International Journal of Computer Vision 69(1), 145–156, 2006
© 2006 Springer Science + Business Media, LLC. Manufactured in The Netherlands.
DOI: 10.1007/s11263-006-6859-3

Geodesic Remeshing Using Front Propagation

GABRIEL PEYRÉ
CMAP, École Polytechnique, UMR CNRS 7641
peyre@cmapx.polytechnique.fr

LAURENT D. COHEN
CEREMADE, Université Paris Dauphine, UMR CNRS 7534
Place du Marechal de Lattre de Tassigny, 75775 Paris cedex 16, France
cohen@ceremade.dauphine.fr

Received April 12, 2004; Revised November 12, 2004; Accepted May 11, 2005

First online version published in May, 2006

Abstract. In this paper, we propose a complete framework for 3D geometry modeling and processing that uses only fast geodesic computations. The basic building block for these techniques is a novel greedy algorithm to perform a uniform or adaptive remeshing of a triangulated surface. Our other contributions include a parameterization scheme based on barycentric coordinates, an intrinsic algorithm for computing geodesic centroidal tessellations, and a fast and robust method to flatten a genus-0 surface patch. 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.

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 known 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, perceptual learning, and finite element remeshing) 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 and some contributions about the Fast Marching algorithm and Voronoi diagrams on surfaces.

In Section 3 we will expose a greedy algorithm for manifold sampling and remeshing, which iteratively adds points to find a mesh that has a uniform or adaptive distribution of vertices on the surface. Figure 1 shows some results of our remeshing procedure.

In Section 4 we will expose two applications of our geodesic sampling strategy: the construction of a geodesic centroidal tesselation, and a fast flattening scheme.

In Section 5 we will show the whole pipeline 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

Geodesic Computations. Distances computation on manifolds is a complex topic, and a lot of algorithms have been proposed such as *Chen* and *Han* shortest path method (Chen and Hahn, 1990) which is of quadratic complexity. *Kimmel* and *Sethian*'s *Fast Marching* algorithm (Kimmel and Sethian, 1998) allows finding numerically the geodesic distance from a given point on the manifold, in $O(n \log(n))$ in the number of vertices. They deduce minimal geodesics between two given points. Some direct applications of geodesic computations on manifolds have been proposed, such as in (Kimmel and Sethian, 2000), which applies the Fast Marching algorithm to obtain *Voronoi diagram* and *offset curves* on a manifold.

Surface Remeshing. Huge 3D datasets often arise from surfaces reconstructed in medical imaging for exemple. This reconstruction task can be performed using algorithms from algorithmic geometry, e.g. (Delingette, 1999) or deformable models see (McInerney and Terzopoulos, 1996; Osher and Paragios, 2003). These 3D models can also be acquired from multiple stereo views, e.g. (Fua, 1997), or other industrial applications. These algorithms often produce meshes with a large amount of redundant vertices, and triangulations with poor quality. Thus these meshes must undergo a remeshing process.

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).

Ideas similar to our greedy solution for sampling a manifold (see Section 3.1) 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)).

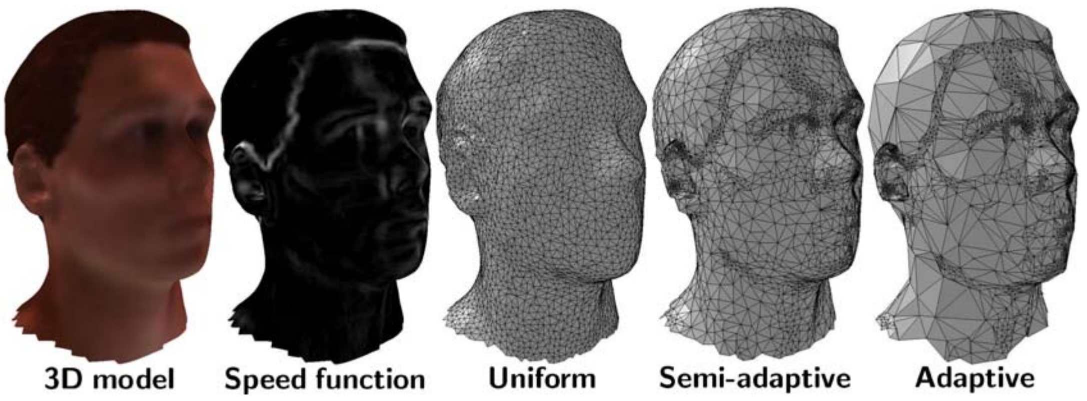

Figure 1. Remeshing of a 3D model using increasing weight for the speed function.

2. Geodesic-Based Building Blocks

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 $\mathscr{S}$ such that $P > 0$. If we have two points $x_0, x_1 \in \mathscr{S}$, the weighted geodesic distance between x_0 and x_1 is defined as

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

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 $U(x) \overset{\text{def.}}{=} d(x_0, x)$ with an accurate and fast algorithm, this minimization can be reformulated as follows. The level set curve $\mathscr{C}_t \overset{\text{def.}}{=} \{x \backslash U(x) = t\}$ propagates following the evolution equation $\frac{\partial \mathscr{C}_t}{\partial t}(x) = \frac{1}{P(x)} \overrightarrow{n_x}$, where $\overrightarrow{n_x}$ is the exterior unit vector normal to the curve at x, and the function U satisfies the nonlinear *Eikonal* equation:

$$||\nabla U(x)|| = P(x). \quad (2)$$

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

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:

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

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. To get more accurate geodesic distance on meshes of bad quality, one can use higher order approximations, e.g. (Manay and Yezzi, 2003), which can be extended to triangulations using a local unfolding of each 1-ring. Figure 2 shows the calculation of a geodesic path computed using a gradient descent of the distance function.

