

in coherence with the physical law of conservation property for diffusion processes. In words, we want to balance the $\mathbf{h}$ values in the information diffusion process, i.e. guaranty that if we reduce the value at one location it will increase elsewhere accordingly, as in a fluid which particles are neither created nor deleted. This guarantees the stability of the numerical computations.

Furthermore, this is a "punctual" integral operator in the sense its magnitude is zero except at $\mathbf{x}$. Considering a quadratic semi-norm (i.e. its variance or inertia), it writes:

$$\int_{\mathbb{R}^n - B(\mathbf{x},\varepsilon)} \sigma_{ij}(\mathbf{x}-\mathbf{y})^2 \, d\mathbf{y} = 0 \tag{10}$$

where $B(\mathbf{x},\varepsilon)$ is a ball around $\mathbf{x}$ of radius ε, which can be as small as possible, but must be excluded, in order (10) to be well-defined because the product of two distributions is not necessarily a distribution.

Optimal approximation of the diffusion operators. Based on this remark and following [23,7,30], we approximate the differential operator by an integral operator of the form:

$$[\Delta^*_{\mathbf{L}}\mathbf{h}(\mathbf{x})]_k = \sum_l \left[\int_{\mathbb{R}^n} \sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y})\, \mathbf{h}^l(\mathbf{y})\, d\mathbf{y}\right] - \bar{\sigma}^\epsilon_{kl}(\mathbf{x})\, \mathbf{h}^l(\mathbf{x}) \tag{11}$$

with $\bar{\sigma}^\epsilon_{kl}(\mathbf{x}) = \int_{\mathbb{R}^n} \sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y})\, d\mathbf{y}$

For such an approximation to be well-defined*****, the integral must be convergent. Here, to obtain this property, we assume that $\sigma^\epsilon(\mathbf{x},\mathbf{y})$ has a bounded support S, *included in a ball* $B(\mathbf{x},\epsilon)$ *of radius* $\epsilon > \varepsilon$, i.e. $S \subset B(\mathbf{x},\epsilon)$.

The 2nd term allows to verify the conservation property, as soon as the kernel is symmetric, i.e.

$$\sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y}) = \sigma^\epsilon_{kl}(\mathbf{y},\mathbf{x}) \tag{12}$$

***** In addition, this operator must belong to a well-defined functional space. Here following [7] again, we consider a subset of the distributions: the Sobolev function space $W^{s,\infty}(\mathbb{R}^n)$. In fact, we are going to obtain solutions which are ordinary differentiable functions, so that it is only a formal point.

.

It appears as an operator which *computes a weighted mean value around* $\mathbf{x}$ *minus the balanced value at* $\mathbf{x}$.

It is easy to verify that, if $\sigma^{\epsilon}_{kl}(\mathbf{x},\mathbf{y}) = \sigma_{kl}(\mathbf{x}-\mathbf{y})$ then $[\Delta^{*}_{\mathbf{L}}\mathbf{h}]_k = [\Delta_{\mathbf{L}}\mathbf{h}]_k$ and the "exact" operator $\Delta_{\mathbf{L}}\mathbf{h}$ is a particular case of the "approximate" operator $\Delta^{*}_{\mathbf{L}}\mathbf{h}$.

Requiring, σ^{ϵ}_{kl} to be around each point $\mathbf{x}$ as closed as possible to the "exact" operator σ_{kl}, we formalize this proximity by the following quadratic criterion:

$$\min_{\sigma^{\epsilon}} \int_S \sigma^{\epsilon}_{kl}(\mathbf{x},\mathbf{y})^2 d\mathbf{x}d\mathbf{y} \tag{13}$$

In words, we use a maximal amount of information close to the point: the "sharpest" the operator, the best.

It is easily shown [30] that minimizing (13) is equivalent to choose $\sigma^{\epsilon}_{kl}(\mathbf{x},\mathbf{y})$ as close as possible to "exact" operator $\sigma_{kl}(\mathbf{x}-\mathbf{y})$ for a quadratic distance, related to the semi-norm specified in (10).

We now have a precise definition of the "discrete approximation" of a differential operator.

Please note that, although out of the scope of this paper, the present solution is directly usable for any 1st or 2nd order differential operator (any $\mathbf{M}$ and $\mathbf{L}$) and straightforward to generalize to any differential operator.

Relations between integral and differential operators. In order to relate the differential operator $\Delta_{\mathbf{L}}\mathbf{h}$ with its integral approximation $\Delta^{*}_{\mathbf{L}}\mathbf{h}$. We identify $\Delta_{\mathbf{L}}\mathbf{h}$ with the Taylor expansion of $\Delta^{*}_{\mathbf{L}}\mathbf{h}$ at some order r [31], yielding a bound on the error for our approximation, as a function of ϵ^{r-1} i.e.:

$$||\Delta_{\mathbf{L}}\mathbf{h} - \Delta^{*}_{\mathbf{L}}\mathbf{h}||_{0,\infty} = ||R^{r}\mathbf{h}||_{0,\infty} \leq C\,\epsilon^{r-1}||\mathbf{h}||_{r+1,\infty}$$

for a fixed constant C, since $S \subset B(\mathbf{x},\epsilon)$. This, for a covering of the parameter space by supports $S \subset B(\mathbf{x},\epsilon)$ organized in some mesh, allows to conclude that $\lim_{\epsilon w \to 0}\Delta^{*}_{\mathbf{L}}\mathbf{h} = \Delta_{\mathbf{L}}\mathbf{h}$ and $\lim_{r\to\infty}\Delta^{*}_{\mathbf{L}}\mathbf{h} = \Delta_{\mathbf{L}}\mathbf{h}$ for small

ϵ: in words, this approximation is consistent, with respect to the sampling and the Taylor expansion.

More precisely, we consider the Taylor expansion of $\mathbf{g}(\mathbf{y},\mathbf{x}) = \mathbf{h}(\mathbf{y}) - \mathbf{h}(\mathbf{x})$ with respect to $\mathbf{y} - \mathbf{x}$.

In other words, we consider the following change of variables $\mathbf{d} = \mathbf{y} - \mathbf{x}$ and, say, $\mathbf{s} = (\mathbf{y} + \mathbf{x})/2$ in order to write the expansion with respect to $\mathbf{d}$:

$$\mathbf{g}(\mathbf{y},\mathbf{x}) = \textstyle\sum_{|\alpha|=1}^{r} \frac{\partial^\alpha \mathbf{h}}{\alpha!}(\mathbf{x})\big|_{\mathbf{y}=\mathbf{x}} (\mathbf{y}-\mathbf{x})^\alpha + o(||\mathbf{y}-\mathbf{x}||^r)$$

and a few algebra yields from (11):

$$\left[\Delta^*_{\mathbf{M},\mathbf{L}}\mathbf{h}\right]_k = \sum_{l=1}^{m} \sum_{|\alpha|=1}^{r} \frac{\partial^\alpha \mathbf{h}^l}{\alpha!} \int_{\mathbb{R}^n} \sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y})\,(\mathbf{y}-\mathbf{x})^\alpha\, d\mathbf{y} + [R^r\mathbf{h}]_k$$

where the remainder $R^\epsilon_{kl}\mathbf{h}$ of this expansion may be written using an integral form:

$$[R^r\mathbf{h}]_k = \sum_{l=1}^{m} \sum_{|\alpha|=r+1} \frac{r+1}{\alpha!} \int_{\mathbb{R}^n\times[0,1]} \sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y})\,(\mathbf{y}-\mathbf{x})^\alpha\,(1-u)^r\,\partial^\alpha \mathbf{h}^l(\mathbf{x}+u\,(\mathbf{y}-\mathbf{x}))\,d\mathbf{y}\,du$$

From the assumptions we have raised, because the support is included in a ball of radius ϵ, the remainder is bounded by the standard condition:

$$||R^r\mathbf{h}||_{0,\infty} < C\,\epsilon^{r-1}||\mathbf{h}||_{r+1,\infty} \tag{14}$$

where C is a fixed quantity [7], yielding the previous bound.

