Centroidal Voronoi Diagrams for Isotropic Surface Remeshing

Pierre Alliez [a], Éric Colin de Verdière [b], Olivier Devillers [a], Martin Isenburg [c]

[a] *INRIA, Sophia-Antipolis, France*

[b] *Laboratoire d'informatique de l'École normale supérieure, Paris, France*

[c] *University of North Carolina at Chapell Hill*

Abstract

This paper proposes a new method for isotropic remeshing of triangulated surface meshes. Given a triangulated surface mesh to be resampled and a user-specified density function defined over it, we first distribute the desired number of samples by generalizing error diffusion, commonly used in image halftoning, to work directly on mesh triangles and feature edges. We then use the resulting sampling as an initial configuration for building a weighted centroidal Voronoi diagram in a conformal parameter space, where the specified density function is used for weighting. We finally create the mesh by lifting the corresponding constrained Delaunay triangulation from parameter space. A precise control over the sampling is obtained through a flexible design of the density function, the latter being possibly low-pass filtered to obtain a smoother gradation. We demonstrate the versatility of our approach through various remeshing examples.

Key words: Surface sampling, error diffusion, centroidal Voronoi diagram, constrained Delaunay triangulation, parameterization, optimal cutting, polygonal schema.

⋆ A preliminary version of this paper appeared in the *Proceedings of the International Conference on Shape Modelling and Applications* 2003.

Email addresses: Pierre.Alliez@sophia.inria.fr (Pierre Alliez), Eric.Colin.de.Verdiere@ens.fr (Éric Colin de Verdière), Olivier.Devillers@sophia.inria.fr (Olivier Devillers), isenburg@cs.unc.edu (Martin Isenburg).

URLs: http://www-sop.inria.fr/geometrica/team/Pierre.Alliez/ (Pierre Alliez), http://www.di.ens.fr/~colin/ (Éric Colin de Verdière),

This paper appeared in *Graphical Models*, 67(3) (2005), 204–231.

1 Introduction

Many applications in simulation, visualization, or computer graphics require a model of 3D surface geometry. The most commonly used representation is the triangle mesh. Such meshes can be the result of careful design using modeling software, or may come as an output of a scanning device, associated with reconstruction or computer vision algorithms.

In this paper, we make the assumption that geometric details are captured accurately in the given model. The original mesh can be seen as one particular instance of the surface geometry of interest. We aim at generating new instances (*e.g.*, new triangle meshes) of this surface geometry that better fit user-specified demands on complexity, sampling, regularity, connectivity, gradation, and quality. For example, a laser scanner often performs a uniform sample acquisition while sweeping a line or point-based beam, without any a priori knowledge of the surface geometry. This may lead to under-sampling or oversampling of certain regions, mainly depending on the angle between the laser beam and the surface normal. Such drawbacks are often alleviated by tuning the sampling rate sufficiently high. Note that the same problems arise for surface extraction from volume data using marching cubes algorithms, where choosing a sufficiently fine grid may lead to overly complex models. In general, simplification techniques are then applied to reduce the complexity and better adapt the sampling to geometry. The lack of flexibility found in the (re-)sampling strategies of most simplification algorithms motivates this work.

Our goal is to provide a flexible technique to remesh triangulated surfaces so that the remeshed models are better suited to a subsequent process, *e.g.* visualization, finite element simulation, storage, transmission, or any mesh processing technique. See Figure 1 for example results of the proposed remeshing technique.

1.1 Related Work

Mesh generation received much attention from various interest groups ranging from Computer Graphics over Numerical Analysis to Computational Geometry. Finite element mesh generation usually amounts to finding a partition of a given domain that is optimal according to some criteria related to shape of elements, angles, sizes or complexity (see [2,3]). In most cases only the frontier of the domain has to be given, the goal being to discretize this domain in ac-

`http://www-sop.inria.fr/geometrica/team/Olivier.Devillers/` (Olivier Devillers), `http://www.cs.unc.edu/~isenburg/` (Martin Isenburg).

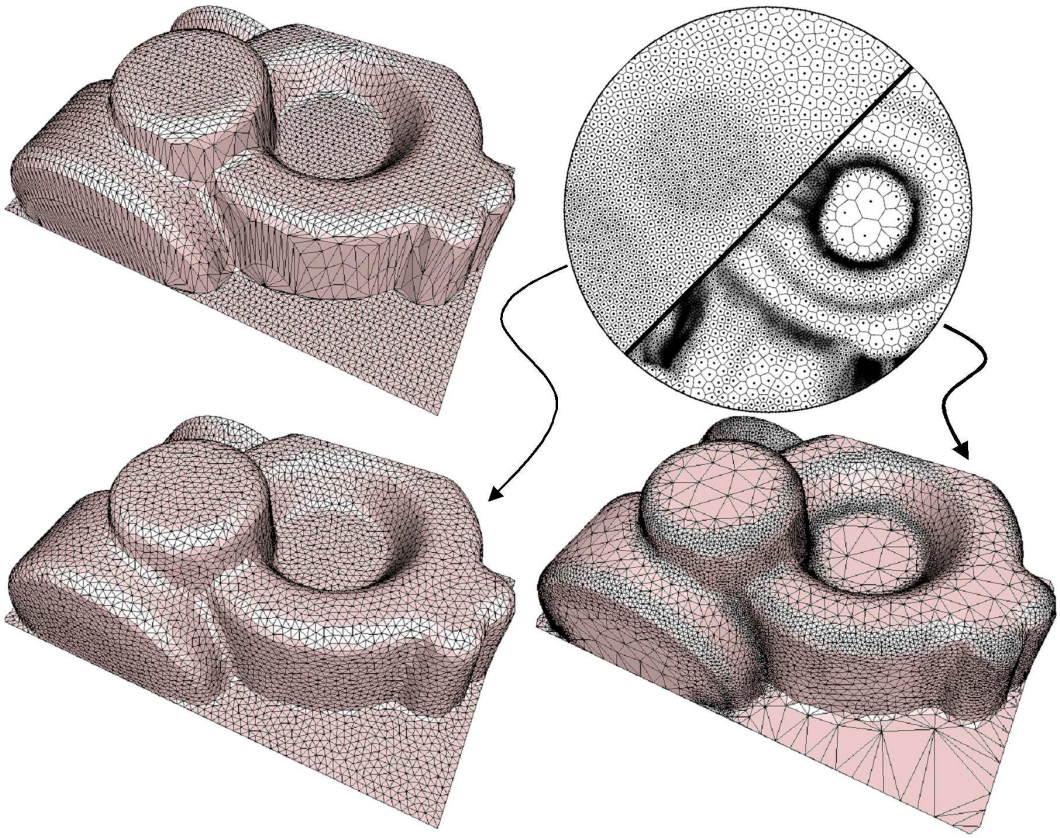

Fig. 1. Top left: the Darmstadt benchmark model. Bottom: two examples of uniform and curvature-adapted remeshing of the Darmstadt benchmark model with 5k and 10k vertices respectively. Top right, two close-ups of the weighted centroidal Voronoi diagram computed in a conformal parameter space to produce the sampling.

cordance with an importance function. Our problem is slightly different since it is dealing with surface *remeshing*. The domain to discretize is now given by an original surface mesh that has to be re-discretized so that the result best matches some user-specified properties.

