Anisotropic Diffusion in ITK

Release 0.00

Jean-Marie Mirebeau[1], Jérôme Fehrenbach[2], Laurent Risser[2], Shaza Tobji[2]

December 17, 2014

[1]University Paris-Dauphine, Laboratory Ceremade, CNRS
[2]Institut de Mathématiques de Toulouse, Université Paul Sabatier, CNRS

Abstract

Anisotropic Non-Linear Diffusion is a powerful image processing technique, which allows to simultaneously remove the noise and enhance sharp features in two or three dimensional images. Anisotropic Diffusion is understood here in the sense of Weickert, meaning that diffusion tensors are anisotropic and reflect the local orientation of image features. This is in contrast with the non-linear diffusion filter of Perona and Malik, which only involves scalar diffusion coefficients, in other words *isotropic* diffusion tensors. In this paper, we present an anisotropic non-linear diffusion technique we implemented in ITK. This technique is based on a recent adaptive scheme making the diffusion stable and requiring limited numerical resources.

Contents

1 Introduction

A digital image is usually a large two or three dimensional array of pixel values (typically scalars or vectors). Image processing methods based on Partial Differential Equations (PDEs) regard images as approximations

of continuous objects, namely functions from an image domain to a pixel space, to which physics-inspired evolution rules can be applied. Among them, Non-Linear Anisotropic Diffusion (NLAD) is a variant of the heat equation, generalized in two regards: *Non-Linearity* and *Anisotropy*.

Anisotropy in diffusion means that the smoothing induced by the PDE can be favored in some directions and prevented in others. This is specified by local eigenvectors and eigenvalues of the diffusion tensor field (see §2). Diffusion coefficients are thus location *and direction* dependent, generalizing the approach of Perona and Malik [PM90] which is only location dependent. Importantly, efficient schemes for anisotropic diffusion have been recently made possible by the breakthrough in [FM13]. This has motivated the development of the ITK module presented in this paper.

Non-Linearity in diffusion means that diffusion tensors are automatically generated from the processed image. We implemented the strategies of Weickert [Wei98] and we give a simple framework for designing extensions and variants, see §3. The implemented filters and their parameters are described in §4. Figures 2 and 3 illustrate their effect on 2D images; Figures 4 and 5 on 3D images; Figures 6 and 7 on color and vector images.

A possible application of NLAD is to enhance a fingerprint image by smoothing tangentially to the lines. Evidence is also plentiful for NLAD relevance in many other image processing applications, but its use has been limited by technical aspects so far. We intend to alleviate such limitations with the present contribution.

Notations: Let $d \in \{2,3\}$ denote the image dimension, let $\Omega \subset \mathbb{R}^d$ be the image domain, and let V be the pixel space (e.g. $[0,1]$ for grayscale, $[0,1]^3$ for color, $\mathbb{R}^d$ for vectors). Throughout the paper, we informally consider an idealized *cartoon* image model, involving a set $\Gamma \subset \Omega$ of *image contours* of dimension $d-1$. The processed image $u : \Omega \to V$ is smooth on $\Omega \setminus \Gamma$, but has discontinuities across Γ, and is overall corrupted by e.g. by additive white noise. A key feature of NLAD is its ability to detect the set Γ and smoothen tangentially to it. Finally, let S_d^+ denote the collection of symmetric positive definite $d \times d$ matrices, and let Id be the identity matrix. To each $D \in S_d^+$ we associate the norm $\|e\|_D := \sqrt{\langle e, De \rangle}$, $e \in \mathbb{R}^d$, where $\langle \cdot, \cdot \rangle$ denotes the standard scalar product on $\mathbb{R}^d$.

