Anisotropic Fast Marching in ITK

Release 0.00

Jean-Marie Mirebeau[1]

June 4, 2015

[1]University Paris-Sud XI, Department of Mathematics, CNRS

Abstract

The Fast Marching algorithm is an efficient numerical method for computing the shortest path between points of a domain in $\mathbb{R}^d$. For that purpose, it solves a front propagation problem, which can be of interest in itself. The method has numerous applications, ranging from motion planning to image segmentation. The unit of length, for computing the path length, may vary on the domain. Motivated by applications, we generalize the algorithm to the case where the unit of length also depends on the path direction. Segmentation methods can take advantage of this flexibility to achieve greater sensitivity and specificity, for a comparable computation time.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/1338]

Contents

[1]Young researcher ANR program: Numerical Schemes using Lattice Basis Reduction. NS-LBR ANR-13-JS01-0003-01. https://sites.google.com/site/anrnslbr/

We develop an extension of the `itk::FastMarchingImageFilter` class, which handles anisotropic measures of path length, or front propagation speeds. Path length is measured locally through a Riemannian metric, or an even more general Finsler metric, instead of a multiple of the standard euclidean norm in the original ITK implementation. The flexibility offered by our extension allows to develop new segmentation methods, or to increase the sensibility and specificity of existing ones. Our implementation of Anisotropic Fast Marching relies on special pixel neighborhoods, illustrated inFigures 1 and 2 and mathematically justified in [8, 7], built using tools from discrete geometry. This operation is transparent to the user, but helps to achieve a good accuracy / numerical complexity compromise. Similar methods can be applied to anisotropic diffusion [4, 9].

1 Path length distances, geodesics, front propagation, and the eikonal PDE.

The geometrical background of the Fast Marching method is the computation of *shortest paths*, also called minimal geodesics, within a compact domain $\Omega \subset \mathbb{R}^d$. Path length is measured locally via a metric[1] $\mathcal{F}$, which associates to each point $z \in \Omega$ a norm $\mathcal{F}_z$ on $\mathbb{R}^d$. The length of a smooth path $\gamma : [0,1] \to \Omega$ is

$$\text{Length}_{\mathcal{F}}(\gamma) := \int_0^1 \mathcal{F}_{\gamma(t)}(\gamma'(t))\, dt.$$

The distance between two points $x, y \in \Omega$ is defined as the length of the shortest path joining these points:

$$\text{D}_{\mathcal{F}}(x,y) := \inf\{\text{Length}_{\mathcal{F}}(\gamma);\ \gamma : [0,1] \to \Omega, \gamma(0) = x, \gamma(1) = y\}.$$

A convenient way to visualize a metric is through the unit balls B_z associated to its norms $\mathcal{F}_z$, $z \in \Omega$, also called Tissot's indicatrix, see Figures 1 and 2

$$B_z := \{u \in \mathbb{R}^d;\ \mathcal{F}_z(u) \leq 1\}.$$

Consider a subset of seeds $X_0 \subset \Omega$, typically a single point $X_0 = \{x_0\}$. The fast marching algorithm allows to estimate the geodesic distance U from the set X_0:

$$U(x) := \min\{\text{D}_{\mathcal{F}}(x_0, x);\ x_0 \in X_0\}$$

The value $U(x)$ is also the minimal time T required to reach the point x, starting from the set X_0, via a path $\gamma : [0,T] \to \Omega$ moving at unit speed: $\mathcal{F}_{\gamma(t)}(\gamma'(t)) \leq 1$, equivalently $\gamma'(t) \in B_{\gamma(t)}$, for all $t \in [0,T]$. In this min-time interpretation, Tissot's indicatrix B_z thus describes the authorized path speeds at point z. Said otherwise, if $z \in \Omega$ and if u is a unit vector $\|u\| = 1$, then the speed limit at z in the direction u is $1/\mathcal{F}_z(u)$.

