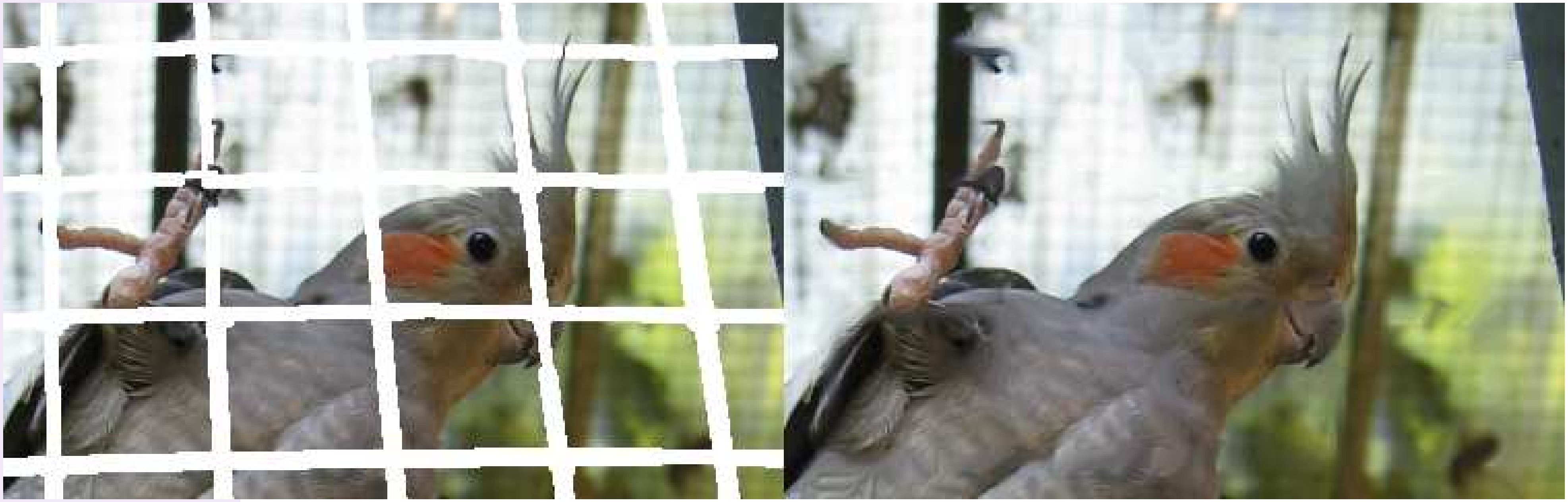
Tensor-Directed Smoothing of Multi-Valued Images with Curvature-Preserving Diffusion PDE's



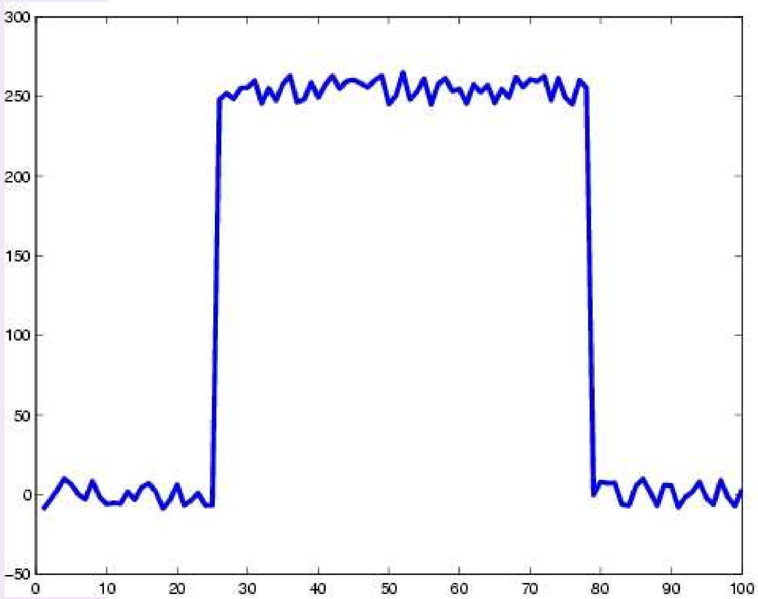
David Tschumperlé

(CNRS UMR 6072 (GREYC/ENSICAEN) - Image Team)

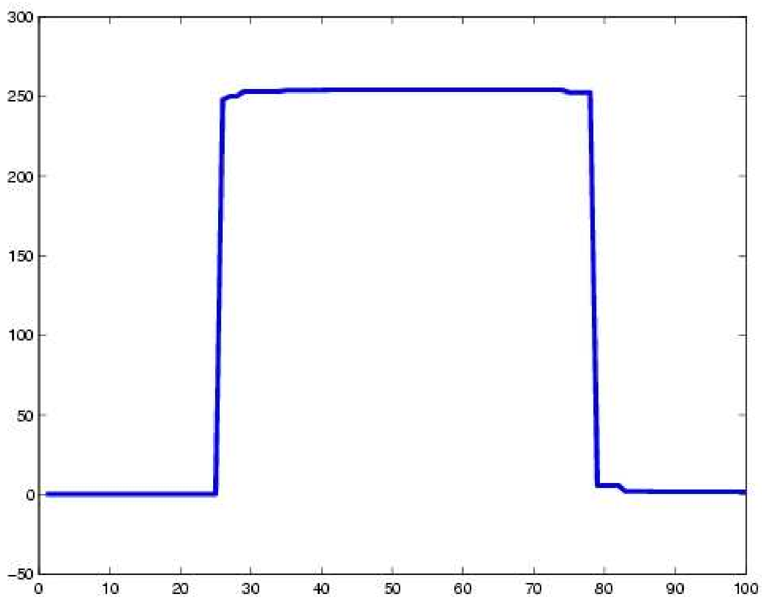
ICISP'2008 - Cherbourg/France - July 2008

Context : Data Regularization

- **Goal :** Transform a noisy signal into a more regular signal, while preserving the important signal features (discontinuities).



1D Noisy Signal



Regularized Signal

⇒ Do the same thing for 2D images.

- **Applications** : Denoising, Data Simplification, Multi-Scale Analysis, Solving ill-posed inverse problems.

What is a “good regularization” process ? (1)

- A “good” regularization process adapts itself to the considered data type as well as to the targeted application. A “best regularization method” does not exist.

Original color image

Regularization 1 (Tikhonov)

Regularization 2 (Total Variation)

Regularization 3 (Tensor-directed)

What is a “good regularization” process ? (2)

Original color image

Regularization 1 (Tikhonov)

Regularization 2 (Total Variation)

Regularization 3 (Tensor-directed)

⇒ Methods based on **non-linear PDE’s** are able to design **flexible** and **customizable** regularization processes.

PDE's formulation

- PDE = Partial Differential Equation $\rightarrow$ Evolution Equation.

- We start from an image $I_{(t=0)}$ which evolves until convergence, or until a finite number of iterations ($t = t_{\text{end}}$) $\Longrightarrow$ Iterative algorithm.

$$\begin{cases} I_{(t=0)} = I_0 \\ \frac{\partial I}{\partial t}_{(x,y)} = \beta^t_{(x,y)} \end{cases}$$

implemented as

$$\begin{cases} \mathtt{I}^{(\mathtt{t}=0)} = \mathtt{I}_0 \\ \mathtt{repeat} \quad \mathtt{I}^{\mathtt{t+dt}}_{(\mathtt{x,y})} = \mathtt{I}^{\mathtt{t}}_{(\mathtt{x,y})} + \mathtt{dt}\ \beta^{\mathtt{t}}_{(\mathtt{x,y})} \\ \mathtt{until} \quad \mathtt{t} < \mathtt{t}_{\mathtt{end}} \end{cases}$$

(for instance, $\beta^t_{(x,y)} = \Delta I_{(x,y)} = \frac{\partial^2 I}{\partial x^2}_{(x,y)} + \frac{\partial^2 I}{\partial y^2}_{(x,y)}$).

- PDE = Partial Differential Equation $\rightarrow$ Evolution Equation.

- We start from an image $I_{(t=0)}$ which evolves until convergence, or until a finite number of iterations ($t = t_{\text{end}}$) $\Longrightarrow$ Iterative algorithm.

$$\begin{cases} I_{(t=0)} = I_0 \\ \frac{\partial I}{\partial t}_{(x,y)} = \beta^t_{(x,y)} \end{cases} \quad \text{implemented as} \quad \begin{cases} \mathtt{I}^{(\mathtt{t}=0)} = \mathtt{I}_0 \\ \mathtt{repeat} \quad \mathtt{I}^{\mathtt{t+dt}}_{(\mathtt{x,y})} = \mathtt{I}^{\mathtt{t}}_{(\mathtt{x,y})} + \mathtt{dt}\ \beta^{\mathtt{t}}_{(\mathtt{x,y})} \\ \mathtt{until} \quad \mathtt{t} < \mathtt{t}_{\mathtt{end}} \end{cases}$$

(for instance, $\beta^t_{(x,y)} = \Delta I_{(x,y)} = \frac{\partial^2 I}{\partial x^2}_{(x,y)} + \frac{\partial^2 I}{\partial y^2}_{(x,y)}$).

- The evolution speed β^t gives the kind of processing done on the data.

- β^t may be obtained via the Euler-Lagrange Equations (gradient descent that minimizes an energy functional), or can be designed more "manually".

- Convolution and Isotropic Diffusion PDE (Koenderink:84, Alvarez-Guichard-etal:92, ...) :

$$I_{(t)} = I_{(t=0)} * G_\sigma \quad \text{where} \quad G_\sigma = \frac{1}{4\pi t}\, e^{-\frac{x^2+y^2}{4t}} \quad \Longleftrightarrow \quad \frac{\partial I}{\partial t} = \Delta I = \mathrm{div}\left(\nabla I\right)$$

- Convolution and Isotropic Diffusion PDE (Koenderink:84, Alvarez-Guichard-etal:92, ...) :

$$I_{(t)} = I_{(t=0)} * G_\sigma \quad \text{where} \quad G_\sigma = \frac{1}{4\pi t}\, e^{-\frac{x^2+y^2}{4t}} \quad \Longleftrightarrow \quad \frac{\partial I}{\partial t} = \Delta I = \operatorname{div}\left(\nabla I\right)$$

- Anisotropic Diffusion PDE's (nonlinear) (Perona-Malik[90], Alvarez [92], ...) :

$$\frac{\partial I}{\partial t} = \operatorname{div}\left(c(\|\nabla I\|)\ \nabla I\right) \qquad \text{with} \ \ c : \mathbb{R} \longrightarrow \mathbb{R}$$

Diffusion PDE's and Image Regularization

- Convolution and Isotropic Diffusion PDE (Koenderink:84, Alvarez-Guichard-etal:92, ...) :

$$I_{(t)} = I_{(t=0)} * G_\sigma \quad \text{where} \quad G_\sigma = \frac{1}{4\pi t}\, e^{-\frac{x^2+y^2}{4t}} \quad \Longleftrightarrow \quad \frac{\partial I}{\partial t} = \Delta I = \mathrm{div}\,(\nabla I)$$

- Anisotropic Diffusion PDE's (nonlinear) (Perona-Malik[90], Alvarez [92], ...) :

$$\frac{\partial I}{\partial t} = \mathrm{div}\,(c(\|\nabla I\|)\ \nabla I) \qquad \text{with} \quad c : \mathbb{R} \longrightarrow \mathbb{R}$$

Noisy Image

Heat Flow ($\frac{\partial I}{\partial t} = \Delta I$)

Perona-Malik ($\frac{\partial I}{\partial t} = \mathrm{div}\,(c_{(.)}\ \nabla I)$)

How to find the best $\beta^t_{(x,y)}$?

- More generally, how to find the "best" possible evolution speed $\beta^t_{(x,y)}$, i.e. the more general and flexible one ?

⇒ 3 principal ways proposed in the literature.

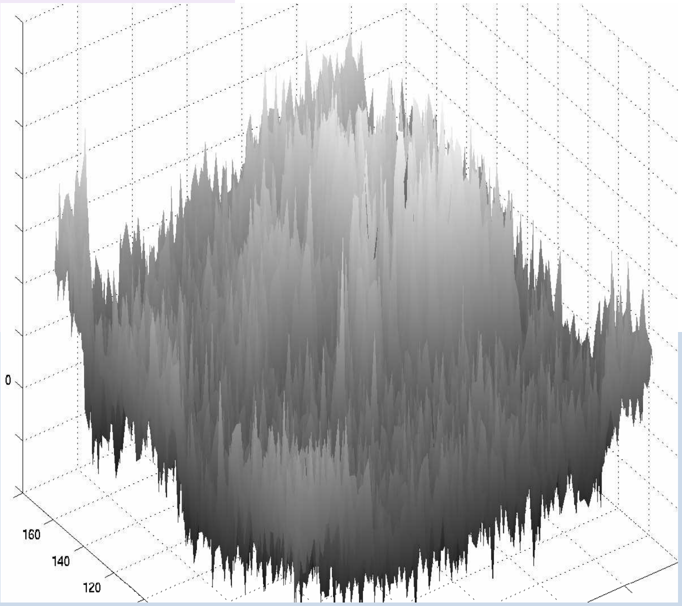
(Alvarez, Aubert, Barlaud, Blanc-Feraud, Blomgren, Charbonnier, Chan, Cohen, Deriche, Kornprobst, Kimmel, Malladi, Munford, Morel, Nordström, Osher, Perona, Malik, Rudin, Sapiro, Sochen, Weickert,...)

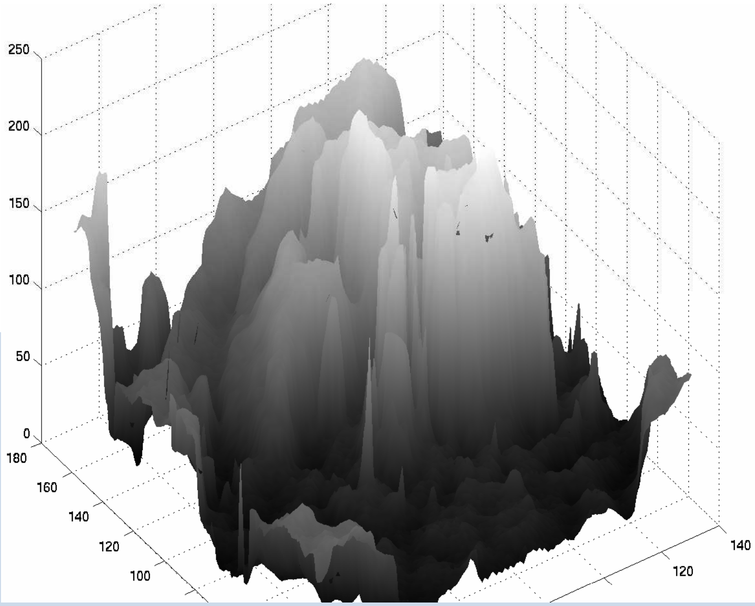
(1) Image Regularization as an Energy Minimization

- Minimizing image variations, expressed as an energy functional $E(I)$:

$$\min_{\mathbf{I}:\Omega\to\mathbb{R}} E(I) = \int_\Omega \phi(\|\nabla I\|)\, d\Omega \quad \overset{\text{(E.L)}}{\Longrightarrow} \quad \frac{\partial I}{\partial t} = \operatorname{div}\left(\frac{\phi'(\|\nabla I\|)}{\|\nabla I\|} \nabla I \right)$$

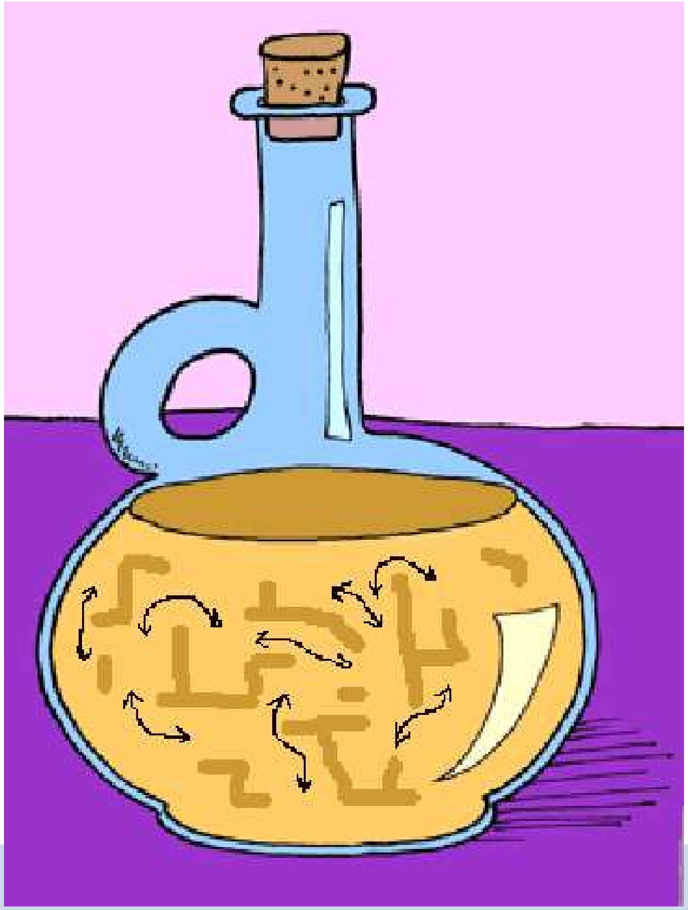
- $E(I)$ can be seen as a global energy depending on a global property of the image (for instance : the area of the image, seen as a surface, $\phi(s) = 1/\sqrt{1+s^2}$)
  $\Rightarrow$ Global Approach.





