Geodesic Methods for Shape and Surface Processing

Gabriel Peyré and Laurent Cohen

Ceremade, UMR CNRS 7534, Université Paris-Dauphine,
75775 Paris Cedex 16, France
{peyre,cohen}@ceremade.dauphine.fr,
WWW home page: http://www.ceremade.dauphine.fr/{~peyre/,~cohen/}

Abstract. This paper reviews both the theory and practice of the numerical computation of geodesic distances on Riemannian manifolds. The notion of Riemannian manifold allows to define a local metric (a symmetric positive tensor field) that encodes the information about the problem one wishes to solve. This takes into account a local isotropic cost (whether some point should be avoided or not) and a local anisotropy (which direction should be preferred). Using this local tensor field, the geodesic distance is used to solve many problems of practical interest such as segmentation using geodesic balls and Voronoi regions, sampling points at regular geodesic distance or meshing a domain with geodesic Delaunay triangles. The shortest path for this Riemannian distance, the so-called geodesics, are also important because they follow salient curvilinear structures in the domain. We show several applications of the numerical computation of geodesic distances and shortest paths to problems in surface and shape processing, in particular segmentation, sampling, meshing and comparison of shapes.

1 Manifold Geometry of Surfaces

In [1], it was shown that finding the weighted distance and geodesic paths to a point leads to fast algorithms for image segmentation. In this chapter, we give a more general framework that is illustrated by different important applications.

This section introduces some basic definitions about local metric (a tensor field) on a Riemannian manifold and the associated notion of geodesic distance and minimal paths. The important point is that the geodesic distance to a set of starting points satisfies a non-linear differential equation, the Eikonal equation, which is solved to compute numerically the geodesic distance.

1.1 Riemannian Manifold

Parametric surface. A parameterized surface embedded in Euclidean space $\mathcal{M} \subset \mathbb{R}^k$ is a mapping

$$u \in \mathcal{D} \subset \mathbb{R}^2 \mapsto \varphi(u) \in \mathcal{M}.$$

This definition can be extended to include surfaces not topologically equivalent to a disk, by considering a set of charts $\{\mathcal{D}_i\}_i$ that overlap in a smooth manner.

A curve is defined in parameter domain as a 1D mapping $t \in [0,1] \mapsto \gamma(t) \in \mathcal{D}$. This curve can be traced over the surface and its geometric realization is $\bar{\gamma}(t) \stackrel{\text{def.}}{=} \varphi(\gamma(t)) \in \mathcal{M}$. The computation of the length of γ in ambient k-dimensional space $\mathbb{R}^k$ follows the usual definition, but to do the computation over the parametric domain, one needs to use a local metric (the first fundamental form) defined as follow.

Definition 1 (First fundamental form)**.** *For a parametric surface φ, one defines*

$$I_\varphi = \left(\langle \frac{\partial \varphi}{\partial u_i}, \frac{\partial \varphi}{\partial u_j} \rangle \right)_{i,j=1,2}.$$

This local metric I_φ defines at each point the infinitesimal length of a curve

$$L(\gamma) \stackrel{\text{def.}}{=} \int_0^1 \|\bar{\gamma}'(t)\| \mathrm{d}t = \int_0^1 \sqrt{\gamma'(t)^{\mathrm{T}} I_\varphi(\gamma(t)) \gamma'(t)} \mathrm{d}t.$$

This fundamental form is an intrinsic invariant that does not depend on how the surface is isometrically embedded in space (since the lengths depend only on this tensor field I_φ). In contrast, higher order differential quantities such as curvature might depend on the bending of the surface and are thus usually not intrinsic (with the notable exception of invariants such as the gaussian curvature).

Riemannian manifold. A parameterized surface is embedded into some Euclidean domain $\mathbb{R}^k$, which allows to define a local metric thanks to the first fundamental form I_φ. It is however possible to consider directly a field of positive definite tensors on a parametric domain $\mathcal{D} = \mathbb{R}^s$ (in practice here $s = 2$ for surfaces or $s = 3$ for volumes). With a slight abuse in notations, we assimilate the resulting abstract surface $\mathcal{M}$ with $\mathcal{D}$. Once again, we consider only surfaces globally parameterized by some Euclidean domain $\mathcal{D}$ and handling generic surfaces requires to split the manifold into overlapping charts.

Definition 2 (Riemannian manifold)**.** *A Riemannian manifold is an abstract parametric space $\mathcal{M} \subset \mathbb{R}^s$ equipped with a metric $x \in \mathcal{M} \mapsto H(x) \in \mathbb{R}^{s \times s}$ positive definite.*

Using the Riemannian metric, one can compute the length of a piecewise smooth curve $\gamma : [0,1] \to \mathcal{M}$

$$L(\gamma) \stackrel{\text{def.}}{=} \int_0^1 \sqrt{\gamma'(t)^{\mathrm{T}} H(\gamma(t)) \gamma'(t)} \mathrm{d}t.$$

At each location x, the Riemannian tensor can be diagonalized as follow

$$H(x) = \lambda_1(x) e_1(x) e_1(x)^{\mathrm{T}} + \lambda_2(x) e_2(x) e_2(x)^{\mathrm{T}} \quad \text{with} \quad 0 \leqslant \lambda_1 \leqslant \lambda_2, \qquad (1)$$

and e_1, e_2 are two orthogonal eigenvector fields. In fact, e_i should be understood as direction (un-oriented) field since both e_i and $-e_i$ are eigenvectors of the tensor. A curve γ passing at location $\gamma(t) = x$ with speed $\gamma'(t)$ has a shorter local length if $\gamma'(t)$ is colinear to $e_1(x)$ rather than any another direction. Hence shortest paths (to be defined in the next section) tend to be tangent to the direction field e_1.

In practice, the Riemannian metric H is given by the problem one wishes to solve. In image processing, the manifold is the image domain $\mathcal{M} = [0,1]^2$ equipped with a metric derived from the image (for instance its gradient). Figure 1 shows some some frequently used geodesic metric spaces:

- *Euclidean space:* $\mathcal{M} = \mathbb{R}^s$ and $H(x) = \mathrm{Id}_s$.
- *2D shape:* $\mathcal{M} \subset \mathbb{R}^2$ and $H(x) = \mathrm{Id}_2$.
- *Isotropic metric:* $H(x) = W(x)\mathrm{Id}_s$, $W(x) > 0$ being some weight function.
- *Parametric surface:* $H(x) = I_\varphi(x)$ is the first fundamental form.
- *Image processing:* given an image $I : [0,1]^2 \to \mathbb{R}$, one can use an edge-stopping weight $W(x) = (\varepsilon + \|\nabla_x I\|)^{-1}$. This way, geodesic curves can be used to perform segmentation since they will not cross boundaries of the objects.
- *DTI imaging:* $\mathcal{M} = [0,1]^3$, and $H(x)$ is a field of diffusion tensors acquired during a scanning experiment. For DTI imaging, the direction field e_1 indicates the direction of elongated fibers of the white matter (see [2]).

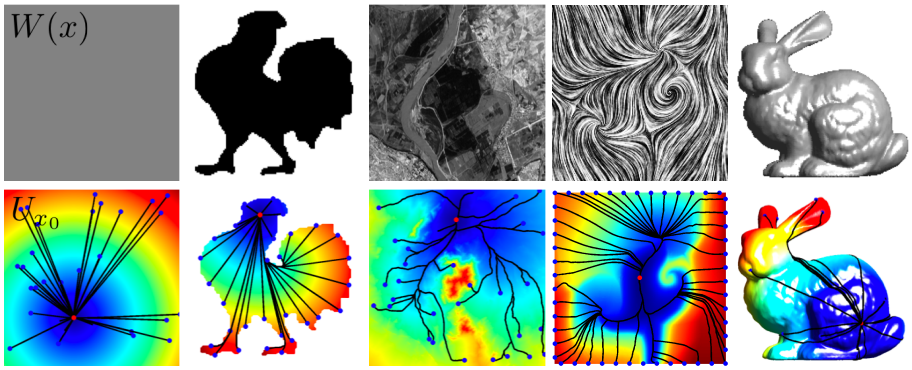


Fig. 1. *Examples of Riemannian metrics (top row) and geodesic distances and curves (bottom row). The blue/red colormap indicates the geodesic distance to the starting point. From left to right: euclidean ($H(x) = \mathrm{Id}_2$ restricted to $\mathcal{M} = [0,1]^2$), planar domain ($H(x) = \mathrm{Id}_2$ restricted to $\mathcal{M} \neq [0,1]^2$), isotropic ($H(x) = W(x)^2\mathrm{Id}_2$ with W computed from the image using (6)), Riemannian manifold metric ($H(x)$ is the structure tensor of the image, see equation (8)) and 3D surface ($H(x)$ corresponds to the first fundamental form).*

The anisotropy of a metric $H(x)$ is defined as

$$\alpha(x) = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} = 2\frac{\sqrt{ab - c^2}}{a + b} \in [0, 1], \qquad \text{for} \qquad H(x) = \begin{pmatrix} a & c \\ c & b \end{pmatrix}. \tag{2}$$

A metric with $\alpha(x)$ close to 1 is highly directional near x, whereas a metric with $\alpha(x) = 1$ is locally isotropic near x.

1.2 Geodesic Distances

The local Riemannian metric $H(x)$ allows to define a global metric on the space $\mathcal{M}$ using shortest paths. This corresponds to the notion of geodesic curves.

Definition 3 (Geodesic distance)**.** *Given some Riemannian space $(\mathcal{M}, H)$ with $\mathcal{M} \subset \mathbb{R}^s$, the geodesic distance is defined as*

$$\forall\, (x, y) \in \mathcal{M}^2, \quad d_{\mathcal{M}}(x, y) \overset{\text{def.}}{=} \min_{\gamma \in \mathcal{P}(x,y)} L(\gamma)$$

where $\mathcal{P}(x, y)$ denotes the set of piecewise smooth curves joining x and y

$$\mathcal{P}(x, y) \overset{\text{def.}}{=} \{\gamma \setminus \gamma(0) = x \quad \textit{and} \quad \gamma(1) = y\}.$$

The shortest path between two points according to the Riemannian metric is called a geodesic. If the metric H is well chosen, then geodesic curves can be used to follow salient features on images and surfaces.