2   Linear Anisotropic diffusion

sec:LAD

Linear Anisotropic Diffusion[1] (LAD), in divergence form, is an elliptic PDE which reads

$$\partial_t u = \mathrm{div}(\mathbf{D}\,\nabla u), \qquad (1)$$

eq:Linea

where $\mathbf{D} : \Omega \to S_d^+$ is a given field of symmetric positive definite diffusion tensors. Eigenvectors of these tensors define preferential diffusion directions, and the eigenvalues their corresponding coefficients. Evolution rule (1) is complemented with an initial condition $u(0,\cdot) = u_0$ at time $t = 0$. If u_0 has pixels of vector type, then their components are treated independently. We use Neumann[2] conditions on the domain boundary $\partial\Omega$, as is common in image processing. LAD is formally a continuous gradient descent for the elliptic energy

$$\mathcal{E}(u) := \int_\Omega \|\nabla u(x)\|_{\mathbf{D}(x)}^2 dx. \qquad (2)$$

eq:Aniso

---

[1] We use here the terminology of Weickert [Wei98]. Perona-Malik diffusion, which uses an adaptive scalar tensor field similar to (4), is in contrast a *Non-Linear Isotropic* Diffusion equation.

[2] Neuman boundary conditions must take into account the geometry defined by the diffusion tensor field. They take the form $\langle \nabla u(x), \mathbf{D}(x)\mathbf{n}(x) \rangle = 0$, where $\mathbf{n}(x)$ denotes the unit outward normal at $x \in \partial\Omega$.

Qualitative effects of LAD strongly depend on the chosen field $\mathbf{D}$ of diffusion tensors. Choosing $\mathbf{D} = \text{Id}$ identically on Ω yields the standard heat equation, which qualitative properties are well known in image analysis: any noise present in the image u is quickly eliminated, but in the meanwhile all image sharp features are blurred.

This undesirable side effect can be limited with a proper choice of diffusion tensors $\mathbf{D} : \Omega \to S_d^+$. Indeed LAD smoothes primarily the image features which contribute strongly to the energy (2). In the spirit of Perona and Malik, one can introduce an *isotropic* but *variable* conductivity $\mathbf{D}(x) = c(x)\,\text{Id}$, with $c(x) \ll 1$ for all x close to the image contours Γ, see introduction. Smoothing is prevented in the neighborhood of Γ, which preserves the contours sharpness, but also traps some noise along them, see Figure 2 (IV). A more elaborate approach is to construct *anisotropic* diffusion tensors $\mathbf{D}(x)$, which favor diffusion *tangentially* to the contours curves Γ, but simultaneously prevent diffusion transversally to these curves and between different image regions. All noise is eliminated, yet image discontinuities are preserved, see Figure 2 (II).

Numerical schemes for LAD are in general non-trivial due to interaction between the anisotropic geometry of the diffusion tensors, and the cartesian structure of the pixel grid. The authors recently developed [FM13] a numerical scheme which handles this interaction using special tools from discrete geometry, named Lattice Basis Reduction (LBR). It provides strong mathematical guarantees (consistency, stability, maximum-principle) for a limited numerical cost.

3 Non-Linear Anisotropic Diffusion.

Linear Anisotropic Diffusion, discussed in §2, requires two main inputs: an image u_0 serving as an initial condition, and a field of diffusion tensors. In order to reduce user input, the diffusion tensors $\mathbf{D}$ can be defined in terms of the filtered image u. The resulting PDE is called non-linear anisotropic diffusion

$$\partial_t u = \text{div}(\mathbf{D}_u \nabla u), \tag{3}$$

complemented, again, with Neumann boundary conditions. Perona and Malik [PM90] suggested to use the following *non-linear isotropic* (i.e. proportional to the identity matrix) tensors

$$\mathbf{D}_u(x,t) := c_u(x,t)\,\text{Id}, \qquad \text{with} \quad c_u(x,t) := \frac{1}{\sqrt{1 + \|\nabla u(x,t)\|^2/\lambda^2}}, \tag{4}$$

where $\lambda > 0$ is a user specified constant. Diffusion is prevented where the conductivity $c_u(x,t)$ is small, in other words where $\|\nabla u(x,t)\|$ is large, such as along the image contours Γ. Perona-Malik diffusion is already available[3] in ITK. It has been the subject of considerable academic and industrial interest revealing that, in spite of its numerous qualities, it is mathematically ill posed, unstable, often leads to unsightly "staircasing" visual artifacts, and is not adequate for oscillating patterns as in Figure 3.