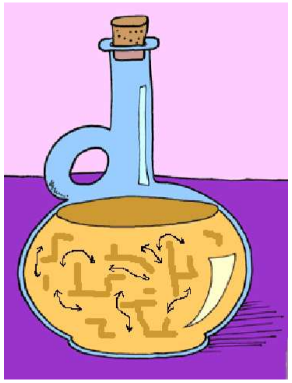
(2) Image Regularization as Pixel Diffusion

- Pixel values are seen as chemical concentrations or temperatures.



(2) Image Regularization as Pixel Diffusion

- Pixel values are seen as chemical concentrations or temperatures.



- Diffusion PDE's modeling a chemical or heat transfer between pixels :

$$\frac{\partial I}{\partial t}_{(x,y)} = \mathrm{div}\left(c_{(x,y)} \nabla I_{(x,y)}\right) \quad \text{or} \quad \frac{\partial I}{\partial t}_{(x,y)} = \mathrm{div}\left(\mathbf{D}_{(x,y)} \nabla I_{(x,y)}\right)$$

- The diffusivity $c_{(x,y)}$ or the diffusion tensor $\mathbf{D}_{(x,y)}$ locally characterize the diffusion process. They often depend on local geometric features of the image (gradients ∇I, edges, corners, etc.), for instance $c = \exp(-\frac{1}{K}\,\|\nabla I\|^2)$ (Perona-Malik).

$\Rightarrow$ Local Approach.

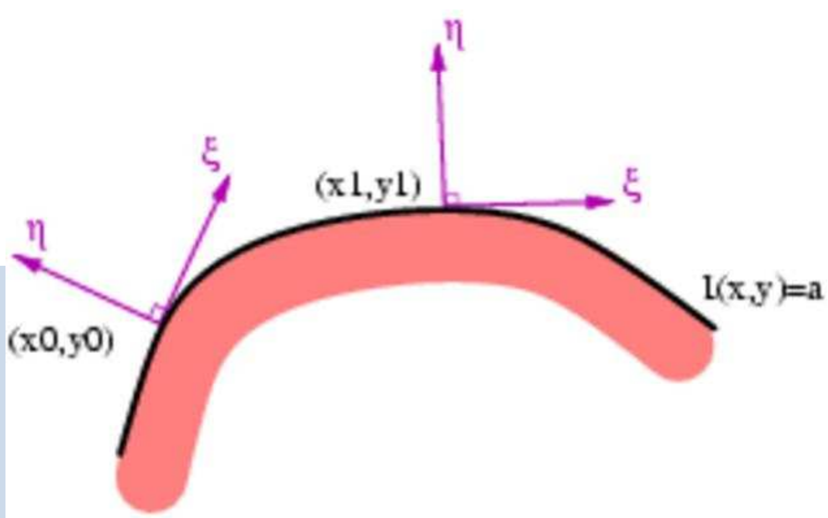
(3) Image Regularization as Oriented 1D Laplacians

- Two simultaneous 1D heat flows, oriented in orthogonal directions $\xi_{(x,y)}$ and $\eta_{(x,y)}$, and weighted by two coefficients $c_{1(x,y)}$ and $c_{2(x,y)} > 0$:

$$\frac{\partial I}{\partial t} = c_1 \frac{\partial^2 I}{\partial \xi^2} + c_2 \frac{\partial^2 I}{\partial \eta^2} \quad \text{where} \quad \eta = \frac{\nabla I}{\|\nabla I\|} \quad \text{and} \quad \xi = \eta^{\perp}$$

- Anisotropic filtering is then done in spatially varying directions.

⇒ Local approach.



Link between these three approaches

- From the global approach to the more local one :

Functional minimization

$$\min_{I:\Omega\rightarrow\mathbb{R}} E(I) = \int_\Omega \phi(\|\nabla I\|)\, d\Omega$$

Divergence expression

$$\frac{\partial I}{\partial t} = \operatorname{div}\left(\frac{\phi'(\|\nabla I\|)}{\|\nabla I\|}\nabla I\right) = \operatorname{div}(c\nabla I)$$

Oriented laplacians

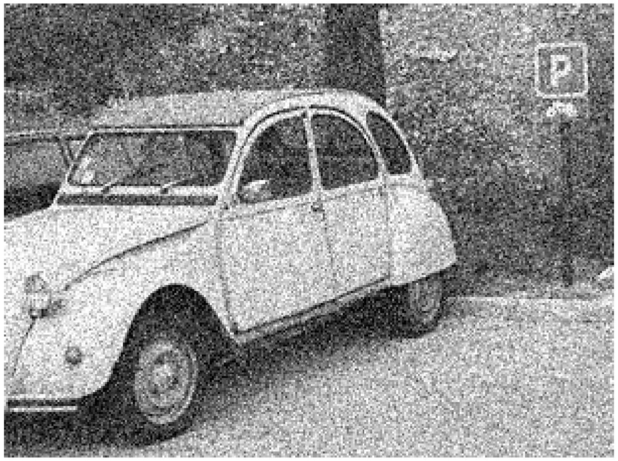
$$\frac{\partial I}{\partial t} = \frac{\phi'(\|\nabla I\|)}{\|\nabla I\|} I_{\xi\xi} + \phi''(\|\nabla I\|)\, I_{\eta\eta}$$
$$= c_1 \frac{\partial^2 I}{\partial \xi^2} + c_2 \frac{\partial^2 I}{\partial \eta^2}$$

- Flexibility : Choosing different ϕ, c, c_1, c_2 leads to different regularization behaviors.

$\Rightarrow$ Oriented Laplacians are the most “flexible” approach, from a local point of view.

Illustration of the oriented Laplacians flexibility

- All results below have been obtained with the Oriented Laplacian PDE, stopped after 20 iterations, using the same time step dt, and $\eta = \nabla I / \|\nabla I\|$.



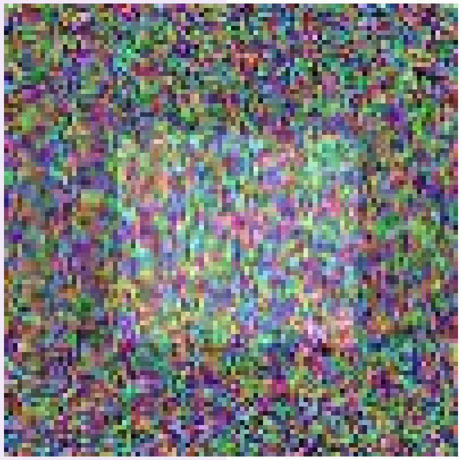
Original image $I_{(t=0)}$

Using $c_1 = \frac{1}{1+\|\nabla I\|}$ and $c_2 = \frac{1}{1+\|\nabla I\|^2}$

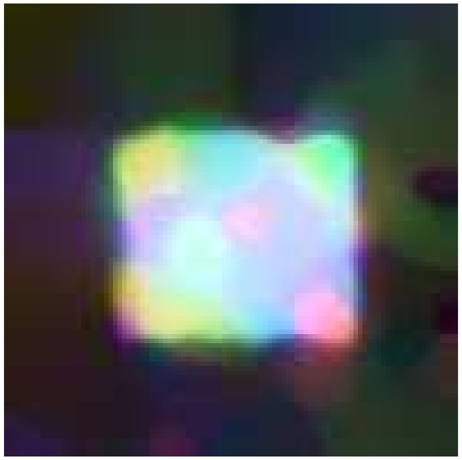
Using $c_1 = c_2 = 1$

Using $c_1 = 1$ and $c_2 = 0$

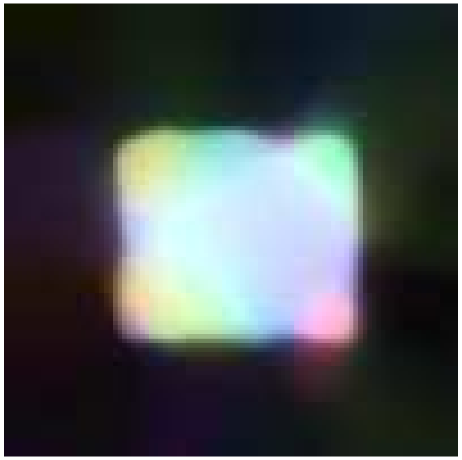
- Image $\mathbf{I} : \Omega \rightarrow \mathcal{N}$ of multi-valued points : vectors ($\mathcal{N} = \mathbb{R}^n$), matrices ($\mathcal{N} = \mathcal{M}_n$).



Color image ($\mathcal{N} = \mathbb{R}^3$)

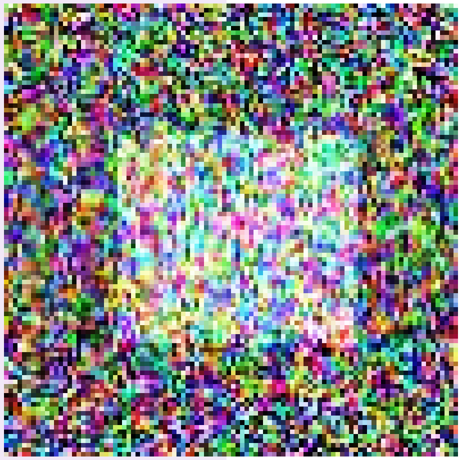


Scalar PDE's applied on each channel

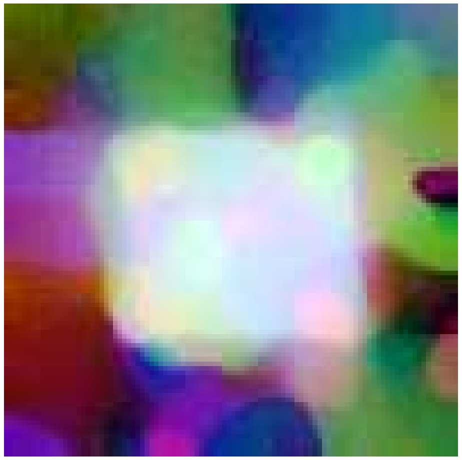


Multi-valued PDE's

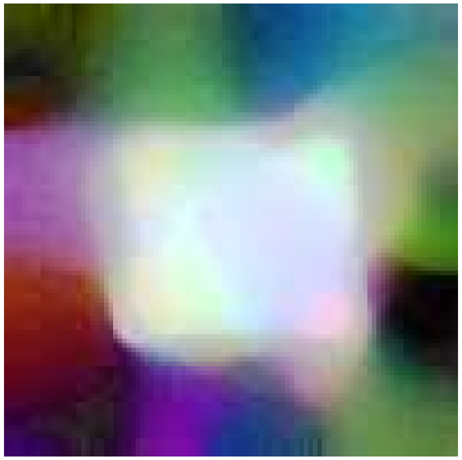
- Image $\mathbf{I} : \Omega \rightarrow \mathcal{N}$ of multi-valued points : vectors ($\mathcal{N} = \mathbb{R}^n$), matrices ($\mathcal{N} = \mathcal{M}_n$).



Color image ($\mathcal{N} = \mathbb{R}^3$)



Scalar PDE's applied on each channel

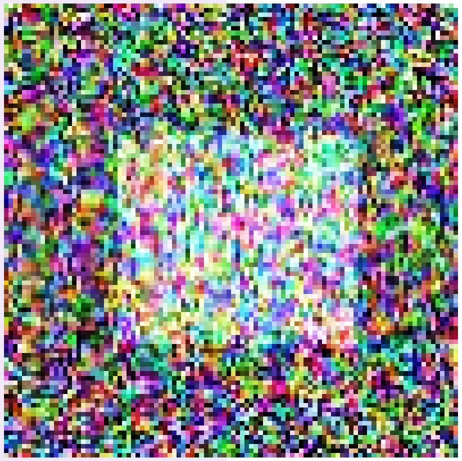


Multi-valued PDE's

(Histogram equalized)

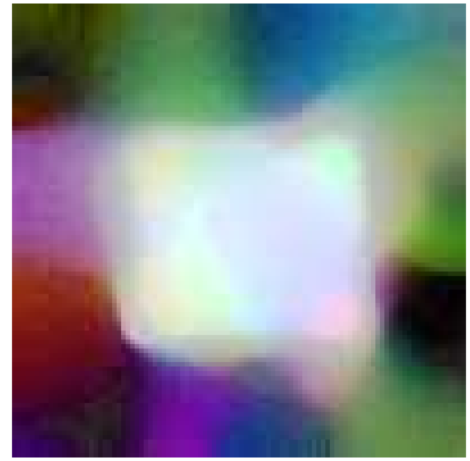
Regularization PDE's and Multi-Valued Images

- Image $\mathbf{I} : \Omega \to \mathcal{N}$ of multi-valued points : vectors ($\mathcal{N} = \mathbb{R}^n$), matrices ($\mathcal{N} = \mathcal{M}_n$).



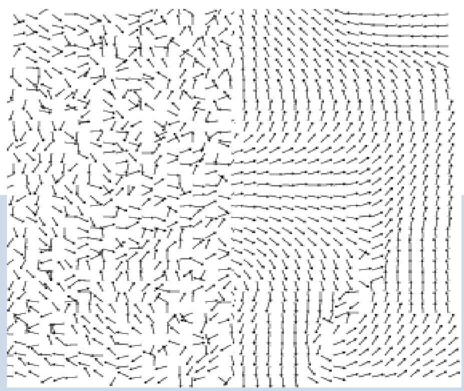
Color image ($\mathcal{N} = \mathbb{R}^3$)

Scalar PDE's applied on each channel



Multi-valued PDE's

Color image



Direction field (+ constraint)

Tensor field (+ constraint)

How to Extend Scalar PDE's to the Multi-Valued Case ?

- How to correctly extend scalar diffusion PDE's to the multi-valued case, without applying them channel by channel ?

⇒ **Introducing 2nd-order Diffusion Tensors and Structure Tensors.**

Introducing Diffusion Tensors

- A second-order tensor is a symmetric and semi-positive definite $p \times p$ matrix. ($p = 2$ for images, $p = 3$ for volumetric images).

- It has p positive eigenvalues λ_i and p orthogonal eigenvectors $\mathbf{u}^{[i]}$:

$$\mathbf{T} = \lambda_1\, \mathbf{u}^{[1]}{\mathbf{u}^{[1]}}^T + \lambda_2\, \mathbf{u}^{[2]}{\mathbf{u}^{[2]}}^T$$

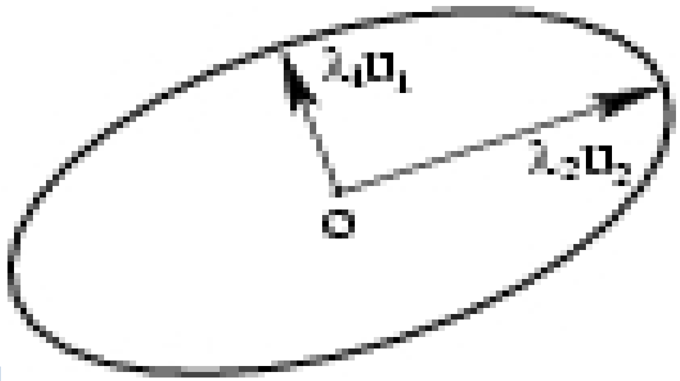
Introducing Diffusion Tensors

- A second-order tensor is a symmetric and semi-positive definite $p \times p$ matrix.
  ($p = 2$ for images, $p = 3$ for volumetric images).

