Geodesic Computations for Fast and Accurate Surface Remeshing and Parameterization

Gabriel Peyré, Laurent D. Cohen

Progress in Nonlinear Differential Equations
and Their Applications, Vol. 63, 157–171

Geodesic Computations for Fast and Accurate Surface Remeshing and Parameterization

Gabriel Peyré and Laurent Cohen

Abstract. In this paper, we propose fast and accurate algorithms to remesh and flatten a genus-0 triangulated manifold. These methods naturally fits into a framework for 3D geometry modeling and processing that uses only fast geodesic computations. These techniques are gathered and extended from classical areas such as image processing or statistical perceptual learning. Using the *Fast Marching* algorithm, we are able to recast these powerful tools in the language of mesh processing. Thanks to some classical geodesic-based building blocks, we are able to derive a flattening method that exhibit a conservation of local structures of the surface.

On large meshes (more than 500 000 vertices), our techniques speed up computation by over one order of magnitude in comparison to classical remeshing and parameterization methods. Our methods are easy to implement and do not need multilevel solvers to handle complex models that may contain poorly shaped triangles.

Keywords. Remeshing, geodesic computation, fast marching algorithm, mesh segmentation, surface parameterization, texture mapping, deformable models.

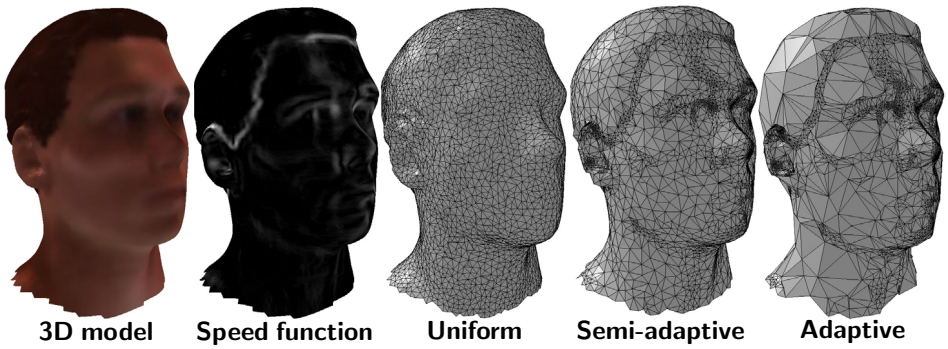

FIGURE 1. Remeshing of a 3D model using increasing weight for the speed function.

1. Introduction

The applications of 3D geometry processing abound nowadays. They range from finite element computation to computer graphics, including solving all kinds of surface reconstruction problems. The most common representation of 3D objects is the triangle mesh, and the need for fast algorithms to handle this kind of geometry is obvious. Classical 3D triangulated manifold processing methods have several well-identified shortcomings: mainly, their high complexity when dealing with large meshes, and their numerical instabilities.

To overcome these difficulties, we propose a geometry processing pipeline that relies on *intrinsic* information of the surface and not on its underlying triangulation. Borrowing from well-established ideas in different fields (including image processing and perceptual learning) we are able to process very large meshes efficiently.

1.1. Overview

In Section 2 we introduce some concepts we use in our geodesic computations. This includes basic facts about the Fast Marching algorithm, and a recently proposed greedy algorithm for manifold sampling.

To flatten each patch of a segmented surface, we will recall some recent advances in perceptual inference learning in Section 3. Combining these techniques with our geodesic computational framework will lead to an elegant solution to the flattening problem for large meshes.

In the conclusion, we will show the two algorithms in action, and see how we can texture large meshes faster than current techniques would otherwise allow. We will then give a complete study of the timings of each part of our algorithm, including a comparison with classical methods.

1.2. Related Work

Surface Remeshing and Finite Elements. Remeshing methods roughly fall into two categories:

- *Isotropic remeshing*: a surface density of points is defined, and the algorithm tries to position the new vertices to match this density. For example the algorithm of *Terzopoulos* and *Vasilescu* [Terzopoulos and Vasilescu, 1992] uses dynamic models to perform the remeshing. Remeshing is also a basic task in the computer graphics community, and [Surazhsky et al., 2003] have proposed a procedure based on local parameterization.
- *Anisotropic remeshing*: the algorithm takes into account the principal directions of the surface to align locally the newly created triangles and/or rectangles. Finite element methods make heavy use of such remeshing algorithms [Kunert, 2002]. The algorithm proposed in [Alliez et al., 2003] uses lines of curvature to build a quad-dominant mesh.

The importance of using geodesic information to perform this remeshing task is emphasized in [Sifri et al., 2003].

Greedy solutions for sampling a manifold (see Section 2.2) have been used with success in other fields such as computer vision (component grouping, [Cohen, 2001]), halftoning (void-and-cluster, [Ulichney, 1993]) and remeshing (Delaunay refinement, [Ruppert, 1995]).

Flattening and Parameterization. The flattening problem can be seen as a particular instance of parameterization. The work of [Eck et al., 1995] first introduces the harmonic formulation for the resolution of the mesh parameterization problem. Most of these classical methods come from graph-drawing theory, and [Floater et al., 2002] gives a survey of these techniques. Authors of [Desbrun et al., 2002] give an in-depth study of the various energies that can be built to flatten a mesh. The flattening algorithm of [Zigelman et al., 2002] is based on methods for finding parameters that reduce a dataset's dimensionality. Such methods have been developed for the purpose of perceptual learning [Tenenbaum et al., 2000, Roweis and Saul, 2000], and we will explain in Section 3 how to exploit these methods to handle a 3D mesh with a large amount of vertices.