The shortest path $\gamma : [0,T]$ reaching a point $x \in \Omega$ from the closest element of X_0 can be recovered by solving an ordinary differential equation, involving the from arrival times U and the dual metric $\mathcal{F}^*$: $\gamma(T) = x$ and

$$\gamma'(t) = \nabla \mathcal{F}^*_{\gamma(t)}(\nabla U(\gamma(t))), \qquad \text{where } \mathcal{F}^*_x(u) := \max_{\mathcal{F}_x(v)=1} \langle u, v \rangle. \tag{1}$$

The distance U from the set X_0 can also be regarded as the arrival time of a front, originating from X_0 and moving at speed $\mathcal{F}^*_z(\mathbf{n}(z))$ at a point z with front outward normal $\mathbf{n}(z)$. The front line at time t is the level set $\{z \in \Omega; U(z) = t\}$. The eikonal PDE (Partial Differential Equation) accounts for this front formulation

$$\begin{cases} \mathcal{F}^*_x(\nabla U(x)) = 1 & x \in \Omega \setminus X_0, \\ U(x_0) = 0 & x_0 \in X_0. \end{cases} \tag{2}$$

[1] Within this paper, the term *metric* is understood in the sense of differential geometry: the data of a norm at each point of a domain. This is obviously distinct from the understanding, as in `itk::Metric`, of a measure of similarity between images.

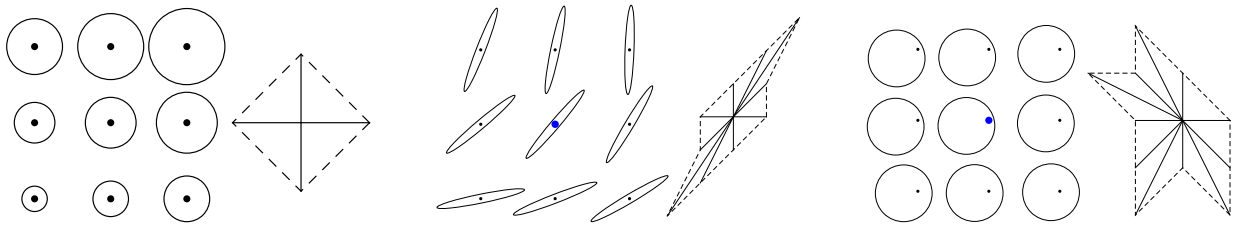

Figure 1: A metric $\mathcal{F}$ on a domain $\Omega \subset \mathbb{R}^d$ is the data of a norm $\mathcal{F}_z$ on $\mathbb{R}^d$ at each point $z \in \Omega$. Tissot's indicatrix is a convenient way to represent metrics, by showing the associated unit ball $B_z := \{u \in \mathbb{R}^2;\ \mathcal{F}_z(u) \leq 1\}$. This is a standard ball of radius $s(z)$ for Isotropic metrics (i,a); a centered ellipsoid for Riemannian metrics (ii,a); and finally an off-centered disk, or ellipse, for the asymmetric Finsler metrics considered here (iii,a). The Fast Marching algorithm requires to construct stencils satisfying a geometric acuteness property [12]. The standard four point stencil is fine for isotropic metrics (i,b), already available in itk, but more sophisticated constructions are required for Riemannian (ii,b) or Finsler (iii,b) metrics.

The metric $\mathcal{F}$ is the main input of the Fast Marching algorithm. In applications to segmentation, it should be chosen so that the shortest paths align with important features of an image of interest. It may have one of the following forms, which offer various amounts of flexibility:

$$\mathcal{F}_z(u) = \|u\|, \qquad \text{(Euclidean Metric)}$$
$$\mathcal{F}_z(u) = \|u\|/s(z), \qquad \text{(Isotropic Metric)}$$
$$\mathcal{F}_z(u) = \sqrt{\langle u, \mathrm{M}(z)u\rangle}, \qquad \text{(Riemannian Metric)}$$
$$\mathcal{F}_z(u) = \sqrt{\langle u, \mathrm{M}(z)u\rangle} - \langle \omega(z), u\rangle. \qquad \text{(Finsler Metric)}$$

We denoted by $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ the standard euclidean norm and scalar product on $\mathbb{R}^d$.

The Euclidean Metric, the simplest and most common one, can be used to find the shortest path in a domain with obstacles. ITK's implementation of the fast marching algorithm is limited to Isotropic Metrics, determined by a scalar field $s : \Omega \to \mathbb{R}_+^*$. Tissot's indicatrix B_z is then an euclidean ball of radius $s(z)$, see Figure 1 (i). In the min-time interpretation, $s(z)$ is the maximum allowed path speed at point z, in any direction. It is also the front normal propagation speed.