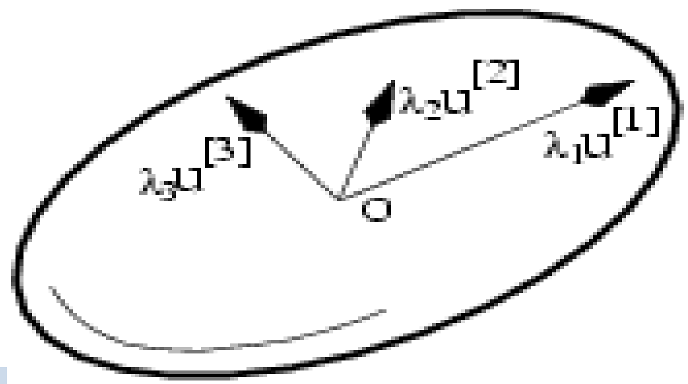
- It has p positive eigenvalues λ_i and p orthogonal eigenvectors $\mathbf{u}^{[i]}$:

$$\mathbf{T} = \lambda_1\, \mathbf{u}^{[1]}\mathbf{u}^{[1]^T} + \lambda_2\, \mathbf{u}^{[2]}\mathbf{u}^{[2]^T}$$

- Representation using ellipses and ellipsoïds :



2×2 Tensor



3×3 Tensor

- Tensors can describe a smoothing process, by telling how much the pixel values diffuse along given orthogonal orientations, i.e. the "geometry" of the smoothing.

Writting Diffusion PDE's using Diffusion Tensors

- Divergence-based diffusion PDE's : (Weickert:98)

$$\frac{\partial I}{\partial t} = \text{div}\left(\mathbf{D}\nabla I\right) \qquad \text{(simple scalar diffusivity when } \mathbf{D}_{(x,y)} = c_{(x,y)}\,\mathbf{Id}\text{)}$$

where $\mathbf{D}$ is a field of diffusion tensors.

Writting Diffusion PDE's using Diffusion Tensors

- Divergence-based diffusion PDE's : (Weickert:98)

$$\frac{\partial I}{\partial t} = \text{div}\left(\mathbf{D}\nabla I\right) \qquad \text{(simple scalar diffusivity when } \mathbf{D}_{(x,y)} = c_{(x,y)}\,\mathbf{Id}\text{)}$$

where $\mathbf{D}$ is a field of diffusion tensors.

- Oriented Laplacians : (Tschumperle-Deriche:02)

$$\frac{\partial I}{\partial t} = c_1 \frac{\partial^2 I}{\partial \xi^2} + c_2 \frac{\partial^2 I}{\partial \eta^2} = \text{trace}\left(\mathbf{TH}\right)$$

where $\mathbf{T} = c_1\, \xi\xi^T + c_2\, \eta\eta^T$ is the diffusion Tensor with eigenvalues c_1, c_2 and eigenvectors ξ, η, and $\mathbf{H}$ is the Hessian matrix : $\mathbf{H}_{i,j} = \frac{\partial^2 I}{\partial x_i \partial x_j}$.

- Divergence-based diffusion PDE's : (Weickert:98)

$$\frac{\partial I}{\partial t} = \text{div}\left(\mathbf{D}\nabla I\right) \qquad \text{(simple scalar diffusivity when } \mathbf{D}_{(x,y)} = c_{(x,y)}\ \mathbf{Id}\ )$$

where $\mathbf{D}$ is a field of diffusion tensors.

- Oriented Laplacians : (Tschumperle-Deriche:02)

$$\frac{\partial I}{\partial t} = c_1 \frac{\partial^2 I}{\partial \xi^2} + c_2 \frac{\partial^2 I}{\partial \eta^2} = \text{trace}\left(\mathbf{TH}\right)$$

where $\mathbf{T} = c_1\, \xi\xi^T + c_2\, \eta\eta^T$ is the diffusion tensor with eigenvalues c_1, c_2 and eigenvectors ξ, η, and $\mathbf{H}$ is the Hessian matrix : $\mathbf{H}_{i,j} = \frac{\partial^2 I}{\partial x_i \partial x_j}$.

⇒ Fields of Diffusion Tensors can define complex (anisotropic) local regularization.

⇒ Separation of the regularization geometry from the diffusion process itself.

What would be "Good" Diffusion Tensors ?

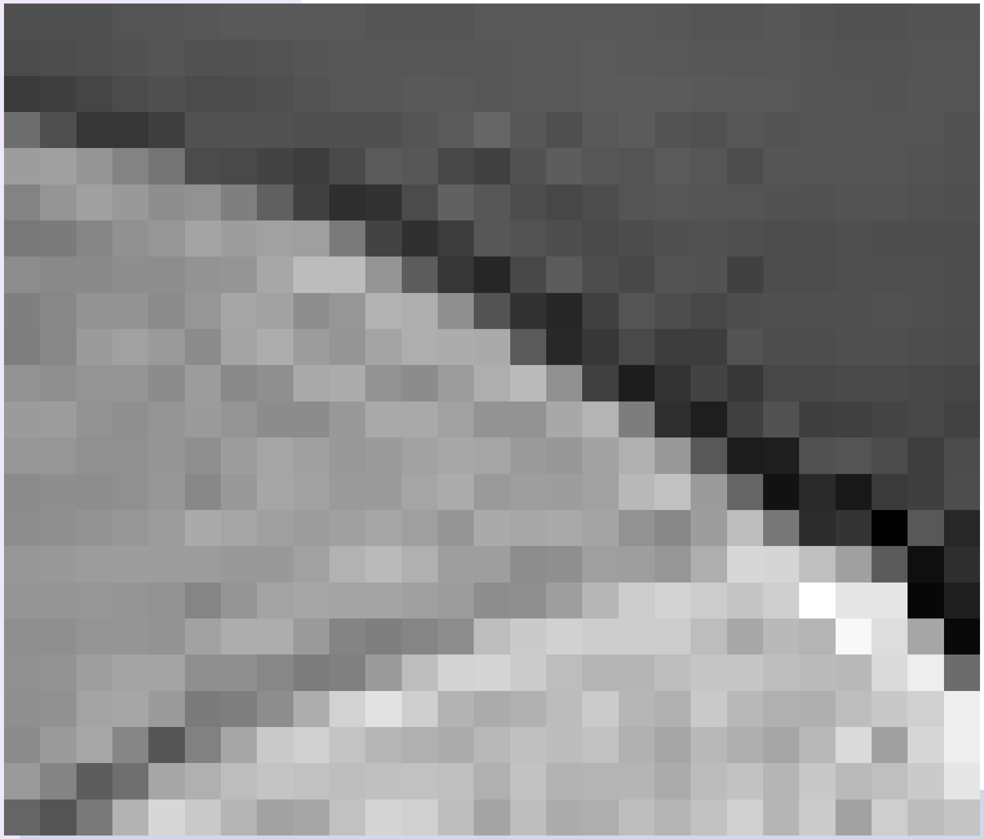
- What is the desired behavior for a regularization algorithm ?

⇒ **Depends on the application !** Common "good" smoothing rules are :

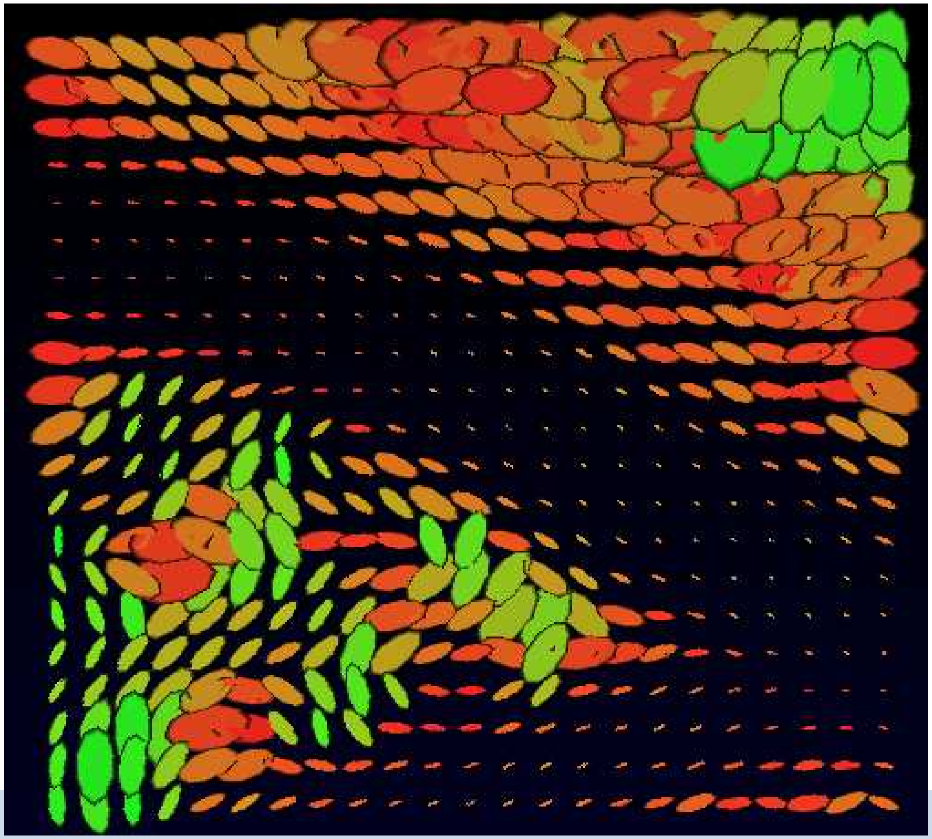
- On a edge, smoothing must be done only along the edge direction *(anisotropic smoothing)* : $\Longrightarrow \quad \mathbf{D}_{(x,y)} \approx \epsilon\, \xi\xi^T$, with $\xi = \frac{\nabla I^{\perp}}{\|\nabla I\|}$.
- On homogeneous regions, smoothing must be done equally in all directions *(isotropic smoothing)* : $\Longrightarrow \quad \mathbf{D}_{(x,y)} \approx \alpha\, \mathbf{Id}$

Modeling Regularization Behavior with Diffusion Tensors

⇒ Tensor field $\mathbf{D} : \Omega \rightarrow \mathrm{P}(2)$ should tell about the desired smoothing directions and smoothing amplitudes that must be locally applied.



Top of the Lena hat



Desired diffusion tensor field $\mathbf{D}$

Designing Diffusion Tensors for Multi-Valued Images

- Goal : Estimate the local geometry of $\mathbf{I} : \Omega \rightarrow \mathbb{R}^n$, a multi-valued image. Can be done by computing the smoothed Structure Tensor Field $\mathbf{G}_\sigma : \Omega \rightarrow \mathrm{P}(2)$:

$$\mathbf{G}_{\sigma_{(x,y)}} = \left(\sum_i \nabla I_i \nabla I_i^T \right) * G_\sigma$$

- Sum of channel by channel structure tensors $\nabla I_i \nabla I_i^T$. Take care of all image variations at the same time, with a notion of incertitude.

$\Rightarrow$ Very nice extension of the notion of “gradient” for multi-valued images.

(Silvano Di-Zenzo:86, Joachim Weickert:98).

Using Structure Tensors in Local Formulations (1)

- When considering local regularization approaches, the diffusion tensor field can be designed directly from the structure tensor $\mathbf{G}_\sigma$:

$$\mathbf{T} = f_1(\lambda_+ + \lambda_-)\ \theta_-\theta_-^T + f_2(\lambda_+ + \lambda_-)\ \theta_+\theta_+^T \quad \text{with} \quad \begin{cases} f_1(s) & = & \frac{1}{1+s^p} \\ f_2(s) & = & \frac{1}{1+s^q} \end{cases}$$

Using Structure Tensors in Local Formulations (2)

- When considering local regularization approaches, the diffusion tensor field can be designed directly from the structure tensor $\mathbf{G}_\sigma$:

$$\mathbf{T} = f_1(\lambda_+ + \lambda_-)\ \theta_-\theta_-^T + f_2(\lambda_+ + \lambda_-)\ \theta_+\theta_+^T \quad \text{with} \quad \begin{cases} f_1(s) & = & \frac{1}{1+s^p} \\ f_2(s) & = & \frac{1}{1+s^q} \end{cases}$$

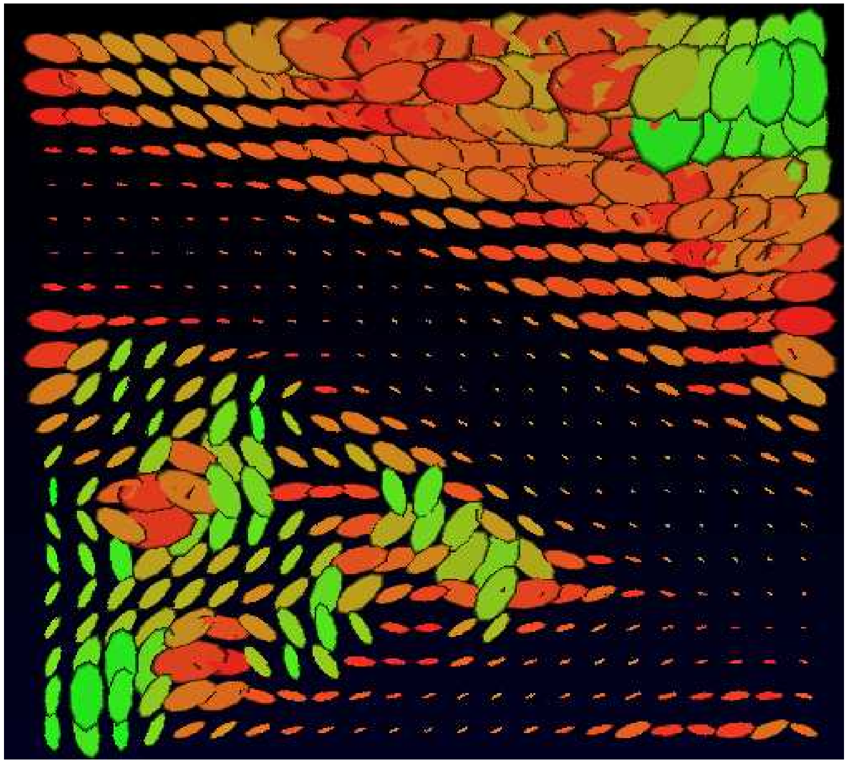
- The smoothing itself is performed by the application of one or several iterations of one of these “locally designed” PDE’s :

$$\frac{\partial I_i}{\partial t} = \operatorname{div}\left(\mathbf{T}\nabla I_i\right) \qquad \mathit{or} \qquad \frac{\partial I_i}{\partial t} = \operatorname{trace}\left(\mathbf{T}\mathbf{H}_i\right)$$

$\Rightarrow$ Most of existing PDE-based regularization methods for multi-valued images fit one of these two equations.

Obtained Diffusion Tensor Field

Top of the Lena hat ($\mathbf{I} : \Omega \rightarrow \mathbb{R}^3$)



Computed diffusion tensor field $\mathbf{T} : \Omega \rightarrow \mathrm{P}(2)$.

- We obtained the desired flexibility in designing different regularization behaviors, while considering all image channels at the same time.

⇒ **So, everything's is OK ?**

Application : Color image restoration

- Color image with real noise (digital snapshot under low luminosity conditions).

Noisy color image

Restored color image

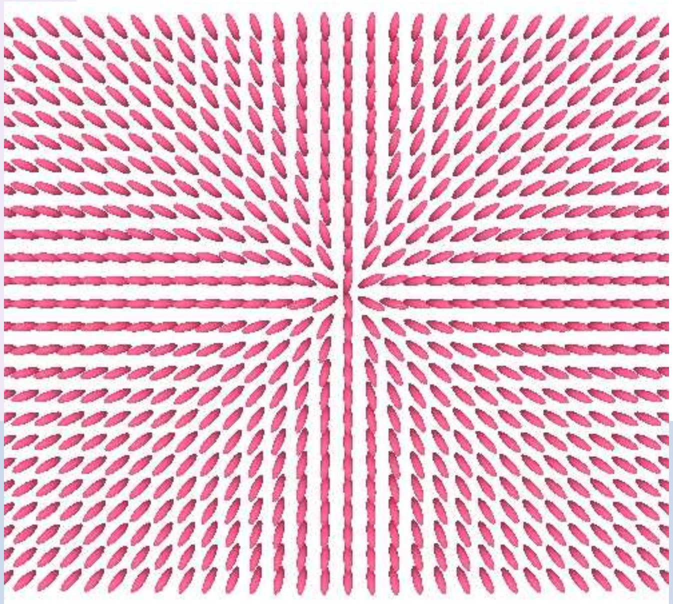
⇒ The geometry of the image has been clearly respected.

But... Is the Smoothing Correctly Achieved ?

- We apply some iterations of one of these generic PDE's, with a synthetic tensor field T on a color image.

$$\frac{\partial I_i}{\partial t} = \mathrm{div}\left(\mathbf{T}\nabla I_i\right) \qquad or \qquad \frac{\partial I_i}{\partial t} = \mathrm{trace}\left(\mathbf{T}\mathbf{H}_i\right)$$

- Ideally, the performed smoothing complies with the diffusion tensor field $\mathbf{T}$:




Tensor-directed PDE applied on a color image.

- **Slow iterative process** : Many iterations needed to get a result that is regularized enough (since $dt \to 0$).