1.1.1 Meshes for Numerical Analysis

Numerical analysts essentially focus on mesh quality, since it impacts the numerical accuracy of computations performed on the mesh elements [2,3]. We distinguish between techniques that use a parameter space and techniques that act on an explicit mesh. The key idea of the first is to partition a parameter domain into a set of adjacent elements that match some specified properties [14,56,26]. The key idea of the second is to progressively adapt an explicit mesh by performing elementary operations [12,48,7,8,23,51,9].

1.1.2 Meshes for Graphics

Computer graphists mainly focus on remeshing for efficient visualization or geometry processing [28,55]. In an early work, Turk [58] proposed a re-tiling technique that resamples an input mesh by applying a relaxation method to a set of points initially placed at random on the mesh: after this operation, the distribution of the points is correct and the original vertices can be removed. We should also mention mesh simplification [25,42,31,62] and refinement [4] methods, which also generate a new mesh starting from a given one. Such schemes primarily aim at adapting the complexity of the mesh to an acceptable level for graphics visualization hardware or simulation algorithms. For efficient mesh processing, most previous works have focused on semi-regular remeshing [40,38,29,37,32], the latter techniques often requiring a first simplification stage. Kobbelt *et al.* focus on feature-sensitive remeshing techniques [60,10,11] to reduce the artifacts produced when converting a given geometry into a triangle mesh.

More recently, Gu *et al.* [27] proposed a technique for regular remeshing of surface meshes. Surfaces of arbitrary genus are first cut into patches, then parameterized using a signal-adapted technique [53], and finally represented as a set of images that store the geometry, the normals and any attributes used for visualization purpose. Such a storage is compact and drastically simplifies the rendering pipeline since all cache indirections found in usual irregular mesh rendering are simply removed. In a recent work, we have proposed an interactive sampling technique [1]. A mesh is decomposed into a set of disk-like patches, and each patch is parameterized. We then measure some geometric and differential quantities and generate a set of maps inserted in a pipeline of signal processing algorithms. The output of this pipeline is a density map, interactively resampled using an error diffusion technique commonly used for gray level image halftoning. Although the technique described in [1] offers a fairly good level of flexibility, its main drawback is the pixel grid layout constraint that turns out to be memory consuming for complex models with a high range of area distortion.

1.2 Goals and Contributions

Based on the above observations, the goal of this paper is to remove the constraint coming from the regular and uniform structure of images for both sampling and optimization. This led us to extend the concept of direct error diffusion onto triangle meshes for sampling. The motivation of this paper being also to formulate the issue of surface sampling with a larger set of Computational Geometry tools at hand, we demonstrate the relevance of building a weighted centroidal Voronoi diagram for distributing a set of samples in

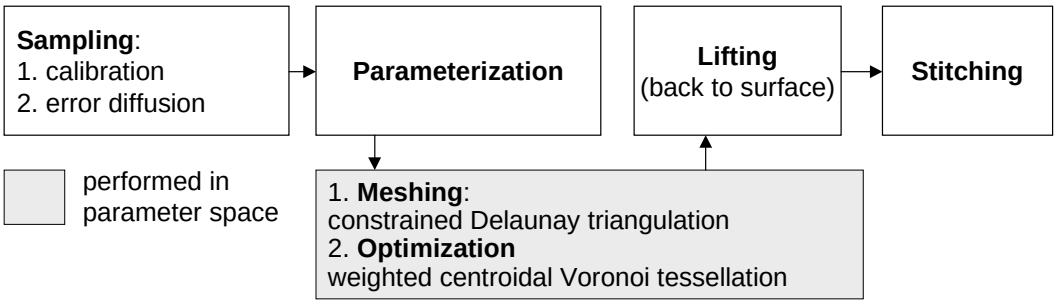

Fig. 2. Remeshing pipeline.

accordance with a specified density function. Another motivation of this paper is to move from the *unit length* paradigm used for numerical analysis [26] to the *unit cell* tiling, well suited for targeted application, *i.e.*, surface shape modeling. The first technique aim at generating meshes with unit edge length measured in a control space metric, while our algorithm tends to partition the surface with unit density *integrated* over the cells of a centroidal Voronoi diagram. The latter property is indeed intricately related to the notion of *isotropic sampling*.

1.3 Overview of the Algorithm

The first stage of our algorithm provides an initial geometry resampling by performing an *error diffusion* process directly over the original triangle mesh (see Section 3). The second stage computes a conformal parameterization of the original model over a planar domain, connects the samples using a constrained Delaunay triangulation built in parameter space, then optimizes the sampling by building a *weighted centroidal Voronoi diagram* in parameter space (see Section 4). Figure 2 shows the tasks requiring a parameter space.

2 Preliminaries

All meshes considered in this paper are presumed to be oriented manifolds of arbitrary genus and possibly with boundary. They are seen as an approximation of a underlying piecewise smooth surface. The input of the algorithm is a set $(\mathcal{M}, \mathcal{F}, d_s, d_f)$, where $\mathcal{M}$ is a triangular mesh, $\mathcal{F}$ is a set of feature edges (defined below), $d_s : \mathcal{M} \to \mathbb{R}$ is an importance function that specifies the *ideal* sampling density for every surface point, and $d_f : \mathcal{F} \to \mathbb{R}$ is an importance function that specifies the density for every point located on a feature edge.

We assume the density functions d_s and d_f to be specified by the user or deduced from differential quantities estimated on the input mesh. In our ex-

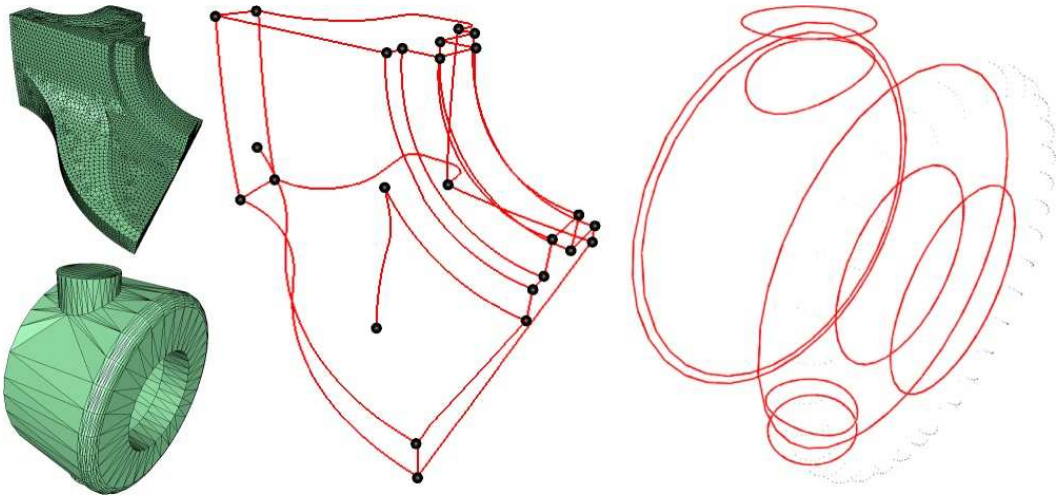

Fig. 3. Two models and their feature skeleton, made of both open and closed backbones.