We describe in the following Coherence Enhancing Diffusion (CED) and Edge Enhancing Diffusion (EED), which are based on more complex tensor constructions introduced by Weickert [Wei98]. Our first ingredient is the Gaussian convolution kernel: given a standard deviation $\sigma > 0$

$$K_\sigma(x) := \frac{1}{\sigma^d} K_1\left(\frac{x}{\sigma}\right), \qquad \text{where} \quad K_1(x) := \frac{1}{(2\pi)^d} \exp\left(\frac{-\|x\|^2}{2}\right). \tag{5}$$

The structure tensor $\mathbf{S}_u : \Omega \to S_d^+$, defined below, is a robust estimator of the gradient direction in an image u, even if this image has oscillating textures. It depends on two small positive parameters: the noise scale σ,

---

[3]Under the name (misleading with our conventions) `itk::GradientAnisotropicDiffusionImageFilter`.

and the feature scale ρ. We denote by $*$ the convolution operator, and by $v \otimes v = vv^{\mathrm{T}}$ the self outer product, which yields a semi-definite symmetric matrix.

$$\mathbf{S}_u := K_\rho * (\nabla u_\sigma \otimes \nabla u_\sigma), \qquad \text{where} \quad u_\sigma := K_\sigma * u. \tag{6}$$

eq:Struc

If u is an image with vector pixels, then $\mathbf{S}_u$ is the *sum* of the structure tensors associated to the components of u. Assume that u is a scalar image, fix a time t and a point $x \in \Omega$, and denote $S := \mathbf{S}_u(x,t)$, $v := \nabla u(x,t)$. Let also $\lambda_1 \leq \cdots \leq \lambda_d$ denote the eigenvalues of S, sorted by increasing magnitude, and $e_1, \cdots, e_d$ the corresponding unit eigenvectors. If u is sufficiently smooth, then the largest eigenvalue approximates the gradient squared norm: $\lambda_d \approx \|v\|^2$, while the corresponding eigenvector approximates the gradient direction: $e_d \approx \pm \frac{v}{\|v\|}$.

Weickert's diffusion tensors $D := \mathbf{D}_u(x,t)$, are defined in terms of this eigen-analysis of the structure tensor $S := \mathbf{S}_u(x,t)$:

$$D = \sum_{1 \leq i \leq d} \mu_i e_i \otimes e_i, \quad \text{where} \quad S = \sum_{1 \leq i \leq d} \lambda_i e_i \otimes e_i. \tag{7}$$

eq:Struc

Smoothing is promoted in the direction e_i if μ_i is large, and prevented if μ_i is small, for any $1 \leq i \leq d$. Weickert's classical constructions are presented in (8) and (11). One should not shy away of designing more complex and application dependent variants; for instance one may want to enhance filaments and tubular structures in 3D data. Three very simple variants (9), (10) and (12) are presented for illustration. All depend on three parameters λ, m, α. The main one, $\lambda > 0$, is an edge detection threshold. The exponent m is typically 2 or 4. The small parameter α, typically $1/100$, determines the condition number of the diffusion tensors.

- Edge Enhancing Diffusion (EED) aims to avoid significant diffusion across the set Γ of image contours, but to allow it anywhere else. Note that for 3D images discontinuity planes will be enhanced, rather than edges. The first diffusion tensor eigenvalue is $\mu_1 = 1$, because the eigenvector e_1 is orthogonal to the image (approximate) gradient direction e_d, hence never transverse to Γ. Other eigenvalues satisfy $\mu_i \approx \alpha \ll 1$ if $\lambda_i - \lambda_1 \gtrsim \lambda$, and $\mu_i \approx 1$ otherwise. The condition $\lambda_i - \lambda_1 \gtrsim \lambda$ indeed suggests that the eigenvector e_i points through an image contour. Precisely[4]: (note that $\mu_1 = 1$)

$$\mu_i := 1 - (1 - \alpha) \exp\left(-\left(\frac{\lambda}{\lambda_i - \lambda_1} \right)^m \right). \tag{8}$$

eqdef:EE

  The choice of Weickert, to set $\mu_1 = 1$, may lead to undesired effects: one always performs diffusion in at least one direction. An undesirable side effect is that the image is blurred close to the angles of its contour set Γ. We believe that such salient features should be preserved, hence we introduce a Conservative variant of EED (cEED) for which μ_1 *can* be small, when appropriate, so as to prevent diffusion around the angles of Γ, see Figure 8 for a comparison. Precisely