The implemented filter `itk::AnisotropicFastMarchingImageFilter` adds support for Riemannian metrics, in dimension 2 and 3. Tissot's indicatrix B_z is then an ellipsoid, which axes and aspect ratio are determined by the eigenvalues and eigenvectors of $\mathrm{M}(z)$, see Figures 1 (ii) and 2. Riemannian metrics are defined through a field $\mathrm{M} : \Omega \to S_d^+$ of symmetric positive definite matrices. In the min-time interpretation, the maximum allowed path speed at z is larger in the direction of some eigenvectors of $\mathrm{M}(z)$, and smaller in the direction of others.

We also implemented a family of asymmetric Finsler metrics, in 2D only. They combine a symmetric part, determined by a positive definite symmetric matrix $\mathrm{M}(z)$, and an asymmetric part determined by a vector $\omega(z)$. Tissot's indicatrix B_z is again an ellipse, but it is *not centered on the origin* if $\omega(z) \neq 0$, see Figure 1 (iii). The authorized speed $1/\mathcal{F}_z(u)$ in a direction u is thus a-priori different from the speed $1/\mathcal{F}_z(-u)$ in the opposite direction $-u$. The distance associated to a Finsler metric is in general asymmetric: $\mathrm{D}_{\mathcal{F}}(x,y) \neq \mathrm{D}_{\mathcal{F}}(y,x)$. This apparent oddity is entirely legitimate in tasks of motion planning (climbing a hill is harder than going down) and image segmentation (the right and left of an object contour, which correspond to the foreground and background, should have different characteristics), see §3.

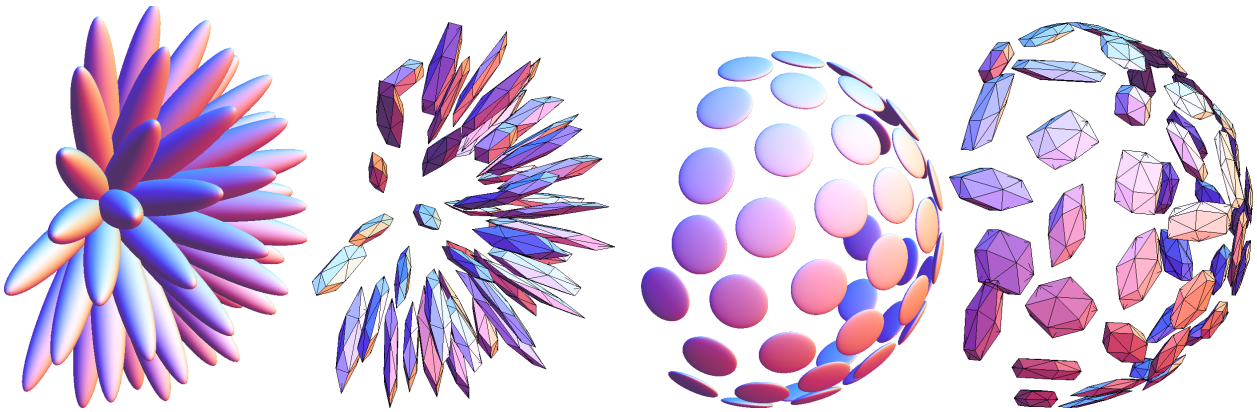

Figure 2: The unit balls $\{u \in \mathbb{R}^3; \mathcal{F}_z(u) \leq 1\}$ associated to a three dimensional Riemannian metric are ellipsoids. They can be needle like (i) or disk like (iii). In order to compute minimal geodesics, the implemented filter discretizes an eikonal PDE using a numerical scheme based on stencils which tend to mimic the ellipsoid shapes (ii) and (iv). They are built of Voronoi vectors with respect to the anisotropic metric [8].