2.2. Extraction of Voronoi Regions

It is possible to start several fronts from points $\{x_1, \dots, x_n\}$ and make them evolve together, as shown on Fig. 3. The areas shown on the surface on the right define the *Voronoi diagram* of the starting

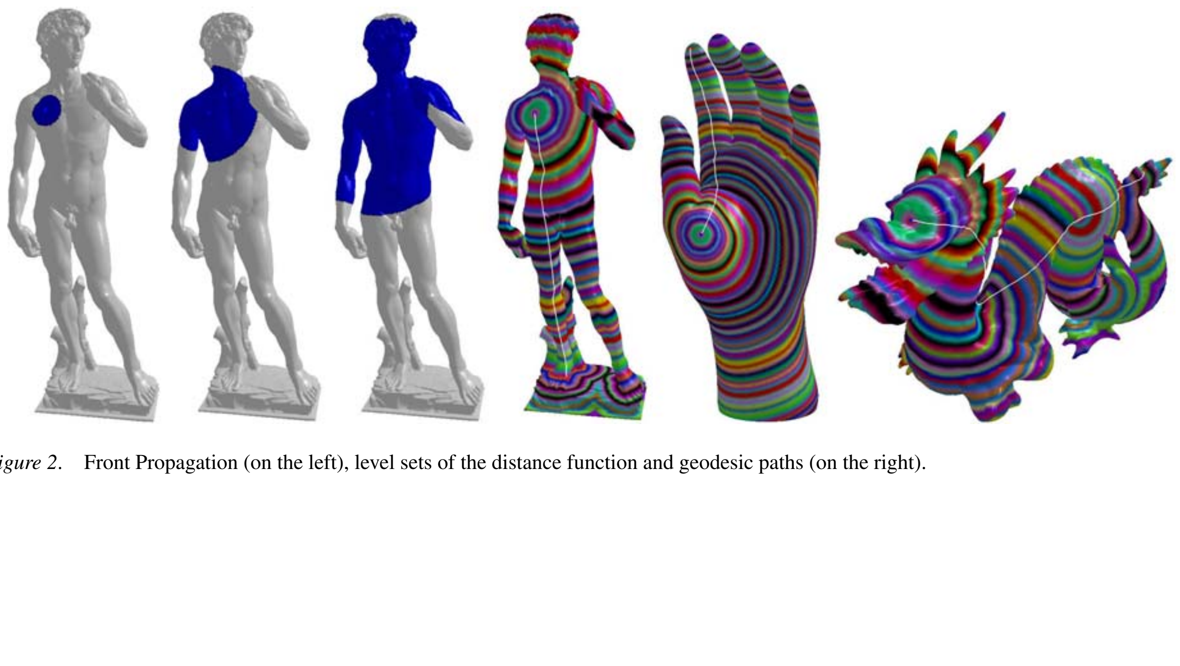

Figure 2. Front Propagation (on the left), level sets of the distance function and geodesic paths (on the right).

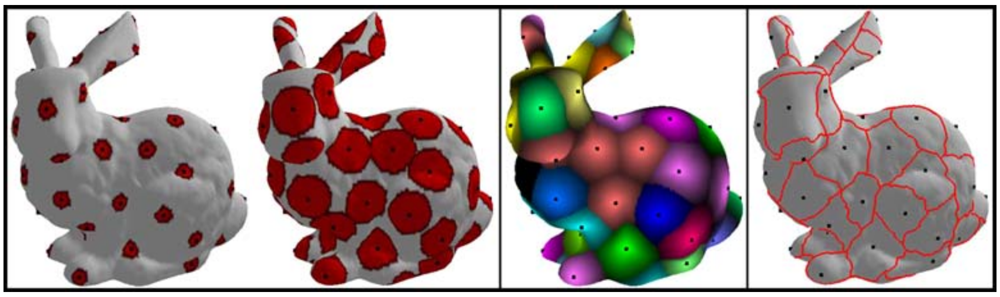
Figure 3. Progression of the fronts, Voronoi diagram, and resulting tessellation.

points, namely the tessellation into the regions, for $i \in \{1, \ldots, n\}$

$$V_i \overset{\text{def.}}{=} \{x \in \mathscr{S} \setminus \forall j \neq i, \quad d(x, x_j) > d(x, x_i)\}.$$

To accurately compute the boundaries of the Voronoi regions, we allow an overlap of the front on one vertex. Suppose a front a arrives at a vertex v_1 with time arrival t_1^a and another front b arrives at a vertex v_2 (connected to v_1) with time t_2^b. Allowing an overlap of the fronts, we record the time arrival t_2^a of a at v_2, and t_1^b of b at v_1. Then the two fronts meet at $(1 - \lambda)v_1 + \lambda v_2$ where $\lambda = \frac{d_2^a - d_1^a + d_1^b - d_2^b}{d_1^b - d_1^a}$.

3. Isotropic Remeshing of a Triangulation

We proposed recently (Peyré and Cohen, 2003) a new method for sampling a 3D mesh 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 introduced 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).

Our approach iteratively adds new vertices based on the geodesic distance on the surface. The result of the algorithm gives a set of vertices uniformly distributed on the surface according to the geodesic distance. Taking into account a local density of vertices will be done in Sections 3.3 and 3.4.