2. Geodesic Remeshing

2.1. Fast Marching Algorithm

The classical Fast Marching algorithm is presented in [Sethian, 1999], and a similar algorithm was also proposed in [Tsitsiklis, 1995]. This algorithm is used intensively in computer vision, for instance it has been applied to solve global minimization problems for deformable models [Cohen and Kimmel, 1997].

This algorithm is formulated as follows. Suppose we are given a metric $P(s)\mathrm{d}s$ on some manifold $\mathcal{S}$ such that $P > 0$. If we have two points $x_0,\ x_1 \in \mathcal{S}$, the weighted geodesic distance between x_0 and x_1 is defined as

$$d(x_0,\, x_1) \stackrel{\text{def.}}{=} \min_{\gamma} \left(\int_0^1 \|\gamma'(t)\| P(\gamma(t)) \mathrm{d}t \right), \tag{1}$$

where γ is a piecewise regular curve with $\gamma(0) = x_0$ and $\gamma(1) = x_1$. When $P = 1$, the integral in (1) corresponds to the length of the curve γ and d is the classical geodesic distance. To compute the distance function $U(x) \stackrel{\text{def.}}{=} d(x_0,\, x)$ with an accurate and fast algorithm, this minimization can be reformulated as follows. The level set curve $\mathcal{C}_t \stackrel{\text{def.}}{=} \{x \setminus U(x) = t\}$ propagates following the evolution equation $\frac{\partial \mathcal{C}_t}{\partial t}(x) = \frac{1}{P(x)} \overrightarrow{n_x}$, where $\overrightarrow{n_x}$ is the exterior unit vector normal to the curve at x, and the function U satisfies the nonlinear *Eikonal* equation:

$$\|\nabla U(x)\| = P(x). \tag{2}$$

The function $F = 1/P > 0$ can be interpreted as the propagation speed of the front $\mathcal{C}_t$.

The Fast Marching algorithm on an orthogonal grid makes use of an upwind finite difference scheme to compute the value u of U at a given point $x_{i,j}$ of a grid:

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

This is a second-order equation that is solved as detailed for example in [Cohen, 2001]. An optimal ordering of the grid points is chosen so that the whole computation only takes $O(N \log(N))$, where N is the number of points.

In [Kimmel and Sethian, 1998], a generalization to an arbitrary triangulation is proposed. This allows performing front propagations on a triangulated manifold, and computing geodesic distances with a fast and accurate algorithm. The only issue arises when the triangulation contains obtuse angles. The numerical scheme presented above is not monotone anymore, which can lead to numerical instabilities. To solve this problem, we follow [Kimmel and Sethian, 1998] who propose to "unfold" the triangles in a zone where we are sure that the update step will work. Figure 2 shows the calculation of a geodesic path computed using a gradient descent of the distance function.

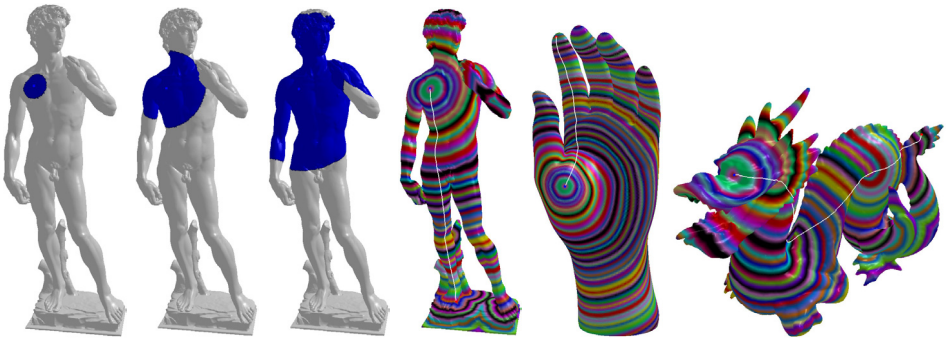

FIGURE 2. Front Propagation (on the left), level sets of the distance function and geodesic path (on the right).

2.2. A Greedy Algorithm for Uniformly Sampling a Manifold

A new method for sampling a 3D mesh was recently proposed in [Peyré and Cohen, 2003] that follows a farthest point strategy based on the weighted distance obtained through Fast Marching on the initial triangulation. This is related to the method introduce in [Cohen, 2001]. A similar approach was proposed independently and simultaneously in [Moenning and Dodgson, 2003]. It follows the *farthest point* strategy, introduced with success for image processing in [Eldar et al., 1997] and related to the remeshing procedure of [Chew, 1993].

This approach iteratively adds new vertices based on the geodesic distance on the surface. Figure 3 shows the first steps of our algorithm on a square. The

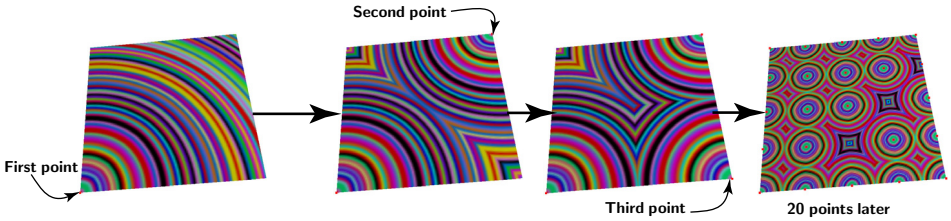

FIGURE 3. An overview of the greedy sampling algorithm.

result of the algorithm gives a set of vertices uniformly distributed on the surface according to the geodesic distance.