1.1 Anisotropy ratio of the metric

The anisotropy ratio of a norm $\mathcal{F}_z$ and of a metric $\mathcal{F}$ are defined by

$$\kappa(\mathcal{F}_z) := \max_{\|u\|=\|v\|=1} \frac{\mathcal{F}_z(u)}{\mathcal{F}_z(v)}, \qquad \kappa(\mathcal{F}) := \max_{z\in\Omega} \kappa(\mathcal{F}_z).$$

The number $\kappa(\mathcal{F}_z)$ reflects how much the norm $\mathcal{F}_z$ deviates from isotropy, in other words distinguishes between directions u, v, around the point z. Euclidean and isotropic metrics satisfy $\kappa(\mathcal{F}) = 1$ (no local distortion), whereas $\kappa(\mathcal{F}) > 1$ for anisotropic Riemannian and Finsler metrics. Indeed Euclidean and Isotropic metrics treat all directions equally, whereas the newly supported Riemannian and Finsler metrics offer the flexibility to define privileged directions and orientations, see Figure 1.

Our algorithm handles well metrics with anisotropy ratio in the range $\kappa(\mathcal{F}) \in [1,20]$. In the case of a Riemannian metric, the ratio of the largest eigenvalue $\lambda(z)$ to the smallest one $\mu(z)$ of the positive definite tensor $M(z)$ should therefore satisfy $\lambda(z)/\mu(z) \leq 20^2 = 400$, at all points z of the image domain.

The theoretical complexity of our filter is logarithmic in the anisotropy ratio: $O((1+\ln\kappa(\mathcal{F}))^\alpha N \ln N)$, where $\alpha \in [1,3]$ depends on the metric type. In practice, computation time is $3\times$ to $5\times$ longer than with ITK's standard isotropic fast marching, independently of the metric. This is a significant improvement over earlier approaches [12, 2] to anisotropic Fast-Marching, which makes the method practical for applications.

2 Filter Usage

The implemented filter usage involves three successive steps. First: the construction of the metric $\mathcal{F}$, under the form of an image which pixels are norms. Second: the propagation of a front. Third: the extraction of the desired geodesics.

```
// 1. Declare a norm type, and construct a metric.
typedef itk::Riemannian2DNorm<float> NormType; // Alternative types
typedef itk::Image<NormType, NormType::Dimension> MetricType;
MetricType::Pointer metric = MetricType::New();
/* TO DO : Resize, Allocate and Fill the metric as desired */

// 2. Declare and initialize the Fast Marching object
typedef itk::AnisotropicFastMarchingImageFilter<NormType> FMType;

// Front origin is defined in a NodeContainer, inherited from itk::FastMarchingImageFilter
FMType::NodeType node;
FMType::IndexType index; index.Fill(0); const double value=0;
node.SetValue(value); // value and index are user defined.
node.SetIndex(index);

FMType::NodeContainer::Pointer seeds = FMType::NodeContainer::New();
seeds->Initialize();
seeds->InsertElement(0,node); // Several seeds may be inserted

// Fast marching filter
FMType::Pointer fm = FMType::New();
fm->SetGenerateUpwindGradient(true); // Required to extract geodesics
fm->SetInput(metric);
fm->SetTrialPoints(seeds);
fm->Update();
// fm->GetOutput() contains the front arrival times.

// 3. Extract desired geodesics
FMType::PointType start; /* TO DO : Choose geodesic start point coordinates */
std::vector<FMType::PointType> geodesic;
geodesic.push_back(start);
fm->Geodesic(geodesic);
```

Figure 3: Typical filter usage

2.1 Construction of the metric

A metric $\mathcal{F}$ is the data of a norm $\mathcal{F}_z$, at each point z of a domain Ω. It is represented numerically by an image which pixels are norms. The corresponding path speeds are illustrated on Figure 1, and some usage cases are described in §3.

The template parameter `TComponent` should be a floating point type. The mathematical validity of a constructed norm can be tested via the method `IsDefinite()`.

- Norm types `itk::Riemannian2DNorm<TComponent>` and `itk::Riemannian3DNorm<TComponent>`. Used to define two and three dimensional Riemannian metrics, see §1. They publicly inherit from `itk::SymmetricSecondRankTensor<TComponent,VDimension>`, where `VDimension` respectively equals 2 or 3. The norm $F(u) = \sqrt{\langle u, Mu\rangle}$ is valid iff the tensor M is positive definite.

- Norm type `itk::Finsler2DNorm<TComponent>`.
  Used to define two dimensional Finsler metrics, see §1. Data members are a tensor M, of type `itk::SymmetricSecondRankTensor<TComponent,2>`, and a vector ω, of type `itk::Vector<TComponent,2>`, accessible via the methods `GetM()` and `GetOmega()`. The norm $F(u) = \sqrt{\langle u, Mu\rangle} - \langle \omega, u\rangle$ is valid iff M is positive definite, and $\langle \omega, M^{-1}\omega\rangle < 1$.