3.1. A Greedy Algorithm for Uniformly Sampling a Manifold

We now describe how to automatically build an evenly spaced set of points on a triangulated surface. A first point x_1 is chosen at random on the mesh and its geodesic distance map U_1 computed by fast marching. A more elaborate choice consists in replacing this random point by the point with maximum distance from it.

Then we assume we have already computed a set of points $S_n = \{x_1, \ldots, x_n\}$, together with U_n the geodesic distance map to S_n. To add a new point x_{n+1}, we simply select a point on the manifold that is furthest away from S_n, meaning that it has maximal value of U_n. To compute the new distance map U_{n+1}, we use the fact that $U_{n+1} = \min(U_n, U_{x_{n+1}})$, where we have noted $U_{x_{n+1}}$ the distance map to x_{n+1}. So we simply need to update U_n by starting a front from x_{n+1} (using the Fast Marching algorithm exposed in Section 2) and to confine it on the set $\{x \ ; \ U_{x_{n+1}}(x) \leq U_n(x)\}$. This assures that the whole remeshing process roughly takes less than $O(N \log(N)^2)$ operations.

At each iteration, the new point x_{n+1} needs not to be a vertex of the original mesh. It can be positioned accurately by interpolating the distance map. To be more precise, it happens most often that the point with maximum distance is located in a triangle where three different fronts meet. We simply compute the intersection of each pair of fronts along each edge as it is described in Section 2.2. We then choose for x_{n+1} the center of mass of the three intersection points.

We choose to stop the algorithm either when the last added point x_{n+1} satisfies $U_n(x_{n+1}) \leq \delta$, where δ is a given threshold, or when a given number of points have been distributed. Figure 4 shows the first steps of our algorithm on a square surface.

3.2. Calculation of the Geodesic Triangles

Once we have found the complete set S_{n_0}, we must determine which vertices to link together to obtain our

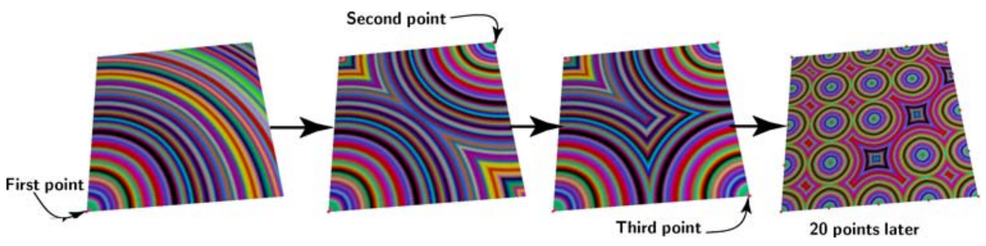

Figure 4. An overview of our greedy algorithm.

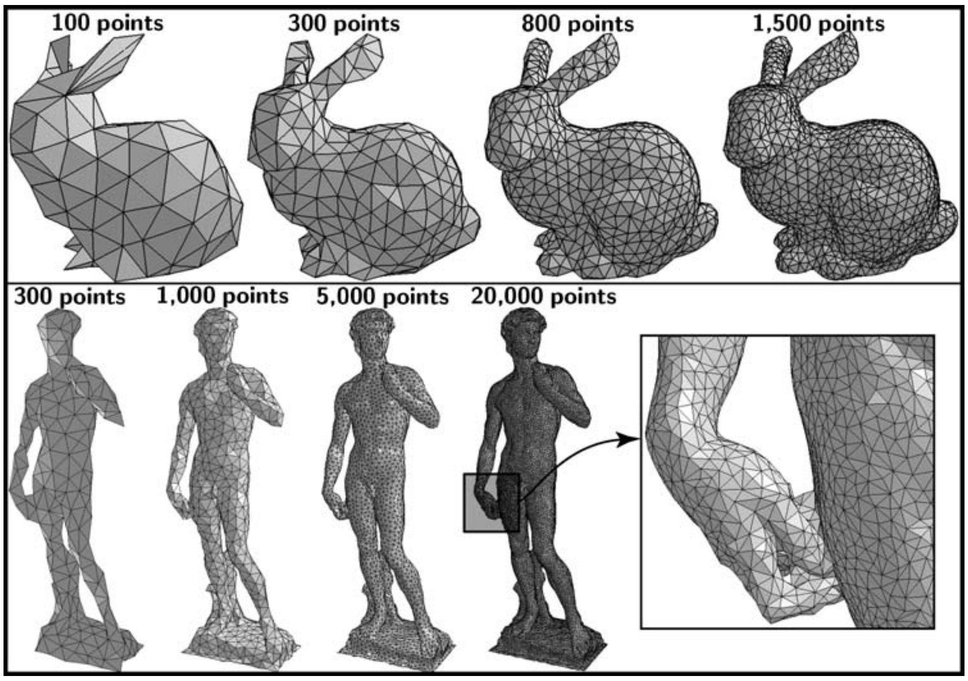

Figure 5. Geodesic remeshing with an increasing number of points.

new triangulation $\mathscr{T}$ which is built incrementally during the algorithm. To that end, during the point distribution process we keep track of *saddle points* (see Cohen (2001)), which are vertices v that satisfy these two criterions:

- When the value of $U(v)$ is set by the Fast Marching algorithm, two fronts coming from different base points x_i and x_j must meet for the *first* time at v (see Cohen (2001)).
- Adding edge $[x_i,\ x_j]$ to the new triangulation $\mathscr{T}$ must keep the triangulation valid (e.g. the edge must have at most two adjacent faces).