Once we have found enough points, we can link them together to form a *geodesic Delaunay triangulation*. This is done incrementally during the algorithm, and leads to a powerful remeshing method.

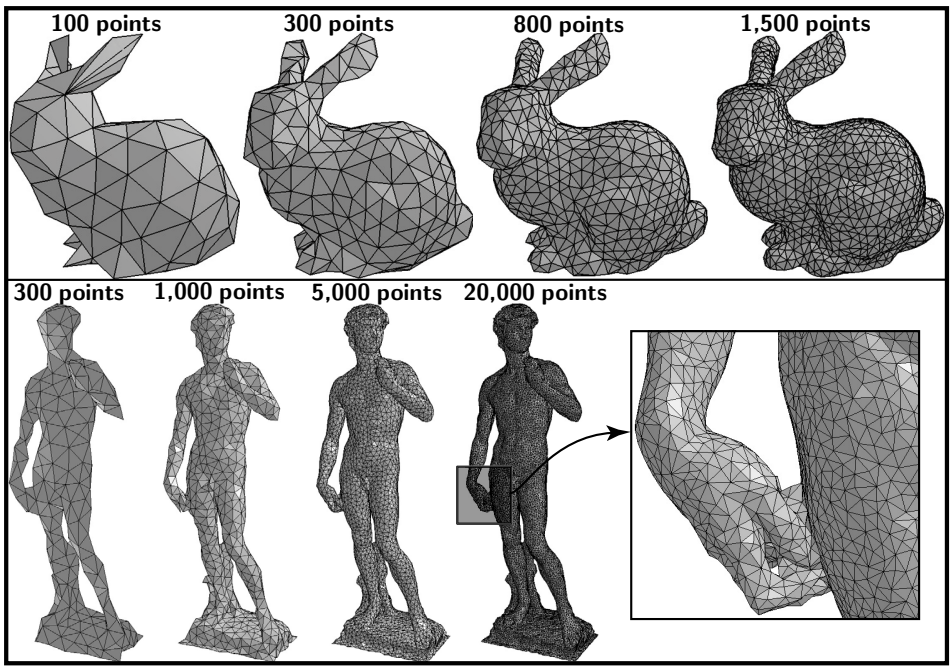

FIGURE 4. Geodesic remeshing with an increasing number of points.

Figure 4 shows progressive remeshing of the bunny and the David. In order to have a valid triangulation, the sampling of the manifold must be dense enough (for example 100 points is not enough to capture the geometry of the ears of the bunny). A theoretical proof of the validity of geodesic Delaunay triangulation can

be found in [Leibon and Letscher, 2000], and more precise bound on the number of points is derived in [Onishi and Itoh, 2003]. Note that our algorithm works with manifolds with boundaries, of arbitrary genus, and with multiple connected components.

2.3. Adaptive Remeshing

In the algorithm presented in Sections 2.2, the fronts propagate at a constant speed which results in uniformly spaced mesh. To introduce some adaptivity in the sampling performed by this algorithm, we use a speed function $F = 1/P$ (which is the right-hand side of the Eikonal equation (2)) that is not constant across the surface. Figure 5 shows the progressive sampling of a square using a speed function with two different values. The colors show the level sets of the distance function U to the set of points.

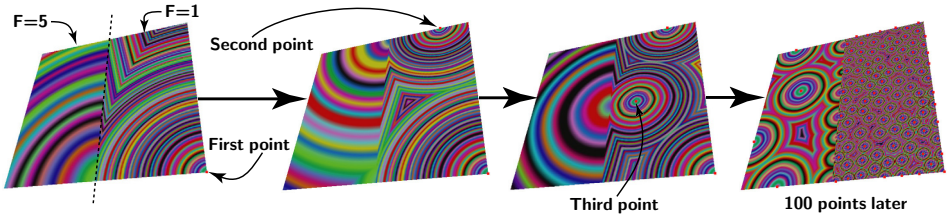

FIGURE 5. Iterative insertion of points in a square.

When a mesh is obtained from range scanning, a picture I of the model can be mapped onto the 3D mesh. Using a function F of the form $F(x) = (1 + \mu|\nabla(I(x))|)^{-1}$, where μ is a user-defined constant, one can refine regions with high variations in intensity. On Figure 1, one can see a 3D head remeshed with various μ ranging from $\mu = 0$ (uniform) to $\mu = 20/\max(|\nabla(I(x))|)$ (highly adaptive).

The local density of vertices can also reflect some geometric properties of the surface. The most natural choice is to adapt the mesh in order to be finer in regions where the local curvature is larger. The evaluation of the curvature tensor is a vast topic. We used a robust construction proposed recently in [Cohen-Steiner and Morvan, 2003]. Let us denote by $\tau(x) \stackrel{\text{def.}}{=} |\lambda_1| + |\lambda_2|$ the total curvature at a given point x of the surface, where λ_i are the eigenvalues of the second fundamental form. We can introduce two speed functions $F_1(x) \stackrel{\text{def.}}{=} 1 + \varepsilon\tau(x)$ and $F_2(x) \stackrel{\text{def.}}{=} \frac{1}{1+\mu\tau(x)}$, where ε and μ are two user-defined parameters. Figure 6 (a) shows that by using function F_1, we avoid putting more vertices in regions of the surface with high curvature. The speed function F_1 can be interpreted as an "edge repulsive" function. On the other hand, function F_2 could be called "edge attractive" function, since it forces the sampling to put vertices in region with high curvature such as mesh corners and edges. Figure 6 (b) shows that this speed function leads to very good results for the remeshing of a surface with sharp

features, which is obviously not the case for the "edge repulsive" speed function (Figure 6 (a)).