- Norm type `itk::ExtendedNorm<TPrimaryNorm>`.
  Defines a $d+1$ dimensional norm type, from a d dimensional norm type `TPrimaryNorm`[2]. Data members are a d-dimensional norm F_d, of type `TPrimaryNorm`, and a scalar value α, accessible via the

[2] Only Riemannian2DNorm and Riemannian3DNorm are currently supported as primary norms.

methods `GetPrimaryNorm()` and `GetScalar()`. The $d+1$-dimensional norm F_{d+1} is mathematically defined by

$$F_{d+1}(x_1,\cdots,x_{d+1}) = \sqrt{F_d(x_1,\cdots,x_d)^2 + \alpha x_{d+1}^2}.$$

The norm F_{d+1} is valid iff F_d is valid and α is positive.

2.2 Front propagation, and geodesic extraction

The implemented *anisotropic* Fast Marching class inherits[3] from ITK's classical isotropic Fast Marching class, hence the initialization syntax is identical, see Figure 3.

Our filter implements a fast and robust method for extracting geodesics, by solving the ODE (1), which takes advantage of the upwind gradients generated during the front propagation, see [8] Appendix A. It is parameterless and guaranteed to terminate. Alternatively, one may bypass the proposed method and rely on standard ODE solvers for (1), such as Runge-Kutta methods which are often more accurate but require setting an adequate termination criterion. Given a reference to an `std::vector` containing single continuous index (resp. physical point), the method `Geodesic` appends to the vector a sequence of continuous indices (resp. physical points): the vertices of a polygonal path approximating the minimal geodesic to the closest seed.

2.3 Command line syntax

From the command line, the syntax is `AnisotropicFastMarchingLBR`

- *Metric filename.* The image describing the metric to be used by the Fast-Marching algorithm. An image of symmetric second rank tensors is converted into a Riemannian metric. In general, the metric type is determined by the number of the number of components per pixel.
- *Seeds filename.* This file describes the set X_0 from which the front propagation starts, see (2). It can be a one dimensional image which pixels are points, or a txt file with one point per line, coordinates separated by spaces.
- *Tips filename.* The set of points from which geodesics are to be computed. Same format as the seeds.
- *Output geodesic filename.* Where to store geodesics, with same format as the seeds and tips. In the case of several tips, hence several geodesics, a number is appended to the filename. (e.g. geo.txt $\rightarrow$ geo_1.txt, geo_2.txt, ...)
- *Output image filename (Optional).* Where to store the distance function U, solution of (2).

3 Application examples

We illustrate the flexibility offered by Riemmanian and Finsler metrics in some typical use cases of the Fast Marching algorithm: motion planning, image segmentation, and 3D tubular structure extraction.

[3] More precisely, it inherits from a slight variant `itk::FastMarchingImageFilter_Modified` which essentially does *not* check that `MetricType::PixelType` is convertible to double (this is obviously not true with an anisotropic `NormType`), and declares a protected accessor to the private variable `m_TrialHeap`.

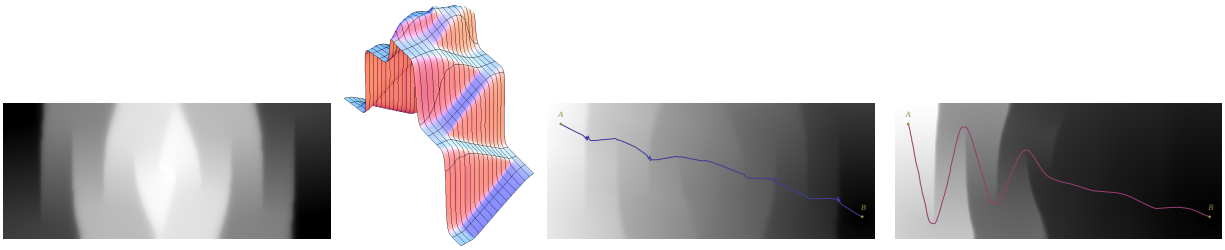

Figure 4: Motion planning example §3.1. (i) Input image, regarded as a height map. (ii) 3D surface view. (iii) Riemannian distance map and shortest path between points A and B (iv) Finsler distance map and shortest path from A to B. The latter path follows the winding road on the way up, but goes straight on the way down, a common strategy for mountain hikers.