- **Problems with Divergence formulations :**

  - Non-unicity of the tensor field : $\exists \mathbf{D}_1 \neq \mathbf{D}_2, \quad \text{div}(\mathbf{D}_1 \nabla I) = \text{div}(\mathbf{D}_2 \nabla I)$.
  - Tensor shapes not always representative of the intuitive smoothing behavior :

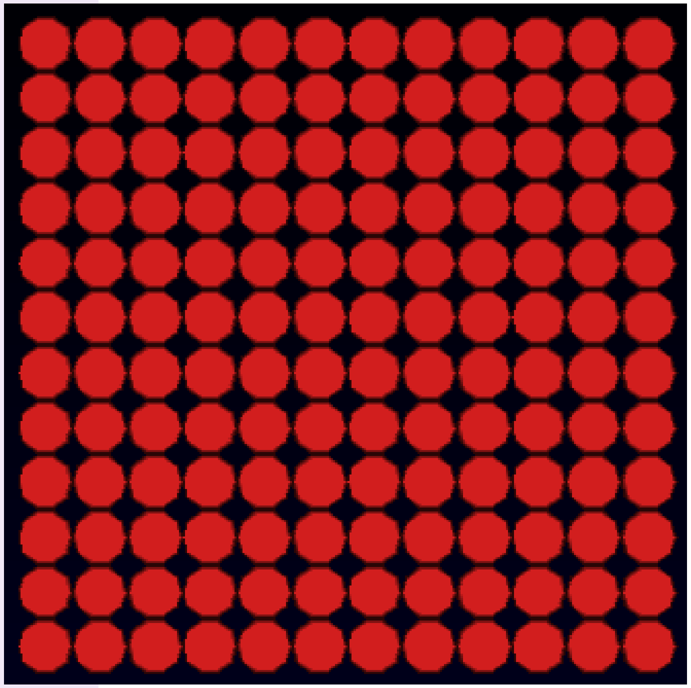
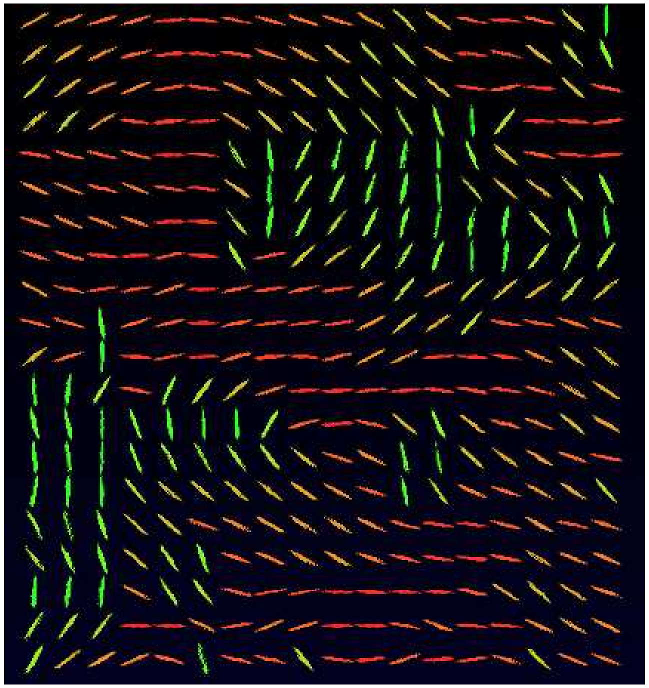
$$\mathbf{D}_1 = \mathbf{Id} \quad \text{and} \quad \mathbf{D}_2 = \frac{\nabla I \nabla I^T}{\|\nabla I\|^2} \qquad \Rightarrow \qquad \frac{\partial I}{\partial t} = \Delta I.$$

  - More generally :

$$\mathbf{D}_1 = \alpha \xi \xi^T + \beta \eta \eta^T \quad \text{and} \quad \mathbf{D}_2 = \beta \eta \eta^T \quad \Rightarrow \quad \text{div}\left(\mathbf{D}_1 \nabla I\right) = \text{div}\left(\mathbf{D}_2 \nabla I\right)$$

with $\eta = \frac{\nabla I}{\|\nabla I\|}$ and $\xi = \eta^{\perp}$.

Non-unicity of Diffusion Tensors

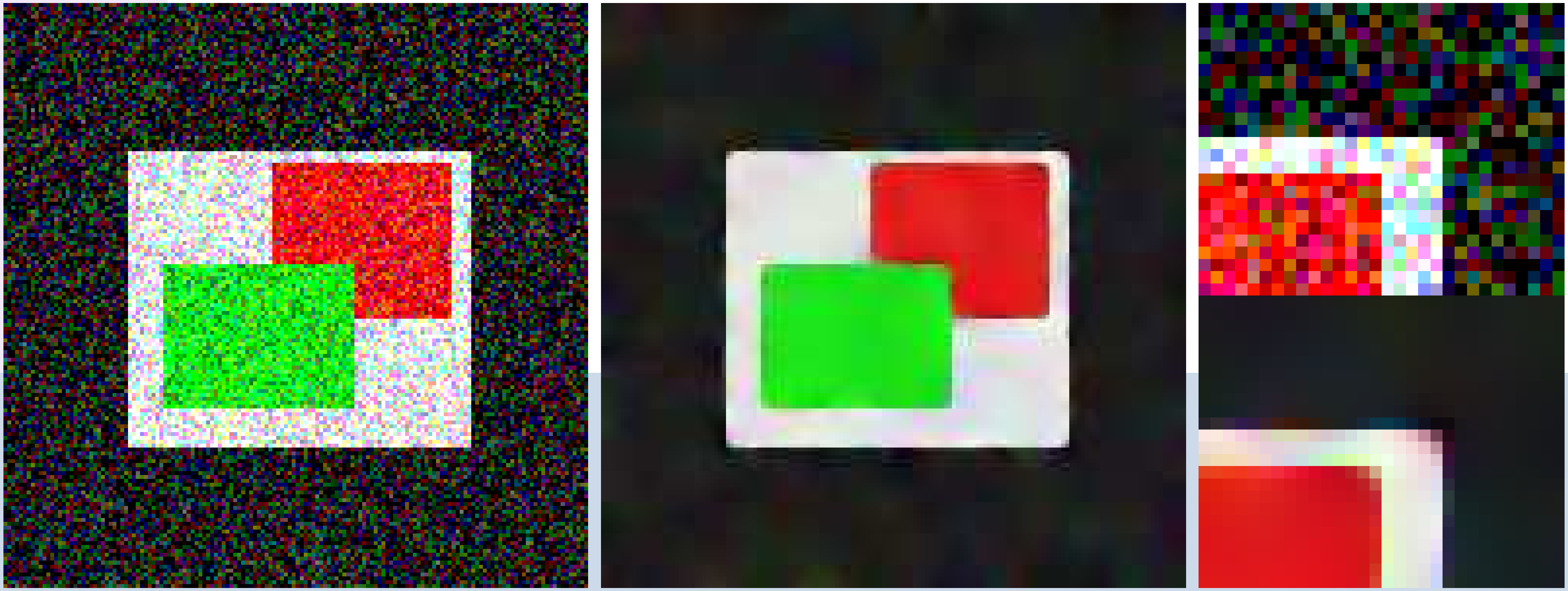
$\mathbf{D}_1 =$  and $\mathbf{D}_2 =$ 

gives the same result  (heat flow)

Issues Encountered with Classical Formulations

- **Problems with Trace formulations :**
  - Better respect of the considered tensor-valued geometry.
  - But tends to over-smooth high-curvature structures (corners) :

$$\frac{\partial I_i}{\partial t} \approx \alpha \frac{\partial^2 I}{\partial \xi^2} \quad \text{on image countours} \quad \Rightarrow \quad \text{Problems at corners !}$$

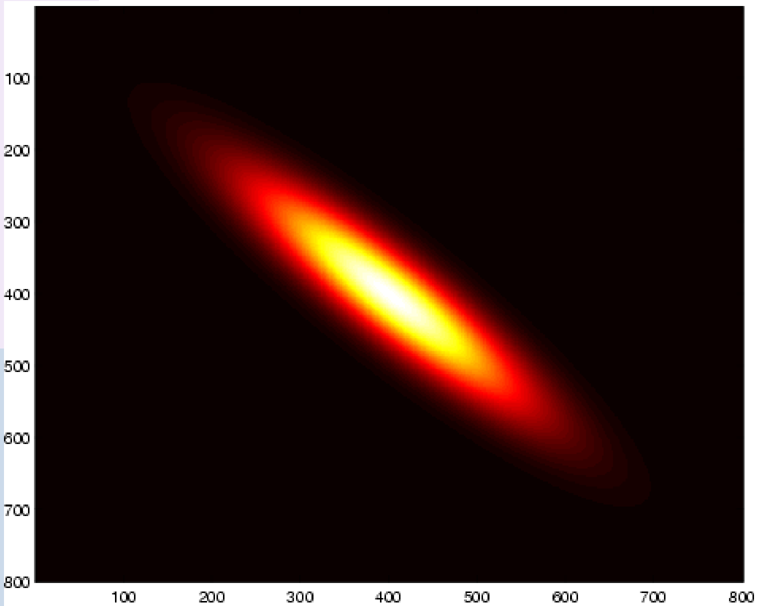


A Geometrical Interpretation of $\mathrm{trace}\,(\mathbf{TH})$

$$\frac{\partial I_i}{\partial t} = \mathrm{trace}\,(\mathbf{TH}_i)$$

- If $\mathbf{T}$ is a constant tensor, the solution at time t is a convolution of the image $\mathbf{I}$ by an oriented Gaussian kernel $\mathbf{G}^{[\mathbf{T},t]}$:

$$I_{i_{(t)}} = I_{i_{(t=0)}} * G^{[\mathbf{T},t]} \qquad \text{with} \qquad G^{[\mathbf{T},t]}(x,y) = \frac{1}{4\pi t}\, e^{-\frac{\mathbf{x}^T \mathbf{T}^{-1} \mathbf{x}}{4t}}$$

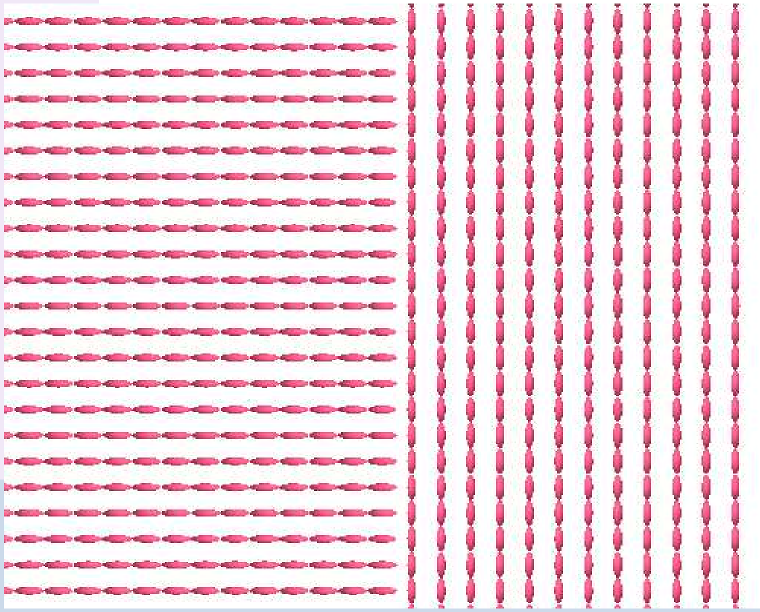


A Geometrical Interpretation of $\mathrm{trace}\,(\mathbf{TH})$

$$\frac{\partial I_i}{\partial t} = \mathrm{trace}\,(\mathbf{TH}_i)$$

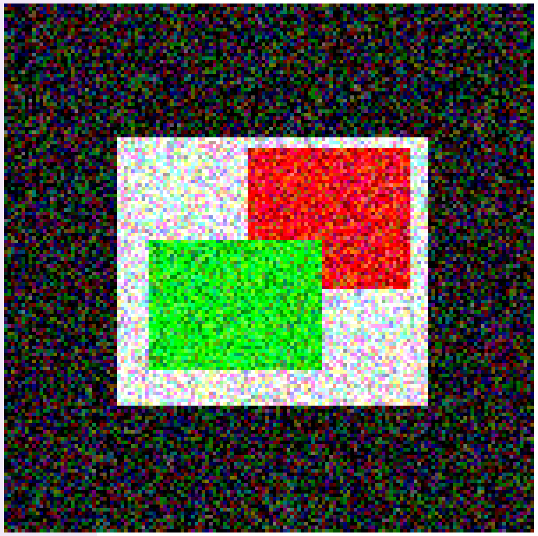
- If $\mathbf{T}$ is a non-constant tensor field : Geometrical Interpretation in terms of local filtering, using gaussian kernels that are temporally and spatially varying.

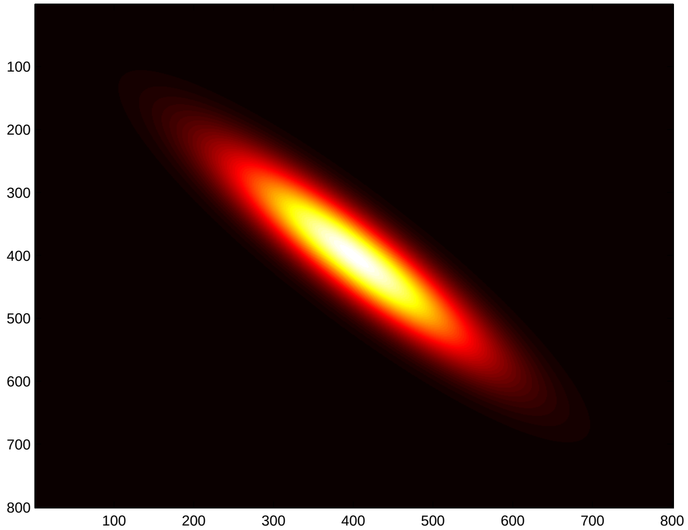
(See also 'Short Time Kernels' by Sochen-Kimmel-etal:01).

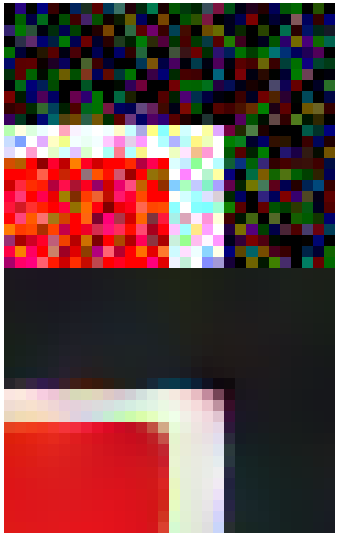


Issues encountered with the trace formulation

- On curved image structures, the structure tensor is often not so well directed.
- Even with a small smoothing, rounded corners appear after several iterations.





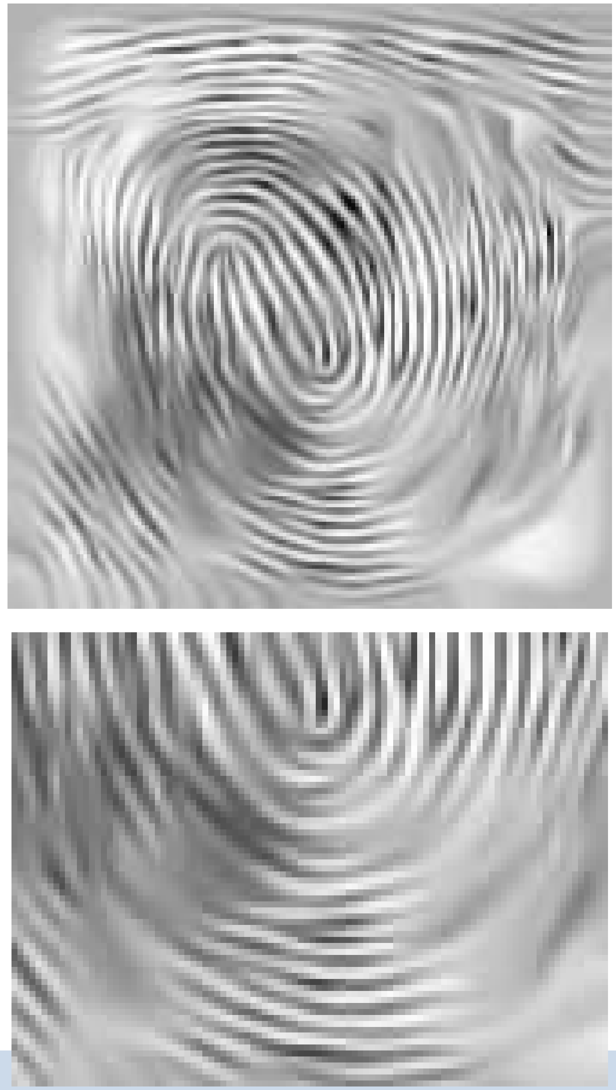


⇒ Needs for specific PDE's avoiding smoothing of structures having high curvatures.