$$\mu_i := 1 - (1 - \alpha) \exp\left(-\left(\frac{\lambda}{\lambda_i} \right)^m \right). \tag{9}$$

eqdef:cE

  If all eigenvalues are set equal $\mu_1 = \cdots = \mu_d$, then the diffusion tensors are *isotropic*, in other words scalar multiples of the identity. The following isotropic variant of EED is close in spirit to the Perona-Malik model: diffusion is prevented in the neighborhood of the image contours Γ, regardless of direction. This construction is implemented purely for comparison with the anisotropic ones, and does not take advantage of the innovative numerical scheme developed by the authors

$$\mu_i := 1 - (1 - \alpha) \exp\left(-\left(\frac{\lambda}{\lambda_d} \right)^m \right). \tag{10}$$

eqdef:Is

---

[4] Actually, Weickert uses $\mu_1 = 1$ together with (9) for $i > 1$. This results in discontinuous diffusion tensors, which is not advisable from a mathematical standpoint, hence the formula (8).

```
// Usage with uchar pixel type, cast to double.
typedef Image<unsigned char,3> ImageType;
typedef Image<double,3> FPImageType;
typedef CastImageFilter<ImageType, FPImageType> CastFilterType;
typedef CoherenceEnhancingDiffusionFilter<FPImageType> CEDFilterType;

CastFilterType::Pointer castFilter = CastFilterType::New();
CEDFilterType::Pointer cedFilter = CEDFilterType::New();
cedFilter->SetInput(castFilter->GetOutput());
```

```
// Usage with RGB pixel type, cast to Vector
typedef Vector<float,3> VectorType;
typedef RGBPixel<unsigned char> RGBType;
struct CastFunctorType {
    VectorType operator()(const RGBType & x){
        VectorType result; for(int i=0; i<3; ++i) result[i]=x[i]; return result;}
};
typedef Image<RGBType, 2> ImageType;
typedef Image<VectorType, 2> FPImageType;
typedef UnaryFunctorImageFilter<ImageType, FPImageType, CastFunctorType> CastFilterType;
typedef CoherenceEnhancingDiffusionFilter<FPImageType, float> CEDFilterType;
```

Figure 1: The provided filters expect image pixels of scalar or vector pixel type. A preliminary cast is required for integral pixel types (left), or non-Vector pixel types such as RGB (right). The provided executable (or ".cxx" file) performs these casts automatically.

- Coherence Enhancing Diffusion (CED) prevents diffusion except along local image structures which have a coherent direction. The diffusion tensor eigenvalues satisfy $\mu_i \approx \alpha \ll 1$, *unless* if $\lambda_d - \lambda_i \gtrsim \lambda$ in which case $\mu_i \approx 1$. The condition $\lambda_d - \lambda_i \gtrsim \lambda$ indeed suggests that e_d points through an image feature, and that e_i points tangentially to it. Precisely: (note that $\mu_d = \alpha$)

$$\mu_i := \alpha + (1-\alpha)\exp\left(-\left(\frac{\lambda}{\lambda_d - \lambda_i}\right)^m\right). \tag{11}$$

The above formula often leads to false positives: at a position with large gradients, but without a clear preferred direction, one may very well have $\lambda_d - \lambda_i \gg \lambda$ (for instance if $\lambda_d = 100\lambda$ and $\lambda_i = 95\lambda$). A more reliable coherence detector is $\lambda_m - \lambda_i \gg \lambda + \lambda_i$, which leads to a Conservative variant of CED (cCED), see Figure 8 for a comparison. Precisely: (note that $\mu_d = \alpha$)

$$\mu_i := \alpha + (1-\alpha)\exp\left(-\left(\frac{\lambda + \lambda_i}{\lambda_d - \lambda_i}\right)^m\right). \tag{12}$$

We emphasize that the distinction between EED and its conservative variant cEED (resp. CED and cCED) is rather subtle, and mostly located around image contour corners, as evidenced on Figure 8. In other illustrations, we only show the conservative variant, which is slightly better at preserving detail.