Definition 4 (Geodesic curve)**.** *A geodesic curve $\gamma \in \mathcal{P}(x, y)$ is such such that $L(\gamma) = d_{\mathcal{M}}(x, y)$.*

A geodesic curve between two points might not be unique, think for instance about two anti-podal points on a sphere. In order to perform the numerical computation of geodesic distances, we fix a set of starting points $\mathcal{S} = (x_k)_k \subset \mathcal{M}$ and consider only distance and geodesic curves from this set of points.

Definition 5 (Distance map)**.** *The distance map to a set of starting points $\mathcal{S} = (x_k)_k \subset \mathcal{M}$ is defined as*

$$\forall\, x \in \mathcal{M}, \quad U_{\mathcal{S}}(x) \overset{\text{def.}}{=} \min_k d(x, x_k).$$

The main theorem that characterizes the geodesic distance is the following, that replaces the optimization problem of finding the minimum distance by a non-linear partial differential equation.

Theorem 1 (Eikonal equation)**.** *If the metric H is continuous, then for any $\mathcal{S} \subset \mathcal{M}$, the map $U_{\mathcal{S}}$ is the unique viscosity solution of the Hamilton-Jacobi equation*

$$\|\nabla_x U_{\mathcal{S}}\|_{H(x)^{-1}} = 1 \qquad \textit{with} \qquad \forall\, k, \quad U_{\mathcal{S}}(x_k) = 0, \tag{3}$$

where $\|v\|_A = \sqrt{v^{\mathrm{T}} A v}$.

It is important to notice that, even if the metric $x \mapsto H(x)$ is a smooth function, the distance function $U_{\mathcal{S}}$ might not be smooth (it exhibit gradient discontinuities). This is why the machinery of viscosity solution is needed to give a sense to the solution of the Hamilton-Jacobi equation. See for instance [3] for an introduction to viscosity solutions.

Once the distance map $U_{\mathcal{S}}$ has been computed by solving the Eikonal equation (3), one can extract a geodesic joining any point x to its closest point $x_k \in \mathcal{S}$ using a gradient descent on the function $U_{\mathcal{S}}$.

Theorem 2 (Gradient descent)**.** *The geodesic curve γ between x and its closest point in $\mathcal{S}$ solves*

$$\gamma'(t) = -\frac{H(\gamma(t))^{-1}\nabla_{\gamma(t)}U_{\mathcal{S}}}{\|H(\gamma(t))^{-1}\nabla_{\gamma(t)}U_{\mathcal{S}}\|} \qquad \textit{with} \qquad \gamma(0) = x.$$

The geodesic curve γ extracted using this gradient descent is parameterized with unit speed since $\|\gamma'\| = 1$, so that $\gamma : [0, T] \to \mathcal{M}$ where $T = d_{\mathcal{M}}(x, x_k)$.

Figure 2 shows examples of geodesic curves computed from a single starting point $\mathcal{S} = \{x_1\}$ in the center of the image $\mathcal{M} = [0, 1]^2$ and a set of points on the boundary of $\mathcal{M}$. The geodesics are computed for a metric $H(x)$ whose anisotropy $\alpha(x)$ (defined in equation (2)) is decreasing, thus making the Riemannian space progressively closer to the Euclidean space.

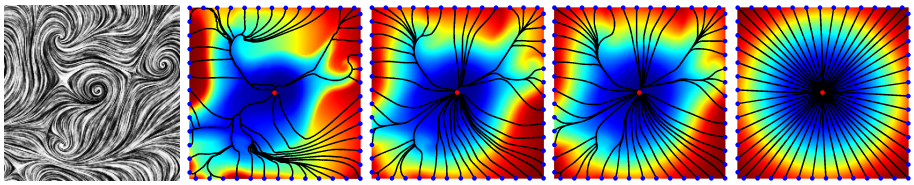


Fig. 2. *Examples of geodesics for a tensor metric with an decreasing anisotropy α (see equation (2) for a definition of this parameter). The tensor field $H(x)$ is computed from the structure tensor of f as defined in equation (8), its eigenvalues fields $\lambda_i(x)$ are then modified to impose the anisotropy α.*

For the particular case of an isotropic metric $H(x) = W(x)^2\mathrm{Id}_2$, the geodesic distance and the shortest path satisfies

$$\|\nabla_x U_{\mathcal{S}}\| = W(x) \qquad \text{and} \qquad \gamma'(t) = -\frac{\nabla_x U_{\mathcal{S}}}{\|\nabla_x U_{\mathcal{S}}\|}. \tag{4}$$

This corresponds to the Eikonal equation, that has been used to compute minimal paths weighted by W [4].

2 Numerical Computations of Geodesic Distances

In order to make all the previous definitions effective in practical situations, one needs a fast algorithm to compute the geodesic distance map $U_{\mathcal{S}}$. This

section details Fast Marching algorithms based on front propagation that enable to compute the distance map by propagating the distance information from the starting points in $\mathcal{S}$.

The basic Fast Marching algorithm and several extensions are exposed in the book on Fast Marching methods [5]. For other applications to computer graphics and image processing one can see [6] and [7]. The recent book [8] treats all the details of the geometry of non-rigid surfaces, including geodesic distance computation and shape comparison. One can also see the two books [9, 10] that contain review articles with some applications of Fast Marching and geodesic methods and in particular [1].

2.1 Front Propagation Algorithms

Depending on the properties of the metric, one needs to consider several algorithms, that all rely on the idea of front propagation. This family of algorithms allows to sort the computations in such a way that each point of the discretization grid is visited only once. This ordering is feasible for distance computation because the distance value of a grid point only depends (and can be computed) from a small number of points having only smaller distances. If one can sort the grid points with increasing distance, then one gets a coherent ordering of the computations. Of course, this is not that easy since this distance ordering would require the knowledge of the solution of the problem (the distance itself). But depending on the application, it is possible to devise a selection rule that actually select at each step the correct grid point.

A front propagation labels the points of the grid according to a state

$$S(x) \in \{Computed, Front, Far\}.$$

During the iterations of the algorithm, a point can change of label according to

$$Far \mapsto Front \mapsto Computed.$$

Computed points $S(x) = Computed$ are those that the algorithm will not consider any more (the computation of $U_{\mathcal{S}}(x)$ is done for these points). Front points $S(x) = Front$ are the points being processed (the value of $U(x) \approx U_{\mathcal{S}}(x)$ is well defined but might change in future iterations). Far points $S(x) = Far$ are points that have not been processed yet.

In practice, a front propagation algorithm requires three key ingredients:

- Given a point x in the grid, a local set of neighbors $Neigh(x)$ connected to x.
- A priority $\mathcal{P}(x)$ among points x in the front, that allows to select the point to process at a given iteration. In most application, this priority is computed as the current value of the distance $\mathcal{P}(x) \stackrel{\text{def.}}{=} U(x)$. Section 3.2 shows how to change this priority in order to speed up computations.
- A procedure $x \mapsto Update(x) \in \mathbb{R}$ that computes the distance value $U(x)$ approximating $U_{\mathcal{S}}(x)$ knowing the value $U(x)$ for computed point and an approximate value for points in the front. This procedure usually solves some kind of equation that discretizes the Eikonal equation (3) one wishes to solve.

Listing 1 gives the details of the front propagation algorithm that computes a distance map U approximating $U_{\mathcal{S}}(x)$ on a discrete grid. The following section details for actual implementations of the $Update$ procedure for different metrics.

The numerical complexity of this scheme is $O(n \log(n))$ for a discrete set of n points. This is because all the points are visited (tagged *Computed*) once, and the selection of $\min \mathcal{P}$ from the front points takes at most $\log(n)$ operations with a special heap data structure (although in practice it takes much less and the algorithm is nearly linear in time).

1. *Initialization:* $\forall\, x \in \mathcal{S}, U(\mathcal{S}) \leftarrow 0$, $S(x) \leftarrow Front$, $\forall\, y \notin \mathcal{S},\ S(y) \leftarrow Far$.
2. *Select point:* $x \longleftarrow \underset{S(z)=Front}{\operatorname{argmin}} \mathcal{P}(z)$.
3. *Tag:* $S(x) \leftarrow Computed$.
4. *Update neighbors:* for all $y \in Neigh(x)$,
   – If $S(y) = Far$, then $S(y) \leftarrow Front$ and $U(y) \leftarrow Update(y)$.
   – If $S(y) = Front$, then $U(y) \leftarrow \min(U(y), Update(y))$.
   – Recompute the priority $\mathcal{P}(y)$.
5. *Stop:* If $x \neq x_1$, go back to 2.

Table 1: Front propagation algorithm.

2.2 Eikonal Equation Discretization

On a square grid. The classical Fast Marching algorithm, introduced by Sethian [5], is a fast procedure to solve the Eikonal equation (3) for an isotropic metric $H(x) = W(x)^2 \mathrm{Id}_s$ for a uniform regular grid that discretizes $[0,1]^s$. We recall this procedure for a planar domain $s = 2$ although it can be extended to any dimension.

In order to capture the viscosity solution of an Hamilton Jabobi equation, one cannot use standard finite differences because of the apparition of shocks and singularities in the solution of the equation. One needs to choose, at each grid point, the optimal finite difference scheme (differentiation on the left or on the right to approximate d/dx for instance). This optimal differentiation should be chosen in the direction where the solution of the equation decreases. This is called an upwind finite difference scheme, and on a 2D grid with spacing h it leads to find $u = Update(x)$ at a grid point $x = x_{i,j}$ that is the smallest solution of

$$\begin{aligned} &\max(u - U(x_{i-1,j}),\, u - U(x_{i+1,j}), 0)^2 + \\ &\max(u - U(x_{i,j-1}),\, u - U(x_{i,j+1}), 0)^2 = h^2 W(x_{i,j})^2. \end{aligned} \tag{5}$$

The smallest solution of this equation leads to a stable and convergent scheme that can be used in the front propagation algorithm listing 1.

On a triangulation. The classical Fast Marching algorithm is restricted to isotropic metrics on a regular grid. This setting is useful for image and volumetric data processing, but in order to deal with arbitrary Riemmanian surfaces embedded in $\mathbb{R}^k$, one needs to modify equation (5).

Kimmel and Sethian [11] have developed a version of the Fast Marching algorithm for a surface $\mathcal{M} \subset \mathbb{R}^k$ with metric $W(x)$ for x in embedding space $\mathbb{R}^k$. In the continuous setting, a parametric surface $(\mathcal{M}, \varphi)$ embedded with a metric $W(x)$ in ambient space corresponds to a Riemannian manifold with a metric $I_\varphi(\bar{x})W(\varphi(\bar{x}))$ in parameter space $\bar{x} \in \mathbb{R}^2$.