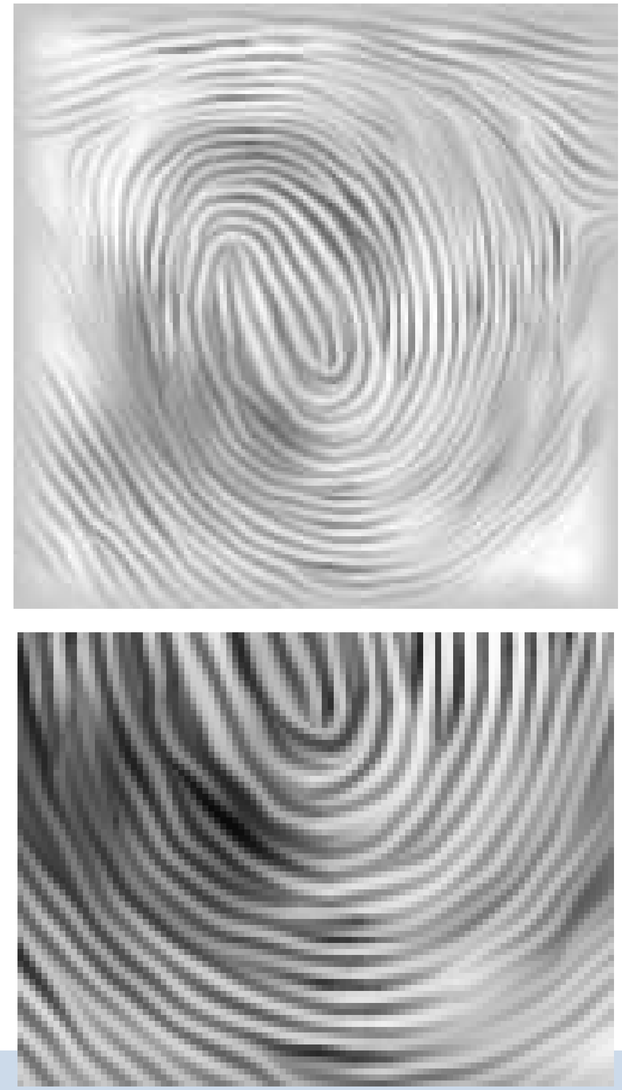
- We want to avoid an explicit curvature computation (perturbed by the noise).

Motivations

Original image



Trace-based PDE (200 iter.)



Curvature-Preserving (200 iter.)

Curvature-preserving constraint

- For the mono-directional case, let us consider the following PDE :

$$\frac{\partial I_i}{\partial t} = \text{trace}\left(\mathbf{w}\mathbf{w}^T\ \mathbf{H}_i\right) + \nabla I_i^T \mathbf{J}_{\mathbf{w}}\mathbf{w}$$

where $\mathbf{J}_{\mathbf{w}} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$ and $\mathbf{H}_i = \begin{pmatrix} \frac{\partial^2 I_i}{\partial x^2} & \frac{\partial^2 I_i}{\partial x \partial y} \\ \frac{\partial^2 I_i}{\partial x \partial y} & \frac{\partial^2 I_i}{\partial y^2} \end{pmatrix}$.

$\Rightarrow$ Classical “Trace” formulation oriented along $\mathbf{w}$

\+ Constraint term depending on the variations of $\mathbf{w}$.

Interpretation of the Constraint Term

- Ths PDE can be written in fact as :

$$\frac{\partial I_i}{\partial t} = \frac{\partial^2 I_i(\mathcal{C}^{\mathbf{X}}_{(a)})}{\partial a^2}_{|a=0} = \Delta^{\mathbf{X}}_{\mathcal{C}} I_i$$

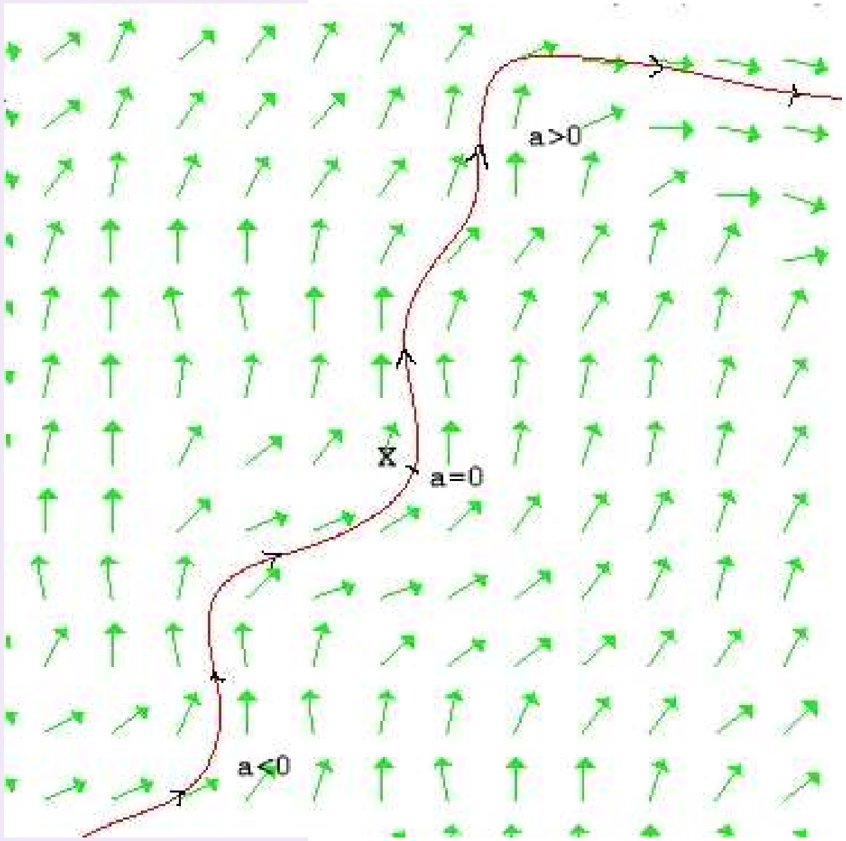
where $\mathcal{C}^{\mathbf{X}}$ is the integral line of $\mathbf{w}$ starting from $\mathbf{X}$, and parameterized as :

$$\mathcal{C}^{\mathbf{X}}_{(0)} = \mathbf{X} \quad \text{and} \quad \frac{\partial \mathcal{C}^{\mathbf{X}}_{(a)}}{\partial a} = \mathbf{w}(\mathcal{C}^{\mathbf{X}}_{(a)})$$

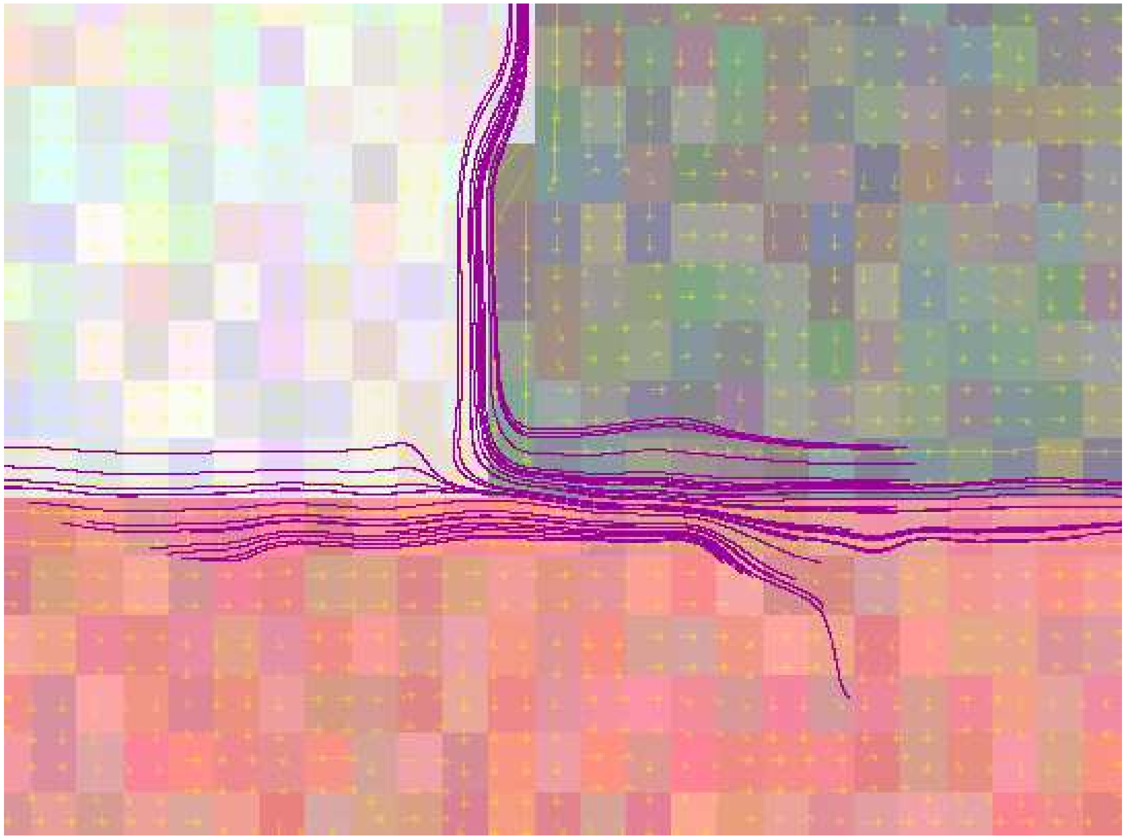
$\Rightarrow$ PDE equivalent to a heat flow on the integral lines of $\mathbf{w}$.

- If $\mathbf{w}$ is chosen to be the directions of the image contours (eigenvector θ_- of $\mathbf{G}_\sigma$), the smoothing will respect the shape of the contour, whatever its curvature is.

Smoothing Along Integral Lines



(a) An integral line $\mathcal{C}^{\mathrm{X}}$



(b) Some integral lines around a triple-junction.

⇒ The performed smoothing will preserve curved structures.

Extension to a Tensor-Based Geometry

- More generaly, we are more interested to a tensor-valued smoothing geometry $\mathbf{T}$ than a vectorial one $\mathbf{w}$.

- We decompose the field $\mathbf{T}$ along all orientations of the plane :

$$\mathbf{T} = \frac{2}{\pi} \int_{\alpha=0}^{\pi} (\sqrt{\mathbf{T}}\, a_\alpha)\, (\sqrt{\mathbf{T}}\, a_\alpha)^T \, d\alpha \quad \text{where } a_\alpha = \left(\begin{array}{cc} \cos\alpha & \sin\alpha \end{array} \right)^T .$$

Extension to a Tensor-Based Geometry

- More generaly, we are more interested to a tensor-valued smoothing geometry $\mathbf{T}$ than a vectorial one $\mathbf{w}$.

- We decompose the field $\mathbf{T}$ along all orientations of the plane :

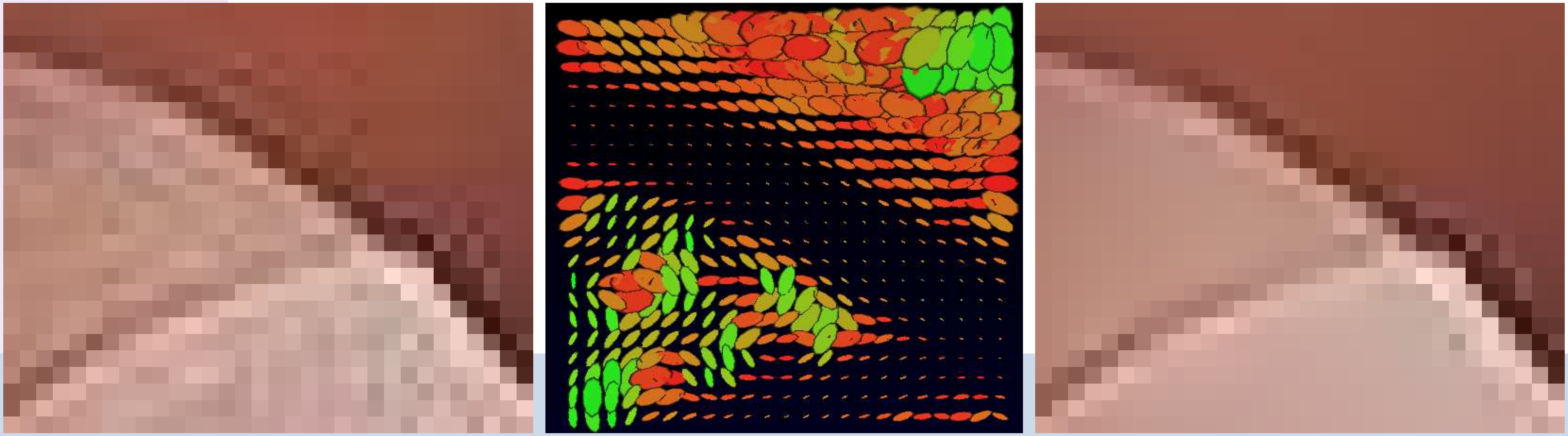
$$\mathbf{T} = \frac{2}{\pi} \int_{\alpha=0}^{\pi} (\sqrt{\mathbf{T}}\, a_\alpha)\, (\sqrt{\mathbf{T}}\, a_\alpha)^T \, d\alpha \quad \text{where } a_\alpha = \left(\begin{array}{cc} \cos\alpha & \sin\alpha \end{array} \right)^T.$$

- This suggests to extend naturally the monodirectional formulation to this tensor-directed one :

$$\frac{\partial I_i}{\partial t} = \text{trace}(\mathbf{T}\mathbf{H}_i) + \frac{2}{\pi} \nabla I_i^T \int_{\alpha=0}^{\pi} \mathbf{J}_{\sqrt{\mathbf{T}} a_\alpha} \sqrt{\mathbf{T}} a_\alpha \, d\alpha$$

Extension to a Tensor-Based Geometry

- Local behavior of the equation :
  - When the tensor $\mathbf{T}$ is isotropic, we are on an homogeneous region : the smoothing is performed with the same strength in all directions a_α.
  - When the tensor $\mathbf{T}$ is anisotropic, we are on an image contour : the smoothing is performed only along this contour (but taking care of its curvature !).

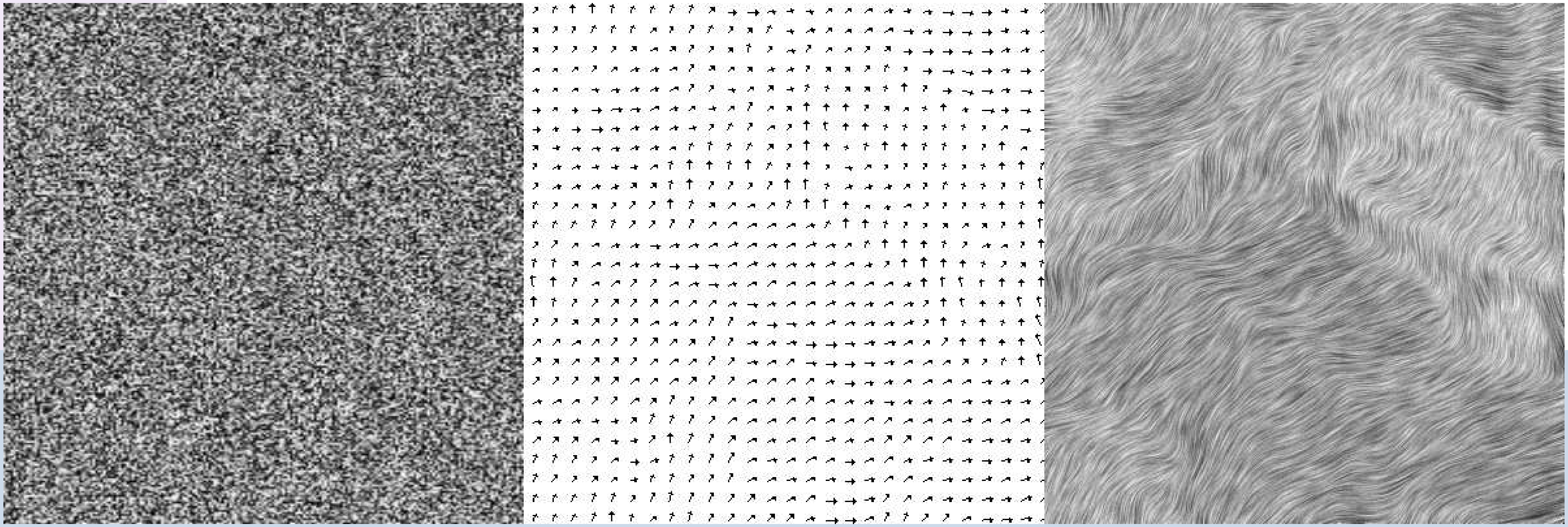


Line Integral Convolutions (LIC's)

- [Cabral & Leedom, 93] : Way to create textured versions of 2D vector fields $\mathcal{F}$.