In this paper, we limit the discussion to synthetic examples, and provide simple metric constructions which should only be used as an heuristic guide or as a starting point for real applications. Indeed, this section does not intend to provide turnkey solutions for complex imaging problems, but to illustrate the flexibility and opportunities offered by anisotropic fast marching. The construction of adequate metrics for image processing tasks is a research area in itself [11, 3].

Throughout this section let $\Omega \subset \mathbb{R}^k$ be a (box) domain, of dimension $k = 2$, or $k = 3$ in the last example. The letter z stands for a generic point of Ω. The image of interest is denoted by $I : \Omega \to \mathbb{R}$, its gradient direction by $\mathbf{n}$, and the orthogonal direction (in dimension 2) by $\mathbf{t}$: for all $z \in \Omega$

$$\mathbf{n}(z) := \nabla I(z)/\|\nabla I(z)\| \qquad\qquad \mathbf{t}(z) := \mathbf{n}(z)^{\perp}.$$

Finally, Id_k denotes the $k \times k$ identity matrix.

3.1 Motion planning

In this example, we regard the image I as a topographical height map, and aim to compute distances and shortest paths on the parametrized surface $\mathcal{S} \subset \Omega \times \mathbb{R} \subset \mathbb{R}^3$ defined by:

$$\mathcal{S} := \{(z, I(z)); z \in \Omega\}.$$

The geodesic surface distance $U(z) = \mathrm{D}_{\mathcal{S}}(z_0, z)$, from a fixed point $(z_0, I(z_0))$ to $(z, I(z))$ along the surface $\mathcal{S}$, is obtained via an anisotropic fast marching on Ω with seed z_0 and Riemannian metric:

$$\mathrm{M}(z) := \mathrm{Id}_2 + \nabla I(z)\, \nabla I(z)^T = \begin{pmatrix} 1 + \partial_x I(z)^2 & \partial_x I(z)\, \partial_y I(z) \\ \partial_x I(z)\, \partial_y I(z) & 1 + \partial_y I(z)^2 \end{pmatrix}.$$

From the perspective of motion planning, the above Riemannian modeling is not entirely satisfying, because most vehicles, human walkers, or robots, move slower and use more energy when climbing uphill than when going downhill. Finsler metrics can account for such asymmetries. For instance they can model the situation where an agent walks at (euclidean) unit speed in any direction, but also simultaneously slides downhill at some speed $-W(z)$, so that authorized path speeds thus form an off-centered disk: $B_z = \{u - W(z); \|u\| \leq 1\}$, as illustrated on Figure 1 (iv). In general, the norm $\mathcal{F}_z$ such that $B_z = \{u - W; u^T M u \leq 1\}$ is constructed as $\mathcal{F}_z :=$ `Finsler2DNorm::TranslatedEllipse(M, W)`. For illustration we choose

$$W(z) := (1 - s(z))\mathbf{n}(z) \qquad\qquad \text{where } s(z) := \frac{1}{1 + \|\nabla I(z)\|^2},$$

so that the uphill speed, in the direction $\mathbf{n}(z)$, is $s(z)$. See Figure 4.

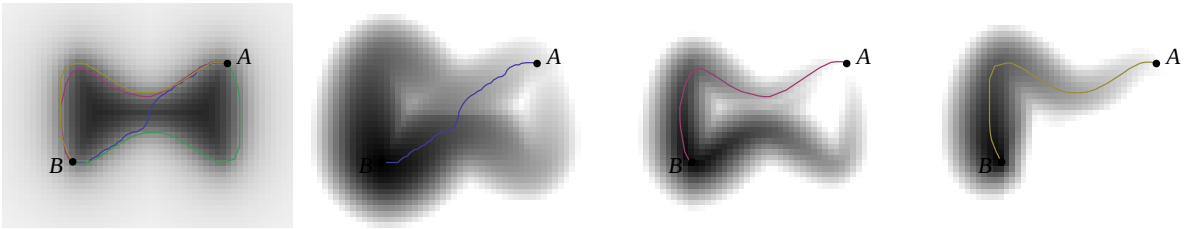

Figure 5: Image Segmentation Example §3.2. (i) Image of interest. Distance map and shortest path from B to A with (i) Isotropic metric, (ii) Riemannian metric (iii) Finsler metric. In the case of the Finsler metric, thanks to asymmetry, the shortest path from A to B (i, green) is distinct from the one from B to A (i and iv, yellow): they correspond to two complementary components the black object (i) boundary.