If we rewrite the diffusion operator with the same notations:

$$[\Delta_{\mathbf{M},\mathbf{L}}\mathbf{h}]_k = \sum_{l=1}^{m}\left[\sum_{\mathbf{e}_j} \mathbf{M}^j_{kl}\,\partial^{\mathbf{e}_j}\mathbf{h}^l + \sum_{\mathbf{e}_i+\mathbf{e}_j} \mathbf{L}^{ij}_{kl}\,\partial^{\mathbf{e}_i+\mathbf{e}_j}\mathbf{h}^l\right]$$

we easily identify the two expressions and obtain:

$$\begin{array}{lll}
r \geq |\alpha| > 2 & \int_{\mathbb{R}^n} \sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y})\,(\mathbf{y}-\mathbf{x})^\alpha\,d\mathbf{y} & = 0_r \quad [\text{C0}] \\
|\alpha| = 2 \;\; 2\,\mathbf{L}^{ij}_{kl}(\mathbf{x}) & - \int_{R^n} \sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y})\,(\mathbf{y}-\mathbf{x})^{\mathbf{e}_i+\mathbf{e}_j}\,d\mathbf{y} & = 0_r \quad [\text{C2}] \\
|\alpha| = 1 \;\; \mathbf{M}^j_{kl}(\mathbf{x}) & - \int_{\mathbb{R}^n} \sigma^\epsilon_{kl}(\mathbf{x},\mathbf{y})\,(\mathbf{y}-\mathbf{x})^{\mathbf{e}_j}\,d\mathbf{y} & = 0_r \quad [\text{C1}]
\end{array}$$

where $0_r = \int_S o(\mathbf{y}-\mathbf{x})^r d\,\mathbf{y}$ is a negligible quantity for sufficiently small ϵ.

Since (here $\int_{B(0,1)} = \frac{\pi^{n/2}}{\Gamma(n/2+1)}$ is the volume of the unit ball) from a few algebra:

$$|\int_{\mathbb{R}^n} \sigma_{kl}^{\epsilon}(\mathbf{x},\mathbf{y})\,(\mathbf{y}-\mathbf{x})^{\alpha}\,d\mathbf{y}| \leq [||\sigma_{kl}^{\epsilon}||_{0,\infty}\ \int_{B(0,1)}/(|\alpha|+1)]\,\epsilon^{|\alpha|+1}$$

thus exponentially decreases with $|\alpha|$, for higher values of $|\alpha|$ the condition [C0] is numerically automatically verified as corresponding to a negligible quantity. It is thus reasonable choice to bound $|\alpha|$ with a fixed value r.

These conditions have the interesting property to be "decoupled" in the sense that they allow, for a given (k,l) to relate each σ_{kl}^{ϵ} to the corresponding $\mathbf{M}_{kl}$ and $\mathbf{L}_{kl}$ independently.

These conditions [C0] [C2] and [C1] allow to identify up to the rth order the two expressions. Among all solutions verifying these constraints, the operator minimizing (13) is going to be chosen.

Reviewing existing solutions. In the literature not only our bounded support solution but at least two other solutions [23,7] have being derived for this integral operator, in the case where $\mathbf{M} = \sum_i \frac{\partial \mathbf{L}}{\partial x^i}$ and for scalar functions only. Contrary to the present approach, these solutions do not correspond to a bounded support S although, indeed, the related integral is still convergent. On the contrary, they are related to cut-off functions defined and integrable in R^n. In other words, the choice of the support is implicitly realized when choosing this cut-off function.

As a comparison with our solution defined on a bounded support, let us briefly review these solutions. Here we consider only a scalar function h and omit the related indices.

The Raviat solution. This solution [23], at the origin of the work of [7], is of the form:

$$\sigma^{\epsilon}(\mathbf{x},\mathbf{y}) = \sum_{ij} \int_{\mathbb{R}^n} L_{ij}(\mathbf{z})\,\frac{\partial \xi(\mathbf{x}-\mathbf{z})}{\partial x_i}\,\frac{\partial \xi(\mathbf{z}-\mathbf{y})}{\partial x_j}\,d\mathbf{z}$$

where the cut-off function ξ verifies $\int_{\mathbb{R}^n} \mathbf{x}^{\alpha}\,\xi(\mathbf{x})\,d\mathbf{x} = \begin{cases} 0 & 1 \leq |\alpha| < r \\ 1 & \alpha = 0 \end{cases}$

This solution is the "more general" in the sense that it has been derived from conditions [C0] [C2] and [C1] (here convergence is now due to the cut-off function profile, not the fact that S is bounded) with a minimum of additional constraints.

Although formally simple, as readable on the formula, this solution is very heavy to implement because a numerical integration is required at each step in order to obtain $\sigma^\epsilon(\mathbf{x},\mathbf{y})$.

The Degond solutions. These are solutions [7] of the form:

$$\sigma^\epsilon(\mathbf{x},\mathbf{y}) = \frac{1}{\epsilon^{n+2}} \sum_{i,j} m_{ij}\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) \psi_{ij}\left(\frac{\mathbf{y}-\mathbf{x}}{\epsilon}\right)$$

(here equivalently we can choose $\frac{1}{2}\left(m_{ij}(\mathbf{x}) + m_{ij}(\mathbf{y})\right)$ instead of $m_{ij}\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right)$) with:

$$\mathbf{m} = \mathbf{L} - \frac{\upsilon}{n+2} trace(\mathbf{L})\, \mathbf{I}_{n\times n} \text{ with } \upsilon \in \{0,1\}$$

If $\upsilon = 0$, we re-obtain $\psi_{ij} = \frac{\partial^2 \xi}{\partial x_i \partial x_j}$ with a cut-off function ξ which corresponds to the previous case

If $\upsilon = 1$, we obtain $\psi_{ij} = x_i\, x_i\, \theta(||\mathbf{x}||)$ where the cut-off function θ verifies:

$$\int_{\mathbb{R}^+} \rho^{n+1+2\,\alpha}\, \theta(\rho)\, d\rho = \begin{cases} \frac{\sigma_{n-1}\,(n-1)}{n\,(n+2)} & \alpha = 1 \\ 0 & 1 < 2\,\alpha \leq r \end{cases}$$

writing $\sigma_n = \frac{n\,\pi^{n/2}}{\Gamma(n/2+1)}$ (this constant defines the surface of the boundaries $\partial B(\mathbf{x},\rho)$ of a ball $B(\mathbf{x},\rho) = \{\mathbf{y}, ||\mathbf{x}-\mathbf{y}|| < \rho\}$ of center $\mathbf{x}$ and radius ρ in $\mathbb{R}^n$, for $\rho = 1$).

Such cut-off functions are calculated for instance considering a profile of the form:

$$\theta(\rho) = e^{-\beta\,\rho^2} \sum_{d=0}^{2\,d=r} a_d\, \rho^{2\,d}$$

with $\beta > 0$ where the constants a_d are easily obtain the linear equations.

This solution is efficient to implement but has some drawbacks: it does not consider a bounded support and among existing solutions it is not an optimal solution, while it does neither consider vectorial maps nor general 1st or 2nd order differential operators.

The bounded support solution. It has been shown [30] that, in the continuous case when the support is bounded, the optimal solution could

be approximated by solutions of the form:

$$\sigma^{\epsilon}_{kl}(\mathbf{x},\mathbf{y}) = \sum_{j} \mathbf{A}_{j\,kl}(\mathbf{x},\mathbf{y})\,\mathbf{M}^{j}_{kl}\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) + \sum_{ij} \mathbf{B}_{ij\,kl}(\mathbf{x},\mathbf{y})\,\mathbf{L}^{ij}_{kl}\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right)$$

where $\mathbf{A}_{j\,kl}(\mathbf{x},\mathbf{y})$ and $\mathbf{B}_{ij\,kl}(\mathbf{x},\mathbf{y})$ are rational functions which only depends on the support S. The poles of these functions approximate the punctual values of the Dirac distribution derivatives. Such approximations are derived automatically using a piece of maple code [30]. As illustrated in Fig. 2, it allows to verify that using a reasonable order $r = 5..8$ for 1st or 2nd order operators allows to obtain a solution which is qualitatively correct. A step further, it allows to verify that the present mechanism generates coherent kernels for various differential operators as illustrated in Fig. 3.

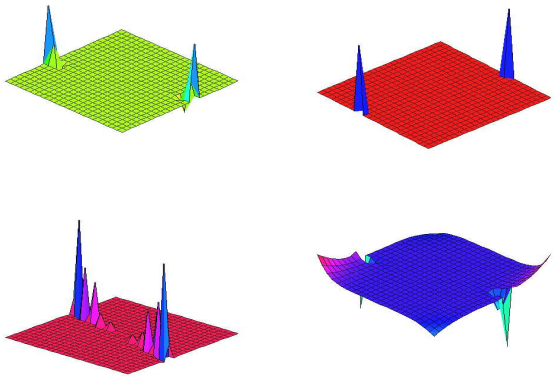


Fig. 2. A few examples of operator 1D-profiles, considering an isotropic second-order derivative, represented in function of (x, y); *top-left view* r = 5, s = 10: we obtain a profile with two poles qualitatively equivalent to the δ'' distribution; *top-right view* r = 8, s = 20: increasing the order of correspondence, a profile closer to δ'' is obtained; *bottom-left view* r = 2, s = 3: when the correspondence is insufficient (r is too small) we obtain a profile which is still qualitatively correct but very "flat"; *bottom-right view* r = 6, s = 10: when considering without any redundancy, the approximation may be slightly biased with spurious effects.