periments, we use discrete differential geometry techniques [46] to approximate the curvature on every vertex or on the wedges surrounding a corner or a crease vertex. We also provide some options for the user to specify a transfer function (typically a gamma function as described in [1]) and the amount of low-pass filtering over the density functions to control the final mesh gradation. Before diffusing the error using the resulting density function, we extract a set of *features edges* from the original mesh, which receive special treatment during processing. Feature edges are of three types:

- the *sharp edges* represent the main features of the object; they are classified using simple dihedral angle thresholding in our implementation, yet a more sophisticated approach could be used (see *e.g.* [33,61] to cite a few);
- the *boundary edges* are incident to exactly one face;
- the *cut edges* are additional edges which we also need to consider separately for parameterization of closed or genus > 0 models (see Section 4.1).

Similarly to Botsch and Kobbelt [10], we then chain the set of incident feature edges as a set of backbones. The resulting feature skeleton will be resampled as a set of curves (see Figure 3).

3 Error Diffusion

Because of its simplicity and efficiency, the error diffusion algorithm received much attention [59,47] since its introduction in 1976 by Floyd and Steinberg [22]. Before describing our algorithm, we review the concept of error diffusion when used for grey level halftoning of images.

3.1 Concept

The core principle of error diffusion in image halftoning consists of processing each pixel of an input signal image according to a path: every pixel is binary quantized according to a given threshold. The signed quantization error is then distributed to its unprocessed neighbors according to error distribution coefficients. Elaborating an error diffusion algorithm consists of finding the best *processing path* and choosing the best *distribution coefficients* so that the sampling spectrum exhibits a so-called blue noise profile (see [47,34] for more details). One of the most appealing property of this concept for our application lies into the global preservation of density offered by error diffusion. Such a property means that the average grey level of the discretized image is closely equal to the one of the original image. As shown in [1], this guarantees an exact vertex budget for any specified sampling.

In our context, the task of sampling consists of distributing a set of samples on the original mesh triangles or feature edges so that they locally match the specified density. If one looks at the density as an input signal to discretize, this task is not far from the issue of image halftoning. As shown in [1], recent error diffusion techniques [47] work fairly well for a limited class of models. In this paper we remove the discrete constraints of the image pixel grid layout by generalizing the concept of error diffusion directly over the original mesh triangles. This task is achieved by organizing a fluency over the mesh triangles for the smooth parts (*i.e.*, 2D error diffusion) and along the backbones of the feature skeleton (*i.e.*, 1D error diffusion).

Before performing error diffusion, the user specifies a number of vertices V to distribute on the original mesh. The integral of the density functions, defined both on smooth parts and on the feature skeleton, represents the amount of density to equally distribute among the V samples. To fit the exact budget we first need to calibrate the sampler.

3.2 Sampling Calibration

Recall that we have a density function d_s and d_f over the surface and the features, respectively, which specify what would be the local ideal distribution of the samples. By summing the density function over the surface and the features, we obtain some quantities with different unit. This requires to define two distinct sampling rates: R_s, expressed in #samples per unit amount of surface density and R_f, in #samples per unit amount of feature density. For uniform sample distribution, the density functions d_s and d_f simplify to equally constants. Uniform sampling means that every surface sample ideally

covers the same amount of area and that every feature sample covers the same length of feature curve. Since we seek isotropic sampling, the triangles can be considered as being ideally equilateral, allowing to deduce the local ideal edge length once each triangle area is known, and therefore link the sampling rates R_s and R_f.

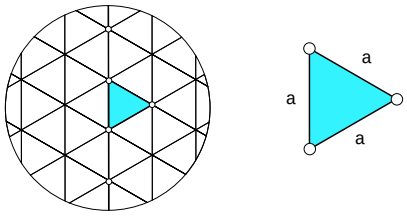

Fig. 4. Uniform plane tiling with triangles.

Let us consider a perfect uniform isotropic tiling of the plane with a triangle mesh (Figure 4). By Euler's formula, the number of triangles T per area unit is equal to twice the number of samples per area unit, namely $T = 2 \cdot R_s$. The ideal area of every triangle is therefore equal to $A = 1/(2R_s)$. From geometry of equilateral triangle we know:

$$A = \frac{a^2}{4}\sqrt{3}$$

where a is the triangle edge length. From the relation $R_f = 1/a$ we deduce:

$$R_s = 2\frac{{R_f}^2}{\sqrt{3}}.$$

We now write the repartition of resources between smooth parts, features and corners to be preserved:

$$R_s \cdot \int_{surface} d_s(u,v)\,du\,dv + R_f \cdot \int_{features} d_f(u)\,du + C = V \qquad (1)$$

where C denotes the total number of corners and V denotes the global vertex budget. Practically, the density functions d_s and d_f are linearly interpolated over the mesh triangles and the feature edges. The number of corners being known, as well as the triangle areas and the feature edges lengths, we deduce R_s and R_f required for the uniform sampling process. The same formula is performed for the non-uniform case after suited conversion.

3.3 Diffusion over Triangles

We replace the pixels of a standard error diffusion technique by the mesh triangles, and organize upon them a fluency that goes through their edges (see

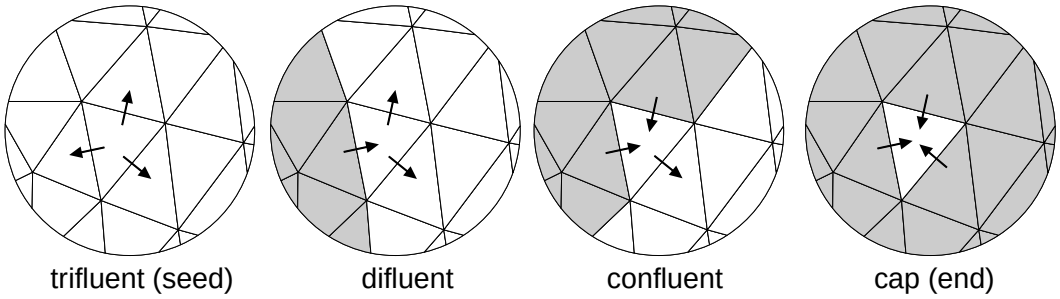

Fig. 5. A fluency is organized over the mesh triangles. Each triangle is classified according to its number of input and output edges during the error diffusion process.