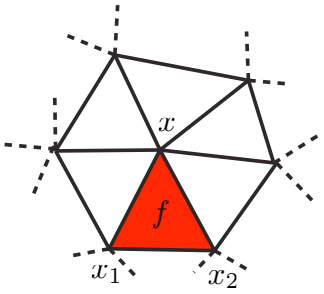
The algorithm of Kimmel and Sethian works on a triangulated mesh and treats the triangles of this mesh as locally flat and equipped with an isotropic metric $W(x)^2$. The same algorithm can be used to process an anisotropic metric $H(x) \in \mathbb{R}^{2\times 2}$ defined on a square lattice (an image), by locally connecting a pixel x to its 4 direct neighbors in order to create 4 adjacent triangles (that are flat). In order to describe the algorithm for these two settings (curved triangulated surface embedded in $\mathbb{R}^k$ and Riemannian manifold with arbitrary metric $H(x)$ for $x \in \mathbb{R}^2$), we solve the Eikonal equation

$$\|\nabla_x U\|_{H^{-1}(x)} = W(x)$$

locally on the triangle faces $f \in F_x$ adjacent to x in order to compute $Update(x) \approx U(x)$.

In order to compute the update value at a given vertex x, the algorithm computes an update value $Update_f(x)$ for each triangle $f \in F_x$ in the face 1-ring around x, $F_x = \{f_1, \ldots, f_k\}$. The resulting Fast Marching update step is defined as

$$Update(x) = \min_{f \in F_x} Update_f(x).$$



In order to derive the expression for $Update_f(x)$, one considers a planar triangle $f = (x, x_1, x_2)$ and denotes $X = (x_1 - x, x_2 - x) \in \mathbb{R}^{2\times 2}$. The known distances are $u = (U(x_1), U(x_2))^{\mathrm{T}} \in \mathbb{R}^2$ and one wishes to solve for $Update_f(x) = p = U(x)$.

The linear interpolation of $U_{\mathcal{S}}$ can be written at the point x_1 and x_2 as

$$\text{for } i \in \{1,2\}, \quad U_{\mathcal{S}}(x_i) \approx \langle g,\, x_i - x \rangle + p \qquad \text{where} \qquad g \approx \nabla_x U_{\mathcal{S}}.$$

With this approximation, equation (3) leads to a quadratic equation

$$\begin{cases} U = X^{\mathrm{T}} g + p\mathbb{I} \\ \|g\|^2_{H^{-1}(x)} = W(x)^2 \end{cases} \quad \Longrightarrow \quad \mathbb{I}^{\mathrm{T}} Q \mathbb{I} p^2 + 2(\mathbb{I}^{\mathrm{T}} Q u)p + (u^{\mathrm{T}} Q u - W(x)^2) = 0.$$

where $\mathbb{I} = (1,1)^{\mathrm{T}} \in \mathbb{R}^2$ and $Q = (XH(x)^{-1}X^{\mathrm{T}})^{-1} \in \mathbb{R}^{2\times 2}$. The only admissible solution to this problem is

$$Update_f(x) = p = \frac{\mathbb{I}^{\mathrm{T}} Q u + \sqrt{(\mathbb{I}^{\mathrm{T}} Q d)^2 + \mathbb{I}^{\mathrm{T}} Q \mathbb{I}(u^{\mathrm{T}} Q u - W(x)^2)}}{\mathbb{I}^{\mathrm{T}} Q \mathbb{I}}$$

There is some technical difficulties with this scheme on triangulations that contain obtuse angles or with metric $H(x)$ with a large anisotropy, because the

update procedure might not be monotone anymore. More accurate monotone schemes have been developped, see for instance [12, 13, 2]. We shall ignore these difficulties here and focus on the application of the numerical computation of geodesic distances.

2.3 Examples of propagations

2D isotropic propagation on a square grid. Figure 3 shows some examples of front propagation with the Fast Marching, for an isotropic $H(x) = W(x)^2 \mathrm{Id}_2$. The colored area shows, at some given step of the algorithm, the set of computed points (its boundary being the set of front points). During the iterations, the front propagates outwards until all the grid points are visited. The numerical complexity of this scheme is $O(n \log(n))$ for a grid of n points.

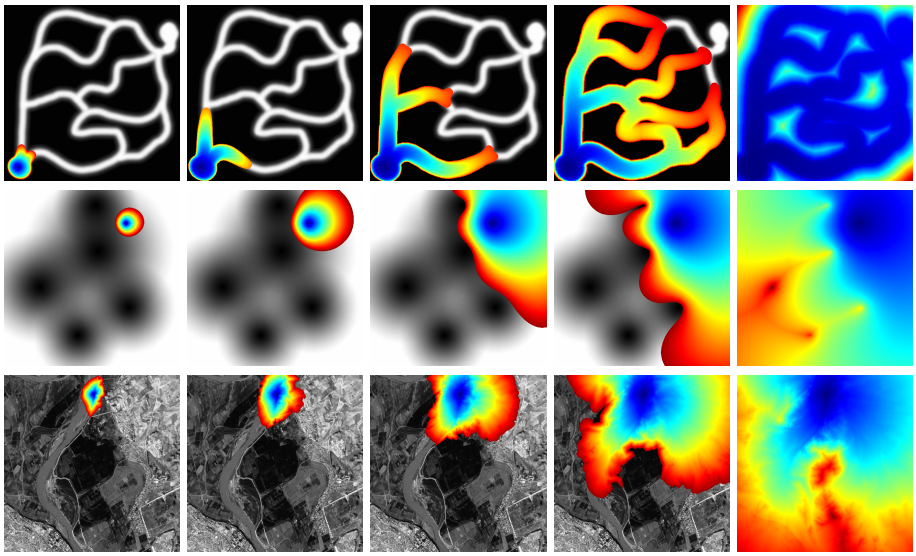


Fig. 3. *Examples of isotropic front propagation. The colormap indicates the values of the distance functions at a given iteration of the algorithm. On rows 1 and 2, the potential W is computed using $W(x) = f(x)$ so that geodesics tend to follow bright regions. On row 3, the potential W is computed using* (6) *where c is chosen to match the intensity of the road to extract.*

Figure 4 shows examples of distance functions to a starting point x_0 with the corresponding geodesics $\gamma(t)$ extracted from some ending point x_1. The front propagation is stopped when $S(x_1) = \mathit{Computed}$ to avoid performing useless computations. The idea of using geodesics in order to extract salient curves in images as been introduced in [14].

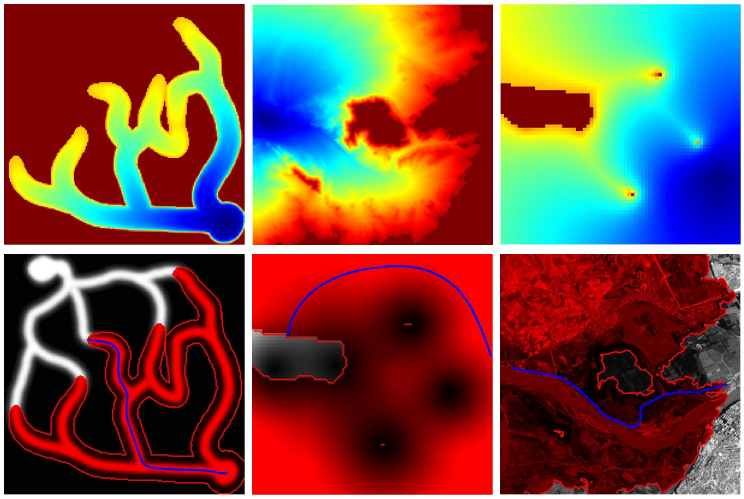


Fig. 4. *Example of distance functions (top row) and geodesics (bottom row).*

In practice, the difficult task is to design a metric W in order have meaningful geodesics. Here are some examples of possible choices, for image processing with an input image f:

– *Pixel value based potential:* in many applications, one simply wishes to extract curves with a constant value c. In this case, one can use a potential like

$$W(x) = \frac{1}{\varepsilon + |f(x) - c|}. \tag{6}$$

Figure 4, left and middle, shows examples of such curves extractions. Also in many applications related to segmentation of tubular shapes, like vessels, we are looking for curves that are located in brighter or darker regions. In this case the potential can be chosen respectively as

$$W(x) = f(x) \qquad \text{or} \qquad W(x) = -f(x) \tag{7}$$

and W should be rescaled to fill the range $[\varepsilon, 1]$.

– *Gradient-based potential:* for application such as edge detection one would like the geodesics to follow regions with high gradients. One can choose a potential such as

$$W(x) = \varepsilon + G_\sigma * \|\nabla_x f\|,$$

where G_σ is a smoothing kernel.

3D isotropic propagation on a square grid. The Fast Marching works the same way in any spacial dimension k and in particular can be used to extract shortest paths in 3D volumetric medical data. Such a volume is a discretization of a mapping $f : [0, 1]^3 \mapsto \mathbb{R}$. Figure 5 shows a 3D display with a semi-transparent mapping that removes more or less parts of the data. The transparency at point

(x, y, z) is defined as $\rho(f(x, y, z))$ where $\rho : [f_{\min}, f_{\max}] \rightarrow [0, 1]$ is the α-mapping. Figure 6 shows in red the front of the Fast Marching propagation, displayed as an isosurface of $U_{\mathcal{S}}$.

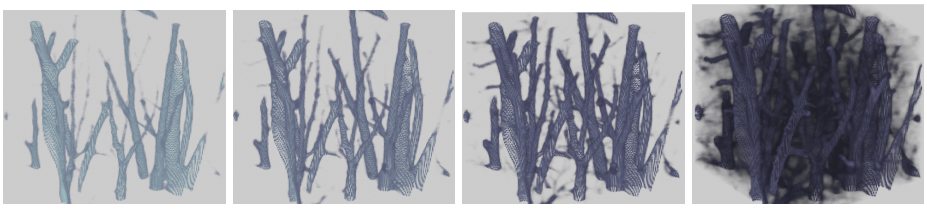


Fig. 5. *Example of semi-transparent display of volumetric data..*

Figure 6 shows some examples of geodesic extraction on a medical image that represents tubular structures (blood vessels) around the heart. The potential $W(x)$ is chosen as $W(x) = (|f(x) - f(x_0)| + \varepsilon)^{-1}$ where x_0 is a point given by the user and supposed to lie inside some vessel. A geodesic follows nicely a vessel since its density is constant and thus the value of f is approximately equal to $f(x_0)$ inside the vessel. Figure 7 shows other application of shortest path to extract tubular structures and centerlines in 3D medical data [15].