4 Implemented filters and their parameters

Our contribution to ITK consists of the following four image filters, which implement the mathematical notions presented in §2 and §3. The figures are produced with the last filter, which implements CED, EED and their variants described §3. The first three filters are its building blocks, but they may be of independent interest for other applications. All filters are multithreaded.

For each filter, the processed image pixels can be of scalar or vector type. In the latter case, the underlying floating point type needs to be specified via the second template parameter of the filter, see Figure 1 (bottom right). Pixels of integral type (resp. RGB pixels) must be cast to floating point (resp. `itk::Vector`) types, see Figure 1. The image dimension must be 2 or 3.

Linear Anisotropic Diffusion (LAD). The filter `itk::LinearAnisotropicDiffusionLBRImageFilter`, requires two inputs: a processed image and a tensor image. Note that non-linear diffusion is achieved through successive linear diffusions, over multiple small time intervals, with regularly updated diffusion tensors. Parameters:

  - *MaxDiffusionTime* specifies the target physical time for the LAD evolution PDE (1). The filter has early abort options, as discussed in the next point, hence one should check the *EffectiveDiffusionTime* at termination.

- *MaxNumberOfTimeSteps*, *RatioToMaxStableTimeStep*. Explicit numerical schemes for diffusion are subject to a Courant-Friedrichs-Levy (CFL) condition, which limits the largest stable time step. This time step, which depends on the input diffusion tensors, is automatically computed by the filter, and the requested time interval $[0, MaxDiffusionTime]$ is split accordingly. Early abort occurs if this splitting exceeds the specified *MaxNumberOfTimeSteps*.

Structure tensor. Filter `itk::StructureTensorImageFilter`. Parameters:

- *NoiseScale* σ, and *FeatureScale* ρ, see (6). Suggested ranges: $\sigma \in [0.5, 3]$, $\lambda \in [2, 10]$, assuming a unit pixel spacing. The lower bounds of these intervals are recommended, unless noise is extremely strong.

- *RescaleForUnitMaximumTrace*. If on, the structure tensors are rescaled: $S'_u = \alpha S_u$, where $\alpha > 0$ is the largest constant such that $\text{Trace}(\mathbf{S}'_u(x)) \leq 1$ for all $x \in \Omega$. (The trace is the sum of the eigenvalues $\lambda_1 + \cdots + \lambda_d$, see (7).) This option is meant to ease the choice of the edge detection threshold λ in CED, EED. One may want to check the variable *PostRescaling* $= \alpha$ at termination.

Non-Linear Anisotropic Diffusion (NLAD). Filter `itk::AnisotropicDiffusionLBRImageFilter`. This class (I) computes structure tensors by invoking the previous filter, (II) performs their eigen-analysis, (III) changes them into diffusion tensors via (7), (IV) runs linear diffusion with the constructed tensors by invoking the first filter. After a limited number of time-steps of linear diffusion, the steps (I-II-III-IV) are repeated so as to update the diffusion tensors, until exhaustion of the prescribed diffusion time. Parameters:

- *DiffusionTime* for which the evolution rule (3) is applied.

- *Adimensionize.* If on, the filter ignores the image pixel spacing information, and sets on the *RescaleForMaximumUnitTrace* option for structure tensor generation. This is intended to ease the choice of *DiffusionTime*, and of the edge detection threshold λ in CED, EED, and variants.

- *MaxTimeStepsBetweenTensorUpdates* is self descriptive. *NoiseScale* and *FeatureScale* are passed for structure tensor generation.

- *EigenValuesTransform* is a virtual method used to construct the diffusion tensor eigenvalues $(\mu_i)_{i=i}^d$ from those of the structure tensors $(\lambda_i)_{i=1}^d$, which are sorted increasingly for convenience. The method must be redefined in a subclass, as in the next filter, else it triggers an exception.

Coherence-Enhancing diffusion and Edge-Enhancing diffusion. The two PDEs, and their variants, are implemented in the same filter `itk::CoherenceEnhancingDiffusionFilter`, which subclasses the Non-Linear Anisotropic Diffusion filter. Parameters:

- *Enhancement* allows to switch between EED, cEED, CED, cCED and Isotropic (10) tensor constructions, by redefining the superclass virtual method *EigenValuesTransform*. The relevant choice depends on the type of image structures that one wants to enhance.