The set of saddle points tells us which vertices x_i and x_j should be linked together to obtain a valid triangulation $\mathscr{T}$. Note that when we update a distance map U_{n+1}, a previously found saddle point v can disappear (if v is reached by the front coming from x_{n+1}), and of course new saddle points can be created.

Figure 5 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.

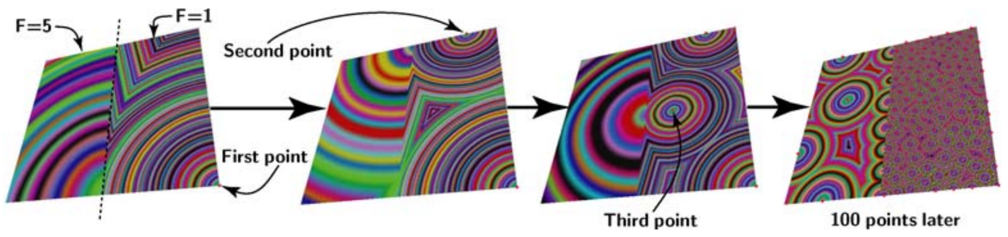

Figure 6. Adaptive remeshing: Iterative insertion of points in a square surface.

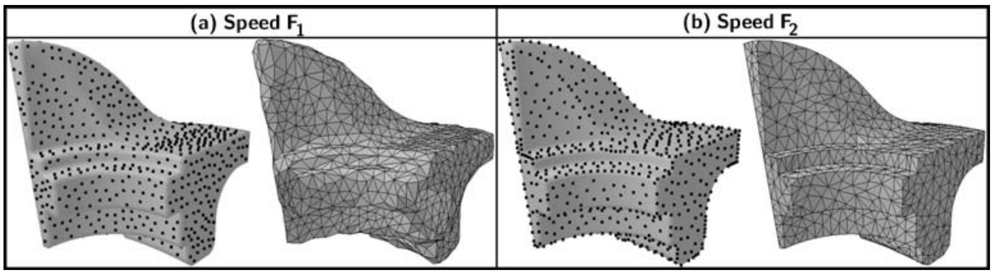

Figure 7. Curvature-repulsive versus curvature-attractive sampling and remeshing.

3.3. Adaptive Remeshing

In the algorithm presented in Sections 3.1 and 3.2, 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) that is not constant across the surface. Figure 6 shows the progressive sampling of a square surface using a speed function with two different values. The colors show the level sets of the distance function U to the set of selected points.

Since vertices are added at maximal values of the geodesic weighted distance, the resulting mesh will be dense in regions with smaller F, and in regions with higher F the mesh will be sparse. This is due to the fact that the algorithm distributes points in such a way that their weighted geodesic distances to neighbors are almost equal. The geodesic distance to vertices in a region with higher value of P is thus smaller. Function F can reflect the need of the user to refine some specific regions with more vertices.

To illustrate our approach, we give an example of a mesh obtained from range scanning. A picture I of the model can be mapped onto the 3D mesh. Using a function F of the form $F(x) = \frac{1}{1+\mu|grad(I(x))|}$, 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 $\mu = 20/\max(|grad(I(x))|)$ (highly adaptive).

3.4. Curvature-Based Remeshing

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 $\tau(x) \overset{\text{def.}}{=} |\lambda_1| + |\lambda_2|$ 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 $F_1(x) \overset{\text{def.}}{=} 1 + \varepsilon\tau(x)$ and $F_2(x) \overset{\text{def.}}{=} \frac{1}{1+\mu\tau(x)}$, where ε and μ are two user-defined parameters. Figure 7(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 more vertices in region with high curvature such as mesh corners and edges. Figure 7(b) shows that this speed function leads to very good results for the remeshing

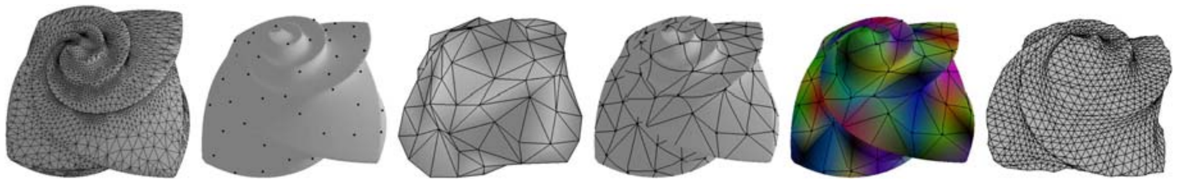

Figure 8. Different steps in mesh parameterization: from left to right original mesh, automatic determination of basis points, determination of associated Delaunay triangulation, calculation of the corresponding geodesic triangles, parameterization interpolation, semi-regular remeshing.

of a surface with sharp features, which is obviously not the case for the "edge repulsive" speed function (Fig. 7(a)).

4. Applications to Mesh Parameterization, Segmentation and Flattening

4.1. Mesh Parameterization

In Peyré and Cohen (2003) we proposed a simple scheme to parameterize an arbitrary triangulated manifold using a coarse triangulation as base domain. This coarse version of the mesh is built using our geodesic remeshing. The parameterization on each corresponding geodesic triangle of the original mesh is built with a geodesic extension of the barycentric coordinates that makes use of Heron formula.

The resulting parameterization is smooth on each triangle of the base domain, and only continuous across the geodesic edges (much more complex method have to be used to get globally smooth parameterization, see (Khodakovsky et al., 2003)). It can be used to build a semi-regular mesh, as shown on Fig. 8, and a simple relaxation scheme can be used to regularize vertices location near the boundaries of geodesic triangles (see (Lee et al., 1998)).