But the key point is that the bounded support solution allows to automatically derive a unbiased solution in the discrete case, as detailed now.

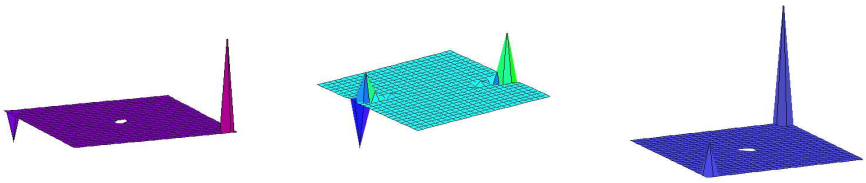


Fig. 3. A few examples of operator 2D-profiles, with r = 3, s = 6, represented in the (x^0, x^1) plane; *left view* approximation of 1st order derivative isotropic operator $\partial^{(1,0)}$ qualitatively equivalent to the corresponding continuous operator $\delta^{(1,0)}$; *middle view* approximation of 2nd order non-isotropic operator $L^{ij}(\mathbf{x}) = \delta^{ij}\, x^0$ and *right view* a 2nd-order non-isotropic operator $L^{ij} = \delta^{ij} + i$, both illustrating how solutions adapt to such profiles.

3.2 Derivation in the discrete case.

Let us consider the computer implementation on 2D or 3D dense data "images", i.e. a regular mesh of hyper-cubes.

The integral approximation must be sampled and becomes a summation:

$$\Delta^*_{\mathbf{L}}\mathbf{h}^k(\mathbf{x_u}) = \sum_l \sum_{\mathbf{y_v}=(v_1,\cdots,v_n)} \sigma^{\epsilon}_{kl}(\mathbf{x_u},\mathbf{y_v})\,\left[\mathbf{h}^l(\mathbf{y_v}) - \mathbf{h}^l(\mathbf{x_u})\right] \text{ with } \mathbf{x_u} = (u_1,\cdots,u_n) \tag{15}$$

Here, we consider without loss of generality, that pixel/voxel hyper-cubes volume is 1.

In this context, a reasonable model [32,18] for a "pixel" or "voxel" signal is to assume that:

$$\mathbf{h}(\mathbf{x_u}) = \int_{\left[u_1-\frac{1}{2}..u_1+\frac{1}{2},\cdots,u_n-\frac{1}{2}..u_n+\frac{1}{2}\right]} \mathbf{h}(\mathbf{x})\, d\mathbf{x}$$

i.e. that the *pixel/voxel value is the average value of the signal* distribution over its domain.

Furthermore, if we consider that $\sigma^{\epsilon}(\mathbf{x},\mathbf{y})$ *is piece-wise constant*, it appears that rewriting the integral operator as a summation is NOT only an approximation but an *unbiased implementation* of these differential operators

since:

$$\Delta^*_{\mathbf{L}} h(\mathbf{x_u}) = \sum_l \int_{\mathbb{R}^n} \sigma^\epsilon_{kl}(\mathbf{x_u}, \mathbf{y}) \left[\mathbf{h}^l(\mathbf{y}) - \mathbf{h}^l(\mathbf{x_u})\right] d\mathbf{y} =$$
$$\sum_l \sum_{\mathbf{y_v}=(v_1,\cdots,v_n)} \sigma^\epsilon(\mathbf{x_u}, \mathbf{y_v}) \int_{\mathbf{y}\in\left[v_1-\frac{1}{2}..v_1+\frac{1}{2},\cdots,v_n-\frac{1}{2}..v_n+\frac{1}{2}\right]} \left[\mathbf{h}^l(\mathbf{y}) - \mathbf{h}^l(\mathbf{x_u})\right] d\mathbf{y} =$$
$$\sum_l \sum_{\mathbf{y_v}=(v_1,\cdots,v_n)} \sigma^\epsilon(\mathbf{x_u}, \mathbf{y_v}) \left[\mathbf{h}^l(\mathbf{y_v}) - \mathbf{h}^l(\mathbf{x_u})\right]$$

As a consequence, with a few algebra, conditions [C0], [C1] and [C2] reduce to:

$$\sum_{\mathbf{v}} \nu^\alpha_{\mathbf{v}}(\mathbf{x_u})\, \sigma^\epsilon_{kl}(\mathbf{x_u}, \mathbf{y_v}) = 0_r + \begin{cases} 0 & 2 < |\alpha| \leq r \\ 2\, \mathbf{L}^{ij}_{kl}(\mathbf{x_u}) & \alpha = \mathbf{e}_i + \mathbf{e}_j \\ \mathbf{M}^j_{kl}(\mathbf{x_u}) & \alpha = \mathbf{e}_j \end{cases} \tag{16}$$

using the notation: $\nu^\alpha_{\mathbf{v}}(\mathbf{x_u}) = \int_{\left[v_1-\frac{1}{2}..v_1+\frac{1}{2},\cdots,v_n-\frac{1}{2}..v_n+\frac{1}{2}\right]} (\mathbf{y} - \mathbf{x_u})^\alpha\, d\mathbf{y}$. Here, since $\mathbf{x_u}$ takes constant values, $\nu^\alpha_{\mathbf{v}}(\mathbf{x_u})$ is a constant.

Similarly, the symmetry condition leads to:

$$\sigma^\epsilon_{kl}(\mathbf{x_u}, \mathbf{z_w}) = \sigma^\epsilon_{kl}(\mathbf{z_w}, \mathbf{y_v})$$

while the optimal criterion proposed in (13) is now of the form:

$$\min \sum_{\mathbf{v}} \sigma^\epsilon_{kl}(\mathbf{x_u}, \mathbf{y_v})^2$$

yielding to a simple quadratic minimization problem with linear constraints (16) with respect to the vector:

$$\Sigma = \left[\sigma^\epsilon_{kl}(\mathbf{x_u}, \mathbf{y_v})_{\mathbf{v}\in[-S..S,..,-S..S]}\right]^T$$

containing the coefficients of this symmetric tensor.

It is immediate to count that in an hyper-cube $[-S..S]^n$ of size $(2\,S+1)^n$, for a given $\mathbf{x_u}$, the symmetric bi-variate operator has $q'_{n,S} = (2\,S+1)^n$ independent coefficients $\sigma^\epsilon_{kl}(\mathbf{x_u}, \mathbf{y_v})$.

Since there are $\frac{(n+d)!}{n!\,d!}$ monomial of degree less or equal d with n variables, we can immediately count the number of equations i.e. $p_{n,r} = \frac{(n+r)!}{r!\,n!} - 1$ in

the general case.

We thus can calculate, in the general case, the minimal size S of the hyper cube, for a given order r and dimension n allowing to solve the linear equations, this table gives this value S for an approximation up to the r-th order, for a 1D, 2D or 3D problem ($n = 1, 2, 3$):

r	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
n = 1	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8
n = 2	1	1	1	2	2	3	3	3	4	4	4	5	5	5	6	6
n = 3	1	1	1	2	2	2	2	3	3	3	4	4	4	4	5	5

The closed form solution is directly obtained from the symbolic calculation `least-square` routine given in appendix B. This solution of the form:

$$\sigma^{\epsilon}_{kl}(\mathbf{x_u}, \mathbf{y_v}) = \sum_j \mathbf{A}_{j\,kl}(\mathbf{x_u}, \mathbf{y_v})\, \mathbf{M}^j_{kl}\,(\mathbf{x_u}) + \sum_{ij} \mathbf{B}_{ij\,kl}(\mathbf{x_u}, \mathbf{y_v})\, \mathbf{L}^{ij}_{kl}\,(\mathbf{x_u})$$

Here $\mathbf{A}_{j\,kl}(\mathbf{x_u}, \mathbf{y_v})$ and $\mathbf{B}_{ij\,kl}(\mathbf{x_u}, \mathbf{y_v})$ are "masks" determined by the previous derivation and which only depends on the pixel/voxel characteristics.