Figure 5). The next task is to find a processing path or tree over the triangles so that any residual error teleport is minimized, and to build a set of distribution coefficients.

For a given processing path, the diffusion algorithm then proceeds as follows for every current face f:

(1) read the total amount of density on f;
(2) deduce the number of samples to distribute on the current face. In the uniform case they are randomly sampled in the current triangle using Turk's technique [57], combined with a low-discrepancy Sobol sequence for efficiency. The non uniform case is correctly handled using an adaptation of Rocchini *et al.* technique [52] after suited discretization of the density function. This number is rounded to the nearest integer value, such a rounding generating a signed quantization error e, translated in amount of density, to diffuse on incident unprocessed faces;
(3) the error e is diffused to the unprocessed faces incident to f through the corresponding edges and proportionally to their length. This heuristic mimics the notion of "geometric aperture" for diffusion;
(4) flag f as processed;
(5) pick the next face to proceed and restart from step 1 if any, stop otherwise.

Finding a processing path is at the heart of any error diffusion sampling algorithm. We now describe three different approaches depending on the genus and number of boundaries of the original triangle mesh to be sampled.

3.3.1 Region Growing Diffusion

This first approach is suited to genus 0 models with at most one boundary. The algorithm picks an arbitrary seed face (on the boundary if there is) and processes by extending a region triangle by triangle without generating any split on its edge boundary. The unprocessed region being topologically equivalent to a disk, it is considered as a triangulated, simple polygon. If this polygon

contains no interior vertex, then it has at least two so-called ear-triangles [45] that we used to expand the processed region. If this polygon contains some interior vertices, then we search for ear-triangles and use one of those if exist. If there is no, we deduce that there exists at least one triangle with exactly one edge incident to the boundary and one interior vertex, used to expand the processed region.

3.3.2 Single-Strip Diffusion

This second approach is suited to arbitrary models with at most one boundary. In a recent work, Eppstein and Gopi [19] describe a way to efficiently compute a strip of an arbitrary triangulated surface without boundary. In other words, the surface is cut along some of its edges such that the resulting mesh is a strip. In the dual graph of the surface, this corresponds to a path which covers all triangles exactly once. Such a path does not exist in all cases; therefore, some triangles have to be split into two triangles. Eppstein and Gopi prove, however, that the increase in the number of triangles in their algorithm is at most 50%, and, in practice less than 2%. The algorithm relies on finding cycles in the dual graph, where each vertex has degree three, using first a perfect matching on this dual graph. It has a running time of $O(n \log^3 n \log\log n)$, where n is the number of triangles.

Although more involved, this elegant result can be directly applied to our problem: the error is simply diffused to the next triangle on the strip. Hence, the fluency starts at the first triangle of the strip, diffuses the error over the strip, and ends on the last triangle with zero quantization error. The splitting of triangles in two halves does not harm: the ideal number of samples to distribute in the two subtriangles is defined to be half of the ideal number of samples in the original triangle. This technique has the interesting property that each triangle receives, by diffusion, a quantization error from (at most) one triangle.

3.3.3 Spanning-Tree Diffusion

This third approach is suited to arbitrary surface with or without boundary. By dualization, we look at the triangles as the vertices of a graph. First, we compute a spanning tree of this connected graph, rooted at an arbitrary triangle. Then we sample the triangles by diffusing the errors from the end leafs to the root. More precisely, assume that each unprocessed triangle t has received a signed error $e(t)$ in the previous part of the algorithm; to sample a given triangle t:

- recursively sample the children of t;
- compute the ideal (float) number of samples to distribute on t, and round it.

The ideal number of samples is computed by summing up the ideal density of t and the result of up to two errors distributed upon it at distinct moments in time during the error diffusion process;
- transmit the excess or loss of samples obtained with the rounding to the father f of t, by increasing or decreasing $e(f)$ accordingly.

We run this process from the root to the leafs of the tree. All unprocessed triangles will be sampled recursively; at the end, the root is sampled and no error has been teleported nor dropped, matching thus the exact vertex budget. Once the smooth parts are sampled, the algorithm moves to the feature skeleton for sampling along its backbones.

3.4 Diffusion over Feature Edges

As for the triangles of surface parts, we design an error diffusion algorithm that distributes some samples along every backbone, the processing path being deduced from the ordering over its halfedges [36]. The diffusion process starts by picking the first edge of a feature backbone and does the following for every current edge:

(1) read the total amount of density on the edge;
(2) from the sampling rate described in Section 3.2, deduce the number of samples to distribute on the current edge. This number is rounded to the nearest integer value, such a rounding generating a signed quantization error e, translated in amount of density, to diffuse on the next unprocessed edge sitting on the current backbone. If the current sampled edge is the last of the processed backbone then we count the number of unprocessed feature edges connected to its end vertex (the latter may be a corner for an open backbone), and distribute the error to each of them with equal repartition. If there is no unprocessed edge connected to the end vertex, we teleport the error to the first edge of the next backbone picked for sampling;
(3) pick the next edge in the current backbone to proceed if any, else move to the next unprocessed backbone while giving priority to any of a backbone connected to the end vertex of the current edge. When possible, the latter heuristic reduces the error teleport;
(4) restart from step 1 if next feature edge exists.

The feature skeleton being possibly not connected, the error diffusion algorithm jumps from one feature backbone to the other and teleports the very last error on another. In a sense this is not consistent with the first goal of diffusing only on nearby areas, yet such a teleport allows to match the exact vertex budget and provides a way to repartition the samples between fea-

tures, corners and surface parts of the original model to remesh since no error is dropped during the process.

Notice that such a diffusion technique offers one direct way to interleave *boundary filtering* with the sampling process. If one considers a smooth model with B boundaries, the feature skeleton is composed of B closed backbones. Each backbone corresponds to a certain amount of density, *i.e.*, a floating point number of samples once converted by the calibration process. After complete diffusion over the backbone, the total number of vertices sampled may be lower than 3, which leads to a closure of the boundary. In other terms, reducing the vertex budget may *filter* some boundaries in a consistent manner with respect to the sampling process. The latter is an appealing feature for applications, similar in spirit to topology filtering as described in [30].

3.5 Discussion

The samples are sitting on the triangles of the original mesh for the smooth parts of the surface, and on the edges of the feature skeleton. The present generalization of direct error diffusion over a triangle mesh does not provide a sampling with a blue-noise profile spectrum such as *e.g.* [47]. This is due to the uniform sampling performed inside each triangle with no dependent probability nor any other sophistication (*i.e.*, white-noise profile), and to the heuristic chosen for distributing the error through the edges. Nevertheless, and thanks to the core principle of error diffusion the resulting sampling is consistent with respect to the specified density function since any error teleport is minimized. The error diffusion thus provides a simple way to *distribute* the samples between boundary loops, features, and connected components of smooth areas. It also serves as the initial guess for building a weighted centroidal Voronoi diagram, which is the mechanism we use to *place* the samples.