4.2. Mesh Segmentation Using Centroidal Tessellation

In Peyré and Cohen (2004b) we have proposed an automatic mesh segmentation method very well adapted to the tessellation of a complex manifold in elementary domains topologically equivalent to a disk. In this method the mesh is cut into regions that best satisfy the following properties:

(C1) Boundaries of the regions agree with sharp features of the surface.

(C2) Regions are as compact as possible (the ratio area/perimeter should be large), enclosing equal areas.

The goal of the algorithm is to build a segmentation $\mathscr{S} = \bigcup_{i=1}^{n} V_i$ of a triangulated manifold $\mathscr{S}$. The V_i will be the Voronoi regions associated with a given set of points $\{v_1, \dots, v_n\}$. Initially, these points are chosen using the sampling algorithm of Section 3.1. Then a geodesic extension of Lloyd algorithm will refine the location of the v_i so that they agree with the geodesic center of mass of each region V_i. Figure 9 shows some iterations of the Lloyd relaxation scheme, and we obtain a geodesic bee-hive segmentation. At each iteration, the center of mass of each region is computed using a gradient descent of the energy

$$E_i(w) \overset{\text{def.}}{=} \int_{x \in V_i} d(x, w)^2 \mathrm{d}s.$$

where ds is the area element on the surface and d is the geodesic distance.

In order to force the boundaries of the regions V_i to follow the discontinuities of the surface, we use the "edge attractive" speed function $F_2(v)$ defined in Section 3.4. This will allow us to "freeze" the front in regions with high curvature. This way the resulting Voronoi regions will have boundaries aligned with sharp features of the surface, and condition (C1) will be satisfied. Figure 10 shows the segmentation we obtain on complex models. In the close-up we can see that the cell boundaries try to follow the edges of the mesh whenever it is possible.

Following the ideas of the Section 3.3, we can use a texture function to modulate the speed function. The resulting segmentation can take into account both the texture intensity and the curvature information, according to the user will. Figure 11 shows the segmentation of a texture computed directly on the triangulated mesh (which can have arbitrary topology). The "segmented function" on the right is the speed function represented on the 2D parameter space together with the boundaries of the Voronoi cells.

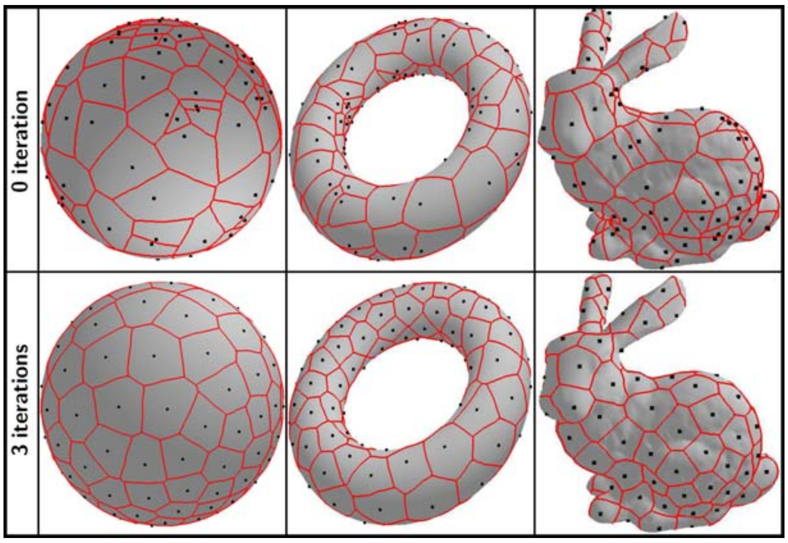

Figure 9. Lloyd iterations on various models.

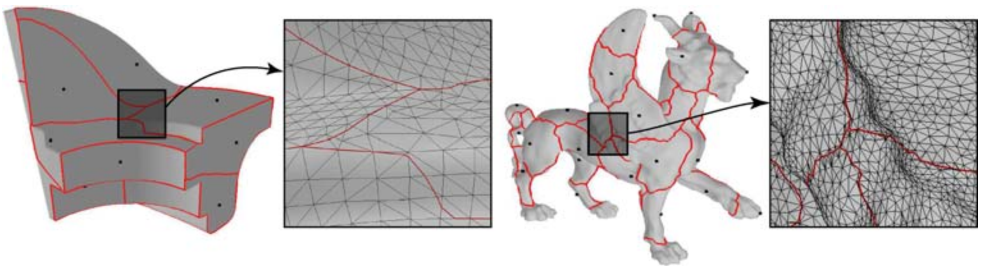

Figure 10. Segmentation of two complex models.

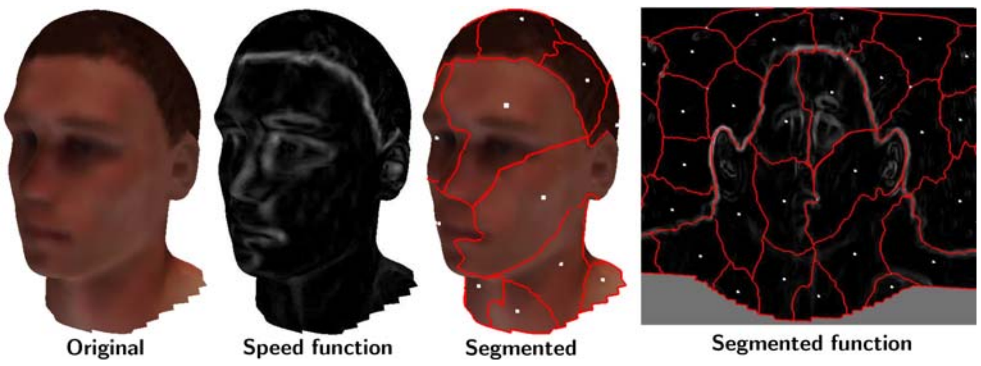

Figure 11. Segmentation of a textured head.