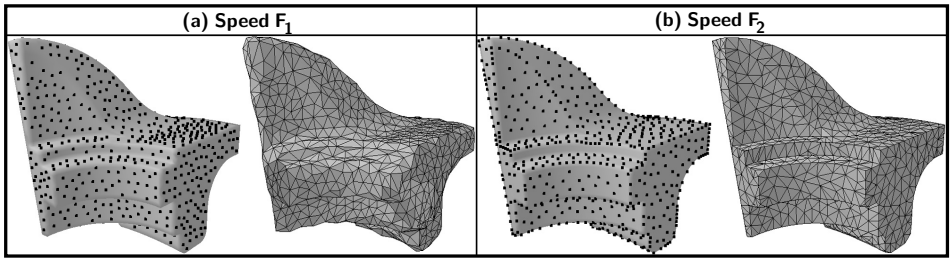

FIGURE 6. Uniform versus curvature-based sampling and remeshing.

3. Fast Geodesic Parameterization

The flattening problem can be seen as a particular instance of the more generic problem of mesh parameterization. Given a genus-0 triangulated manifold $\mathcal{S}$ homeomorphic to a disc, it consists in finding a map $f : \mathcal{S} \to \mathcal{U}$, where $\mathcal{U} \subset \mathbb{R}^2$ is a planar domain.

3.1. Geodesic Flattening and IsoMap

Recently, some nonlinear algorithms for dimensionality reduction have appeared in the community of perceptual manifold learning. The most notable are *IsoMap* [Tenenbaum et al., 2000] and *Locally Linear Embedding* (LLE) [Roweis and Saul, 2000].

Interestingly, the only echo of these techniques in the computer graphics community seems to be the multi-dimensional scaling approach to flattening of [Zigelman et al., 2002]. This method is closely related to IsoMap, and we will see that it shares its main drawbacks.

We start with a given set of points $\{x_1, \ldots, x_n\}$ on our manifold, and we seek $f(x_i) = \widetilde{x_i} \in \mathbb{R}^2$ such that the mapping minimizes some measure of distortion. The most natural constraint is to try to keep the same distance between points, which is exactly what IsoMap is doing by requiring that $d(x_i, x_j) \approx \|\widetilde{x_i} - \widetilde{x_j}\|$, where d stands for (some approximation of) the geodesic distance on the manifold. The method of [Zigelman et al., 2002] is very close to this approach, since it uses the geodesic distance d computed via the Fast Marching algorithm presented in Section 2.1.

The major bottleneck of this method is that it needs to compute all pairwise distances $d(x_i, x_j)$. To overcome this difficulty, the authors of [Zigelman et al., 2002] proposed to restrict the computations to a small set of points, which gives rise to three questions:

- What should be done to speed up computation?
- How should we choose this small set of base points?
- How should we extend the map f from this small set of points to the rest of the mesh?

In the next subsection, we will show how the LLE algorithm can bring a important speed improvement that answers the first question. The answer to the two last questions will be given in Subsections 3.3 and 3.4 respectively, with an extension of LLE to triangulated manifolds.

3.2. Speeding Up Computation with LLE

The LLE algorithm is explained in detail in [Roweis and Saul, 2000]. The goal of the algorithm is to find a low-dimensional embedding in $\mathbb{R}^d$ of a set of points $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^s$, $s > d$. The only parameter of this algorithm is an integer K that measures the size of the neighborhood of each point. We will denote by N_i the K-neighborhood of x_i, that is to say $N_i \stackrel{\text{def.}}{=} \{x_{m(1)}, \ldots, x_{m(K)}\}$, where $x_{m(j)}$ is the j^{th} closest point to x_i for the Euclidean metric. We will briefly recall the two main steps of the procedure:

Step 1: First, for each point x_i, we are looking for some weights $w_{i,j}$ that locally best reconstruct the manifold, from the set N_i only, by minimizing

$$E_1\left(\{w_{i,j}\}_j\right) = \|x_i - \sum_j w_{i,j} x_j\|^2. \tag{3}$$

We further enforce that $w_{i,j} = 0$ if $x_j \notin N_i$, and that $\sum_j w_{i,j} = 1$. This imposes that the reconstruction is both local and invariant under affine transformations. In a Euclidean setting, the minimization (3) requires the introduction of the Gram matrix $C(x)$ defined by

$$(C(x))_{i,j} \stackrel{\text{def.}}{=} \left\langle x - x_{m(i)},\, x - x_{m(j)} \right\rangle \tag{4}$$

The solution of the minimization (3) is then

$$w_{i,j} = \mathbf{1}_{N_i}(x_j) \frac{\sum_q (C(x_i)^{-1})_{k,q}}{\sum_{p,q} (C(x_i)^{-1})_{p,q}}, \quad \text{where} \quad x_j = x_{m(k)}.$$

The value of $\mathbf{1}_{N_i}(x)$ is equal to 1 if $x \in N_i$, and 0 otherwise.