4 Centroidal Voronoi Diagram

Before improving the sampling by building a (weighted) centroidal Voronoi diagram, we first need to *mesh* the samples so that it generates an oriented manifold. The meshing is performed by parameterizing the original mesh onto a planar domain (possibly with holes), then applying 2D constrained Delaunay triangulation over the newly sampled vertices. If the mesh is closed or of genus > 0, such a correspondence between the surface and a planar domain does not exist, and we need to cut the surface first.

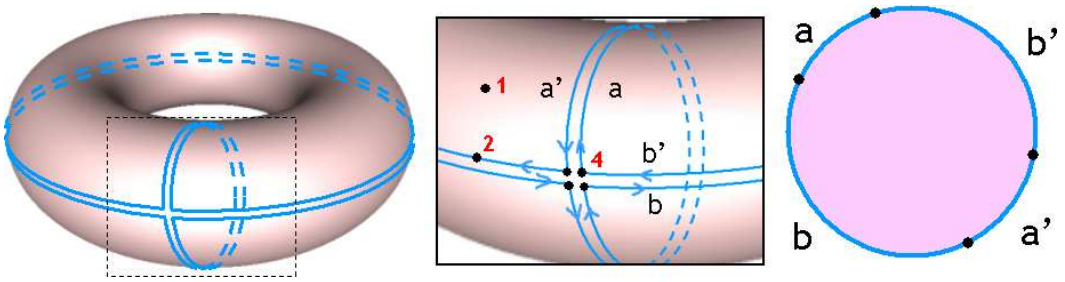

Fig. 6. Left: the cut graph of the torus is composed of two pairs of twin backbones $\{a; a'\}$ and $\{b; b'\}$, intentionally separated for better visualization purpose. Middle: closeup on the branching node of the cut graph. Vertices are labeled according to their multiplicity. Right: the torus has been parameterized on a disk-like domain, 4 instances of the branching node have been generated and will later be merged during the stitching.

4.1 Cutting

We discuss below several ways to cut the original surface into a genus 0 surface. Note that, although the cutting is necessary only at this point of the remeshing process, the edges which have to be cut are computed in the *first* stage of the algorithm, even before the error diffusion process, since they have to be marked as feature edges as described in Section 2. Since all cut edges are considered as features and need to be sampled separately, like the sharp and boundary edges, our goal is to minimize the total length of the cut and to snap corresponding cut graph onto the feature skeleton as much as possible. We first describe our current implementation, then explain how recent theoretical results may help us to improve our cutting algorithm.

4.1.1 Our Current Implementation

In [27], the authors use a simple algorithm to cut the surface into a genus zero surface; after that, they describe a way to improve the cut graph. We implemented a retraction technique similar in spirit to the first part of their method. Note that cutting a mesh along a seaming backbone generates two instances of it in parametric space and one or more instances of the branching nodes encountered in the cut graph (Figure 6). To guarantee a perfect stitching, we temporary split the feature or branching samples into a number of samples equivalent to their multiplicity, during the error diffusion process *and* the univariate Lloyd relaxation along the seaming backbones.

We note that other approaches have been considered to compute a topological decomposition of surfaces with handles. In the special case of the torus embedded in the standard way in $\mathbb{R}^3$ (*i.e.*, without knot), an algorithm is given [24] to compute "longitudinal" and "meridianal" loops splitting the sur-

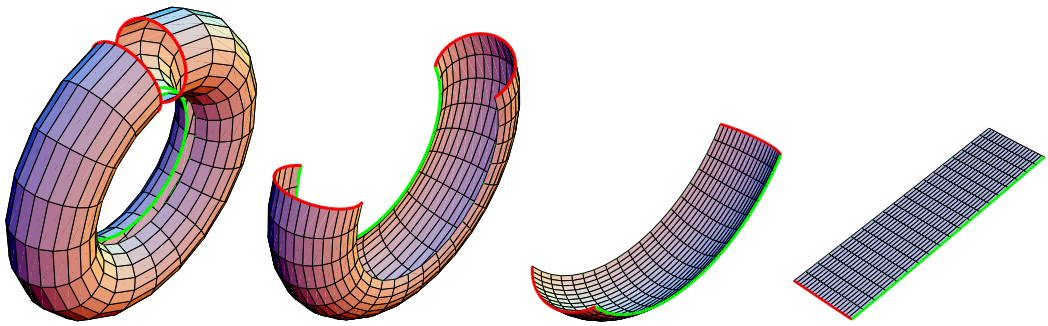

Fig. 7. From an orientable surface to a polygonal schema: the torus.

face of the torus into a disk. When dealing with a surface of this special type, the resulting loops can indeed be used for our purposes.

In the more general setting of volumic objects homeomorphic to the three-dimensional ball with solid handles, there is an algorithm [35] to cut the (volumic) object, obtaining thus a topological ball. This provides a nice way to parameterize solids over a domain of $\mathbb{R}^3$, and in particular decomposes the *surface* of the solid into a genus zero surface with boundaries, which can further be split easily into a topological disk, thus achieving our goal.

All these results can be used to parameterize our mesh into a planar domain. But no control is possible on the quality of the cut (which should be as short as possible, and should match the feature skeleton whenever possible). The next section aims at explaining how some recent techniques enable to cut the surface with theoretical guarantees.

4.1.2 Promising Approaches

Some recent work in Computational Geometry express interest in the problem of cutting a surface along a set of curves to obtain a topological disk. The latter is called a *polygonal schema*, because the edges on its boundary can be pairwise identified to re-obtain the surface; see Figure 7. Such a polygonal schema is *reduced* if the curves are simple loops, meeting at a common vertex v_0, and pairwise disjoint except at this vertex. Note that, for our cutting stage, we can simulate that we work on a surface without boundaries: fill its holes, cut this surface (without entering the filled holes) with any cutting algorithm, then re-open the holes.

Erickson and Har-Peled [20] have focused on finding the shortest polygonal schema of a surface (possibly non-oriented and with boundary). They have proved that this problem is NP-hard, yet they describe a greedy algorithm that outputs a $O(\log^2 g)$-approximation of the minimum schema. To our knowledge, no experiment has been done to see whether it yields a visually acceptable result, and it would be worth implementing it (though it is not so easy) for

our purposes. Because it is desirable to cut the surface along sharp edges, we advise to simulate that these edges are considered shorter.