4.3. Fast Geodesic Flattening

In Peyré and Cohen (2004a) we have proposed a method for flattening a triangulated manifold. This method is very fast and in contrast to traditional methods (see Floater et al. (2002) for a complete survey, and (Gu and Yau, 2003; Khodakovsky et al., 2003) for more complex and global schemes), it does not

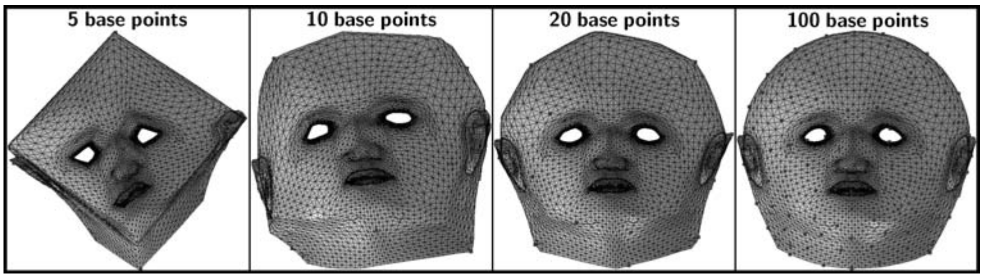

Figure 12. Flattening: influence of the number of base points. The original model is shown on the left of Fig. 13.

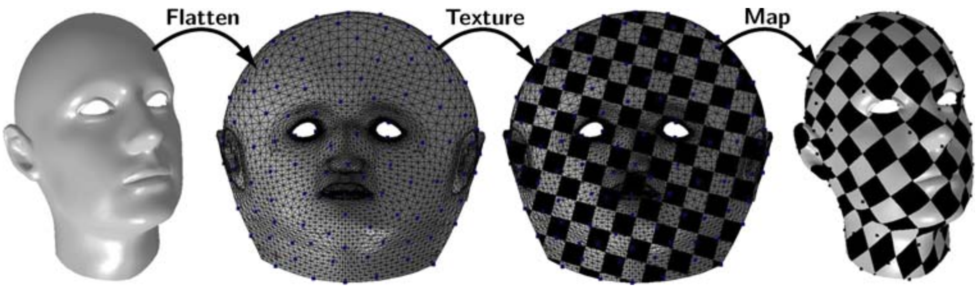

Figure 13. From left to right, the original model, texture on the flattened domain, and on the 3D mesh.

require the solution of a large and ill-conditioned linear system.

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).

The multidimensional scaling approach to flattening of Zigelman et al. (2002) is closely related to IsoMap. In order to speed-up the computation and to achieve a local control over the flattening, we have extended the classical LLE approach to the geodesic setting. Our procedure can be used to flatten a small set of points (chosen using the greedy procedure presented in Section 3.1). We then extend the mapping to the whole mesh using a Nyström integral relation, as already proposed in Bengio et al. (2003). Figure 13 shows the extension of the flattening from this small set of points and figure 13 shows the influence of the number of points. We can then use the resulting mapping to perform texture mapping, see Fig. 13.

5. Results and Discussion

Uniform remeshing. To show the improvement of the quality of the mesh that our uniform remeshing algorithm can bring, we report in Table 1 the average angle of the triangulation together with the standard deviation of the angle repartition.

Adaptive remeshing. To study the behavior of our isotropic remeshing as an approximation procedure, we measure the mean-square Hausdorf distance between the original mesh and a coarse version produced by our method. We also compare the distortion result of our scheme with another greedy procedure, the progres-

Table 1. Average and standard deviation of the angle distribution.

	David (Fig. 5) (700k vert.)	Feline (Fig. 10) (50k vert.)	Bunny (Fig. 5) (35k vert.)	Fandisk (Fig. 10) (6500 vert.)
Original	54° (± 8°)	60° (± 7°)	59° (± 6°)	57° (± 8°)
Remeshing (10% #verts)	58° (± 2°)	60.5° (± 2.5°)	61° (± 4°)	59° (± 4°)
Remeshing (30% #verts)	59° (± 2°)	61° (± 2°)	60° (± 3.5°)	59° (± 3°)