- *Lambda*$= \lambda$, *Exponent*$= m$, *Alpha*$= \alpha$ are the parameters involved in tensor design (8) - (12).

- Suggested parameter ranges. *Adimensionize*=True. *DiffusionTime* $\in [0.5, 5]$, although larger values can be relevant for very strong noise or artistic effects. Edge detection threshold $\lambda \in [10^{-3}, 5 \times 10^{-2}]$, small for complex images with detail, large for simple "cartoon" like images. Finally $m \in [2, 4]$, $\alpha = 0.01$, though these parameters are secondary and have little impact.
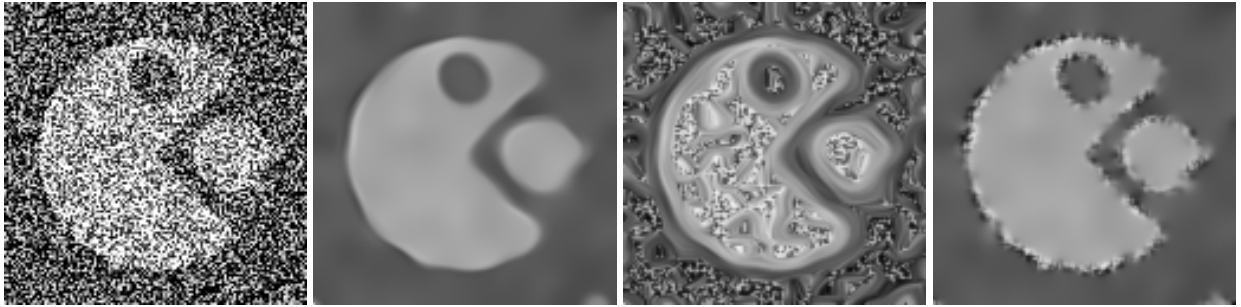
Figure 2: I: Source image. II: cEED is the best pick. III: cCED only affects the directionally coherent parts, the image region contours, which is clearly inadequate. IV: Isotropic (10, Perona-Malik like) diffusion leaves some noise trapped along the image contours. Parameters: $T = 20$, $\lambda = 0.05$, $\sigma = 3$, others default.
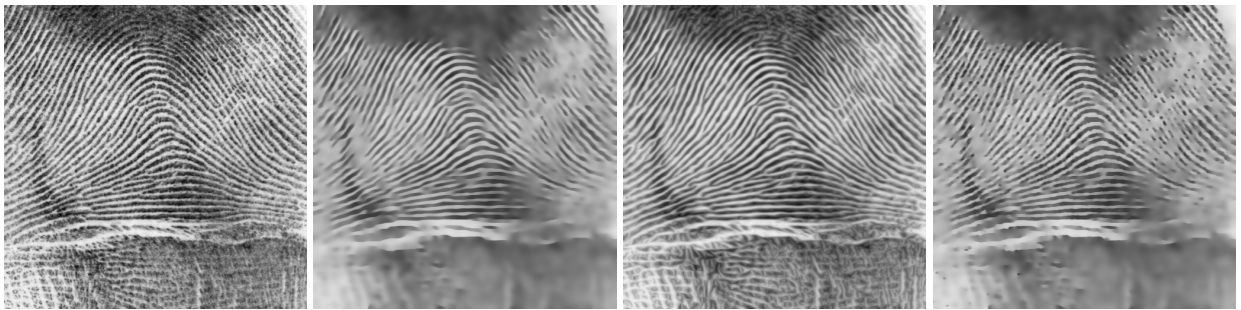