In contrast to the NP-hardness of the above problem, Colin de Verdière and Lazarus [13] give a polynomial algorithm which computes the shortest reduced polygonal schema homotopic to a given reduced polygonal schema (computed *e.g.* with the help of [39]). The algorithm described in [13] consists of iterating *elementary optimization steps*, each of them optimizing the current schema by shortening one loop while maintaining the other ones fixed; stability is reached after a finite number of steps. The loops drawn on the edges of the manifold can partly overlap, yet they are considered as disjoint if we imagine that they are spread apart with a thin space. We believe this algorithm is better suited to our purposes. In particular, in our context, it may be desirable to further generate overlappings of loops on edges, because this decreases the number of distinct cut edges. This can be easily achieved by artificially decreasing the lengths of the edges that contain loops at elementary steps. It remains to describe how we deal with multiple loops on an edge during the parameterization and sampling processes. If several loops go along a given edge, this creates strips of infinitely small area, and no sample should go inside them. In order to achieve this goal, these strips are given a density of zero in parameter space. Moreover, in this case, a given edge of the original mesh can correspond to an arbitrary large (even) number of boundary edges in parameter space, all these edges being sampled consistently before stitching.

4.2 Parameterization

The goal of surface parameterization is to remove the embedding by flattening the original mesh on a plane. Our goal being to build an isotropic sampling technique, we advocate for using a conformal parameterization [49,18] with free boundaries since it is known to be both *angle-preserving* and *locally isotropic* (see [41,15] for more details on the technique). Those two properties are of crucial importance for the meshing and optimization stages since the latter operations are performed only from parameter space. This way every decision on parameter space makes sense with respect to the original mesh, at least for the two preserved properties. Intuitively, locally isotropic and angle-preserving means that a small circle mapped on the surface will be transformed into a circle in parameter space (see Figure 8). Hence a well-shaped triangle in parameter space will not be deformed too much once lifted back into $\mathbb{R}^3$, except for its size. One distortion remains: the area-stretching. This one being isotropic, it can easily be compensated by modifying the density function expressed in parameter space for the optimization stage, as explained in Section 4.4, contrary to a shearing deformation that would be much more difficult

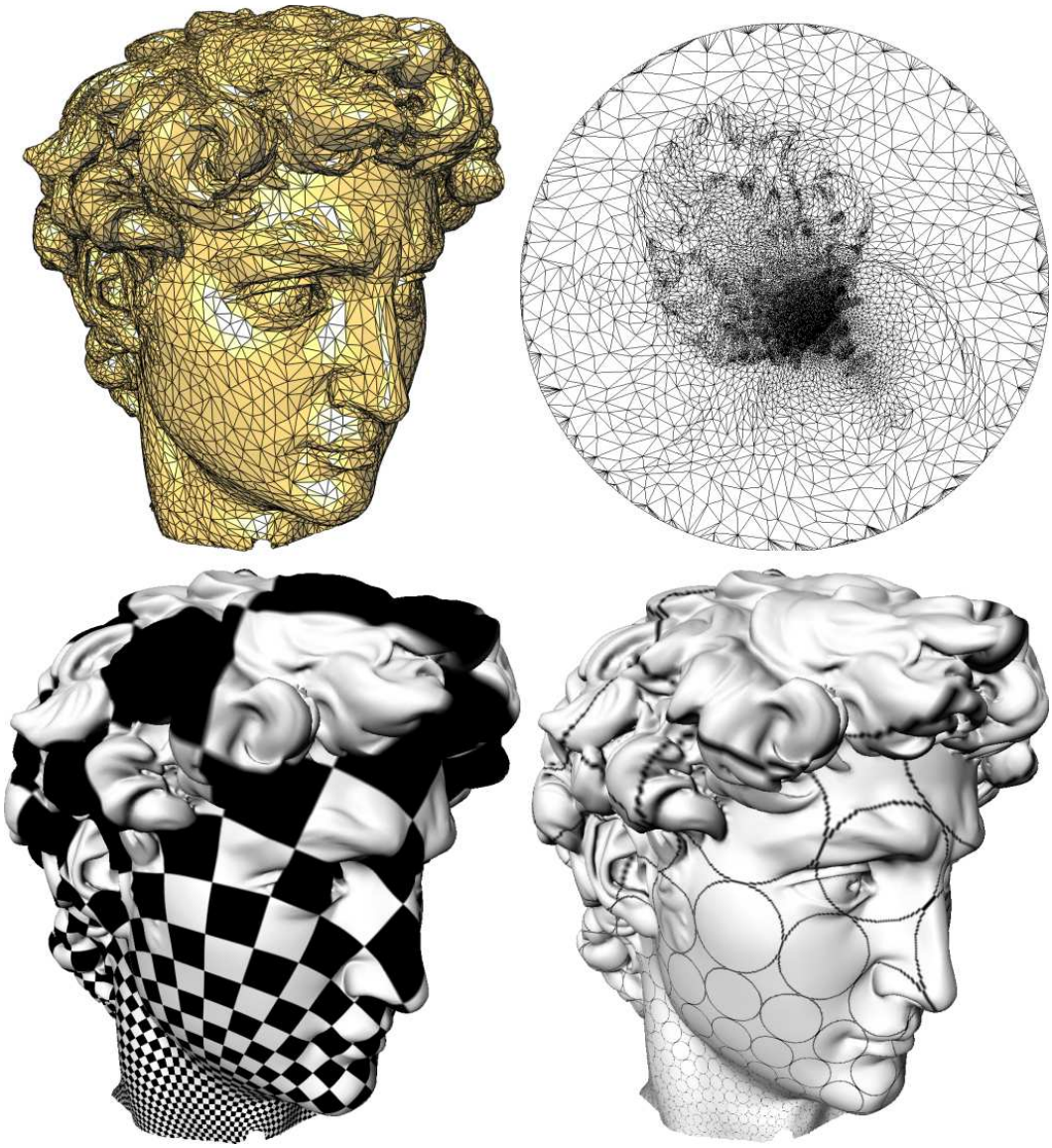

Fig. 8. Conformal parameterization of the *David* head model over a disk-like domain. The two textures mapped onto the model illustrates the two main properties of conformal parameterization: preservation of angles and local isotropy.

to compensate for.

4.3 Meshing

Every sample now lives in parameter space, be it on a corner, on a feature edge or on a face of the original mesh. We perform a 2D constrained Delaunay triangulation [21] in parameter space so that every segment joining two consecutive vertices sampled along a feature backbone is added to the list of constrained edges (see Figure 9).