$\Rightarrow$ From a pure noisy image $\mathbf{I}^{\text{noise}}$, one computes for each pixel $\mathbf{X} = (x, y)$

$$\mathbf{I}^{LIC}_{(x,y)} = \frac{1}{N} \int_{-\infty}^{+\infty} f(p)\, \mathbf{I}^{noise}(\mathcal{C}^{\mathbf{X}}_{(p)})\, dp \qquad \text{where} \qquad \begin{cases} \mathcal{C}^{\mathbf{X}}_{(0)} & = & \mathbf{X} \\ \frac{\partial \mathcal{C}^{\mathbf{X}}_{(a)}}{\partial a} & = & \mathcal{F}(\mathcal{C}^{\mathbf{X}}_{(a)}) \end{cases}$$



- $\frac{\partial I_i}{\partial t} = \text{trace}\left(\mathbf{w}\mathbf{w}^T\ \mathbf{H}_i\right) + \nabla I_i^T \mathbf{J}_\mathbf{w}\mathbf{w}$ can be seen as a $1D$ heat flow on the integral line $\mathcal{C}^\mathbf{X}$.

$\Rightarrow$ Implementation can be done by convolving the data lying on the integral line $\mathcal{C}^\mathbf{X}$ of $\mathbf{w}$ by a Gaussian kernel.

- $\frac{\partial I_i}{\partial t} = \text{trace}\left(\mathbf{w}\mathbf{w}^T\ \mathbf{H}_i\right) + \nabla I_i^T \mathbf{J}_{\mathbf{w}}\mathbf{w}$ can be seen as a $1D$ heat flow on the integral line $\mathcal{C}^{\mathbf{X}}$.

⇒ Implementation can be done by convolving the data lying on the integral line $\mathcal{C}^{\mathbf{X}}$ of $\mathbf{w}$ by a Gaussian kernel.

- Tensor version : $\frac{\partial I_i}{\partial t} = \text{trace}(\mathbf{T}\mathbf{H}_i) + \frac{2}{\pi}\nabla I_i^T \int_{\alpha=0}^{\pi} \mathbf{J}_{\sqrt{\mathbf{T}}a_\alpha}\sqrt{\mathbf{T}}a_\alpha\ d\alpha$ can be implemented with several short LIC computations.

$$\mathbf{I}_{(\mathbf{X})}^{regul} = \frac{1}{N}\int_0^{\pi}\int_{-dt}^{dt} f(a)\ \mathbf{I}^{noisy}(\mathcal{C}_{(\mathbf{X},a)}^{\theta})\ da\ d\theta$$

where $f()$ is a 1D Gaussian function, $N = \int\int f(a)dad\theta$, and dt corresponds to the PDE time step (global smoothing strength for one iteration).

Algorithm Properties

⇒ The maximum principle is verified (only local means of pixel intensities are computed).

Algorithm Properties

⇒ The maximum principle is verified (only local means of pixel intensities are computed).

⇒ Very stable and fast algorithm, compared to classical PDE implementations. The time step (dt) can be very large ($\simeq 50$) while process remains stable.

Algorithm Properties

⇒ The maximum principle is verified (only local means of pixel intensities are computed).

⇒ Very stable and fast algorithm, compared to classical PDE implementations. The time step (dt) can be very large ($\simeq 50$) while process remains stable.

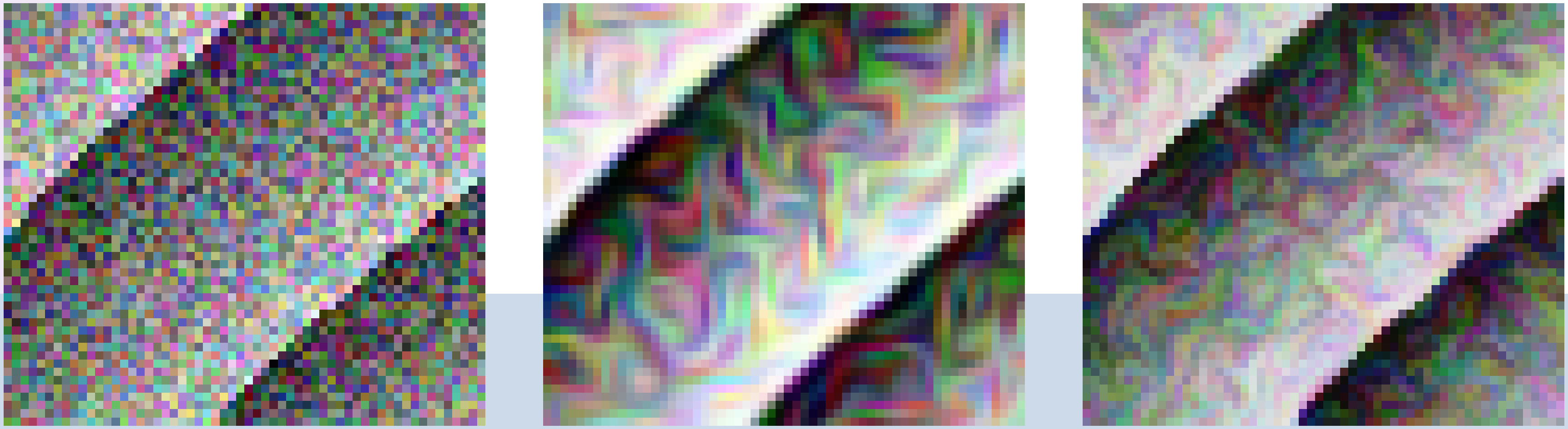
⇒ LIC-based numerical schemes allows a **sub-pixel accuracy** (4^{th}-order Runge-Kutta integration) $\Rightarrow$ **Very good preservation of small structures.**

Algorithm Properties

⇒ The maximum principle is verified (only local means of pixel intensities are computed).

⇒ Very stable and fast algorithm, compared to classical PDE implementations. The time step (dt) can be very large ($\simeq 50$) while process remains stable.

⇒ LIC-based numerical schemes allows a **sub-pixel accuracy** (4^{th}-order Runge-Kutta integration) $\Rightarrow$ **Very good preservation of small structures.**

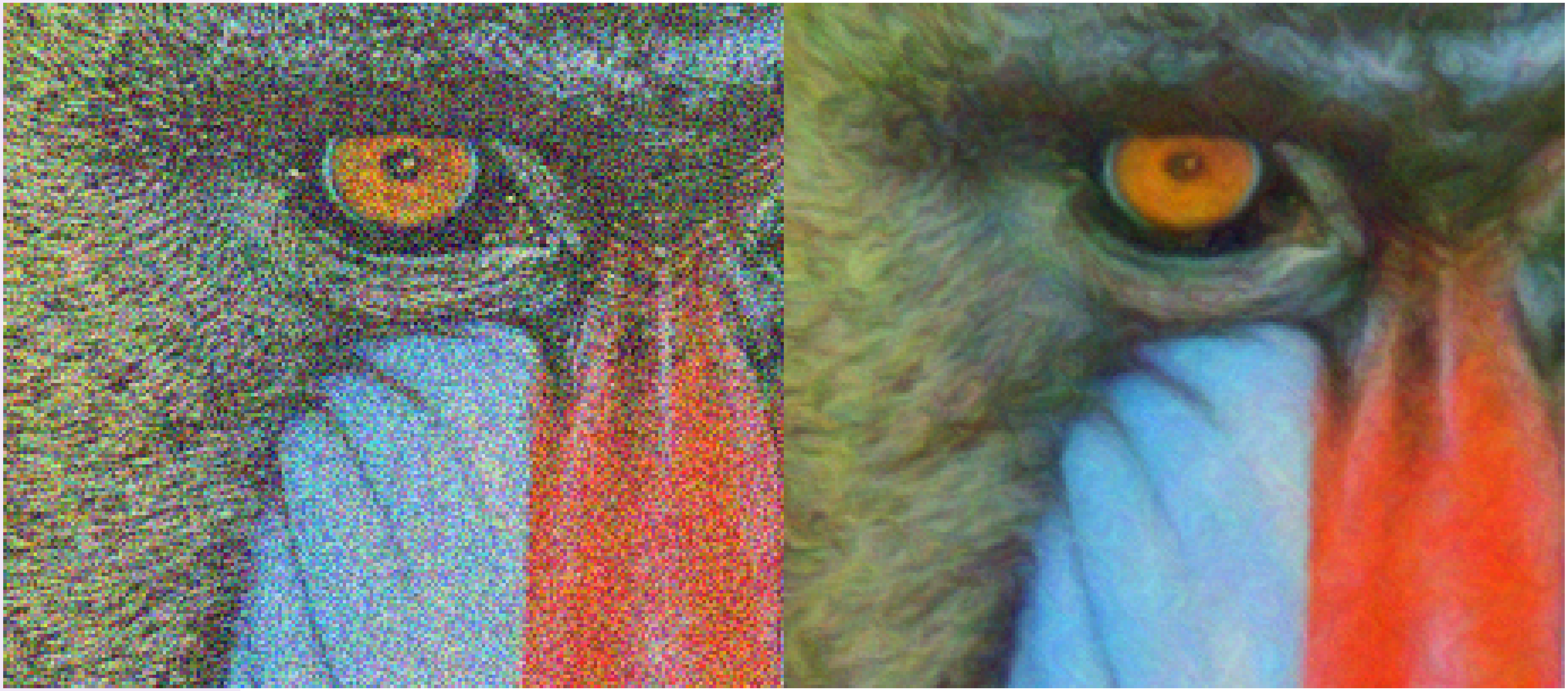


(a) Original image

(b) PDE Regul. (explicit Euler scheme)

(c) LIC-base scheme

Application : Image Denoising



“Babouin” (détail) - 512x512 - (1 iter., 19s)

Application : Image Denoising

“Tunisie” - 555x367

Application : Image Denoising

“Tunisie” - 555x367 - (1 iter., 11s)

Application : Image Denoising

“Tunisie” - 555x367 - (1 iter., 11s)

Application : Image Denoising

“Baby” - 400x375

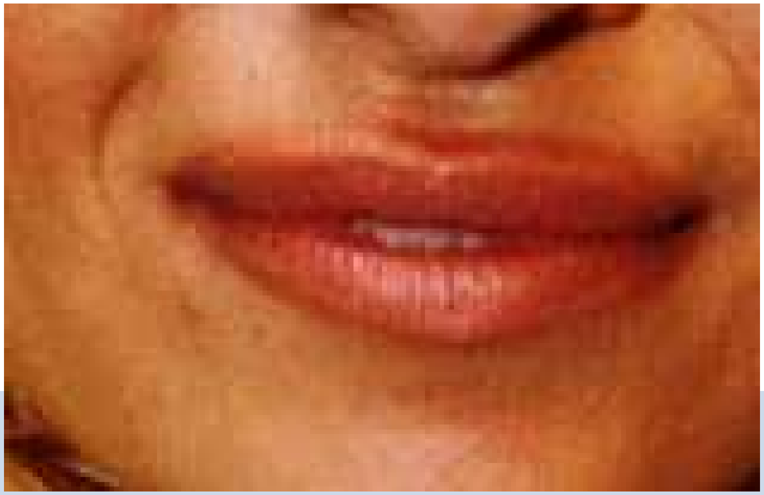
Application : Image Denoising

“Baby” - 400x375 - (2 iter, 5.8s)

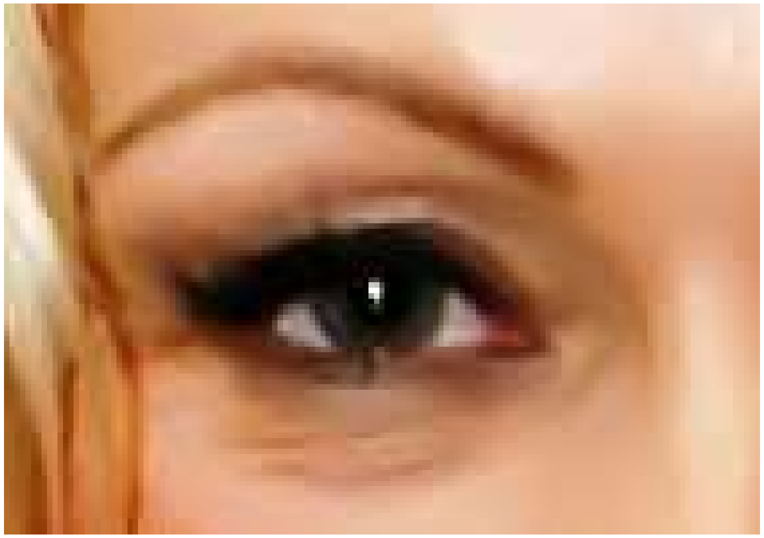
Application : Image Denoising

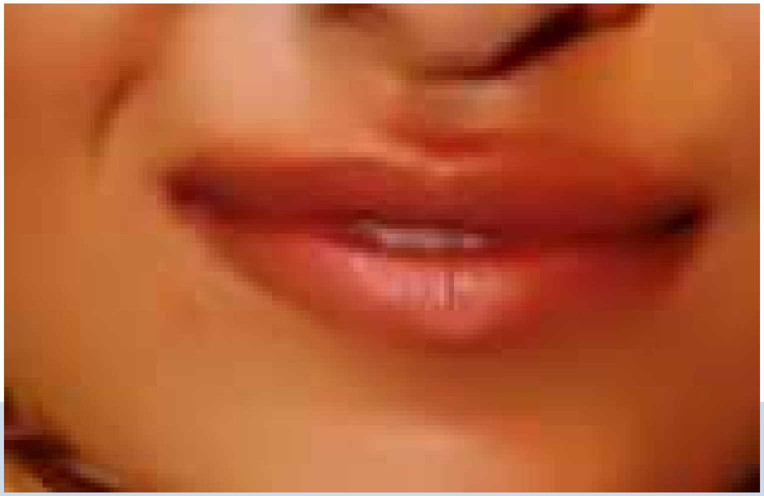
“Baby” - 400x375 - (2 iter, 5.8s)

Application : AWS, Anti Wrinkles System(tm)



Application : AWS, Anti Wrinkles System(tm)





⇒ Perform better than any other wrinkle-cream !

Application : Enhancement of compressed images.

Blocky JPEG Image (10% quality)

Application : Enhancement of compressed images.

Enhanced image

Application : Enhancement of compressed images.

Zoom (Blocky - Enhanced)

Application : Reducing JPEG artefacts

“Flowers” (JPEG, 10% quality).

Application : Creating Painting Effects

“Corail” (1 iter.)

Application : Image Inpainting

“Bird”, original color image.

Application : Image Inpainting

“Bird”, inpainting mask definition.

Application : Image Inpainting

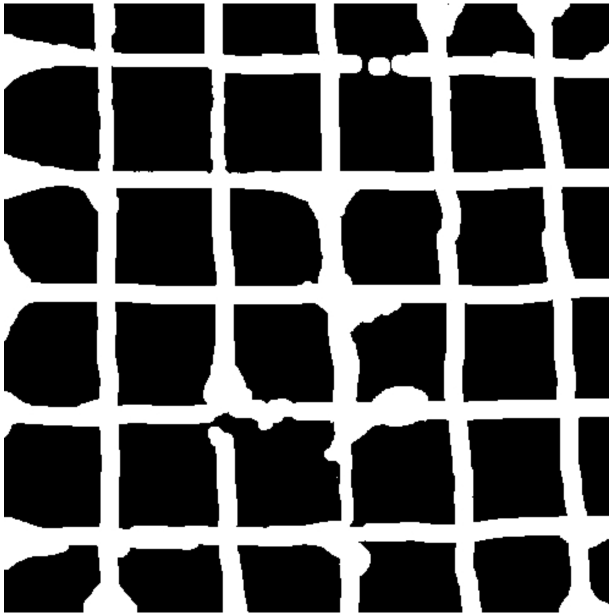
“Bird”, inpainted with our PDE.

Application : Image Inpainting

"Bird", inpainted with our PDE.

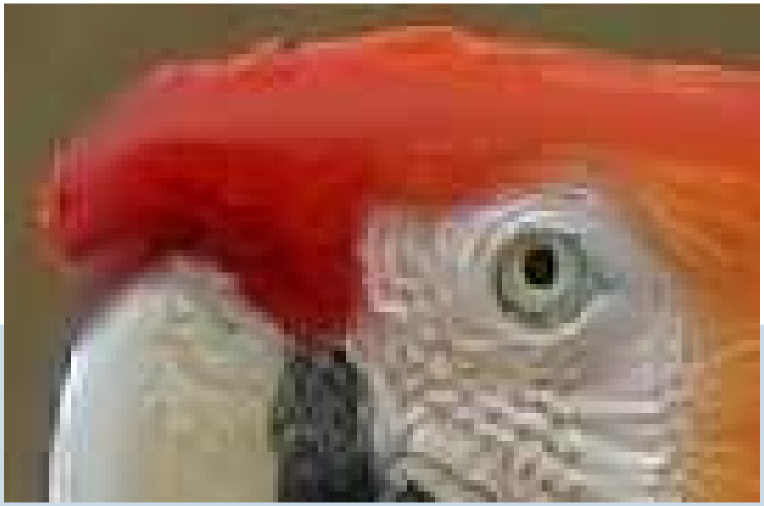
Application : Free the bird !