3.2 Image segmentation

Consider an image, potentially blurred and noisy, of an object which needs to be segmented, represented by $I : \Omega \to \mathbb{R}$. The image gradient, denoised if needed, is strong along the object boundary, and points orthogonally to it. Introduce a speed function increasing with the gradient magnitude: for illustration

$$s(z) := 1 + \|\nabla I(z)\|^2.$$

The object boundary between two given[4] points z_0, z_1, may be extracted using the classical fast marching algorithm as the shortest path between these points with respect to the Isotropic metric $\mathrm{Id}_2 / s(z)^2$.

Anisotropic fast marching allows to improve the robustness and the accuracy of this method. The Riemannian metric defined by the following tensors prescribes a path speed $s(z)$ in the direction $\mathbf{t}(z)$, hence *tangentially* to the object boundary, and in contrast a slower path speed of 1 in the direction $\mathbf{n}(z)$, normal to the object boundary

$$\mathrm{M}(z) := \frac{1}{s(z)^2}\mathbf{t}(z)\mathbf{t}(z)^T + \mathbf{n}(z)\mathbf{n}(z)^T.$$

Finsler metrics, thanks to their asymmetry, can encode the fact that the right and left of the path should have different characteristics, corresponding respectively to the foreground and the background of the segmented object. The metric is defined by the parameters

$$\mathrm{M}(z) := \mathrm{Id}_2, \quad \omega(z) := \left(1 - \frac{1}{s(z)}\right)\mathbf{t}(z).$$

It prescribes a large path speed $1/\mathcal{F}_z(\mathbf{t}(z)) = s(z)$ in the direction of $\mathbf{t}(z)$. In contrast path speed is $1/\mathcal{F}_z(-\mathbf{t}(z)) \approx 1/2$ in the opposite direction $-\mathbf{t}(z)$, and $1/\mathcal{F}_z(\pm\mathbf{n}(z)) = 1$ in the directions $\pm\mathbf{n}(z)$ which are orthogonal to the object boundary. The geodesic active contours introduced in [13, 6] follow a similar idea. Since the constructed metric is asymmetric, the geodesic path $z_0 \to z_1$ is different from the one $z_1 \to z_0$: they correspond to two complementary components the object boundary.

3.3 Tubular structure extraction

Consider an image of dimension $k \in \{2,3\}$, featuring tubular structures (e.g. arteries, axons) which need to be extracted. Assume that a tubularity score $s : \Omega \to \mathbb{R}$, and an field of unit vectors $E : \Omega \to \mathbb{R}^k$ approximating

[4] The provided filter can extract geodesics between prescribed endpoints $z_0, z_1 \in \Omega$, but not closed geodesics, often referred to as active contours, as in [6].

Figure 6: Tubular structure extraction example (3D). (i) Source image. (ii), (iii) Fast Marching front at time $t = 0.7$, and extracted geodesics, with an isotropic metric. (iv), (v) Likewise with the Riemannian metric, at time $t = 1$.

the tube orientation, have been extracted by a preprocessing of the image $I : \Omega \to \mathbb{R}$. Given two points z_0, z_1 at the extremities of a tubular structure, we may extract the centerline as the shortest path with respect to the metric

$$\mathrm{M}(z) := \frac{E(z)E(z)^T}{s(z)^2} + \left(\mathrm{Id}_k - E(z)E(z)^T\right)$$

which encodes a path speed $s(z)$ in the direction of $E(z)$, and an unit speed in orthogonal directions. The alternative isotropic metric, used in Figure 6 (ii), (iii), is simply $\mathrm{Id}_k / s(z)^2$.

Figure 6 illustrates this procedure. Given (i) an image of two fuzzy vessels, we used `itk::HessianRecursiveGaussianImageFilter` to extract the hessian $H(z)$ for all $z \in \Omega$. We let $E(z)$ be the eigenvector associated to the eigenvalue of smallest magnitude, and build[5] an tubularity score $s(z)$ using the eigenvalues of $H(z)$. The anisotropic metric does indeed help the tubular structure extraction, by steering the front evolution in the direction of $E(z)$. As a result, the extracted path is smoother and better aligned with the centerline.