Figure 3: I: Source image. II: cEED is not advised, since it blurs the junctions of the fingerprint lines. III: cCED is the best pick, since it enhances the fingerprint lines, and does not blur the more complex regions. IV: Isotropic (10, Perona-Malik like) diffusion either fails to remove noise, or blurs the fingerprint lines, depending on the image region. Parameters: $T = 20$, $\lambda = 0.02$, others default.
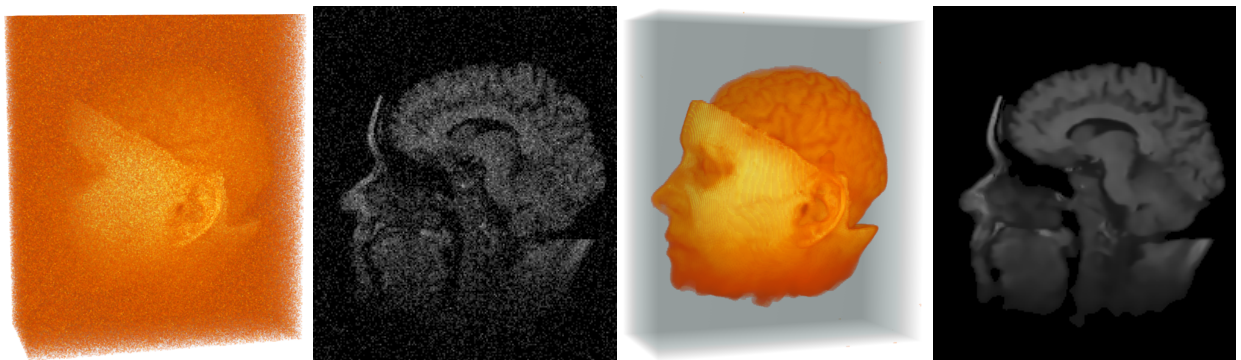

Figure 4: Left: Volume plot and slice of an FMRI of the human skull, with artificially added Gaussian noise of variance 0.01 (data range: $[0, 1]$). In the volume plot, values below 0.08 are shown transparent. Right: Effect of cEED. Parameters: $T = 5$, $\lambda = 0.003$.
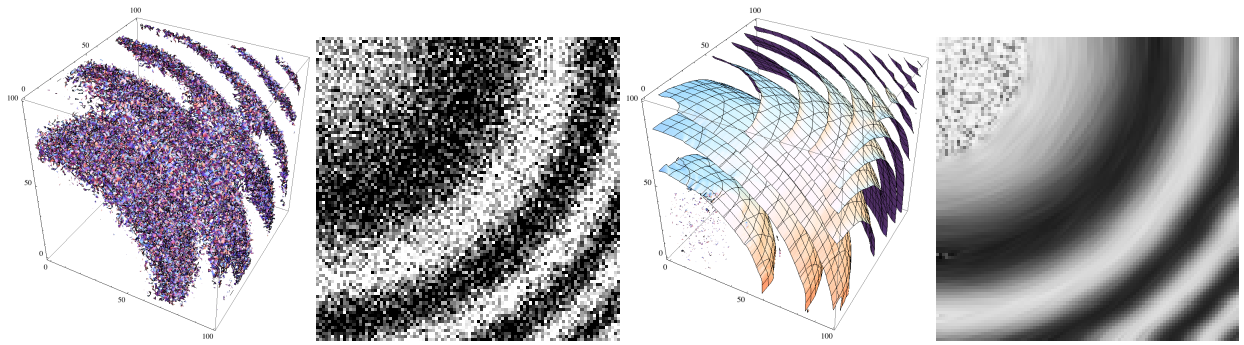
Figure 5: Synthetic function $\cos(\|x\|^3)$, $x \in [0,1]^3$, corrupted with Gaussian noise of variance 0.5. Left: level lines, and slice $\{x = 0.1\}$. Right: effect of CED. By construction, CED barely affects the neighborhood of the origin $(0,0,0)$, where no specific coherent direction can be determined. Parameters: $T = 10$, $\lambda = 0.02$, $\sigma = 4$, $\rho = 10$.   `fig:Cos3`