Original image



Inpainting mask definition

After image inpainting



Application : Image Inpainting

“Chloé au zoo”, original color image.

Application : Image Inpainting

“Chloé au zoo”, inpainting mask definition.

Application : Image Inpainting

“Chloé au zoo”, inpainted with our PDE.

Application : Image inpainting

Original image

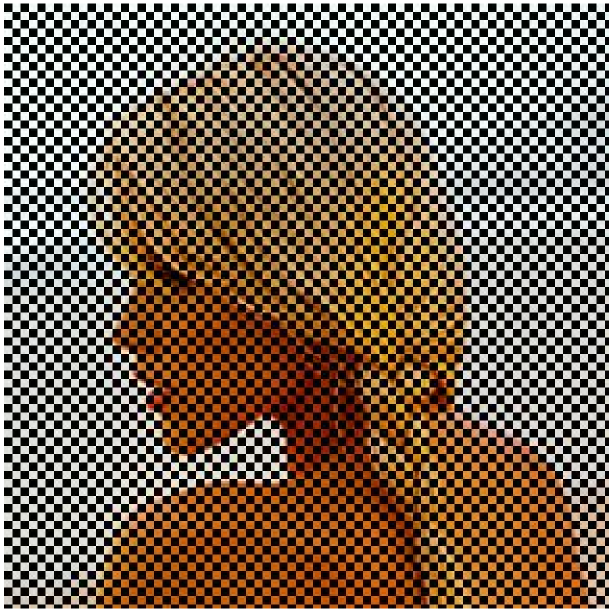
Inpainting mask definition

After image inpainting

Application : Image inpainting

- PDE's used for reconstruction of images with missing data.

Original image



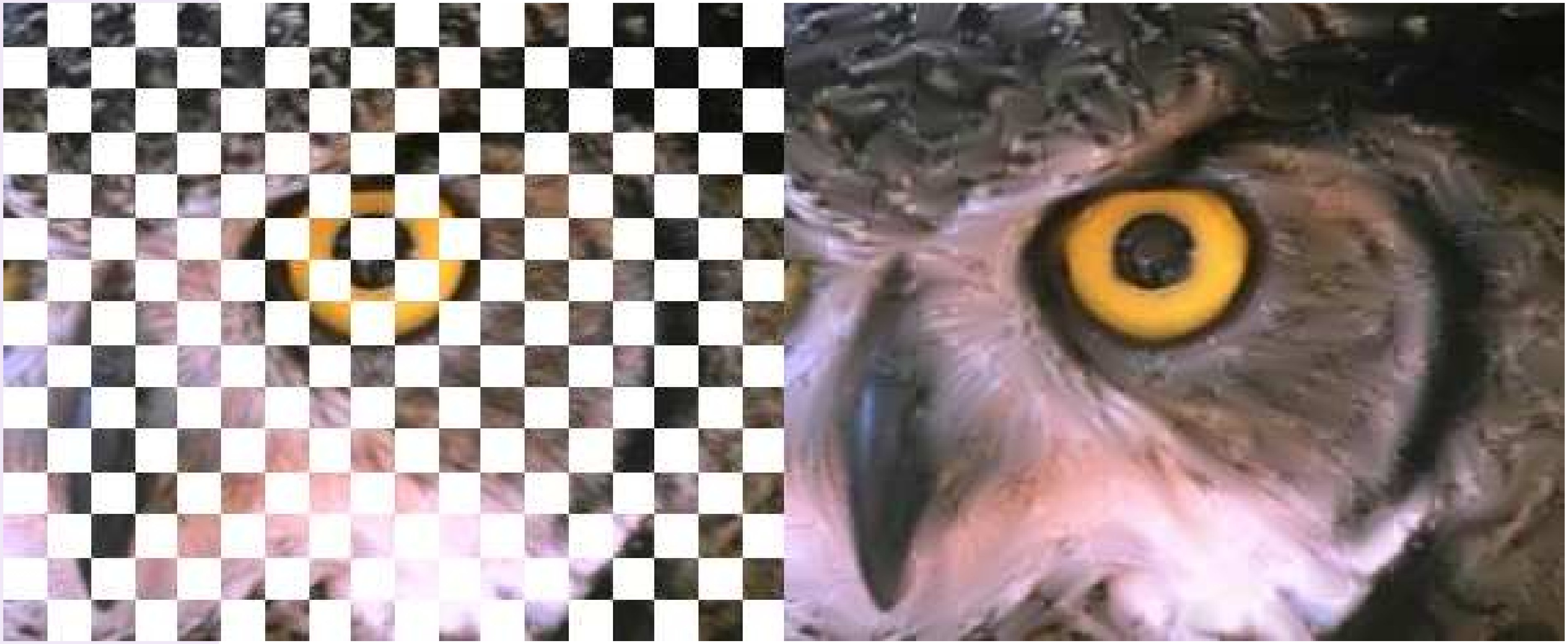
Removing 50% of the data

Reconstruction

⇒ Possible applications in static image compression.

Application : Image inpainting

- PDE's used for reconstruction of images with missing data.



Application : Image Resizing

“Nude” - (1 iter., 20s)

Application : Image Resizing

“Forest” - (1 iter., 5s)

Application : Image Resizing

(c) Details from the image resized by bicubic interpolation.

(d) Details from the image resized by a non-linear regularization PDE.

Application : Image Resizing

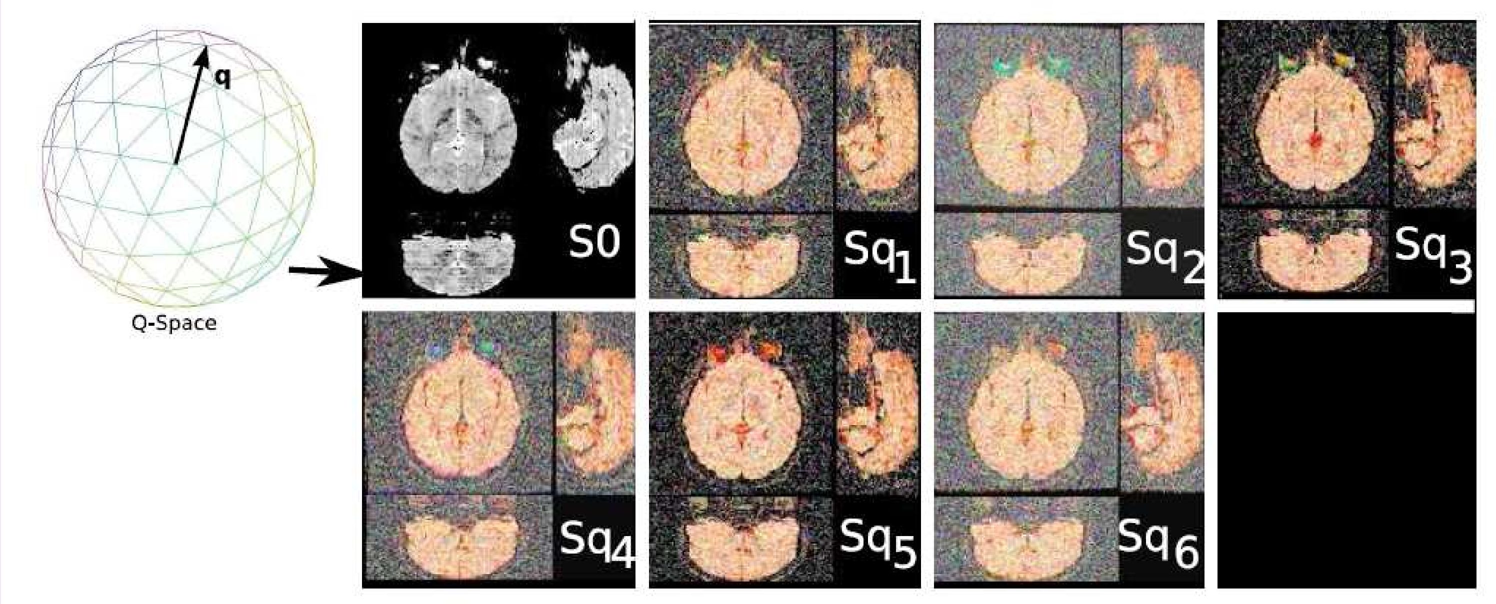
(a) Original color image

(b) Bloc Interpolation (c) Linear Interpolation (d) Bicubic Interpolation (e) PDE/LIC Interpolation

Application : DT-MRI Images

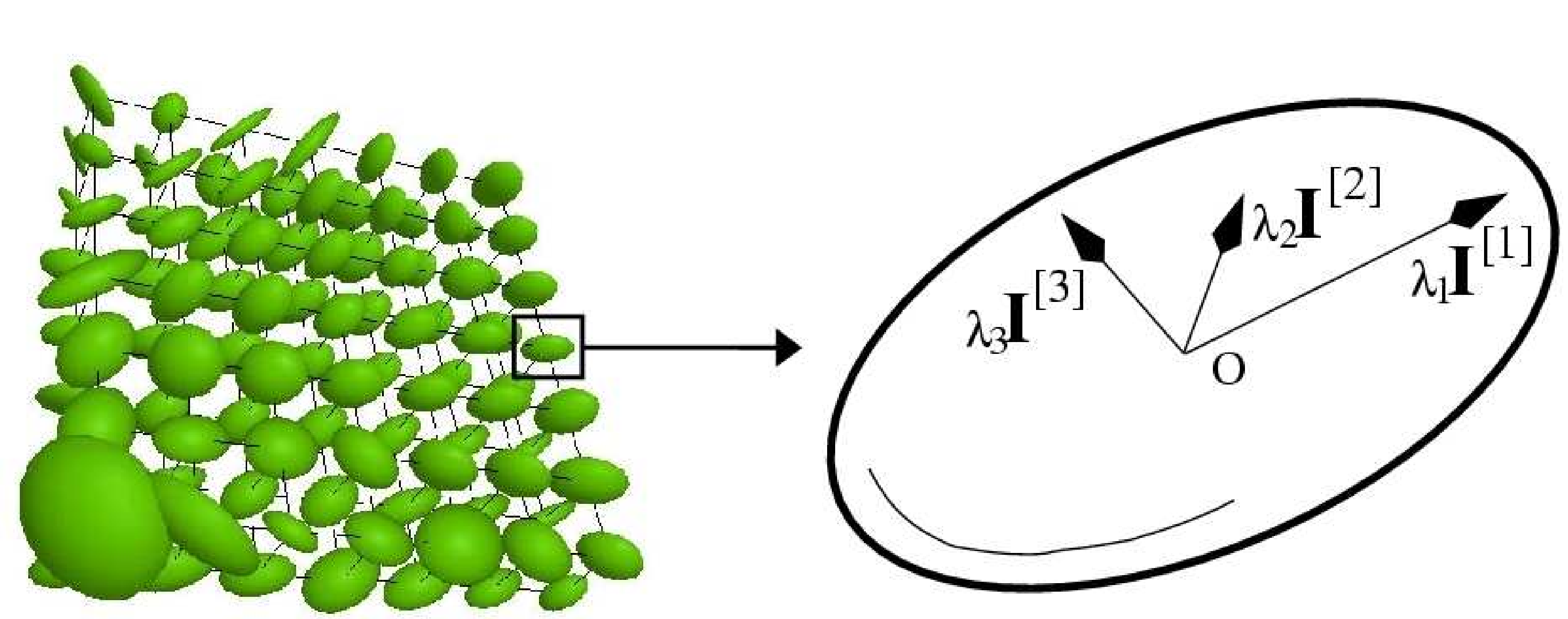
- MRI-based image modality that measures the water molecule diffusion in tissues.

- Acquisition or a set of multiple “raw MRI images, under different magnetic field configurations.



Application : DT-MRI Images (2)

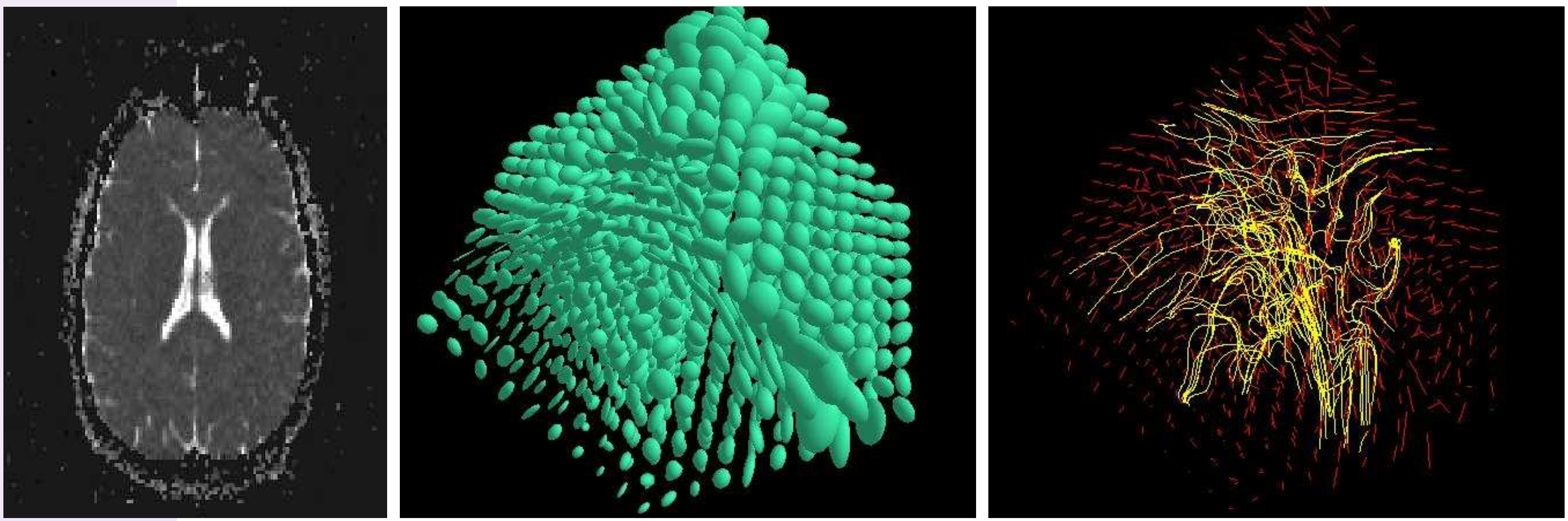
- A volume of Diffusion Tensors can be estimated from these raw images.
- Diffusion tensors represent gaussian models of the water diffusion within voxels, and are 3x3 symetric and positive matrices.
- Representation of a DT-MRI image with a volume of ellipsoids :



Application : DT-MRI Images (3)

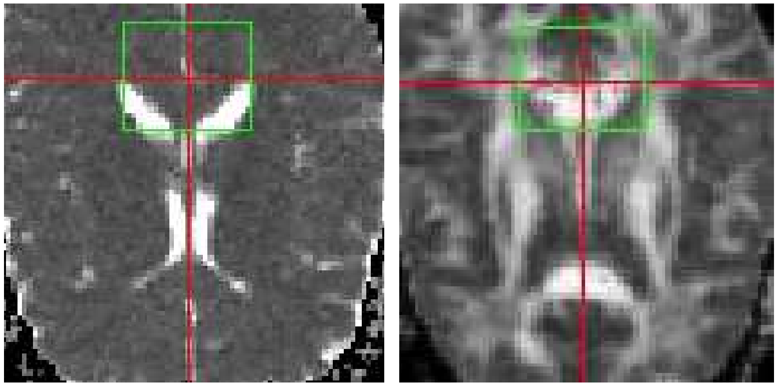
- DT-MRI Images give structural informations on the fibers network in the tissues.

- A fiber map reconstruction can be done by following at each voxel the principal tensor directions.