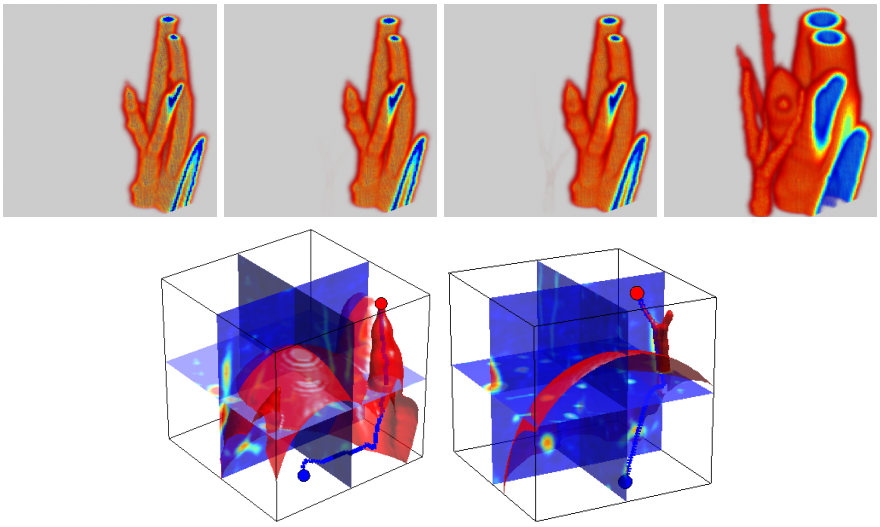


Fig. 6. *Example of volumetric Fast Marching evolution (top row) and geodesic extractions (bottom row).*

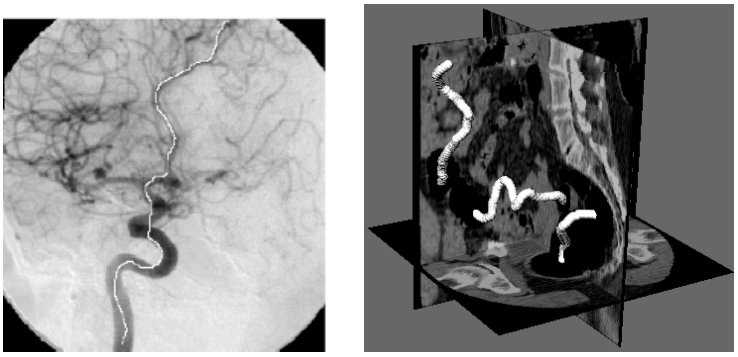


Fig. 7. *Left: vessel extraction. Right: tubular structure extraction.*

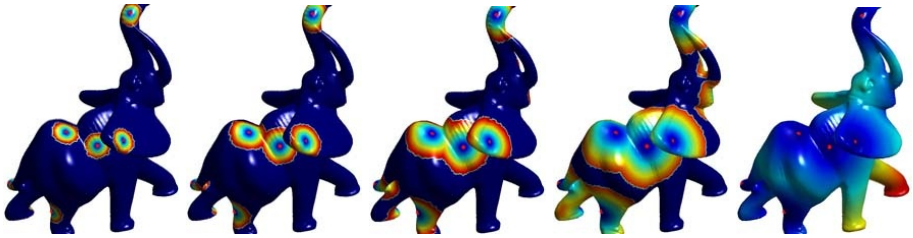


Fig. 8. *Example of Fast Marching propagation on a triangulated mesh.*

Isotropic propagation on a triangulated mesh. Figure 8 shows an example of propagation on a triangulated surface. The colored region corresponds to the points that are computed (its boundary being the front).

The propagation can be started from several starting points $\mathcal{S} = (x_k)_k$ in order to compute the geodesic distance map $U_{\mathcal{S}}$. Figure 9 shows examples of such distances to several points together with geodesics. A geodesic γ links a point x to its closest point in $\mathcal{S}$.

Anisotropic propagation on a square grid. In order to better follow the salient structures of an image f, one can replace the isotropic metric $H(x) = W(x)^2 \mathrm{Id}_s$ (examples are given here in $s = 2$ dimensions) by a fully anisotropic metric $H(x) \in \mathbb{R}^{2\times 2}$ which is a symmetric tensor field. This field might be given by the physical problem, such as the tensor field of DTI imaging [2]. Another option is to infer this field from some input image f.

The local orientation of a feature around a pixel x is given by the vector orthogonal to the gradient $v(x) = (\nabla_x f)^{\perp}$, which is computed numerically with finite differences (using maybe some little smoothing to cancel noise). This local direction information can be stored in a rank-1 tensor $T_0(x) = v(x)v(x)^{\mathrm{T}}$. In order to evaluate the local anisotropy of the image, one needs to average this tensor

$$T(x) = T_0 * G_\sigma(x) \tag{8}$$

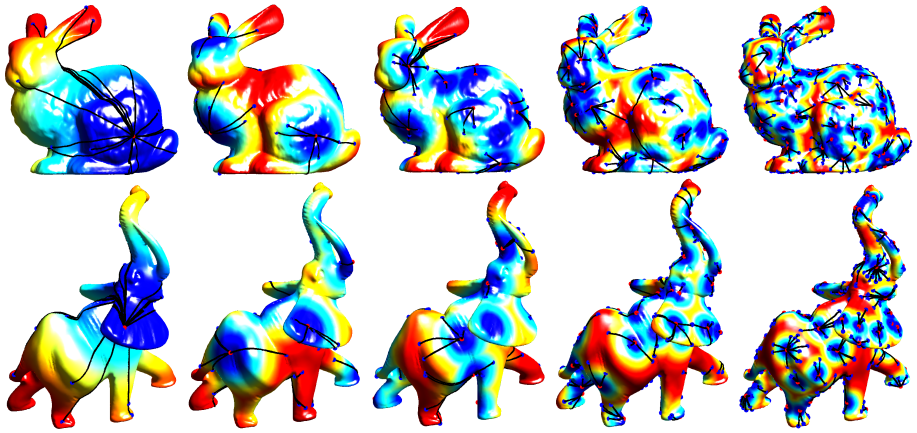


Fig. 9. *Examples of geodesic extraction on a mesh with an increasing number of starting points.*

where the 4 entries of the tensor are smoothed against a gaussian kernel G_σ of width $\sigma > 0$. The metric H corresponds to the so-called structure tensor, see for instance [16]. This local tensor T is able to extract both the local direction of edges and the local direction of textural patterns (see figure 11, left). Another option, that we do not pursue here, is to use the square of the Hessian matrix of f instead of the structure tensor.

In order to turn the structure tensor into a Riemannian metric, one can apply a non-linear mapping to the eigenvalues,

$$T(x) = \mu_1 e_1 {e_1}^{\mathrm{T}} + \mu_2 e_2 {e_2}^{\mathrm{T}} \quad \Longrightarrow \quad H(x) = \psi_1(\mu_1) e_1 {e_2}^{\mathrm{T}} + \psi_2(\mu_2) e_2 {e_2}^{\mathrm{T}}. \quad (9)$$

where ψ_i is a decreasing function, for instance $\psi_i(x) = (\varepsilon + |x|)^{-1}$ for a small value of ε.

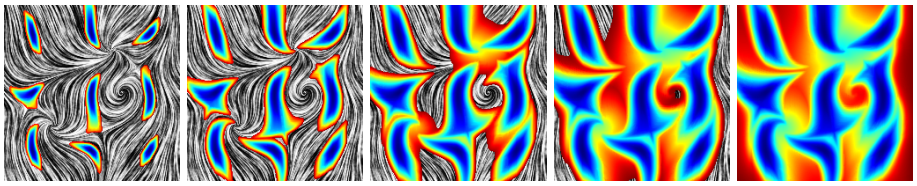


Fig. 10. *Examples of anisotropic front propagation (from 9 starting points). The colormap indicates the values of the distance functions at a given iteration of the algorithm. The metric is computed using the structure tensor, equation* (8)*, of the texture f shown in the background.*

Figure 10 shows an example of Fast Marching propagation using an anisotropic metric $H(x)$. The front propagates faster in the direction of the main eigenvector field $e_1(x)$. Figure 11 shows distance map for a tensor field $H(x)$ whose

anisotropy α is progressively decreased, so that the geodesic distance becomes progressively Euclidean.

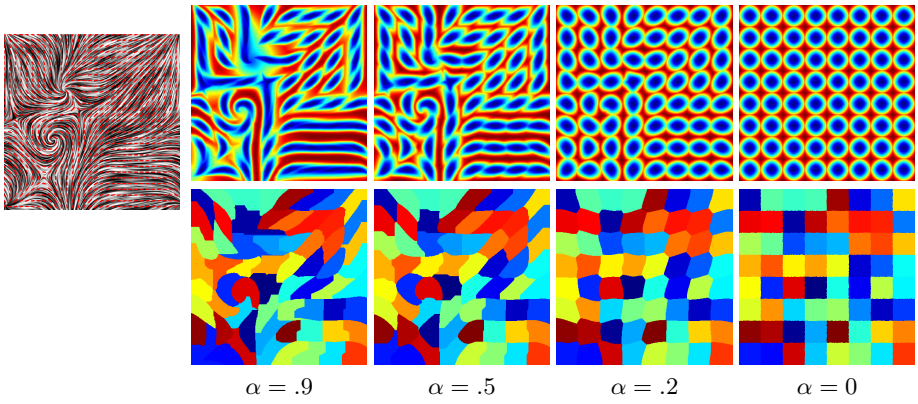


Fig. 11. *Left image: example of texture together with the structure tensor field, computed using equation* (9)*. Right: examples of anisotropic distances (top row) and Voronoi diagrams (bottom row) with a decreasing anisotropy* α *(see equation* (2) *for a definition of this parameter).*

3 Applications and Extensions of Geodesic Distances

3.1 Shape Analysis

In order to analyze the shape of planar objects, one can consider the metric space obtained by restricting the plane to the inside of a planar domain.

Definition 6 (2D shape)**.** *A 2D shape* S *is a connected, closed compact set* $S \subset \mathbb{R}^2$*, with a piecewise-smooth boundary* ∂S*.*

The geodesic distance inside such a shape is obtained by constraining the curve to lie inside S.