Step 2: To reconstruct the manifold in low dimension (here in 2D), we want to solve a global minimization procedure, for $\widetilde{x}_i \in \mathbb{R}^2$:

$$\text{minimize} \quad E_2\left(\{\widetilde{x}_i\}\right) = \sum_i \|\widetilde{x}_i - \sum_j w_{i,j} \widetilde{x}_j\|^2.$$

subject to $\sum \widetilde{x}_i = 0$ and $\sum \|\widetilde{x}_i\|^2 = 1$ to avoid a degenerate solution. To solve this problem, we need to form the matrix $M \stackrel{\text{def.}}{=} (W - \mathrm{Id})^{\mathrm{T}}(W - \mathrm{Id})$, where W is a sparse matrix containing all the weights. The eigenvector of M with lowest

eigenvalue it the constant vector **1** which should be discarded. The d following eigenvectors give us the coordinates of our embedding in $\mathbb{R}^d$ for each point.

The fact that we only need to perform computations on sparse matrices allows an improvement of one order of magnitude over dense procedures such as in [Zigelman et al., 2002].

3.3. Geodesic LLE

In the previous section we saw the classical LLE algorithm in a Euclidean setting. To solve the flattening problem for a mesh, we need to extend these computations to the manifold setting. The following modifications allow such an extension.

Modification 1: The points $\{x_1, \ldots, x_n\}$ should be sampled as uniformly as possible on $\mathcal{S}$. That is why we use the greedy sampling algorithm of Section 2.2 to select these points. To get an adaptive sampling, one could use a varying speed function, as shown in Figure 2.2 (see also [Peyré and Cohen, 2003] for a curvature-based adaptation).

Modification 2: The K-neighborhood N_i of each point should be computed using the geodesic distance and not the Euclidean one. This can be done very quickly using a local front propagation.

Modification 3: The matrix $C(x)$ of equation (4) can not be computed anymore using dot products. Instead, following [Roweis and Saul, 2000] (pairwise LLE), we propose the following formula

$$-2C(x)_{i,j} \stackrel{\text{def.}}{=} d(x_{m(i)}, x_{m(j)})^2 - \frac{1}{K}\sum_{k=1}^{K} d(x_{m(i)}, x_{m(k)})^2 - \frac{1}{K}\sum_{k=1}^{K} d(x_{m(k)}, x_{m(j)})^2 + \frac{1}{K^2}\sum_{k,l=1}^{K} d(x_{m(k)}, x_{m(l)})^2,$$

that only uses geodesic distance information. This formula is equivalent to (4) in the Euclidean setting.

3.4. Extending the Map

The three modifications proposed in the previous section allow us to find the location of $\widetilde{x}_i = f(x_i) = (f_1(x_i), f_2(x_i))^{\mathrm{T}} \in \mathbb{R}^2$ for each base point x_i. To compute the whole map f, we need to interpolate the location of $f(x) = (f_1(x), f_2(x))^{\mathrm{T}}$ for each point $x \in \mathcal{S}$, using the known locations $f(x_i)$.

This problem has been addressed very recently in [Bengio et al., 2003], by recasting it into a unified framework of eigenvector learning, common to many dimensionality reduction methods.

To extend f, we use the fact that vectors $\{f_1(x_i)\}_{i=1}^n$ and $\{f_2(x_i)\}_{i=1}^n$ are eigenvectors of the symmetric matrix $M = (W - \mathrm{Id})^{\mathrm{T}}(W - \mathrm{Id})$ (with eigenvalues λ_1 and λ_2). In the continuous setting, this matrix becomes a symmetric kernel

$\widetilde{M}(x, y)$ for each point x, y in $\mathcal{S}$. Matrix multiplication by M is then replaced by

$$\varphi \mapsto \widetilde{M}\varphi(x) \stackrel{\text{def.}}{=} \int_{\mathcal{S}} \varphi(y)\widetilde{M}(x, y)\mathrm{d}y, \tag{5}$$

where φ is any mapping from $\mathcal{S}$ to $\mathbb{R}$. Using this remark, it is natural to suppose that the continuous maps f_1 and f_2 are eigenfunctions of the operator defined by equation (5) for the same eigenvalues λ_1 and λ_2. This implies that we can compute them using a Nyström-like formula

$$f_1(x) = \frac{1}{\lambda_1}\int_{\mathcal{S}} f_1(y)\widetilde{M}(x, y)\mathrm{d}y \approx \frac{1}{n\lambda_1}\sum_{i=1}^{n} f_1(x_i)\widetilde{M}(x_i, y), \tag{6}$$

and similarly for f_2.

Since the $f_1(x_i)$ are known, we just need to setup our kernel $\widetilde{M}$. The only constraint is that for all $y \in \mathcal{S}$, $\widetilde{M}(x_i, y)$ should be easy to compute, e.g., it should only involve already computed distances such as $d(x_j, y)$ for $x_j \in N_i$. This can be done in a straightforward manner by first setting the weights for y:

$$w(x_i, y) \stackrel{\text{def.}}{=} \mathbf{1}_{N_i}(y)\frac{\sum_q (C(x_i)^{-1})_{k,q}}{\sum_{p,q} (C(x_i)^{-1})_{p,q}} \quad \text{with} \quad y = x_{m(k)},$$

and then defining the kernel:

$$\widetilde{M}(x, y) \stackrel{\text{def.}}{=} w(x, y) + w(y, x) - \sum_k w(x_k, x)w(x_k, y),$$

We can check that for base points, we retrieve the original matrix up to a substraction of the identity, i.e., $\widetilde{M}(x_i, x_j) = \delta_{i,j} - M_{i,j}$. This shift is only here to avoid a singularity along the diagonal and does not modify the computation.