Shortest paths in lifted spaces. Some segmentation methods rely on shortest paths in a $k+1$ dimensional domain $\Omega \times I$. This extended domain is the product of the image domain Ω, with an abstract parameter space I, typically accounting for the radius $r \in [r_*, r^*]$ of some tubular structures [5], or their gray level, or their orientation[6] [10, 1]. The class `ExtendedNorm<TPrimaryNorm>` is intended for this use case.

Conclusion and perspectives

We implemented an Anisotropic Fast Marching Image Filter in ITK, which allows to compute distance maps and minimal geodesics with respect to Riemannian and Finsler metrics, in addition to Isotropic metrics already supported by the subclass `itk::FastMarchingImageFilter`. Numerous applications, including motion planning, image segmentation, and tubular structure extraction, can take advantage of this additional flexibility to provide more robust or accurate results. Computation times are $3\times$ to $5\times$ longer than with the base class, which we regard as a tolerable tradeoff in view of the added functionality. Future work will be devoted to specific metric constructions, and to geodesic active contours.

[5] Specifically, we let $s(z) := \exp((|\lambda_2| - |\lambda_1|)/(|\lambda_1| + \delta))$ where $\lambda_1, \lambda_2, \lambda_3$ are the eigenvalues of $H(z)$, sorted so that $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$. On the tube centerline one has $|\lambda_1| \ll |\lambda_2|$, and the eigenvector associated to λ_1 points in the tube direction. Parameters $\delta = 0.5$, and $\sigma = 0.3$ (for the hessian recursive gaussian filter).

[6] Periodic boundary conditions, which would be adequate suitable the orientation space $[0, \pi[$, are not supported at this time.

References

[1] E J Bekkers, Remco Duits, A Mashtakov, and G R Sanguinetti. Data-driven Sub-Riemannian Geodesics in SE (2). *preprint*. 3.3

[2] Folkmar Bornemann and Christian Rasch. Finite-element Discretization of Static Hamilton-Jacobi Equations based on a Local Variational Principle. *Computing and Visualization in Science*, 2006. 1.1

[3] Da Chen, Laurent D. Cohen, and Jean-Marie Mirebeau. Vessel Extraction Using Anisotropic Minimal Paths and Path Score. In *IEEE International Conference on Image Processing (ICIP 2014)*, 2014. 3

[4] Jérôme Fehrenbach and Jean-Marie Mirebeau. Sparse Non-negative Stencils for Anisotropic Diffusion. *Journal of Mathematical Imaging and Vision*, 2013. (document)

[5] H Li and A Yezzi. Vessels as 4-D curves: Global minimal 4-D paths to extract 3-D tubular surfaces and centrelines. *IEEE Transactions on Medical Imaging*, 2007. 3.3

[6] J Melonakos, E Pichon, S Angenent, and A Tannenbaum. Finsler Active Contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008. 3.2, 4

[7] Jean-Marie Mirebeau. Efficient fast marching with Finsler metrics. *Numerische Mathematik*, 2013. (document)

[8] Jean-Marie Mirebeau. Anisotropic Fast-Marching on cartesian grids using Lattice Basis Reduction. *SIAM Journal on Numerical Analysis*, 2014. (document), 2, 2.2

[9] Jean-Marie Mirebeau, Jérôme Fehrenbach, Laurent Risser, and Shaza Tobji. Anisotropic Diffusion in ITK. *The Insight Journal*, 2015. (document)

[10] M Pechaud, R Keriven, and G Peyré. Extraction of tubular structures over an orientation domain. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, 2009. 3.3

[11] Gabriel Peyré, Mickael Péchaud, Renaud Keriven, and Laurent D. Cohen. *Geodesic Methods in Computer Vision and Graphics*. 3

[12] James A. Sethian and Alexander Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms. *SIAM Journal on Numerical Analysis*, 2003. 1, 1.1

[13] Christopher Zach, Liang Shan, and Marc Niethammer. *Globally Optimal Finsler Active Contours*. Springer Berlin Heidelberg, 2009. 3.2