Figure 6: I: Detail of the Lena image. II: cEED removes noise and preserves most image detail. III: cCED enhances the flow structure of the hat plumes, without much affecting the rest of the image. In particular it does not remove noise. IV: Isotropic diffusion (10, Perona Malik like) removes noise in the interior of the image regions, but leaves some noise close to image contours such as the border of Lena's cheek. Parameters: $T = 2$, $\lambda = 0.003$, $m = 4$, others default.   `fig:Lena`
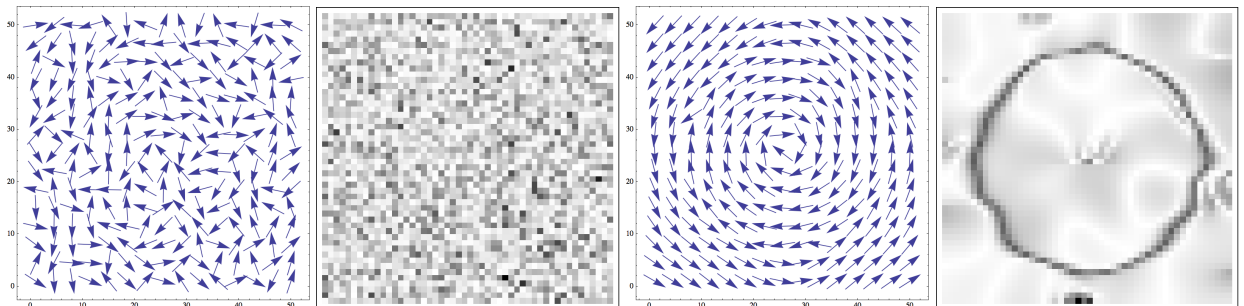

Figure 7: Left: Directions and norms (black=0, white=1) of the vector field $v(x) := \text{sign}(\|x\| - 1)\, x^\perp/\|x\|$ on $[-1.3, 1.3]^2$, degraded by gaussian noise of variance 2, on a 50×50 grid. Norms of these vectors (black=0, white=1). Right: Effect of cEED. Note that the streamlines are better reconstructed, and that the norms vanish along the (approximate) circle $\|x\| = 1$ where $v(x)$ changes sign so that there is a cancellation effect (and likewise close to the vector field singularity at the center). Parameters: $T = 10$, $\lambda = 0.05$, others default.   `fig:Vect`
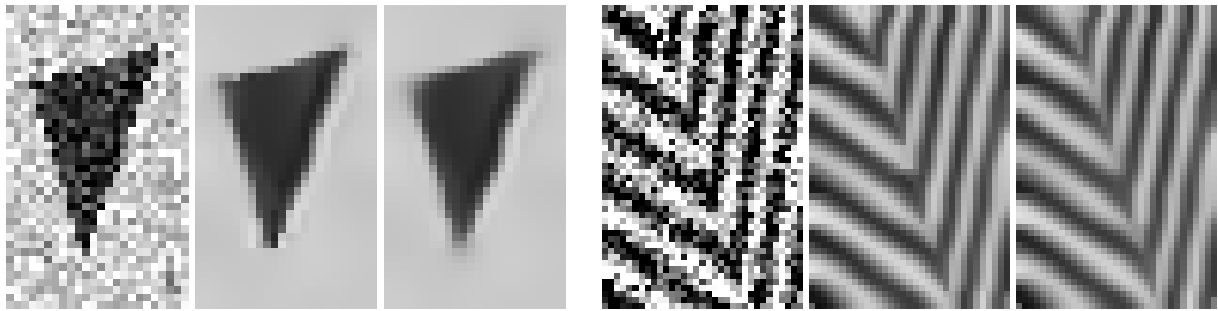
Figure 8: Left: (I) source image, (II) effect of cEED, (III) effect of EED. Note that the triangle corners are better preserved with cEED. Parameters: $T = 5$, $\lambda = 0.03$. Right: (I) source image, (II) effect of cCED, (III) effect of CED. Note that cCED better preserves contrast along where the two fronts meet. Parameters: $T = 20$, $\lambda = 0.05$.

`fig:Tria`

References

`ehrenbach:2013ut`

[FM13] Jérôme Fehrenbach and Jean-Marie Mirebeau. Sparse Non-negative Stencils for Anisotropic Diffusion. *Journal of Mathematical Imaging and Vision*, 2013. 1, 2

`Perona:1990ej`

[PM90] P Perona and J Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990. 1, 3

`1998anisotropic`

[Wei98] Joachim Weickert. *Anisotropic diffusion in image processing*. Teubner Stuttgart, 1998. 1, 1, 3