This shows that we can extend the map f to a new point x using only some local distance information between x and its neighborhood in $\{x_1, \ldots, x_n\}$. Furthermore, most of the time, this information is already available from previous front propagations performed to flatten $\{x_1, \ldots, x_n\}$.

Figure 7 shows the flattening of one half of a human head. Even with a large patch that contains holes, our method gives very good results (no face flip) with only 100 base vertices.

Figure 8 shows the influence of the number of base points on the flattening. Even with only 20 points, the resulting embedding is nearly smooth except at the border of the mesh, and with 100 points, we get a perfectly smooth flattening.

4. Results and Discussion

Texturing of a Complex Model. To perform texture mapping on a complex 3D mesh, a segmentation step is required to first cut the model into disk-shaped charts. Although the study of this step is outside the scope of this paper, we note that the notion of Voronoi cells is often to perform mesh partition, as introduced

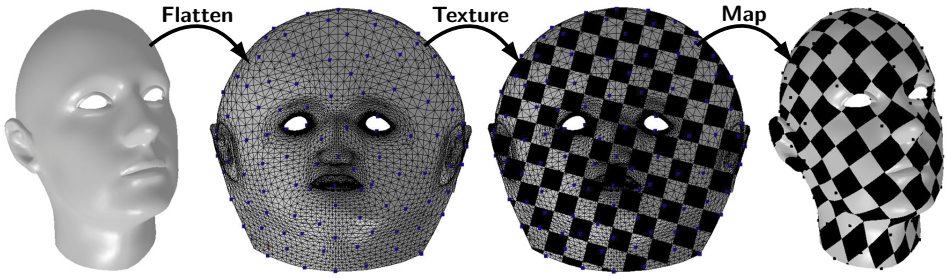

FIGURE 7. The original model, texture on the flattened domain, and on the 3D mesh.

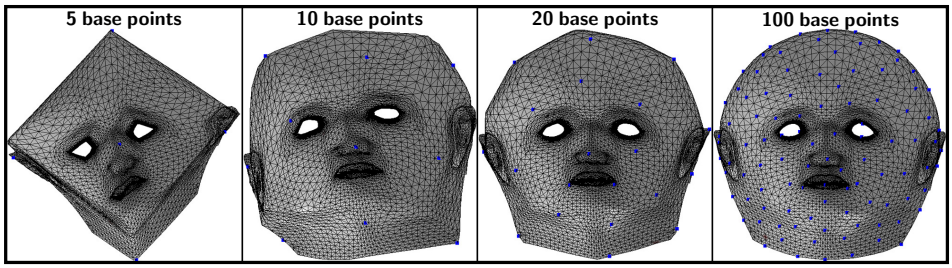

FIGURE 8. Influence of the number of base points. The original model is shown on the left of Figure 7

in [Eck et al., 1995]. We choose to use a scheme based on a weighted geodesic distance [Peyré and Cohen, 2004], since its continuous nature is clearly related to our flattening approach.

On Figure 9 one can see the whole pipeline in action. This includes first a centroidal tessellation of the mesh, then the extraction and flattening of each cell, and lastly the texturing of the model.

Computation Times. For our tests of the flattening procedure, we have chosen to use a fixed number of points (200 points), since the geometric complexity of the meshes was almost constant. The parameterization of [Desbrun et al., 2002] is implemented using the boundary-free formulation (Neumann condition).

Table 1 shows the complexity of the algorithms mentioned in the paper, for a mesh of 10k vertices. The constant A is the number of steps in the gradient descent for the localization of the intrinsic center of mass, which is about $A = 8$ for 10k vertices. The constant B represent the number of base points, which is $n/100$ in our tests. This clearly shows the speed up that Geodesic LLE can bring over global methods such as [Zigelman et al., 2002]. This is confirmed by the running times reported in table 2. For large meshes, the stability of our method is an advantage over the approaches based on large linear system such as [Desbrun et al., 2002], for which it is difficult to ensure the convergence of the conjugate gradient.

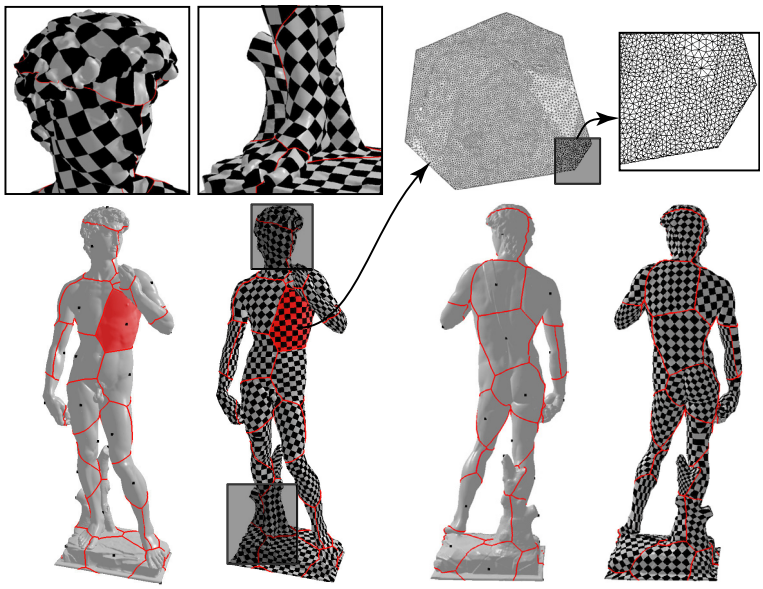

FIGURE 9. Texturing of the David.