As an illustration, for an isotropic 2D operator ($\mathbf{M}^i = 0$ and $\mathbf{L}^{ij} = \delta^{ij}$) we obtain the masks reported in Fig. 4. Although this does not bring much with respect to usual operators, it allows to verify the coherence of the obtained results.

$$\begin{bmatrix} .1048 & .1271 & .1048 \\ .1271 & .0725 & .1271 \\ .1048 & .1271 & .1048 \end{bmatrix} \qquad \begin{bmatrix} .02233 & .03182 & .03693 & .03182 & .02233 \\ .03182 & .05337 & .06462 & .05337 & .03182 \\ .03693 & .06462 & .03695 & .06462 & .03693 \\ .03182 & .05337 & .06462 & .05337 & .03182 \\ .02233 & .03182 & .03693 & .03182 & .02233 \end{bmatrix}$$

Fig. 4. Isotropic 2D-masks obtained for $r = 2$ or 3 and $s = 1$ (*left array*) and $s = 2$ (*right array*). In both cases, we obtain a contribution decreasing with the eccentricity, and isotropic in the horizontal/vertical directions as expected. Since we have considered an integral over the domain without excluding the domain center, there is also a (small) contribution of the central pixel. These masks have been normalized in the sense that $\bar{\sigma} = 1$.

Similarly, an anisotropic 2D-operator is reported in Fig. 5 showing how we directly obtain the convolution mask from the "feedback" vector. Here, for simplicity, we assume that the gradient is defined along the edge normal, i.e. in a unique direction $\mathbf{n}$.

In order to understand the behavior of this mask, let us consider a normalized feedback vector, i.e. $||\hat{\mathbf{n}}||^2 = 1$, with e.g. $\hat{n}_x = 0$, yielding: $\begin{bmatrix} .1048 & .5053 & .1048 \\ -.2511 & .07255 & -.2511 \\ .1048 & .5053 & .1048 \end{bmatrix}$

It is obvious that this 2nd order anisotropic derivation mask corresponds to a second order derivative in the y direction and a smoothing in the x direction as expected.

$$\begin{bmatrix} .1048\,||\hat{\mathbf{n}}||^2 + .3782\,\hat{n}_x\,\hat{n}_y & .5053\,\hat{n}_y^2 - .2511\hat{n}_x^2 & .1048\,||\hat{\mathbf{n}}||^2 - .3782\,\hat{n}_x\,\hat{n}_y \\ .5053\,\hat{n}_x^2 - .2511\hat{n}_y^2 & .07255\,||\hat{\mathbf{n}}||^2 & .5053\,\hat{n}_x^2 - .2511\hat{n}_y^2 \\ .1048\,||\hat{\mathbf{n}}||^2 - .3782\,\hat{n}_x\,\hat{n}_y & .5053\,\hat{n}_y^2 - .2511\hat{n}_x^2 & .1048\,||\hat{\mathbf{n}}||^2 + .3782\,\hat{n}_x\,\hat{n}_y \end{bmatrix}$$

Fig. 5. An example of anisotropic 2D-mask in the direction $\hat{\mathbf{n}} = (\hat{n}_x, \hat{n}_y)$ obtained for $r = 2$ or 3 and $s = 1$. See text for details.

These discrete implementations of the present diffusion operator is experimented in the next section.

Implementing the iteration scheme. As revisited in (5), for a control variable $\nu \in]0..1]$, we immediately obtain an iterative scheme to solve the regularization equation.

In the present context, simply solving the linear equation (5) with respect to $\mathbf{h}(\mathbf{x_u})$, from a few algebra left to the reader, the iterative scheme may be written:

$$\begin{aligned} \mathbf{h}^o(\mathbf{x_u}) = \textstyle\sum_{kl} \Upsilon_k^o \, \Big[\mathbf{h}^k(\mathbf{x_u}) + \nu \, \Big[\mathbf{\Lambda}_l^k \, \bar{\mathbf{h}}^l(\mathbf{x_u}) + \\ \Big[\textstyle\sum_{\mathbf{v},\mathbf{y_v} \in S} \sigma_{kl}^{\epsilon}(\mathbf{x_u}, \mathbf{y_v}) \, \mathbf{h}^l(\mathbf{y_v})\Big]^T\Big]\Big] \end{aligned} \tag{17}$$

with $\begin{cases} \Upsilon = [1 + \nu \, [\Lambda + \bar{\sigma}]]^{-1} \\ \bar{\sigma} = \sum_{\mathbf{v},\mathbf{y_v} \in S} \sigma(\mathbf{x_u}, \mathbf{y_v}) \end{cases}$ while the reader will easily verify that if $\nu > 0$ the fixed point of this iterative scheme is the Euler-Lagrange equation solution, while if $\nu \to 0$ the scheme converges (not necessarily towards the exact solution but, in the worst case, towards a sub-optimal solution) as used in [18].

In practice, we start with $\nu = 1$ and divide this value, say by 2, until from

one step to another the criterion (13) decreases, as detailed and experimented in [33].

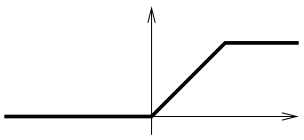
4 Experimental illustrations

4.1 Experimenting with a image de-noising problem

Following [5] we propose a 1st experimentation related to image restoration, more precisely image de-noising when high noise levels with fine scale details are present. The goal is thus to "smooth" an image reducing the noise and preserving the edges, in fact the shape of the main objects. Here $\mathbf{h}$ is thus the image intensity (the red, green and blue channels of the color image).

The goal of the present experiment is to show how the anisotropic diffusion mechanism could be easily implemented, providing efficient results. In (1), we thus simply consider the diffusion term and no input related term (i.e. $\mathbf{\Lambda} = 0$): the relation with the input is simply due to the fact the output initial value is always the input (as made explicit in (4)) which then evolve according to the diffusion mechanism.

As in (8), we choose a diffusion mechanism with (i) isotropic diffusion (thus $\mathbf{L} \equiv Id$) for small gradient magnitudes and (ii) anisotropic diffusion along the edge's tangent for higher magnitudes (thus $\mathbf{L} \equiv \mathbf{g}^{\perp}(\mathbf{g}^{\perp})^T$ where $\mathbf{g}^{\perp}$ is the edge tangent, orthogonal to the edge normal $\mathbf{g}$). Here, the balance between isotropic and anisotropic diffusion is implemented via a simple thresholded profile, as made explicit in Fig. 6.



$$\nu(x) = \begin{cases} 0 & x \leq 0 \\ x & 0 < x < 1 \\ 1 & x \geq 1 \end{cases}$$

Fig. 6. A simple progressive thresholded profile, see text for details

More precisely, considering a threshold s on the gradient magnitude, we

use the Cottet-Ayyadi formula:

$$\mathbf{L}(t) = T * \left[\frac{3}{2} \nu \left(1 - \frac{||\mathbf{g}||^2}{s^2} \right) Id + \nu \left(\frac{||\mathbf{g}||^2}{s^2} \right) \frac{\mathbf{g}^{\perp} (\mathbf{g}^{\perp})^T}{||\mathbf{g}||^2} \right] \text{ with } \mathbf{g} = S * Dh$$

where $\mathbf{g} = S * Du$ is a spatial smoothing version (a local Gaussian filtering in a 3x3 window) of the gradient and $\mathbf{L}(t) = T * \mathbf{F}$ a temporal smoothing****** (an 1st order recursive filter, thus with an exponential profile) of the diffusion tensor. The role of this smoothing is to stabilize the feedback between the estimation of the filtered intensity and the diffusion tensor in which the intensity is re-injected via the local gradient computation. The very interesting result is that the final result very weakly (in fact the difference is often not observable) depends on the temporal smoothing window because at the convergence the diffusion tensor is constant thus invariant with respect to temporal smoothing. A typical window of 2-5 samples seems suitable. A step further, the spatial smoothing S has a weak influence on the final result since it only indirectly acts on the output. However suppressing the temporal or the spatial smoothing kills the mechanism stability and spurious results may occur.

Yet another step further, aside from using the Cottet-Ayyadi formula reported here, we have also experimented a few variants (e.g.: forget the $\frac{3}{2}$ ratio in the formula which seems to have no influence or uses not a progressive profile $\nu()$ but a Heaviside profile with almost negligible variation for the experimented data set). Doing this, we experimentally verified two facts:
- thanks to the fact we can automatically re-implement a diffusion scheme given any $\mathbf{L}$ formula it is a very simple task to check several solutions,
- only the gradient magnitude threshold s has a real influence on the result (it can be sued to tune the level of details which is preserved or filtered).