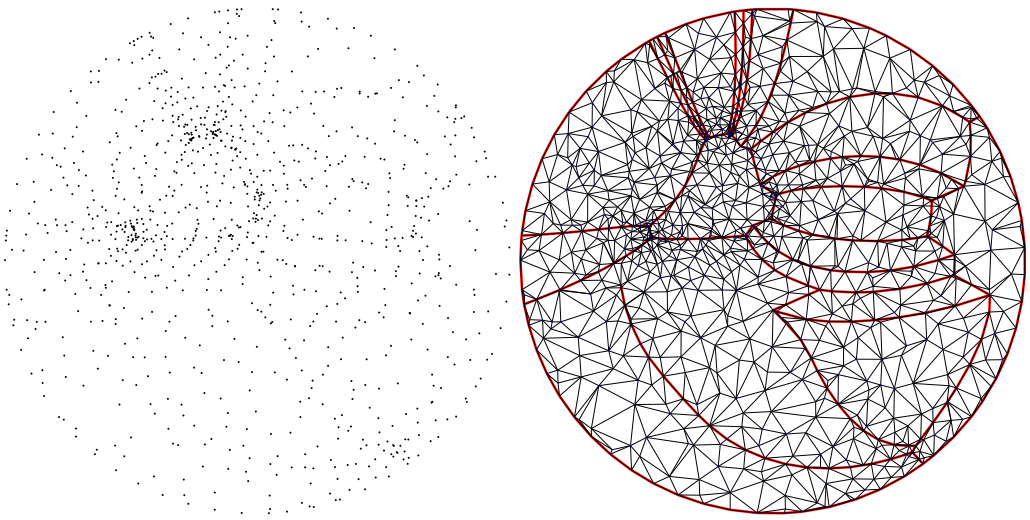

Fig. 9. Constrained Delaunay triangulation.

4.4 Construction

With a Delaunay triangulation, we can already deduce an initial Voronoi Diagram [6]. From this, we aim at building a *weighted centroidal* Voronoi diagram [17] to improve the initial sampling obtained by error diffusion. A weighted centroidal Voronoi diagram is a Voronoi diagram such that the associated sites coincide with the center of mass of the corresponding Voronoi cells. In the current state of the algorithm, we know the density function and the position of the samples in parameter space. The goal is to determine the final locations of the samples so that they coincide with the centroids of the corresponding Voronoi regions; the centroids have to be evaluated over the density function expressed in *parameter space* rather than on the surface.

Similarly in spirit to [1], we compute a stretching factor for every vertex as a weighted averaging over the stretching factors of its incident faces. The weight being equal to the area of every face measured in parameter space, the stretching factor s of a vertex v is equal to: $s(v) = \sum area(f_i) / \sum area_{uv}(f_i)$, where the sums iterate over its incident faces and any subscript uv denotes a quantity measured in parameter space. The corrected density expressed on a vertex in parameter space is now the product of its stretching factor by its density expressed on the surface: $d_{uv}(v) = s(v) \cdot d(v)$. Such a density function is linearly interpolated over the vertices in parameter space and used for weighting the centroidal Voronoi diagram.

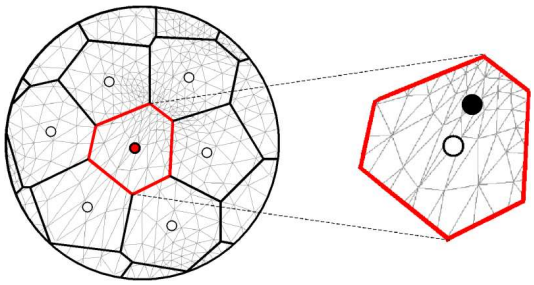

Fig. 10. Left: the Voronoi cell surrounding a sample overlaps the original mesh. Right: closeup on the intersection computed between a Voronoi cell and the original mesh. The new centroid (filled) of the cell is computed as a weighted contribution of the centroids computed on the linearly interpolated polygons.

4.4.1 2D Lloyd Relaxation

One way to build such a diagram is to use Lloyd's relaxation method [17]. The Lloyd algorithm is a deterministic, fixed point iteration [44]. Given a density function and an initial set of n sites, it consists of the following steps:

(1) build the Voronoi diagram corresponding to the n sites;
(2) compute the centroids of the n Voronoi regions with respect to the density function expressed in parameter space, and move the n sites onto their respective centroid;
(3) repeat steps 1 and 2 until satisfactory convergence is achieved.

Computing the centroid (*i.e.*, center of mass) of every cell is the most delicate part of the algorithm. Let us recall that the original mesh is now parameterized and that a density function lives in parameter space. The new mesh has been created using constrained Delaunay triangulation over the new samples obtained by error diffusion. The Voronoi regions are computed over the new samples and we aim at computing their centroids evaluated over the density function sitting over the *original* mesh. This requires to compute the intersection between each cell and the set of overlapped triangles, then compute the center of mass of the set of resulting polygons on which the corrected density function is piecewise linear (see Figure 10). Figure 11 illustrates several iterations of the Lloyd algorithm over the *mushroom* model uniformly sampled with 3k vertices. The variable density compensates for any area distortion due to parameterization, and the final tiling tends to make each cell cover the same amount of density. Notice how the main improvement is already achieved after 10 iterations of Lloyd algorithm, the sampling being then polished with additional iterations. As described in Section 6, the temporal coherence of Lloyd's algorithm is exploited for speed improvement.

Clipping the Voronoi cells with constrained edges [43] allows to disconnect two smooth regions separated by a backbone during the computation of the centroid. This leads to a nice quality of the sampling in the vicinity of the features.