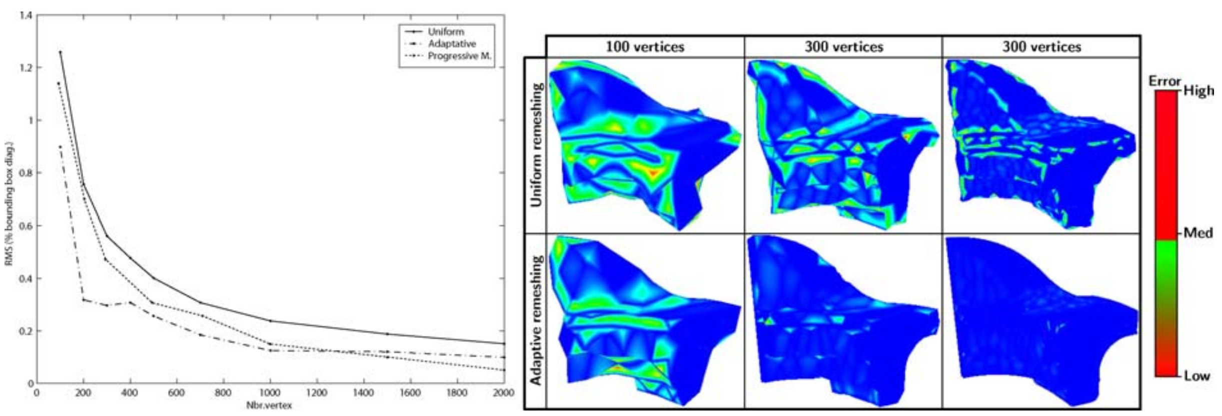

Figure 14. Remeshing error. Left: distorsion curve, right: graphic display of the location of the error.

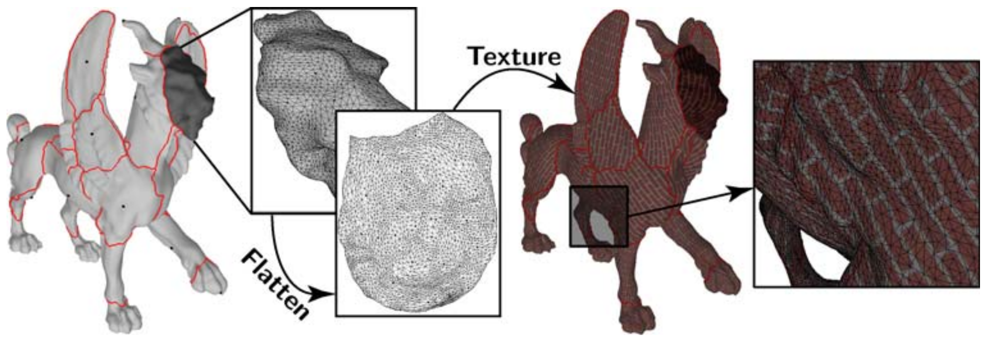

Figure 15. Texturing of the feline model.

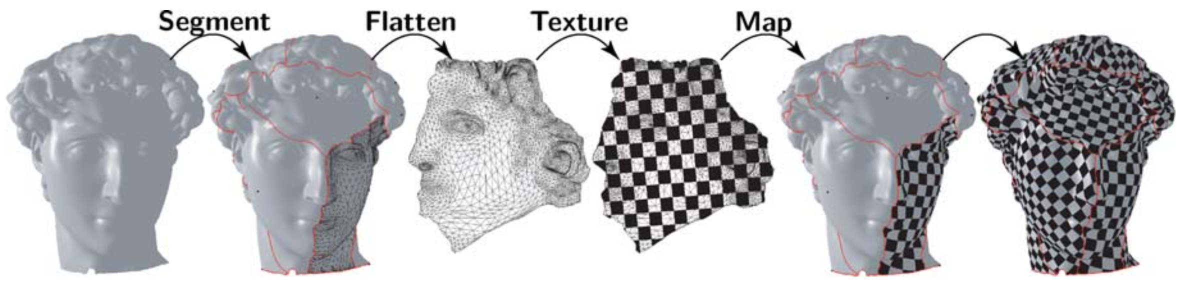

Figure 16. An overview of our pipeline. The mesh is first segmented using a weighted geodesic centroidal tessellation. Each resulting patch is then flattened using the Geodesic LLE procedure. At last, we can perform texture mapping on each base domain.

sive mesh (Hoppe, 1996). On the left of Fig. 14 one can see the decreasing of the error with the number of vertices, and on the right a display of the location of the error. This clearly shows the strength of our scheme for models with sharp features.

Texturing of a Complex Model. On Figs. 15 and 16 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. Table 2 shows the complexity of the algorithms mentioned in the paper. 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 10 k vertices. The constant B represent

Table 2. Complexity of the algorithms.

	F. Marching	Greedy sampling	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 (10k vert.)	$2s$	$10s$	$6s$	$55s$	$28s$

the number of base points, which is 100 in our tests. This clearly shows the speed up that Geodesic LLE can bring over global methods such as Zigelman et al. (2002). On a mesh of 700k vertices, the whole pipeline (i.e. segmentation, sampling and flattening) takes 740s for Geodesic LLE, 1320s for the classical method of Desbrun et al. (2002) and several hours for Zigelman et al. (2002).

Our future works include a theoretical study of the quality of the embedding given by our algorithms. We also would like to analyze experimentally the quality of the whole pipeline. A good way of evaluating the efficiency of such a scheme is to use its output to perform mesh compression. The mesh atlas provided by our algorithm is an ideal pre-processing step for performing wavelet transform in parameter space, in a fashion similar to Sander et al. (2003).

6. Conclusion

We have described a complete pipeline for 3D mesh resampling, segmentation and flattening. The main tool that allows to have a fast algorithm is the fast marching on a triangulated mesh, together with some improvements we added. We introduced a fast algorihtm 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. The first stage of the pipeline is the segmentation of the mesh into a set of genus-0 patches. Our contribution there includes a geodesic extension of the Lloyd algorithm that is able to construct a geodesic centroidal tessellation. This iterative algorithm takes into account curvature information of the surface and is very well suited to building a set of base domains for mesh flattening. The second stage of our pipeline is a geodesic flattening procedure. 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, P., Cohen-Steiner, D., Devillers, O., Levy, B., and Desbrun, M. 2003. Anisotropic Polygonal Remeshing. *ACM Transactions on Graphics. Special Issue for SIGGRAPH Conference*, pp. 485–493.