Definition 7 (Geodesic distance in S)**.** *The geodesic distance in* S *for the uniform metric is*

$$d_S(x,y) \overset{\text{def.}}{=} \min_{\gamma \in \mathcal{P}(x,y)} L(\gamma) \qquad \textit{where} \qquad L(\gamma) \overset{\text{def.}}{=} \int_0^1 |\gamma'(t)| \mathrm{d}t.$$

where $\mathcal{P}(x,y) \subset S$ *are the paths with starting point* x *and ending point* y*.*

Figure 12 shows examples of shapes together with the geodesic distance to a starting point. The geodesic curve is the union of segments inside S and pieces of the boundary ∂S.

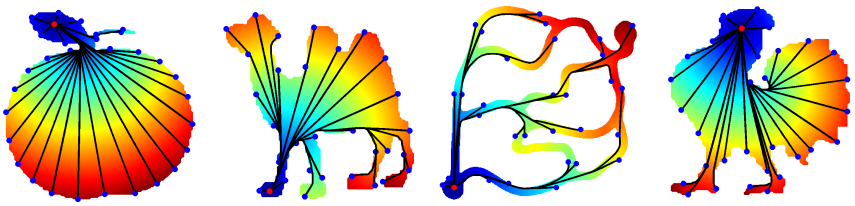


Fig. 12. *Geodesics inside a 2D shape.*

The geodesic distance can be used to define several functions on the 2D shape. This section studies the eccentricity of a shape, as introduced by [17] to perform shape recognition.

Definition 8 (Eccentricity)**.** *The eccentricity* $E_S : \mathcal{M} \mapsto \mathbb{R}$ *is*

$$E_S(x) \stackrel{\text{def.}}{=} \max_{y \in S} d_S(x,y) = \max_{y \in \partial S} d_S(x,y).$$

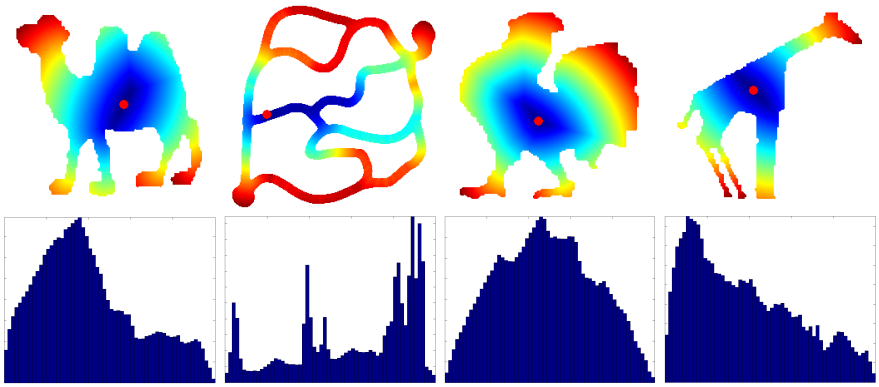


Fig. 13. *Example of eccentricity* E_S *and corresponding histograms* h_S*.*

Figure 13 (top row) shows several examples of eccentricity. The colormap indicates in blue points with small eccentricity.

The points for which the minimum in the definition of E_S is obtained are called eccentric. The set of eccentric point is denoted as $\mathcal{E}(S)$.

Definition 9 (Eccentric points)**.** *An eccentric point* $x \in \mathcal{E}(S)$ *satisfies* $\exists y \in S,\ E_S(y) = d(x,y)$.

These eccentric points define regions of influence which perform a segmentation of the shape as follow

$$S = \bigcup_{x \in \mathcal{E}(S)} \{y \in S \setminus \mathrm{E}_S(y) = d(x,y)\}.$$

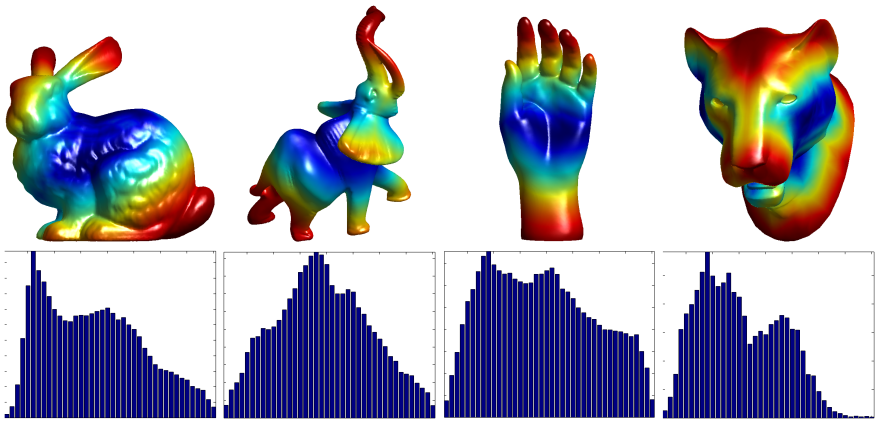


Fig. 14. *Example of eccentricity and corresponding histograms for 3D surfaces.*

These eccentric points are in fact located along the boundary.

Theorem 3 (Location of eccentric points)**.** *One has* $\mathcal{E}(S) \subset \partial S$.

A more general definition of eccentricity allows to replace the maximum by a weighted average of geodesic distances.

Definition 10 (α-eccentricity)**.** *The* α *eccentricity of some shape* S *is defined as*

$$E_S^\alpha(x) \overset{\text{def.}}{=} \left(\int_S d_S(x,y)^\alpha \mathrm{d}y \right)^{1/\alpha}.$$

This eccentricity allows to generalize the notion of gravity center to the geodesic setting.

Definition 11 (Euclidean gravity center)**.** *The Euclidean gravity center is*

$$\operatorname*{argmin}_x \int_S \|x-y\|^2 \mathrm{d}y.$$

The α*-eccentric center is*

$$\operatorname*{argmin}_x E_S^\alpha(x).$$

Remark 1. For $\alpha = 2$, the eccentric center is called geodesic gravity center (and equivalent to the Euclidean center in the case of an uniform metric).

Having defined a function such as E_S inside a shape S, one can collect information about the shape using the histogram of that function.

Definition 12 (Descriptors)**.** *The eccentricity histogram descriptor* $h_S \in \mathbb{R}^m$ *of a shape is*

$$\forall\, i = 1, \ldots, m, \quad h_S(i) = \frac{1}{|S|} \# \left\{ x \in S \setminus \frac{i-1}{m} \leqslant \frac{E_S(x) - \min(E_S)}{\max(E_S) - \min(E_S)} < \frac{i}{m} \right\}.$$

In particular, one can compare shapes by measuring the distance between the histograms

$$\delta(h,\tilde{h})^2 \stackrel{\text{def.}}{=} \sum_{i=1}^{m} (h(i)-\tilde{h}(i))^2.$$

These histograms are invariant if one modifies a shape isometrically. In the plane, geodesic isometry of shapes are not interesting since they are rotations and translations. One can however consider approximate isometries such as articulations, that are useful to model deformations of planar shapes, as defined in [18].

Definition 13 (ε-articulated object)**.** *An articulated object S can be split as*

$$S = \bigcup_{i=1}^{m} S_i \bigcup_{i\neq j} J_{ij},$$

(a disjoint union) with $diam(J_{ij}) \leqslant \varepsilon$.

Definition 14 (Articulation)**.** *An articulation is a mapping between two articulated shapes S, S' such that*

$$f : S \to S' = \bigcup_{i=1}^{m} S'_i \bigcup_{i\neq j} J'_{ij}$$

is rigid on $S_i \mapsto S'_i$.

The eccentricity is approximately invariant for shapes that are modified by articulation.

Theorem 4 (Articulation and isometry)**.** *If f is an articulation, then*

$$|d_S(x,y) - d_{S'}(x,y)| \leqslant m\varepsilon \quad \textit{and} \quad |E_S(x) - E_{S'}(x)| \leqslant m\varepsilon.$$

Starting from a shape library $\{S_1,\ldots,S_p\}$, one can use the shape signature h_S to do shape retrieval using for instance a nearest neighbor classifier, as shown in table 2. Figure 15 shows examples of typical shape retrievals. More complex signatures can be constructed out of geodesic distances and un-supervised recognition can also be considered. We refer to [17] for a detailed study of the performance of shape recognition with eccentricity histograms. In a similar way, the eccentricity can be used to perform 3D surface retrieval, using the histograms displayed in figure 14.

1. *Dataset:*shapes $\{S_1,\ldots,S_p\}$ (binary images).
2. *Preprocessing:* compute eccentricity descriptors h_{S_i}.
3. *Input:* shape S.
4. *Retrival:* return $i^\star = \operatorname{argmin}_i \delta(h_S, h_{S_i})$.

Table 2: Shape retrieval process.

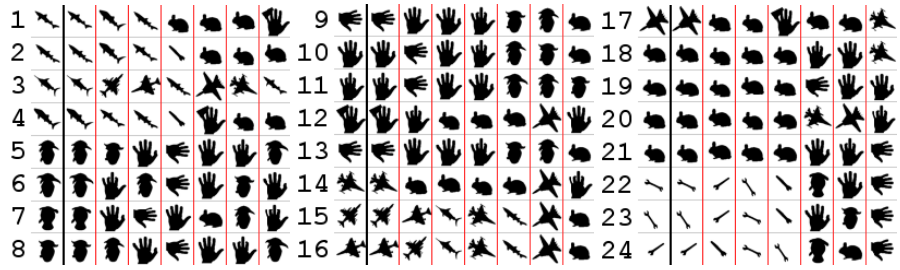


Fig. 15. *Examples of shape recognitions. The shape on the left is the input S and the second shape in each raw is $S_{i^\star}$.*

3.2 Heuristically Driven Propagation

The various implementations of the front propagation algorithm, pseudo-code 1, use a simple priority $\mathcal{P}(x) = U_{x_0}(x)$, where $U(x) \approx d_{\mathcal{M}}(x_0, x)$ is the current value of the distance to the starting point. This strategy leads to an isotropic grow of the front which enforces the exploration of a large area of the computational grid. The advantage of using this priority is that it does not favor any points and thus produces provable valid approximations of geodesic distance (both on a graph with Dijkstra and on a square/triangular grid with Fast Marching).