Results are shown in Fig. 7 and Fig. 8 while a demonstration applet is visible on the web: the `http://www-sop.inria.fr/odyssee/imp/ima/doc-files/ar` applet allows to play with the smoothing parameter experimenting its very

****** In [5], the authors propose a dynamic regularization tensor $\mathbf{L}(t)$ solution of the differential equation:

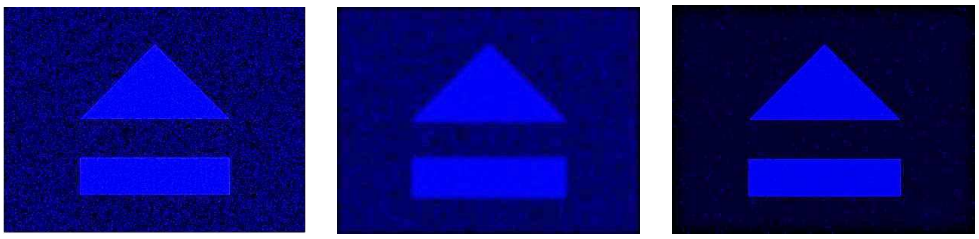
$$\frac{\partial \mathbf{L}(t)}{\partial t} + \mathbf{L}(t) = \mathbf{F}(t) \Leftrightarrow \mathbf{L}(t) = \int_{-\infty}^{t} \mathbf{F}(\tau)\, e^{\tau - t} d\tau$$

which strictly corresponds to an exponential temporal smoother.

weak influence and the gradient threshold parameter much more significant. The results reported here are reproducible in this applet.

Raw *Isotropic* *Anisotropic*

Fig. 7. An example of result using anisotropic diffusion (right image), as implemented using the proposed method. The original image is on the left. As a comparison, a Gaussian filtering (isotropic diffusion) is shown in the middle.



Raw *Isotropic* *Anisotropic*

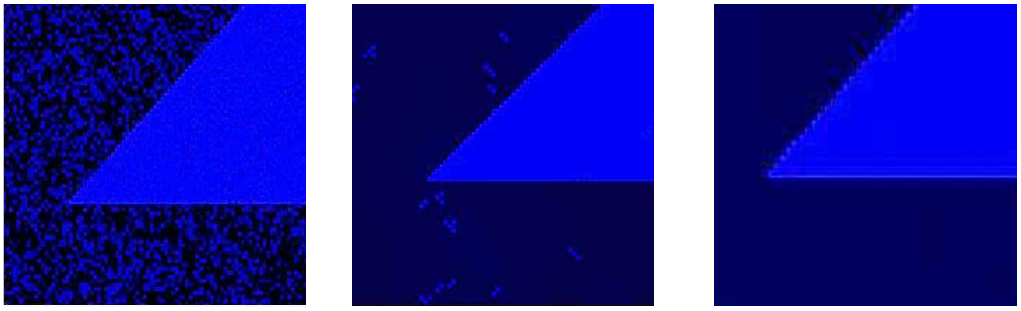
Fig. 8. Another example of result using anisotropic diffusion (right image), as implemented using the proposed method. The original image is on the left. As a comparison, a Gaussian filtering (isotropic diffusion) is shown in the middle.

We have also observed that it was almost always sufficient to implement the diffusion kernel in a 5x5 window (thus a bit less that what was expected from the theory predicting a bias with windows less than 7x7) although with huge noise levels the use of a higher window has a positive influence, as visible in Fig.9.

4.2 Experimenting with a motion estimation problem

Video RGB image sequences

As a second illustration of the previous derivations, we consider sequences of color RGB images (the color being represented using the red, green and blue channels) with the goal of computing the motion between two consecutive frames. Here, we have a standard video camera with two interlaced frames with a 20 msec delay between them. We choose to compute



Raw *Anisotropic* (5x5 kernel) *Anisotropic* (7x7 kernel)

Fig. 9. Showing what happens when using anisotropic filtering in the limit case where 90% of noise has been introduced in the image. In this limit case, the use of large kernels is relevant: in the 2nd case the noise is really eliminated in the black parts of the picture, whereas spurious zones remained in the 1st case.

the inter-frame motion, as illustrated in Fig. 10. We do not compute the motion between two consecutive images because, with standard low-cost acquisition devices, the acquisition rate is not fixed (it depends on the computer load) and is usually slow (typically 1-5Hz with more than 10 pixels of disparity between two consecutive images for common mobile objects) yields occlusions, variation in object aspects, etc.. Computing the retinal motion between two frames of a single image, thus with a fixed 20 msec delay, is preferable. However, it requires to work with small quantities thus numerically not very stable, so that the present regularization framework is very useful in this case. This is thus a relevant application of our formalism.

In order to compare our results with other motion estimation methods, we also consider the "Otte" [21] image sequence as shown in Fig. 11, which is of common use for benchmarking such algorithms.

Image gradient of RGB images

At a given image location $\mathbf{m} = (x, y)^T$ with a RGB image intensity $\mathbf{I} = (red, green, blue)^T$, the image contrast and gradient is decomposed using

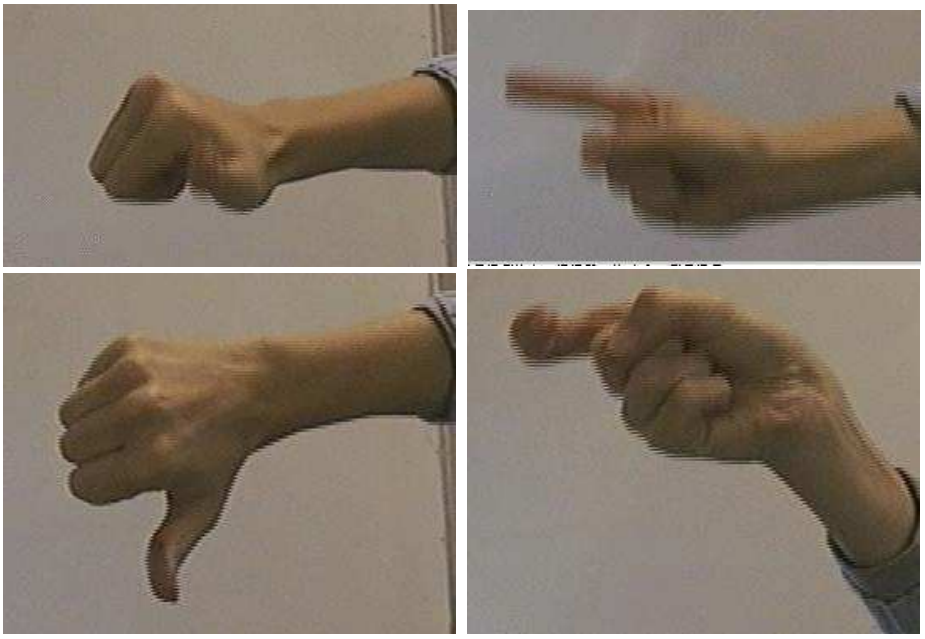


Fig. 10. A few examples of interlaced images with a hand displacement, as used in this experimentation: the inter-frame displacement is visible at the hand boundary.

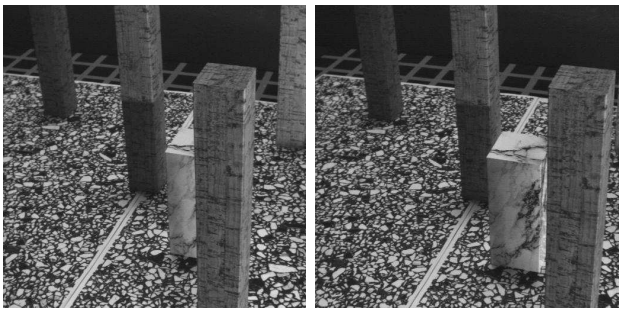


Fig. 11. Two images of the "Otte" image sequence, of common use to benchmark motion estimation algorithms.

the following singular value decomposition:

$$\mathbf{G} = \frac{\partial \mathbf{I}}{\partial \mathbf{m}} = c\,\mathbf{i_n}\,\mathbf{n}^T + c'\,\mathbf{i_t}\,\mathbf{t}^T$$
$$\text{with} \tag{18}$$
$$\mathbf{n} = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \text{ and } \mathbf{t} = \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix}$$

where $c \geq c' \geq 0$ are the singular values, while $\mathbf{i_n} \in \mathbb{R}^3$, $\mathbf{i_t} \in \mathbb{R}^3$ are the orthonormal RGB singular vectors and $\mathbf{n}$ and $\mathbf{t}$ the retinal singular vectors.