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

Chen, J. and Hahn, Y. 1990. Shortest Path on a Polyhedron. *Proc. 6th ACM Sympos*, Comput Geom pp. 360–369.

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

Cohen, L. 2001. Multiple contour finding and perceptual grouping using minimal paths. *Journal of Mathematical Imaging and Vision*, 14(3):225–236.

Cohen, L.D. and Kimmel, R. 1997. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision* 24(1):57–78.

Cohen-Steiner, D. and Morvan, J.-M. 2003. Restricted delaunay triangulations and normal cycles. In *Proc. 19th ACM Sympos*, Comput. Geom. pp. 237–246.

Delingette, H. 1999. General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146.

Desbrun, M., Meyer, M., and Alliez, P. 2002. Intrinsic parameterizations of surface meshes. *Eurographics Conference Proceedings*, 21(2):209–218.

Eldar, Y., Lindenbaum, M., Porat, M., and Zeevi, Y. 1997. The farthest point strategy for progressive image sampling. *IEEE Trans. on Image Processing*, 6(9):1305–1315.

Floater, M.S., Hormann, K., and Reimers, M. 2002. Parameterization of manifold triangulations. *Approximation Theory X: Abstract and Classical Analysis*, pp. 197–209.

Fua, P. 1997. From multiple stereo views to multiple 3-D Surfaces. *International Journal of Computer Vision*, 24(1):19–35.

Gu, X. and Yau, S.-T. 2003. Global conformal surface parameterization. In *Proc. ACM Symposium on Geometry Processing*, 2003 pp. 127–137.

Hoppe, H. 1996. Progressive meshes. In *Proc. ACM SIGGRAPH* 1996 pp. 99–108.

Khodakovsky, A., Litke, N., and Schröder, P. 2003. Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics. Special Issue for SIGGRAPH Conference*, pp. 350–357.

Kimmel, R. and Sethian, J. 1998. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci.*, 95(15):8431–8435.

Kimmel, R. and Sethian, J.A. 2000. Fast voronoi diagrams on triangulated surfaces. In *Proc. of the 16th European Workshop on Comp. Geom.*, (EUROCG-00). pp. 1–4.

Kunert, G. 2002. Towards anisotropic mesh construction and error estimation in the finite element method. *Numerical Methods in PDE*, 18:625–648.

Lee, A.W.F., Schröder, P., Sweldens, W., Cowsar, L., and Dobkin, D. 1998. MAPS: Multiresolution adaptive parameterization of surfaces. *Computer Graphics* 32(Ann. Conf. Series), 95–104.

Leibon, G. and Letscher, D. 2000. Delaunay triangulations and voronoi diagrams for riemannian manifolds. *ACM Symposium on Computational Geometry*, pp. 341–349.

Manay, S. and Yezzi, A. 2003. Second-order models for computing distance transforms. In *Proc. IEEE Variational*, *Geometric and Level Set Methods 2003*, pp. 105–112.

McInerney, T. and Terzopoulos, D. 1996. Deformable models in medical image analysis: A survey. *Medical Image Analysis* 1(2):91–108.

Moenning, C. and Dodgson, N.A. 2003. Fast marching farthest point sampling. In *Proc. EURO-GRAPHICS*, 2003.

Onishi, K. and Itoh, J. 2003. Estimation of the necessary number of points in Riemannian Voronoi diagram. In *Proc. CCCG* 2003, pp. 19–24.

Osher, S. and Paragios, N. 2003. *Geometric Level Set Methods in Imaging, Vision, and Graphics.* Springer-Verlag New York, Inc.

Peyré, G. and Cohen, L.D. 2003. Geodesic remeshing using front propagation. In *Proc. IEEE Variational, Geometric and Level Set Methods 2003*, pp. 33–40.

Peyré, G. and Cohen, L.D. 2004a. Geodesic computations for fast and accurate surface flattening. *Preprint CMAP*.

Peyré, G. and Cohen, L.D. 2004b. Surface Segmentation Using Geodesic Centroidal Tesselation. In *Proc. IEEE 3D Data Processing Visualization Transmission 2004*, pp. 995–1002.

Roweis, S. and Saul, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Ruppert, J. 1995. A delaunay refinement algorithm for quality 2-Dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585.

Sander, P., Wood, Z., Gortler, S., Snyder, J., and Hoppe, H. 2003. Multi-chart geometry images. *Proc. Symposium on Geometry Processing* 2003 pp. 146–155.

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

Sifri, O., Sheffer, A., and Gotsman, C. 2003. Geodesic-based surface remeshing. In *Proc. 12th International Meshing Roundtable*, pp. 189–199.

Surazhsky, V., Alliez, P., and Gotsman, C. 2003. Isotropic remeshing of surfaces: A Local Parameter-ization Approach. In *Proc. 12th International Meshing Roundtable*.

Tenenbaum, J.B., de Silva, V., and Langford, J.C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Terzopoulos, D. and Vasilescu, M. 1992. Adaptive meshes and shells: Irregular triangulation, dis-continuities, and hierarchical subdivision. In *Proc. IEEE CVPR 92. Champaign, Illinois*, pp. 829–832.

Tsitsiklis, J. 1995. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control*.

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, G., Kimmel, R., and Kiryati, N. 2002. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Trans. on Visualization and Computer Graphics*, 8(1):198–207.