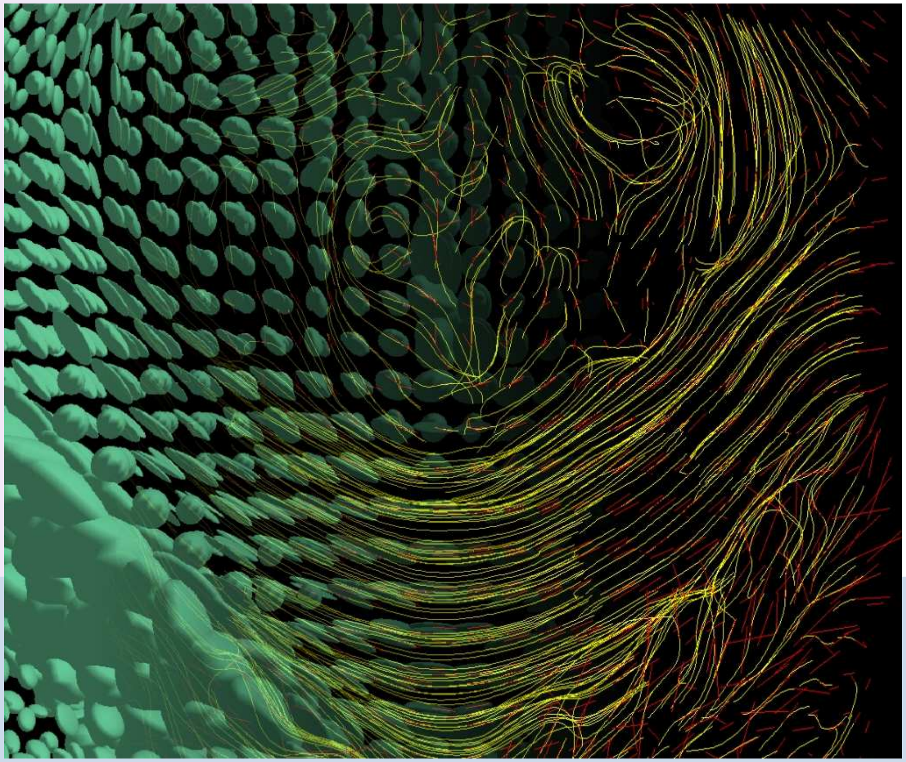


- The regularization of these DT-MRI images can be necessary to compute more coherent fiber networks (original images are very noisy)

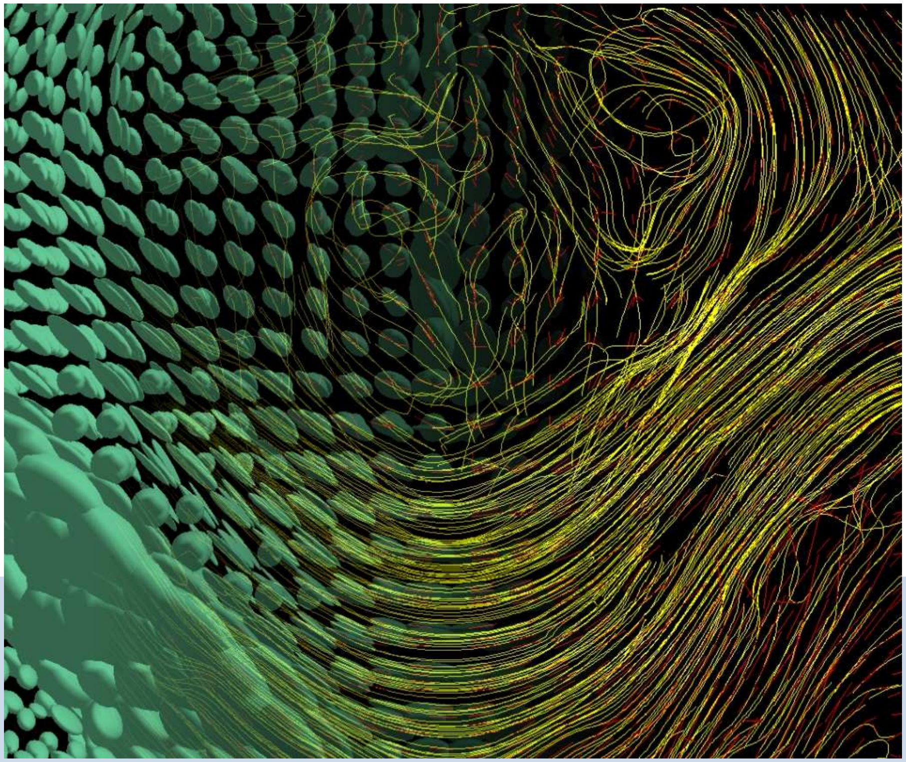
Fiber tracking on real data



(a) Average diffusivity (left) and Fractional Anisotropy (right)

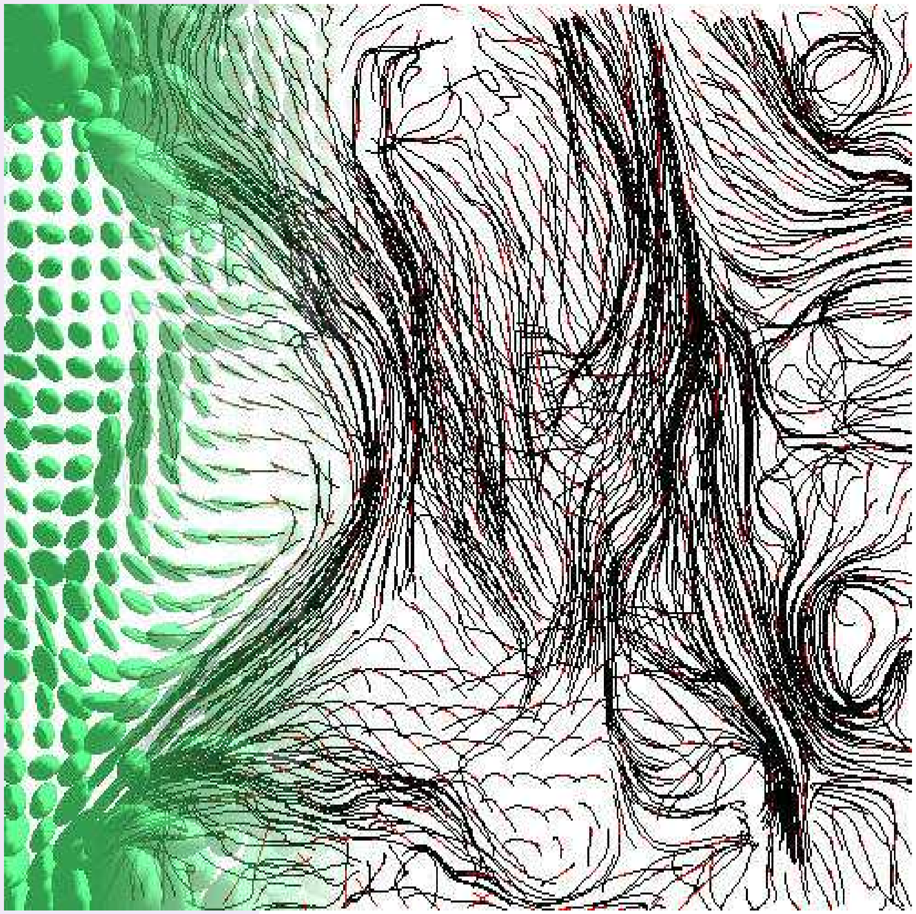


(b) Original tensors and computed fibers



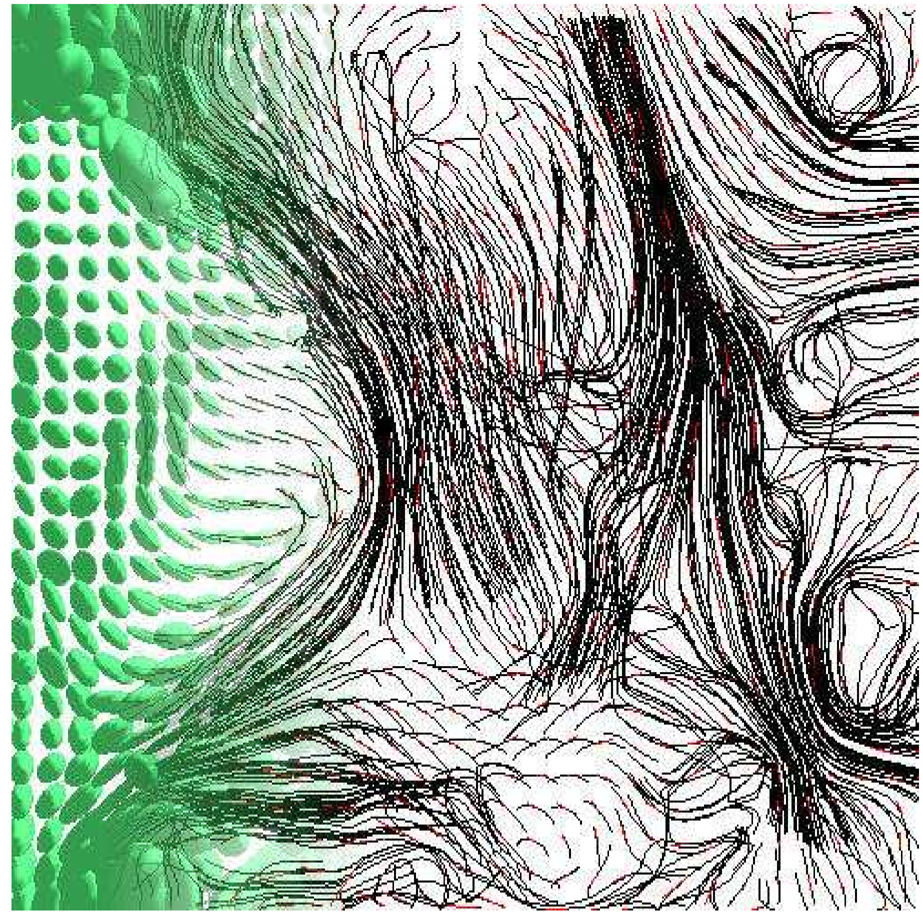
(c) Regularized tensors and computed fibers

Fiber Scale space (1)



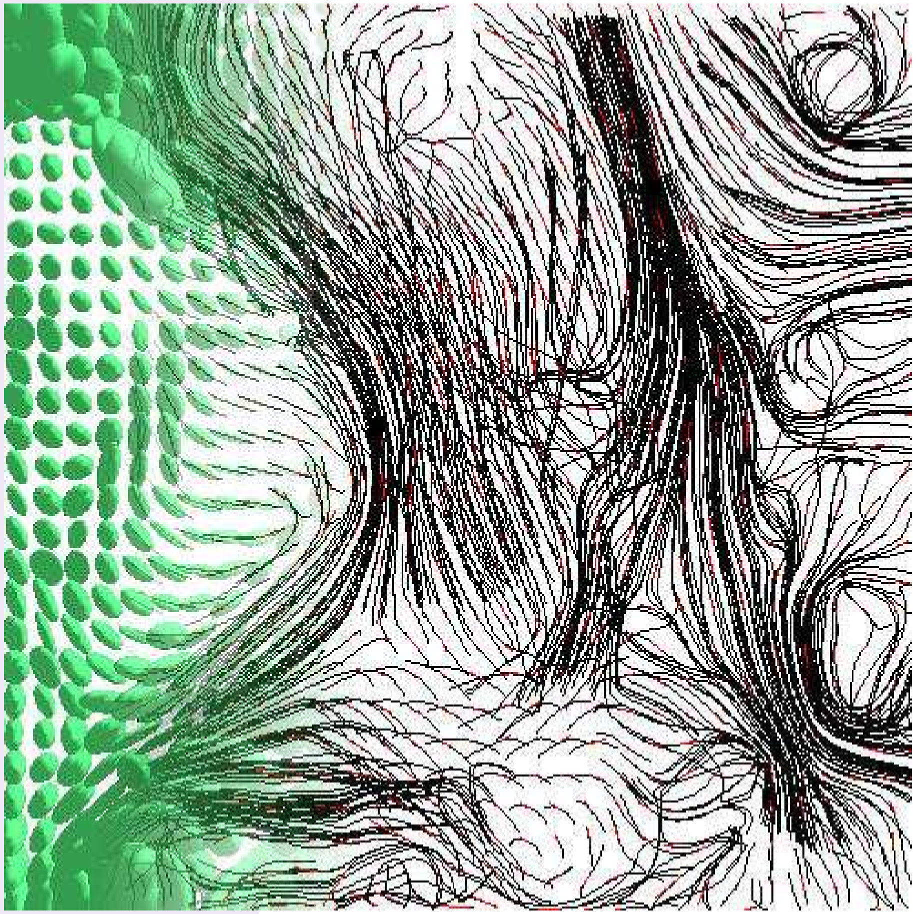
Tensors (left) & Fibers (right)

(Original data)

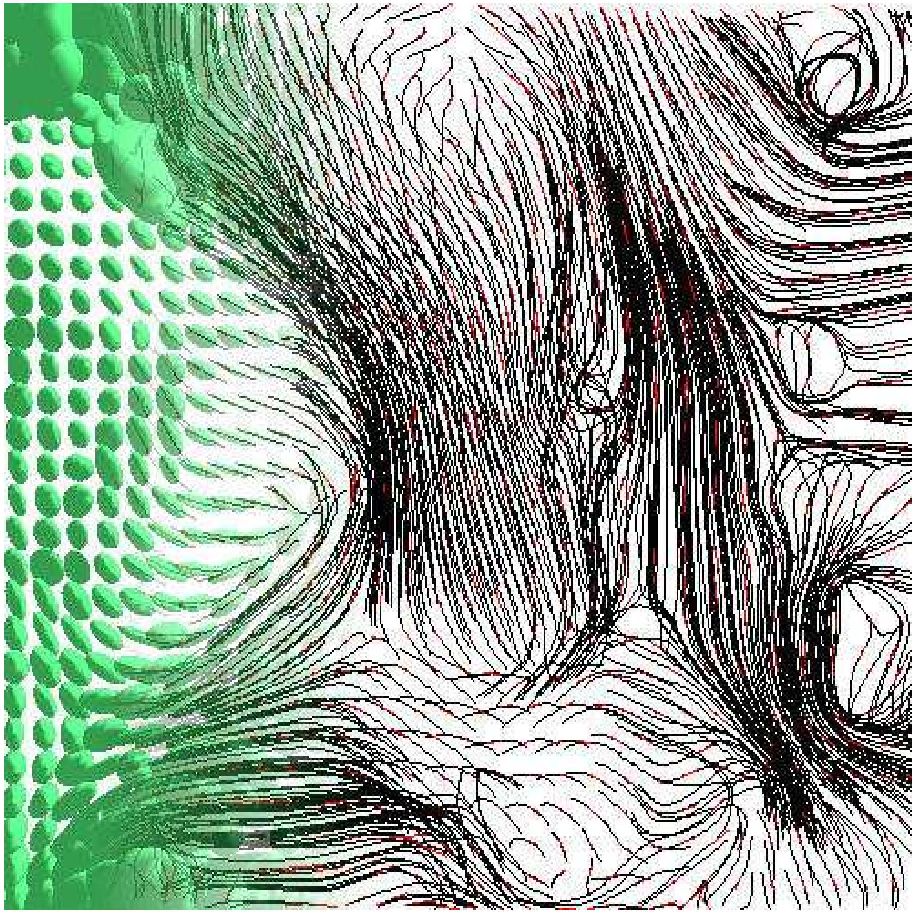


Regularized volume (after 20 it.)

Fiber Scale space (2)



Regularization after 20 it.



Regularization after 40 it.

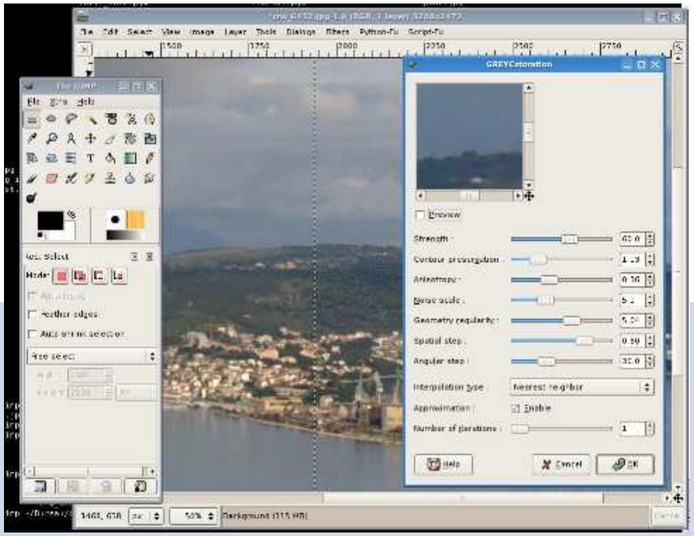
⇒ Scale-space model of the fiber network.

Conclusion

- Generic Multi-valued and Tensor-driven PDE's for the Regularization of Multi-Valued Images.

- Try it by yourself ! Experiments are reproducible. Source code (C++) is open.
  http://cimg.sourceforge.net/
  http://cimg.sourceforge.net/greycstoration/

- A plug-in for GIMP is also available, with a nice GUI.



Thanks for your attention !

Questions ?