The c magnitude corresponds to the highest gradient magnitude, for a

given direction.

More precisely, along an edge, we expect to have only one direction of intensity variation. This means that $c' \simeq 0$ thus $\mathbf{G} \simeq c\,\mathbf{i_n}\,\mathbf{n}^T$ and (i) θ corresponds to the edge orientation in the image, (ii) c corresponds to the edge magnitude and (iii) the unary vector $\mathbf{i_n}$ the edge color orientation in the RGB space.

This is a straightforward generalization of edge orientation and magnitude definitions of monochromatic images.

This definition is interesting because it does not depend upon the color space, providing it is in linear relation with the RGB intensity. More precisely, any linear transformation of the RGB channel, say $\mathbf{I}' = \mathbf{M}\,\mathbf{I}$, yields $\mathbf{G}' = \mathbf{M}\,\mathbf{G}$ and, thanks to the SVD properties, the same orientation θ is calculated, while the edge magnitude and color orientation is now given in the new color space.

Initial motion map estimation

The retinal motion estimation $\mathbf{h} : \mathbb{R}^2 \to \mathbb{R}^2$ associates to a retinal point $\mathbf{x} = (u, v)^T$ a displacement $\mathbf{h}(u, v) = (\dot{u}, \dot{v})^T$. In our implementation, the input $\bar{\mathbf{h}}$ is obtained considering a standard multi-scale correlation operator. The sub-pixelic displacement and the displacement least-square precision (i.e. covariance in a statistical framework) is computed using a 2nd order approximation of the inter-correlation profile (see, in `http://www-sop.inria.fr/od` the `imp.ima.util.Correl` class for more details). This operator provides a relevant initial estimate (see e.g. [16] for a review).

However, this method has several drawbacks. In uniform regions there is no local motion information. Along edges this information is only available in a direction normal to the edge. In spurious regions (e.g. with pseudo-periodic patterns) this estimation may be wrong. The regularization mechanism should help correcting these caveats.

Following the methodology proposed in [20], we refine this initial estimate using the present discrete implementation and considering the linearized diffusion operator proposed in (8). We thus simply choose $D\hat{\mathbf{h}} = c\,\mathbf{n}$ as

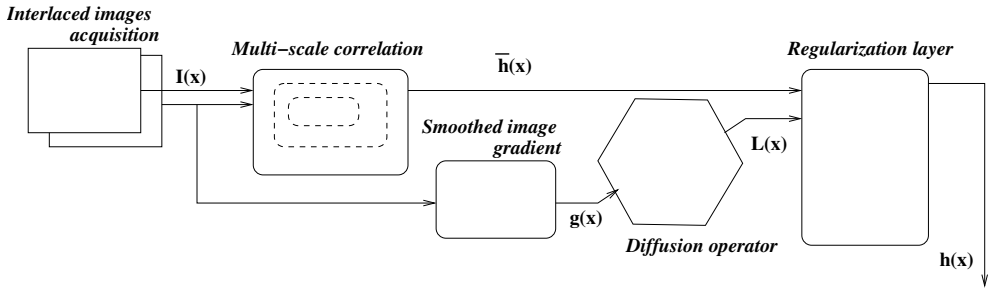


Fig. 12. Schematic description of our implementation (see text for details).

obtain from (18), since we want to diffuse the information in uniform areas but not across edges.

More precisely we choose:

$$\mathbf{L}(\widehat{\mathbf{h}}) = \nu \left(1 - \frac{c^2}{s^2}\right) Id + \nu \left(\frac{c^2}{s^2}\right) \mathbf{n}^{\perp} (\mathbf{n}^{\perp})^T$$

while the input related weight $\mathbf{\Lambda}$ is obtained from the correlation module, which allows to estimate not only the local displacement but also give an indicator of its precision in relation of how "flat" is the correlation profile around its minimum.

A typical experimental results is proposed in Fig. 13 for the "Otte" sequence, where results with isotropic and anisotropic diffusion has been show. Although similar, anisotropic diffusion is qualitatively closer to the ground truth: since the displacement is a pure translation the motion amplitude is higher for closer objects and the focus of expansion corresponds to what was expected [21].

Raw *Isotropic* *Anisotropic*

Fig. 13. A result obtained on the "Otte" sequence. *Raw* is what is obtained by the correlation module; *Isotropic* is, for comparison, what is obtained using an isotropic diffusion; *Anisotropic* is what is obtained by the present implementation.

Similarly, experimental results for interlaced images with a hand displacements are proposed in Fig. 14

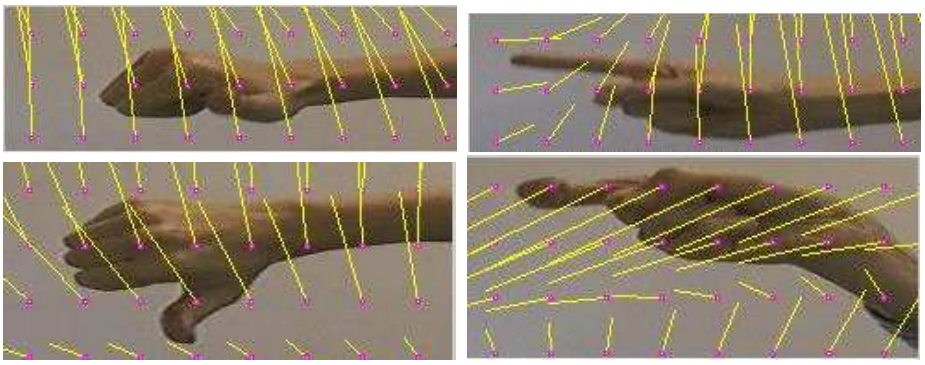


Fig. 14. A few examples of hand displacement estimation using the anisotropic regularized displacement field.

Although qualitative, this small experiment allows to verify the coherence of our derivations.

A step further we have been able to quantify the estimation as follows: since in the Otte sequence it is relatively easy to estimate by hand the displacements for, say, the plane corresponding to the floor (this planar patch in translation generates a quadratic motion field, see e.g. [16]). We thus have been able to compare a few values with what has been obtained by the present estimations. In this case, we obtained a precision of 15% which is the order of magnitude of .. the manual estimation precision.

4.3 Application to early-vision biological or hardwired networks.

As discussed in [4] following [5] another application of this method is the fact we can represent the processing as a biological neural network which implements such visual calculations. In this context, we simply assume that a neuron code a value at a given point $\mathbf{p}_u$ of the cortical map. Values at other points are set to zero, yielding:

$$\sigma^{\epsilon}_{kl}(\mathbf{x},\mathbf{y}) = \textstyle\sum_{uv} \sigma^{uv}_{kl}\, \delta(\mathbf{x}-\mathbf{p}_u, \mathbf{y}-\mathbf{p}_v)$$

where $\mathbf{p}_u$ are the neuron localization in the map, while $\delta(\mathbf{x},\mathbf{y})$ is the vector $2\,n$ Dirac distribution. This sampling model, related to a so called "particle process" [7], assumes that each unit computes a discrete set of map values. It is often used in the absence of a priori geometric knowledge on the unit

distribution, which is the case for neuronal units. Here, equation (5), is now of the form:

$$\frac{\partial \mathbf{h}_t^k}{\partial t}(\mathbf{p}_u) = -\sum_l \left[\mathbf{\Lambda}_l^k \left[\mathbf{h}_t^l - \bar{\mathbf{h}}_t^l \right](\mathbf{p}_u) - \sum_v \sigma_{kl}^{uv} \mathbf{h}_t^l(\mathbf{p}_v) \right] \tag{19}$$

and its implementation has the same architecture and is not more complex than any other neuronal networks, including Hopfield networks. See e.g. [3] for details about the biological plausibility of such linear and multiplicative computational steps.

A step further (see [4] for details), given a cortical map computation parametrized by $\mathbf{\Lambda}$ and $\mathbf{L}$, the implementation of the computation σ is straightforward in this context using a robust Hebbian rule of the form:

$$\sigma_{t+1} = \sigma_t - \gamma$$

for a large class of approximate values of γ. Biological networks can thus easily implement this rule. This kind of linear learning rule is a particular case of Hebbian learning rule (see [10,22] for an experimental discussion and [13] for a theoretical development). More precisely, this derivation is in coherence with [34] where the biologically plausible implementation of nonlinear operation such as minimum computation or comparisons between inputs is detailed.