In order to reduce the computational burden, one could think about using more aggressive ordering of the front that favors some specific direction in the front. The hope is that the front would advance faster in the direction of the goal x_1 one wishes to reach. Ultimately, one would like the front to explore only points along the geodesic $\gamma \in \mathcal{P}(x_0, x_1)$ joining the starting point to the ending point.

If one has an oracle: $V(x) \approx d(x_1, x)$ that estimates the remaining geodesic distance from the current point x to the end x_1, one can use as priority map

$$\mathcal{P}(x) = U(x) + V(x).$$

The map V is called a heuristic since the exact distance $d(x_1, x)$ is not available in practice. The value of a good heuristic close to the real distance is revealed by the following theorem.

Theorem 5 (Geodesic segment)**.** *The function $\psi(x) = d(x_0, x) + d(x_1, x)$ is minimal and constant $\psi(x) = d(x_0, x_1)$ along the geodesic path joining x_0 and x_1.*

In the setting of graph theory, the Dijkstra algorithm can be replaced by the A* (A-star), [19], which uses a heuristic to speed up computations. The following theorem proves the validity of this approach.

Theorem 6 (A* validity)**.** *If the heuristic satisfies $V(x) \leqslant d(x_1, x)$, then the curve $\gamma \in \mathcal{P}(x_0, x_1)$ extracted from the front propagation, algorithm 1, is a geodesic between x_0 and x_1.*

Over a continuous domain, one can invoke a similar (but weaker) theorem.

Theorem 7 (Explored area)**.** *If the heuristic satisfies* $V(x) \leqslant d(x_1, x)$*, then the geodesic* $\gamma \in \mathcal{P}_1(x_0, x_1)$ *between* x_0 *and* x_1 *satisfies*

$$\{\gamma(t) \setminus t \in [0,1]\} \subset \{x \setminus \mathcal{P}(x) = U(x) + V(x) \leqslant \mathcal{P}(x_1)\}\,.$$

This theorem shows why it is important to estimate the geodesic distance by below, since otherwise the region explored by the algorithm might not contain the true geodesic.

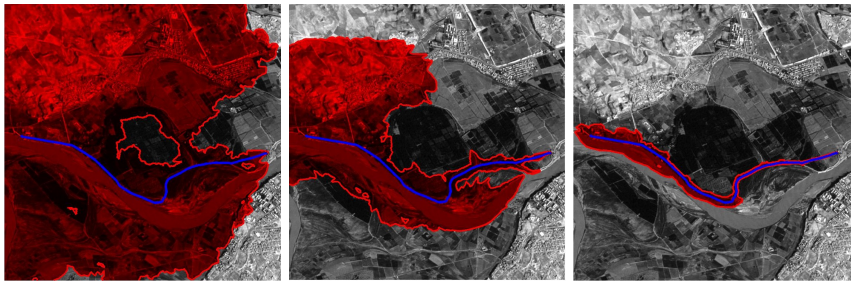


Fig. 16. *Example of propagations with a priority* $\mathcal{P}(x) = U(x) + \lambda V(x)$ *for* $\lambda = 0, 0.5, 0.9$.

Figure 16 shows examples of heuristics that approximate the true remaining distance by bellow. One can see how the explored area of the propagation progressively shrinks while containing the true geodesic. Such a heuristic is however impossible to use in practice since one does not have direct access to the remaining distance during the propagation.

Many strategies can be used to estimate a heuristic. For instance, on a Riemannian metric $(\mathcal{M}, H(x))$, one could use

$$V(x) = \rho \|x - x_1\| \qquad \text{where} \qquad \rho = \min_{x \neq 0, v \neq 0} \|v\|_{H(x)}.$$

In this case, ρ is the minimum eigenvalue of all the tensors $H(x)$. This heuristic estimates the geodesic distance with a Euclidean distance and satisfies $V(x) \leqslant d(x_1, x)$.

For a propagation on a graph (A* algorithm) that is embedded in Euclidean space according to $i \in V \mapsto x_i \in \mathbb{R}^k$, one could also define

$$\forall\, i \in V, \quad V(i) = \|x_{i_1} - x_i\|,$$

where i_1 is the index of the ending point. This heuristic also satisfies $V(i) \leqslant d(i_1, i)$.

These Euclidean heuristics performs poorly on spaces that are not relatively flat. In order to compute more accurate heuristic, we use an expression of the geodesic distance as a minimization.

Theorem 8 (Reversed triangular inequality). *For all* $(x, y) \in \mathcal{M}$, *one has*

$$d(x,y) = \sup_z \Big(|d(x,z) - d(z,y)| \Big).$$

If one restricts the minimum to a small subset of landmark points $\{z_1, \ldots, z_n\} \subset \mathcal{M}$, one can define the following approximate distance

$$\tilde{d}_{z_1 \ldots z_n}(x,y) = \sup_{k=1 \ldots n} \Big(|d_k(x) - d_k(y)| \Big),$$

This kind of approximation has been used first in graph theory [20] and it is defined in a continuous setting in [21]. This leads to a heuristic $V(x) = \tilde{d}(x, x_1)$ that has the following properties.

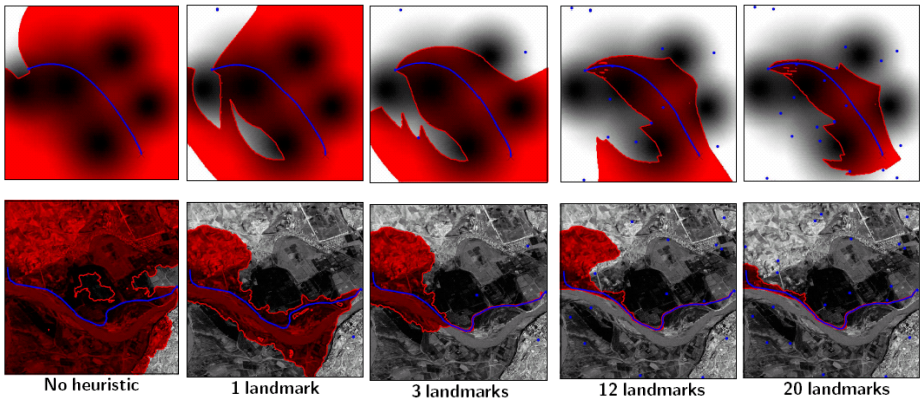


Fig. 17. *Heuristically driven propagation in 2D with an increasing number of landmark points.*

Theorem 9 (Convergence of heuristic). *One has* $\tilde{d} \leqslant d$ *and* $\tilde{d} \overset{n \to +\infty}{\longrightarrow} d$.

In a numerical application that requires the extraction of many geodesics in real time over a large domain, one can pre-compute (off-line) the set of distance maps to the landmarks $\{d(x, z_i)\}_{i=1}^m$. At run time, this set of distances is used to compute the heuristic and speed up the propagation. Figure 17 shows how the quality of the heuristic increases with the number of landmarks. Figure 18 shows an application to geodesic extraction on 3D meshes.

4 Surface Sampling

In order to acquire discrete samples from a continuous surface, or to reduce the number of samples of an already acquired mesh, it is important to be able

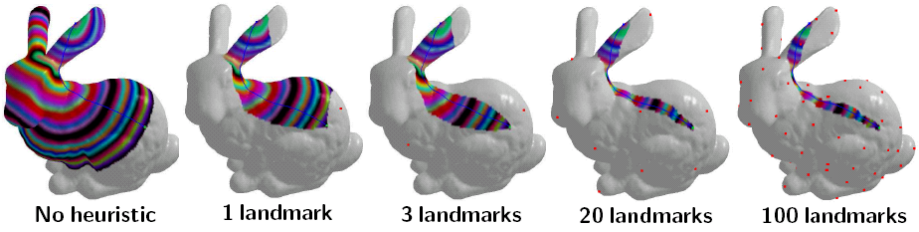


Fig. 18. *Heuristically driven propagation on a 3D mesh with landmark points.*

to seed evenly a set of points on a surface. This is relevant in numerical analysis to have a good accuracy in computational simulations, or in computer graphics to display 3D models with a low number of polygons. In practice, one typically wants to enforce that the samples are approximately at the same distance from each other. The numerical computation of geodesic distances is thus a central tool, that we are going to use both to produce the sampling and to estimate the connectivity of a triangular mesh.

4.1 Farthest Point Sampling

A sampling of a Riemannian surface $\mathcal{M}$ is a set of points $\{x_1, \ldots, x_n\} \subset \mathcal{M}$. If the surface is parameterized by $\varphi : [0,1]^2 \mapsto \mathcal{M}$, the easiest way to compute a sampling is to seed points regularly over the parametric domain

$$\forall\, (i,j) \in \{1, \ldots, \sqrt{n}\}^2, \quad x_{i,j} = \varphi(i/\sqrt{n}, j/\sqrt{n}).$$

This strategy performs poorly if the mapping φ introduces heavy geodesic distortion and the sampling might not be regular any more for the geodesic metric on the surface. In order to ensure the quality of a sampling, one can use the notion of a well separated covering.

Definition 15 (ε-covering)**.** *A sampling $\{x_1, \ldots, x_n\} \subset \mathcal{M}$ is an ε-covering if*

$$\bigcup_i B_\varepsilon(x_i) = \mathcal{M} \qquad \textit{where} \qquad B_\varepsilon(x) \stackrel{\text{def.}}{=} \{y \setminus d_{\mathcal{M}}(x,y) \leqslant \varepsilon\}\,.$$

Definition 16 (ε-separated)**.** *A sampling $\{x_1, \ldots, x_n\} \subset \mathcal{M}$ is ε-separated if*

$$\max(d_{\mathcal{M}}(x_i, x_j)) \leqslant \varepsilon.$$

The farthest point sampling algorithm is a simple greedy strategy able to produce quickly a good sampling. This algorithm has been introduced in image processing to perform image approximation [22]. It is used in [23] together with geodesic Delaunay triangulation (to be defined in the next section) to do surface remeshing. The detection of saddle points (local maxima of the geodesic distance) is used in [24] to perform perceptual grouping.