	F. Marching	Greedy sampl.	1 Lloyd iter.	Zigelman02	Geodesic LLE
Complexity	$n\log(n)$	$n\log(n)^2$	$An\log(n)$	$Bn\log(n)+B^3$	$n\log(n)+B^2$
Times	$2s$	$10s$	$6s$	$55s$	$28s$

TABLE 1. Complexity of the algorithms

Nbr.vertices	[Zigelman et al., 2002]	[Desbrun et al., 2002]	Geodesic LLE
1,000	$7s$	$3s$	$5s$
10,000	$55s$	$25s$	$28s$
100,000	$440s$	$210s$	$150s$
700,000	$2160s$	$1320s$	$740s$

TABLE 2. Comparison of flattening algorithms

Discussion. The complete texturing of the David mesh (700,000 vertices) shown on Figure 9 clearly enlightens the strengths of our approach:

- The resulting flattening map is smooth, with no face flip (at least on this model). This can be seen on the close-up of the flattened domain.
- The whole texturing procedure takes $740s$, which shows an important speed up with respect to previous methods.
- Our scheme is more local than the flattening procedure of [Zigelman et al., 2002], but it does not reach the per-vertex resolution of classical methods such as [Desbrun et al., 2002]. This enables both fast computations and respect of

	M_1	M_2	M_3
[Desbrun et al., 2002] (conformal)	$E_c = 0.9$ $E_a = 1.2$	$E_c = 1.5$ $E_a = 3$	$E_c = 2.5$ $E_a = 10.4$
[Desbrun et al., 2002] (authalic)	$E_c = 1.4$ $E_a = 0.6$	$E_c = 3.0$ $E_a = 1.1$	$E_c = 8.3$ $E_a = 3.5$
[Zigelman et al., 2002] (MDS)	$E_c = 0.8$ $E_a = 0.9$	The flattening is not valid	
Our scheme (GeodesicLLE)	$E_c = 1.1$ $E_a = 0.9$	$E_c = 1.7$ $E_a = 1.6$	$E_c = 6.5$ $E_a = 5.5$

TABLE 3. Area and angular distortion for various schemes

the small scale variations (bumps or noise), which is not the case of [Zigelman et al., 2002].

Notice that there is no theoretical guarantee on the validity of the flattening. The only cases where face flips occurs is on patches with huge isoperimetric distortion. We believe that this is not a real issue since such degenerate cases can be easily detected and fixed (for example by subdividing the region). It is important to note that classical methods also face similar problems. In [Levy et al., 2002] a cut is performed to accelerate the convergence of the system resolution and ease the parameterization in regions with a sock-like shape. In practice however, our segmentation algorithm ensures that patches that need to be flattened do not contain high curvature variations and the whole process performs very well with no face flip.

Distorsion Measures. To support our claim that our flattening scheme performs a trade-off between conservation of area and conservation of angle, we have performed some test (see table 3). We used 3 finger-like meshes M_1, M_2 and M_3 with increasing isoperimetric distortion. On each face x we compute the eigenvalues (s_1, s_2) of of the Jacobian of the parameterization map (linearly evaluated). Locally, conformality is characterized by $s_1 = s_2$ and conservation of area by $s_1 s_2 = 1$. As a conformal metric, we use

$$E_c(M)^2 = \frac{1}{A} \sum_x \left| \frac{s_1(x)}{s_2(x)} + \frac{s_2(x)}{s_1(x)} - 2 \right|^2 A(x)$$

and as an equi-areal metric we use

$$E_a(M)^2 = \frac{1}{A} \sum_x \left| s_1(x)s_2(x) + \frac{1}{s_1(x)s_2(x)} - 2 \right|^2 A(x)$$

where $A(x)$ is the area of a face x, A is the total area.

5. Conclusion

We have described new algorithms to perform the remeshing and the flattening of a genus-0 triangulated manifold. The main tool that allows having a fast algorithm is the fast marching on a triangulated mesh, together with some improvements we added. We have presented a fast algorithm for remeshing of a surface with a uniform or adaptive distribution. This is based on iteratively choosing the farthest point according to a weighted distance on the surface. We introduced a geodesic version of Locally Linear Embedding that is able to perform fast computations on a given set of points, and to extend the embedding to the rest of the mesh in a transparent manner. The resulting flattening is smooth and achieves a desirable trade-off between conservation of angle and area.

References

[Alliez et al., 2003] Alliez, P., D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun: 2003, 'Anisotropic Polygonal Remeshing'. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference* pp. 485–493.

[Bengio et al., 2003] Bengio, Y., J.-F. Paiement, and P. Vincent: 2003, 'Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering'. *Proc. NIPS 2003*.

[Chew, 1993] Chew, L. P.: 1993, 'Guaranteed-Quality Mesh Generation for Curved Surfaces'. *Proc. of the Ninth Symposium on Computational Geometry* pp. 274–280.

[Cohen, 2001] Cohen, L.: 2001, 'Multiple Contour Finding and Perceptual Grouping Using Minimal Paths'. *Journal of Mathematical Imaging and Vision* **14**(3), 225–236.

[Cohen and Kimmel, 1997] Cohen, L.D. and R. Kimmel: 1997, 'Global Minimum for Active Contour Models: A Minimal Path Approach'. *International Journal of Computer Vision* **24**(1), 57–78.