But there is another issue here: not only biological neural-network but also hardwired implementations of such non-linear mechanism could easily rely on this derivation mechanism to implement real-time versions of such operator, since the hardware design can be as simple as what is proposed in this paragraph.

Let us finally mention that when considering several computation maps, with feed-forward but also feed-backs connections, it is possible to guaranty the stability of the whole set of computation maps [4] under weak conditions, this result being inspired by the cortical maps architecture.

5 Conclusion

An unbiased implementation of differential operators used in the implementation of diffusion processes required in regularization mechanisms has been proposed, using a method similar to [32] for ordinary image derivative operators.

It is designed to derive "vectorial" operators, i.e. to regularize vectorial maps. An interesting extension of the present work would be to consider not only vectorial but quantities with more complex properties such as transformation groups or diffusion tensors [28], e.g. following the track initiated by [15]: representing the parameter space by implicit equations and minimizing the same criterion but with constraints.

Here, the proposed scheme has the following advantage with respect to usual frameworks: it provides a way to "automatically" derived unbiased discrete implementation of the continuous equations. Here "automatic" means a small computer tools which save some time (and some bugs :) when obtaining numerical schemes.

Our work relies on the fact that a pixel/voxel averages the "image" intensity over its surface/volume. This somehow restrictive assumption can be easily generalized if one is able to calibrate the image formation onto the sensor. For instance, generalization to any linear convolution operator should be straightforward.

A step further, we have considered bounded supports for the integral operator (contrary to previous approaches) and introduced the idea of deriving an approximation as closed as possible to the original operator.

References

[1] L. Alvarez and J. Morel. Formalization and computational aspects of image analysis. *Acta Numerica*, pages 1–59, 1994.

[2] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, volume 147 of *Applied Mathematical Sciences*. Springer-Verlag, Jan. 2002.

[3] G. Bugmann. Biologically plausible neural computation. *Biosystems*, 40:11–19, 1997.

[4] J. Bullier, R. Deriche, O. Faugeras, D. Fieze, P. Girard, R. Guyonneau, P. Kornprobst, T. Papadopoulo, S. Thorpe, and T. Vieville. Rivage feedback during visual integration : towards a generic architecture. Technical Report 5451, INRIA, 2004.

[5] G.-H. Cottet and M. E. Ayyadi. A Volterra type model for image processing. *IEEE Transactions on Image Processing*, 7(3), Mar. 1998.

[6] R. Courant. *Calculus of Variations*. New York, 1946.

[7] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. *Mathematics of Computation*, 53(188):485–525, 1989.

[8] R. Deriche and O. Faugeras. Les EDP en Traitement des Images et Vision par Ordinateur. *Traitement du Signal*, 13(6), 1996.

[9] R. Deriche, O. Faugeras, G. Giraudon, T. Papadopoulo, and R. Vaillant. Four applications of differential geometry to computer vision. In G. Orban and H.-H. Nagel, editors, *Artificial and Biological Vision Systems*, Basic Research Series, pages 93–141. Springer–Verlag, 1993.

[10] R. Durbin, C. Miall, and G. Mitchinson, editors. *The computing neuron*. Addison-Wesley, 1989.

[11] L. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. Proceedings of the American Mathematical Society, 1998.

[12] O. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993.

[13] W. Gerstner and W. M. Kistler. Mathematical formulations of hebbian learning. *Biol Cybern*, 87:404–415, 2002.

[14] G. Hermosillo. *Variational Methods for Multimodal Image Matching*. PhD thesis, Universit de Nice-Sophia Antipolis, May 2002.

[15] G. Hermosillo, C. Chefd'hotel, and O. Faugeras. Variational methods for multimodal image matching. *The International Journal of Computer Vision*, 50(3):329–343, Nov. 2002.

[16] T. Huang and A. Netravali. Motion and structure from feature correspondences: A review. *Proc. IEEE*, 82(2):252–268, Feb. 1994.

[17] D. Hubel. *L'oeil, le cerveau et la vision : les tapes crbrales du traitement visuel*. L'univers des sciences. Pour la science, 1994.

[18] P. Kornprobst, R. Peeters, T. Viville, G. Malandain, S. Mierisova, S. Sunaert, O. Faugeras, and P. V. Hecke. Superresolution in MRI and its influence in statistical analysis. Technical Report 4513, INRIA, July 2002.

[19] A. Leonard. Vortex methods for flow simulations. *J. Comput. Phys*, 37:289–335, 1980.

[20] H. Nagel and W. Enkelmann. An investigation of smoothness constraint for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:565–593, 1986.

[21] M. Otte and H. Nagel. Optical flow estimation: Advances and comparisons. In J.-O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision*, volume 800 of *Lecture Notes in Computer Science*, pages 51–70. Springer–Verlag, 1994.

[22] R. Rao and T. J. Sejnowski. Spike-timing-dependent hebbian plasticity as temporal difference learning. *Neural Comput.*, 13(10):2221–2237, 2001.

[23] P. Raviat. An analysis of the particle methods. In F. Brezzi, editor, *Numerical Methods in Fluid Dynamics*, volume 1127 of *Lecture Notes in Math.*, pages 243–324. Springer Verlag, Berlin, 1985.

[24] L. Schwartz. *Thorie des distributions*. Hermann, 1957.

[25] A. Tikhonov. Regularization of incorrectly posed problems. *Soviet. Math. Dokl.*, 4:1624–1627, 1963.

[26] D. Tschumperlé. *PDE's Based Regularization of Multivalued Images and Applications*. PhD thesis, Universit de Nice-Sophia Antipolis, Dec. 2002.

[27] D. Tschumperlé and R. Deriche. Vector-valued image regularization with PDE's : A common framework for different applications. In *IEEE Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin (United States), June 2003.

[28] D. Tschumperl and R. Deriche. Constrained and unconstrained PDE's for vector image restoration. In I. Austvoll, editor, *Proceedings of the 10th Scandinavian Conference on Image Analysis*, pages 153–160, Bergen, Norway, June 2001.

[29] T. Viéville. Biologically plausible regularization mechanisms. In *8th ICCNS*. Boston University, 2004.

[30] T. Viville. Biologically plausible regularization mechanisms. RR 4625, INRIA, 2002.

[31] T. Viville. Towards biologically plausible regularization mechanisms. RR 4965, INRIA, 2003.

[32] T. Viville and O. D. Faugeras. Robust and fast computation of unbiased intensity derivatives in images. In G. Sandini, editor, *Proceedings of the 2nd ECCV*, pages 203–211, Santa-Margherita, Italy, 1992. Springer–Verlag.

[33] T. Viville, D. Lingrand, and F. Gaspard. Implementing a multi-model estimation method. *The International Journal of Computer Vision*, 44(1), 2001.

[34] A. J. Yu, M. Giese, and T. Poggio. Biophysiologically plausible implementations of maximum operation. *Neural Computation*, 14(12), 2003.

Acknowledgments: *Olivier Faugeras* is gratefully acknowledged for some powerful ideas at the origin of this work and *Pierre Kornprobst* for precious explanations and some advice. This work has been realized within the scope of the RIVAGe project.

A Derivation of the diffusion tensor in the nonlinear case

Considering the Euler-Langgarne equation of (7), since:

$$[\Delta_{\Phi,\mathbf{L}}\mathbf{h}]^i = \operatorname{div}\left(\Phi'(\varphi(D\mathbf{h}))D\varphi(D\mathbf{h})\right)) = 2\operatorname{div}\left(\Phi'(\varphi_{\mathbf{L}}(D\mathbf{h}))\sum_j \mathbf{L}_{ij}\, Dh^j\right).$$

using the fact that $\operatorname{div}(\alpha\mathbf{v}) = D\alpha\cdot\mathbf{v} + \alpha\operatorname{div}\mathbf{v}$., the right-hand side is equal to twice

$$\Phi^{''}(\varphi_{\mathbf{L}}(D\mathbf{h}))D\varphi_{\mathbf{L}}(D\mathbf{h})\cdot\sum_j \mathbf{L}_{ij}\, Dh^j + \Phi'(\varphi_{\mathbf{L}}(D\mathbf{h}))\operatorname{div}(\sum_j \mathbf{L}_{ij}\, Dh^j).$$