Table 3 gives the details of this iterative algorithm. In particular, note that the update of the distance $d(x)$ to the set of already seeded points goes faster at

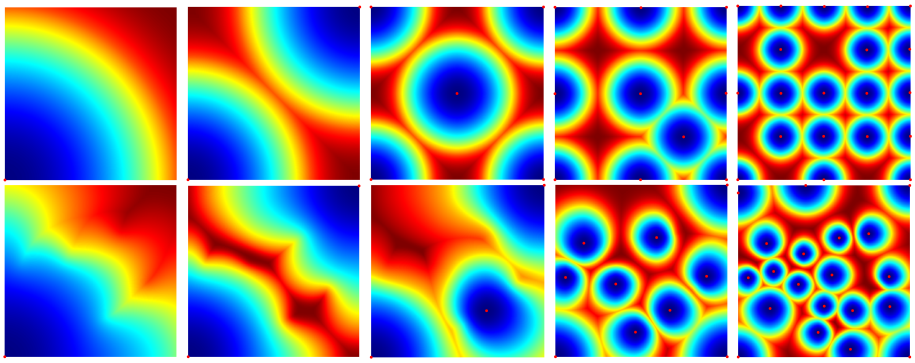


Fig. 19. *Examples of farthest point sampling (the colormap indicates the distance function to the seeds).*

each iteration since the domain of update is smaller when the number of points increases.

1. *Initialization:* $x_1 \leftarrow$random, $d(x) \leftarrow d_{\mathcal{M}}(x_1, x)$, set $i = 1$.
2. *Select point:* $x_{i+1} = \operatorname{argmax}_x d(x)$, $\varepsilon = d(x_{i+1})$.
3. *Local update of the distance:* $d(x) \leftarrow \min(d(x), d_{\mathcal{M}}(x_{i+1}, x))$. This update is restricted to the set of points $\{x \setminus d_{\mathcal{M}}(x_{i+1}, x) < d(x)\}$.
4. *Stop:* If $i < n$ or $\varepsilon > \varepsilon_0$, set $i \leftarrow i + 1$ and go back to 2.

Table 3: Farthest point sampling algorithm.

The output sampling of the algorithm enjoys the property of being a well separated covering of the manifold.

Theorem 10 (Farthest seeding properties)**.** *The farthest point sampling* $\{x_1, \ldots, x_n\}$ *is an* ε*-covering that is* ε*-separated for*

$$\varepsilon = \max_{i=1,\ldots,n} \min_{j=1,\ldots,n} d_{\mathcal{M}}(x_i, x_j).$$

Note however that there is no simple control on the actual number of samples n required to achive a given accuracy ε. We refer to [25] for an in-depth study of the approximation power of this greedy sampling scheme.

Figure 19 shows examples of farthest point sampling with a uniform (top row) and a spatially varying isotropic metric $W(x)$ (bottom row). One can see that this scheme seeds more points in areas where the metric W is large. One can thus control the sampling density by modifying the metric W.

4.2 Triangulations

Having computed, for instance with farthest points, a sampling $\{x_i\}_{i \in V} \subset \mathcal{M}$, the next step is to compute some connectivity between the samples in order

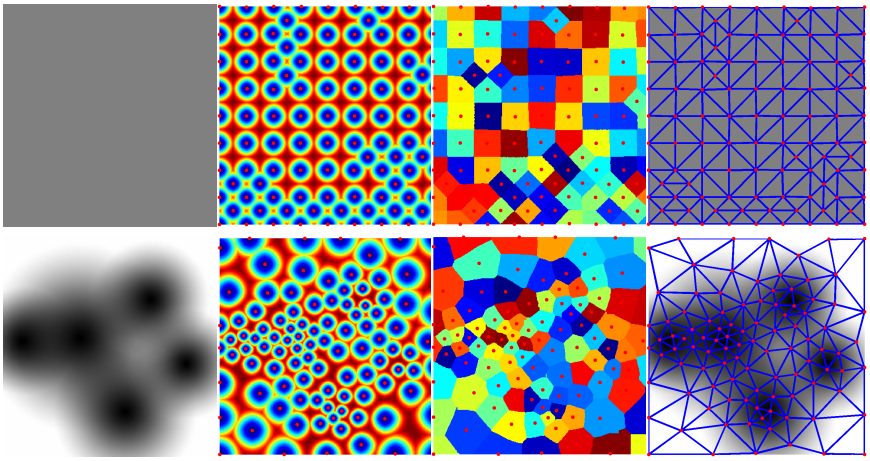


Fig. 20. *Examples of sampling and triangulations with an isotropic metric* $H(x) = W(x)^2 \mathrm{Id}_2$. *The sampling is denser in the regions where the metric is small (dark).*

to build a graph, or even better, a triangulation. The problem of surface remeshing has been studied extensively in computer graphics, see the survey [26]. This section explains a solution based on the geodesic Delaunay triangulation.

The following definition generalizes the notion of an Euclidean Voronoi diagram, to an arbitrary surface.

Definition 17 (Voronoi segmentation)**.** *The Voronoi segmentation of a sampling* $\{x_i\}_{i \in V} \subset \mathcal{M}$ *is*

$$\mathcal{M} = \bigcup_i V_i \qquad \text{with} \qquad (V_i)_{i=1}^m \overset{\text{def.}}{=} \textit{Voronoi}_{\mathcal{M}}(\{x_i\}_i)$$

where

$$V_i \overset{\text{def.}}{=} \{x \setminus \forall\, j \neq i,\ d_{\mathcal{M}}(x, x_i) \leqslant d_{\mathcal{M}}(x, x_j).\}$$

Each Voronoi cell V_i is thus composed of points that are closer to x_i than to any other sampling point. The boundary between two adjacent cells V_i and V_j is thus a piece of curve at equal distance between x_i and x_j. One can then compute the graph dual to a given partition, which joins together pair of adjacent cells. This leads to the notion of Delaunay graph.

Definition 18 (Geodesic Delaunay graph)**.** *The Delaunay graph* (V, E) *of a sampling* $\{x_i\}_{i \in V} \subset \mathcal{M}$ *is defined for* $V = \{1, \ldots, n\}$ *as*

$$E = \{(i, j) \in V \setminus \partial V_i \cap \partial V_j \neq 0\}\,.$$

The main interest of this Delaunay graph is that, if the number of points is large enough to capture the topology of the surface (for instance at least 4 points are needed on a sphere), then one gets a valid triangulation.

Fig. 21. *Example of Voronoi segmentations for an increasing number of seeding points.*

Theorem 11. *For a large enough number of points, the Delaunay graph is a valid triangulation.*

This theorem means that one can find a set of faces F such that (V, E, F) is a triangulated mesh. One can see [27] for a theoretical study of geodesic Delaunay triangulations.

4.3 Examples of Meshing and Remeshing

This Delaunay triangulation can thus be used to perform a geodesic meshing or re-meshing of any Riemannian surface, as explained in [23].

Figure 20 shows examples of Voronoi segmentations on the plane for various isotropic Riemannian metrics $W(x)$. The Delaunay graph allows to define a planar mesh of points evenly sampled according to the metric. Figure 21 shows examples of Voronoi cells on a surface embedded in $\mathbb{R}^3$.

Instead of using a constant or an isotropic metric $W(x)$, one can use a fully anisotropic metric $H(x) \in \mathbb{R}^{2\times 2}$. The local dominant eigenvector $e_1(x)$ given in the decomposition (1) of the tensor gives the local preferred direction of the triangles and the anisotropy $\lambda_1(x)/\lambda_2(x)$ describe how much the triangles should be stretched in this direction. Figure 22 shows an example of meshing with a metric of decreasing anisotropy. Figure 23 shows an anisotropic farthest point meshing with an increasing number of sampling points.

In order to mesh the interior of a planar shape $S \subset \mathbb{R}^2$, one can use the Euclidean metric inside the shape and compute a geodesic Delaunay triangulation. Some care should be made during the algorithm so that the boundary of the domain is included in the delaunay triangulation. This requires splitting boundary edges if they disappear from the Delaunay graph during the algorithm.

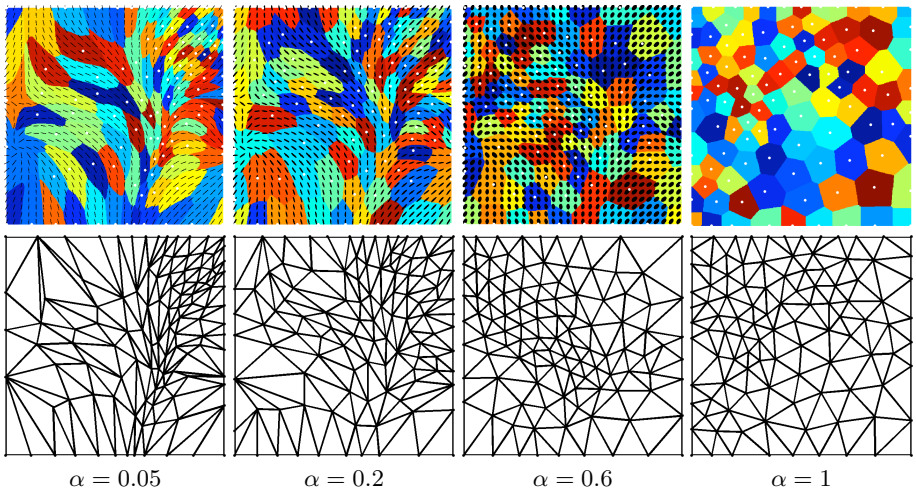


Fig. 22. *Meshing of a square with a metric of decreasing anisotropy of a same synthetic tensor field. Top: Voronoi diagrams, tensor fields and points added by the algorithm (last image is the Euclidean case). Bottom: resulting meshes.*

Figure 24 shows some examples of shape meshing with this uniform metric. This triangulation is however very close to the usual definition of a planar Euclidean Delaunay triangulation. In contrast, one can use a non-uniform metric $W(x)$ and compute a sampling inside the shape that conforms itself to this density. Figure 24 shows a sampling and meshing that uses a metric $W(x) = (\varepsilon + d(x, \partial S))^{-1}$ that tends to seed more points on the boundary of the shape S.

Figure 25 shows an example of uniform remeshing of a 3D surface acquired from medical imaging with an increasing number of points. Figure 26 shows how one can adapt the density by defining a non-constant isotropic metric on the surface.

An option to compute this metric is to use a texture mapped on the surface. Starting from some parametric surface: $\varphi : \mathcal{D} \subset [0,1]^2 \rightarrow \mathcal{M}$, a texture T is a mapping $T : [0,1]^2 \rightarrow \mathbb{R}$. It allows to define an isotropic metric using for instance an edge adaptive function

$$\forall\, x \in \mathcal{D},\ H(x) = \psi_T(x)\mathrm{Id}_2.$$

where the edge-based stopping function is $\psi_T(x) = (\|\nabla_x T\| + \varepsilon)^{-1}$. Figure 27 shows examples of remeshing with a texture-adapted metric with a decreasing value of ε (increasing adaptivity).