[Cohen-Steiner and Morvan, 2003] Cohen-Steiner, D. and J.-M. Morvan: 2003, 'Restricted Delaunay Triangulations and Normal Cycles'. *Proc. 19th ACM Sympos. Comput. Geom.* pp. 237–246.

[Desbrun et al., 2002] Desbrun, M., M. Meyer, and P. Alliez: 2002, 'Intrinsic Parameterizations of Surface Meshes'. *Eurographics conference proceedings* **21**(2), 209–218.

[Eck et al., 1995] Eck, M., T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle: 1995, 'Multiresolution Analysis of Arbitrary Meshes'. *Computer Graphics* **29**(Annual Conference Series), 173–182.

[Eldar et al., 1997] Eldar, Y., M. Lindenbaum, M. Porat, and Y. Zeevi: 1997, 'The Farthest Point Strategy for Progressive Image Sampling'. *IEEE Trans. on Image Processing* **6**(9), 1305–1315.

[Floater et al., 2002] Floater, M. S., K. Hormann, and M. Reimers: 2002, 'Parameterization of Manifold Triangulations'. *Approximation Theory X: Abstract and Classical Analysis* pp. 197–209.

[Kimmel and Sethian, 1998] Kimmel, R. and J. Sethian: 1998, 'Computing Geodesic Paths on Manifolds'. *Proc. Natl. Acad. Sci.* **95**(15), 8431–8435.

[Kunert, 2002] Kunert, G.: 2002, 'Towards Anisotropic Mesh Construction and Error Estimation in the Finite Element Method'. *Numerical Methods in PDE* **18**, 625–648.

[Leibon and Letscher, 2000] Leibon, G. and D. Letscher: 2000, 'Delaunay triangulations and Voronoi diagrams for Riemannian manifolds'. *ACM Symposium on Computational Geometry* pp. 341–349.

[Levy et al., 2002] Levy, B., S. Petitjean, N. Ray, and J. Maillot: 2002, 'Least Squares Conformal Maps for Automatic Texture Atlas Generation'. In: ACM (ed.): *Special Interest Group on Computer Graphics – SIGGRAPH'02, San-Antonio, Texas, USA*.

[Moenning and Dodgson, 2003] Moenning, C. and N.A. Dodgson: 2003, 'Fast Marching Farthest Point Sampling'. *Proc. EUROGRAPHICS 2003*.

[Onishi and Itoh, 2003] Onishi, K. and J. Itoh: 2003, 'Estimation of the necessary number of points in Riemannian Voronoi diagram'. *Proc. CCCG*.

[Peyré and Cohen, 2003] Peyré, G. and L.D. Cohen: 2003, 'Geodesic Remeshing Using Front Propagation'. *Proc. IEEE Variational, Geometric and Level Set Methods 2003*.

[Peyré and Cohen, 2004] Peyré, G. and L.D. Cohen: 2004, 'Surface Segmentation Using Geodesic Centroidal Tesselation'. *Proc. 3D Data Processing Visualization Transmission 2004*.

[Roweis and Saul, 2000] Roweis, S. and L. Saul: 2000, 'Nonlinear Dimensionality Reduction by Locally Linear Embedding'. *Science* **290**(5500), 2323–2326.

[Ruppert, 1995] Ruppert, J.: 1995, 'A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation'. *Journal of Algorithms* **18**(3), 548–585.

[Sethian, 1999] Sethian, J.: 1999, *Level Sets Methods and Fast Marching Methods*. Cambridge University Press, 2nd edition.

[Sifri et al., 2003] Sifri, O., A. Sheffer, and C. Gotsman: 2003, 'Geodesic-based Surface Remeshing'. *Proc. 12th International Meshing Roundtable* pp. 189–199.

[Surazhsky et al., 2003] Surazhsky, V., P. Alliez, and C. Gotsman: 2003, 'Isotropic Remeshing of Surfaces: a Local Parameterization Approach'. *Proc. 12th International Meshing Roundtable*.

[Tenenbaum et al., 2000] Tenenbaum, J.B., V. de Silva, and J.C. Langford: 2000, 'A Global Geometric Framework for Nonlinear Dimensionality Reduction'. *Science* **290**(5500), 2319–2323.

[Terzopoulos and Vasilescu, 1992] Terzopoulos, D. and M. Vasilescu: 1992, 'Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierarchical Subdivision'. In: *Proc. IEEE CVPR '92*. Champaign, Illinois, pp. 829–832.

[Tsitsiklis, 1995] Tsitsiklis, J.: 1995, 'Efficient Algorithms for Globally Optimal Trajectories'. *IEEE Trans. on Automatic Control*.

[Ulichney, 1993] Ulichney, R.: 1993, 'The Void-and-Cluster Method for Generating Dither Arrays'. *Proc. IS&T Symposium on Electronic Imaging Science & Technology, San Jose, CA* **1913**(9), 332–343.

[Zigelman et al., 2002] Zigelman, G., R. Kimmel, and N. Kiryati: 2002, 'Texture Mapping Using Surface Flattening via Multi-dimensional Scaling'. *IEEE Trans. on Visualization and Computer Graphics* **8**(1), 198–207.

Gabriel Peyré
CMAP, École Polytechnique, UMR CNRS 7641
e-mail: peyre@cmapx.polytechnique.fr

Laurent Cohen
CEREMADE, Université Paris Dauphine, UMR CNRS 7534
e-mail: cohen@ceremade.dauphine.fr