The term $D\varphi_{\mathbf{L}}(D\mathbf{h})$ is given by

$$D\varphi_{\mathbf{L}}(D\mathbf{h}) = \sum_{i,j} D\left((Dh^j)^T\mathbf{L}_{ij}Dh^i\right),$$

and using the fact that $D(\mathbf{u}\cdot\mathbf{v}) = (D\mathbf{u})\,\mathbf{v} + (D\mathbf{v})\,\mathbf{u}$:

$$D\left((Dh^j)^T\mathbf{L}_{ij}Dh^i\right) = \mathbf{H}_j\mathbf{L}_{ij}Dh^i + \mathbf{L}_{ij}\mathbf{H}_iDh^j + D\mathbf{L}_{ij}(Dh^i,\, Dh^j),$$

where $\mathbf{H}_i$, $\mathbf{H}_j$ are the Hessians of h^i and h^j and $D\mathbf{L}_{ij}$ is the derivative of the matrix $\mathbf{L}_{ij}$. Hence we have

$$D\varphi_{\mathbf{L}}(D\mathbf{h}) = \sum_i \left((\mathbf{H}_i\mathbf{L}_{ii} + \mathbf{L}_{ii}\mathbf{H}_i)Dh^i + D\mathbf{L}_{ii}(Dh^i, Dh^i)\right) +$$

$$\frac{1}{2}\sum_{i,j,i\neq j}(\mathbf{H}_j\mathbf{L}_{ij}+\mathbf{L}_{ij}\mathbf{H}_j)Dh^i+(\mathbf{H}_i\mathbf{L}_{ij}+\mathbf{L}_{ij}\mathbf{H}_i)Dh^j+D\mathbf{L}_{ij}((Dh^i, Dh^j)+(Dh^j, Dh^i))$$

In the case where $\mathbf{L} = Id$ so that $\varphi_{\mathbf{L}}(D\mathbf{h}) = ||D\mathbf{h}||^2$ and $\mathbf{L}_{ij} = \delta_{ij}Id_{n\times n}$, we have

$$D\varphi_{\mathbf{L}}(D\mathbf{h}) = 2\sum_j \mathbf{H}_jDh^j$$

and

$$\frac{1}{2}\left[\Delta_{\Phi,Id}\mathbf{h}\right]_i = \Phi'\left(||D\mathbf{h}||^2\right)\,\Delta h^i + 2\,\Phi''\left(||D\mathbf{h}||^2\right)Dh^i\,\cdot\sum_j \mathbf{H}_jDh^j$$

If we assume further that $\Phi(||D\mathbf{h}||^2) = \sum_j \Phi_j(||Dh^j||^2)$ things simplify even further:

$$[\Delta_{\Phi,Id}\mathbf{h}]^i = \operatorname{div}\left(\Phi_i'(||Dh^i||^2)Dh^i\right)$$

and the resulting equations are *decoupled*:

$$[\Delta_{\Phi,Id}\mathbf{h}]^i = \Phi_i'\left(||Dh^i||^2\right)\,\Delta h^i + 2\,\Phi_i''\left(||Dh^i||^2\right)||Dh^i||^2\,\boldsymbol{\eta}_i^T\,\mathbf{H}_i\,\boldsymbol{\eta}_i\,, \quad \text{(A.1)}$$

where $\boldsymbol{\eta}_i = \frac{Dh^i}{||Dh^i||}$.
Furthermore, in this case all components h^i of $\mathbf{h}$ are decoupled.

B Maple code used for symbolic derivations

This functions allows to automatically derive the kernel to implement given a diffusion operator. Maple standard functions allows automatic code generation (e.g. in C) from this function output.

```
# Define the kernel of the discrete approximation of a diffusion operator,
#  returning a gross equation
sigma := proc(
 M, L, # .. defined by the 1st order coeffs M(u) and 2nd order coeffs L(u,v)
       #    as functions of x,
 x :: vector, y :: vector, # .. returning an operator sigma(x, y),
 r :: integer,             # .. identify up to the r-th order,
 s :: integer              # .. for an hyper-cube [-s .. s]^n is of size s^n,
)
  option remember: local sigma, n, v, i, j, c, z, S, B, X, Y, A, D, U, V:
  n := vectdim(x):

  lprint("# Define the sigma symmetric matrix"):
  X := array(['sigma[v]'$v=1..(2*s+1)^n]):

  lprint("# Calculate the [C1] [C2] and [C0] conditions"):
  c := d ->
    sum('intCube(pow(y, d), y, array(indixes(v, s, n))) *
    (-sigma[v]/(1+linalg[norm](array(indixes(v, s, n)), 2)^2))', v=1..(2*s+1)^n):
  B := array([op(evalf(
  {'M(i) - c(map(dirac, [$1..n], i))'$i=1..n,
   ''2*L(i,j) - c(map((a,b,c)->dirac(a,b)+dirac(a,c), [$1..n], i, j))'$j=i..n'$i=1..n
  } union map(c, `union`(op(map(indexes, {$3..r}, n))))])):
  Y:= array([op(map(proc(e) local t: coeffs(e, indets(e), t): t end,
{'M(i)'$i=1..n,''L(i,j)'$j=i..n'$i=1..n}) minus {1} )]):

  lprint("# Calculate the SVD and its pseudo-inverse"):
  A := linalg[jacobian](B, X):
  B := evalm(B - A &* X):
  lprint("svd ..");
  D := convert(evalf(Svd(A, U, V)), list):
  lprint("  done");
  D := matrix(coldim(A), rowdim(A),
   (i,j) -> if (i = j) and (D[i] > D[1] / 10000) then 1 / D[i] else 0 fi):
  B := convert(evalm(V &* D &* transpose(U) &* B), vector):
  z := proc(u)
   if type(u, algebraic) and not type(u, {constant,name}) then map(procname, u)
   elif type(u, constant) and (abs(u) < 1e-6) then 0
   else u fi
  end:
  B := map(z, array(evalf(
    map((v,B,s,n) -> B[v]/(1+linalg[norm](array(indixes(v, s, n)), 2)^2),
```

```
    [$1..(2*s+1)^n], op(B), s,n))):
  S := factor(z(sum(B[i],i = 1 .. (2*s+1)^n))):
  if type(S, constant) then z := S: S := 1 fi:
  if type(S, `+`) and
   nops(map(u-> if type(u, `*`) then op(1,u) else u fi, convert(S,set))) = 1 then
   if type(op(1,S), `*`) then z := op(1,op(1,S)) else z := op(1,S) fi: S := S / z
  fi:
  if type(z, constant) then
    B := array(map((v,B,z,s,n)->B[v]/z, [$1..(2*s+1)^n], op(B), z,s,n)):
  fi:

  lprint("# Format the output"):
  if vectdim(Y) = 0 then
    [op(X), `=`, op(B), `/`, S]
  else
    c := (e, v) -> coeff(subs(v=DUMMY_VARIABLE,e),DUMMY_VARIABLE):
    A := matrix(vectdim(X), vectdim(Y), (i, j) -> c(B[i], Y[j])):
    if convert(evalm(B - A &* Y), set) <> {0} then ERROR(SpuriousForm) fi:
    [op(X), `=`, op(A), `&*`, op(Y), `/`, S]
  fi
end:
```

The following macros have been used for the symbolic derivations proposed in this paper.

```
# Return the extremum of a quadratic criterion c with linear constraints ctr
leastsquare := proc(c, ctr :: set, vars :: set)
  local l, v, r:
  l := {'cat(_Z_,i)'$i=1..nops(ctr)}: v := vars union l:
  r := solve(
   convert(linalg[grad](
    c + sum('cat(_Z_,i)'*ctr[i],i=1..nops(ctr)), convert(v, list)), set),v):
  if r <> NULL then map((u, l) -> if not member(op(1,u), l) then u fi, r, l) fi
 end:

# Calculate the power of vector x with respect to multi-indices
pow := (x :: vector, d :: list(integer)) ->
 convert(map((i, x, d) -> x[i]^d[i], [$1..min(vectdim(x),nops(d))], x, d), `*`):

# Integrate an expression e over a canonic hyper-cube
intBox := proc(e, x :: vector)
 option remember: local r, i:
 r := e: for i to vectdim(x) do r := int(r, x[i]=-1/2..1/2) od
end:

# Return the set of multi-indexes d = [d_1 .. d_n] with r = sum(d_i, i=1..n)
indexes := proc(r :: integer, n :: integer)
  option remember:
  if r = 0 then
   {[0$i=1..n]}
  else
   map((d, n) -> op(
    map((i, n, d) ->
     map((j, i, d) ->
      if i = j then d[j] + 1 else d[j] fi,
     [$1..n], i, d),
    {$1..n}, n, d)),
   {op(procname(r-1, n))}, n)
  fi
end:

# Define the kronecker symbol
dirac := (a, b) -> if a = b then 1 else 0 fi:
```