Conclusion

This chapter has reviewed several applications of Riemannian metrics in computer vision and graphics. In particular, the use of geodesic distances and shortest paths is useful in many areas of these fields. The design of adapted isotropic

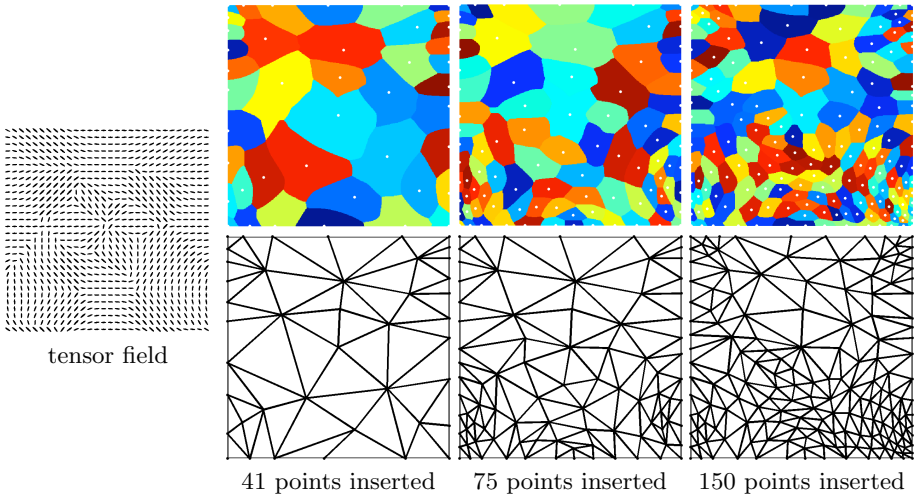


Fig. 23. *Anisotropic meshing of a square with an increasing number of points.*

or anisotropic metrics allows to solve efficiently segmentation, sampling, meshing and recognition problems with fast algorithms.

References

1. Cohen, L.D.: Minimal paths and fast marching methods for image analysis. In: Handbook of Mathematical Methods in Computer Vision, N. Paragios and Y. Chen and O. Faugeras Editors, Springer. (2005)
2. Prados, E., Lenglet, C., Pons, J.P., Wotawa, N., Deriche, R., Faugeras, O.D., Soatto, S.: Control theory and fast marching methods for brain connectivity mapping. In: Proc. CVPR 2006. (2006)
3. Crandall, M.G., Ishii, H., Lions, P.L.: User's guide to viscosity solutions of second order partial differential equations. Bull.Amer.Math.Soc **27** (1992) 1
4. Cohen, L.D., Kimmel, R.: Global Minimum for Active Contour models: A Minimal Path Approach. International Journal of Computer Vision **24**(1) (Aug. 1997) 57–78
5. Sethian, J.: Level Sets Methods and Fast Marching Methods. 2nd edn. Cambridge University Press (1999)
6. Osher, S.J., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer (2002)
7. Kimmel, R.: Numerical Geometry of Images: Theory, Algorithms, and Applications. Springer (2004)
8. Bronstein, A., Bronstein, M., Kimmel, R.: Numerical Geometry of Non-Rigid Shapes. Springer (2007)
9. Osher, S.J., Paragios, N.: Geometric Level Set Methods in Imaging, Vision, and Graphics. Springer-Verlag (July 2003)
10. Paragios, N., Chen, Y., Faugeras, O.D.: Handbook of Mathematical Models in Computer Vision. Springer (2005)
11. Kimmel, R., Sethian, J.: Computing Geodesic Paths on Manifolds. Proc. Natl. Acad. Sci. **95**(15) (1998) 8431–8435

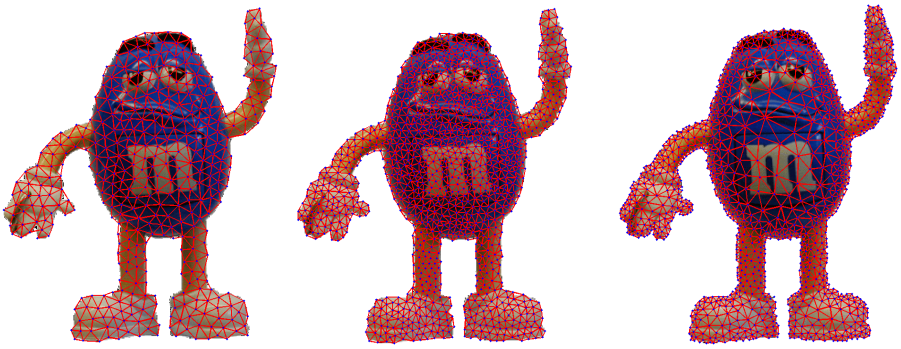


Fig. 24. *Shape meshing with an increasing number of points. Left and center: uniform meshing, right: adaptive meshing.*

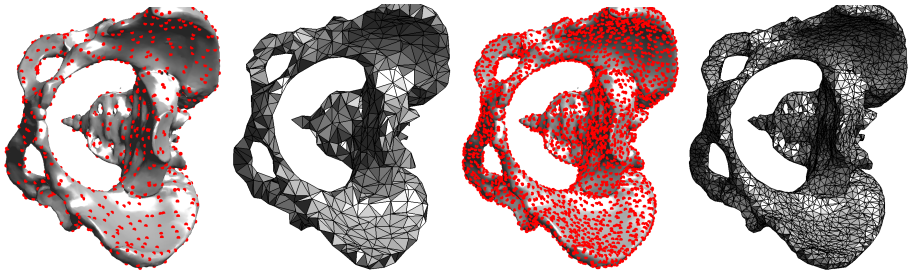


Fig. 25. *Geodesic remeshing with an increasing number of points.*

12. Spira, A., Kimmel, R.: An efficient solution to the eikonal equation. Interfaces and Free Boundaries **6**(3) (2004) 315–327
13. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Weighted distance maps computation on parametric three-dimensional manifolds. Journal of Computational Physics, accepted (2007)
14. Kimmel, R., Amir, A., Bruckstein, A.M.: Finding shortest paths on surfaces using level sets propagation. IEEE Trans. on PAMI **17**(6) (1995) 635–640
15. Deschamps, T., Cohen, L.: Fast Extraction of Minimal Paths in 3D Images and Applications to Virtual Endoscopy. Medical Image Analysis **5**(4) (December 2001)
16. Kothe, U.: Edge and junction detection with an improved structure tensor. In: Proc. DAGM03. (2003) 25–32
17. Ion, A., Peyré, G., Haxhimusa, Y., Peltier, S., Kropatsch, W.G., Cohen, L.: Shape matching using the geodesic eccentricity transform. In: Proceedings of OAGM'07. (2007)
18. Ling, H., Jacobs, D.W.: Using the inner-distance for classification of articulated shapes. In: CVPR 2005, 20-26 June 2005, San Diego, CA, USA. (2005) 719–726
19. Nilsson, N.: Problem-solving Methods in Artificial Intelligence. McGraw-Hill, New York (1971)
20. Goldberg, A.V., Harrelson, C.: Computing the shortest path: A* search meets graph theory. Technical Report MSR-TR-2004-24 (2004)

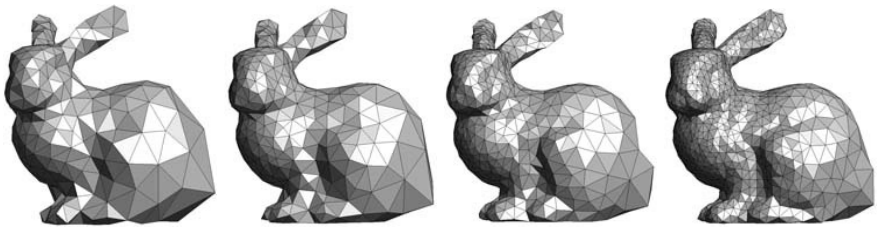


Fig. 26. *Adaptive remeshing with a linearly increasing density.*

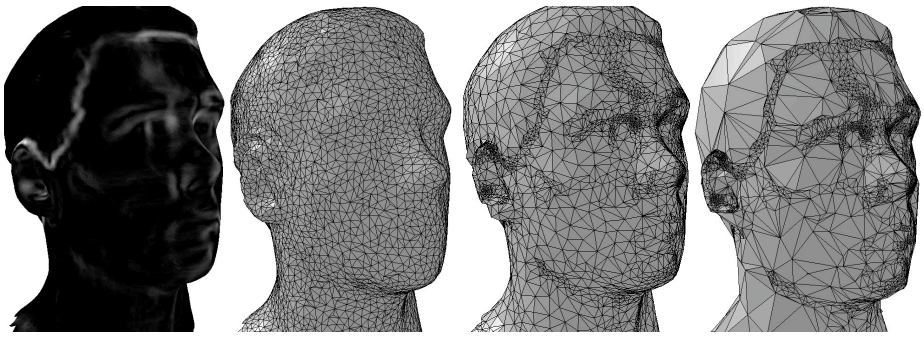


Fig. 27. *Adaptive remeshing with a density given by a texture.*

21. Peyré, G., Cohen, L.: Heuristically driven front propagation for fast geodesic extraction. International Journal for Computational Vision and Biomechanics **1**(1) (2007)
22. Eldar, Y., Lindenbaum, M., Porat, M., Zeevi, Y.Y.: The farthest point strategy for progressive image sampling. IEEE Trans. Image Processing **6**(9) (September 1997) 1305–1315
23. Peyré, G., Cohen, L.D.: Geodesic remeshing using front propagation. Int. J. Comput. Vision **69**(1) (2006) 145–156
24. Cohen, L.: Multiple Contour Finding and Perceptual Grouping Using Minimal Paths. Journal of Mathematical Imaging and Vision **14**(3) (May 2001) 225–236
25. Clarkson, K.L.: Building triangulations using epsilon-nets. In Kleinberg, J.M., ed.: STOC, ACM (2006) 326–335
26. Alliez, P., Ucelli, G., Gotsman, C., Attene, M.: Recent advances in remeshing of surfaces. In: AIM@SHAPE repport. (2005)
27. Leibon, G., Letscher, D.: Delaunay triangulations and voronoi diagrams for riemannian manifolds. In: SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry, New York, NY, USA, ACM (2000) 341–349