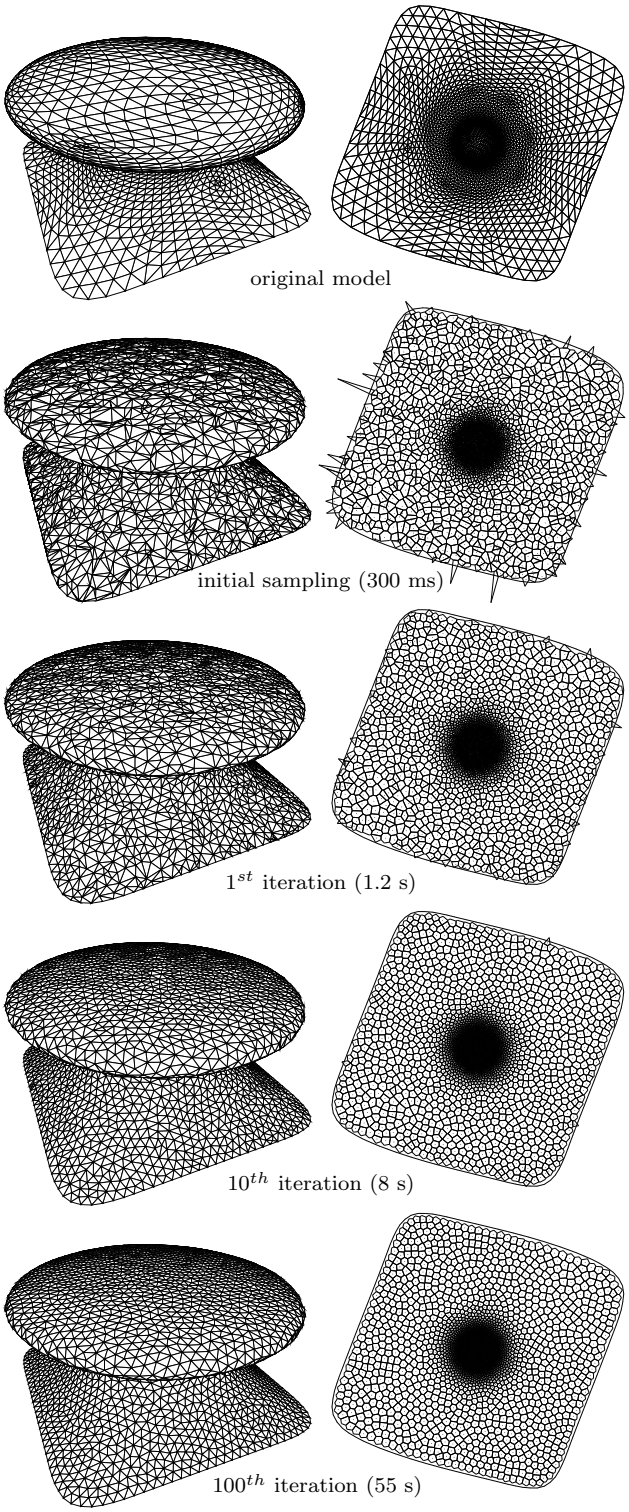

Fig. 11. Top row: original *mushroom* model to resample and its boundary-free parameterization on a disk-like domain. Row 2: initial uniform sampling of 3k vertices generated by direct error diffusion (300 ms). Row 3: first iteration of Lloyd algorithm with a variable density coming from the area distortion due to parameterization (1.2 s). Row 4: 10^{th} iteration of Lloyd algorithm (8 s). Row 5: 100^{th} iteration of Lloyd algorithm (55 s).

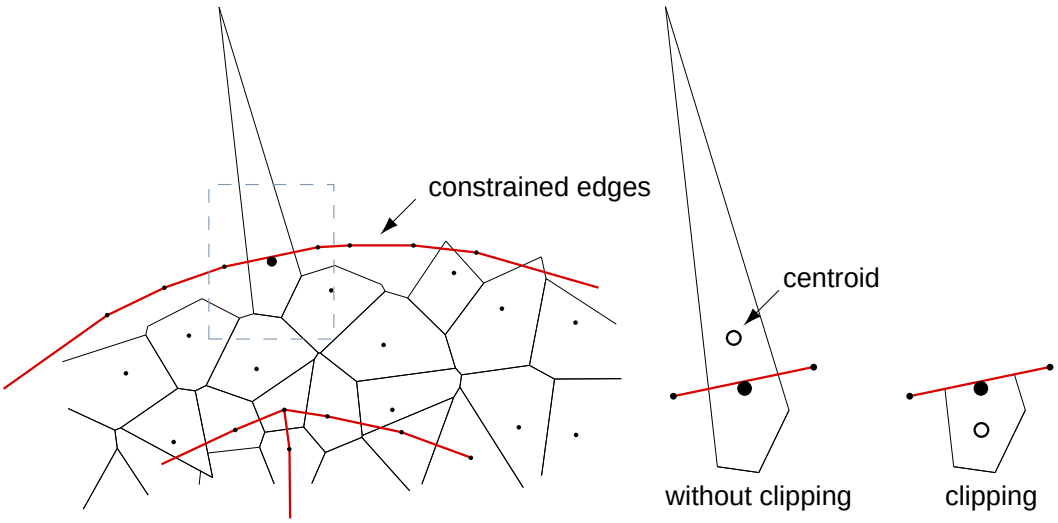

Fig. 12. Left: a Voronoi diagram in parameter space with a feature skeleton. All the cells are drawn according to the circumcircle property. Computing the centroid without clipping by the constraints makes the sampling inconsistent, while the effect of clipping is to repulse the samples from the boundary or sharp edges, the centroid being computed on the truncated cell. A constrained edge separating two samples thus acts as a barrier [43] annihilating their mutual influence as expected.

The latter is obtained through a non-symmetric behavior of the algorithm in the sense that the boundary and the feature backbones influence the surface samples but the inverse is not true. Intuitively, two samples incident in the Voronoi diagram but separated by a feature are not influencing each other anymore, and the samples closed to a boundary or from a feature backbone are rather repulsed by the constraints (see Figure 12). Indeed, clipping a cell by the set of constraints may remove some regions from the computation of the centroid, making the Lloyd relaxation consistent with respect to the constraints.

4.4.2 1D Lloyd Relaxation

The goal is to distribute on each feature backbone a number of samples so that each of them covers the same amount of density. To proceed, we parameterize each backbone independently on a segment without any length distortion, then apply univariate Lloyd relaxation on the density function sitting on every parameterized feature backbone. A special care is taken for seaming backbones, the twin samples being reflected on opposite halfedges to guarantee a perfect stitching during the lifting. For the sake of consistency with the asymmetric influence between feature and surface samples, the univariate relaxation described here is applied first, then the feature samples are not moved any more during the 2D Lloyd relaxation process previously described.

4.5 Lifting and Stitching

The lifting stage restores back the embedding by locating every vertex in its associated triangle in parameter space and computing its barycentric coordinates. We then project back every ordinary sample onto its corresponding triangle in $\mathbb{R}^3$ and every feature sample onto its corresponding feature edge or corner. For closed or genus > 0 objects, a stitching stage is required. It amounts to merging each set of twin vertices with a multiplicity > 1 generated for meshing in parameter space. This welds the model and wraps up the complete remeshing pipeline. For the sake of clarity we postponed the implementation details to Section 6.

5 Results

Figure 13 and 14 illustrate a uniform remeshing of the Michelangelo David head model. The model considered for remeshing has a complexity of 25k vertices, and the new remeshed model has 50k vertices. After 100 iterations of Lloyd relaxation performed in parameter space, the weighted centroidal Voronoi diagram provides a satisfactory uniform sampling after lifting. Running times are the following on a PIII 1GHz: 7 s for the mesh parameterization, 400 ms for differential geometry analysis, 850 ms for sampler calibration, 2.8 s for error diffusion, and 26 minutes for polishing the sampling using 100 iterations of Lloyd algorithm. Note that starting from iteration 20 the result is already very similar to the present one. As shown by Figure 11, our technique is in a sense progressive in terms of sampling quality since the Lloyd algorithm progressively improves the repartition of the density function among all Voronoi cells. Figure 15 illustrates three curvature-adapted remeshings of the David head using 30k vertices. The curvature-adapted behavior is obtained by specifying a variable $\gamma > 0$ to the transfer function x^γ applied over the initial density function, related to absolute mean discrete curvature. Increasing γ from 0 (Figure 14) to 2.5 on our examples allows us to vary continuously from uniform to curvated-adapted remeshing, while matching the exact desired vertex budget.

Figure 16 illustrates uniform remeshing of the bottomless *fandisk* model with 10k vertices. The Lloyd algorithm optimizes the sample positions both along backbones (univariate relaxation) and on the surface. The repartition of samples between corners, backbones and smooth parts exactly matches the vertex budget thanks to the sampler calibration described in Section 3.2. A few close-ups nearby the features show the consistent behavior due to cell clipping with the constrained edges.

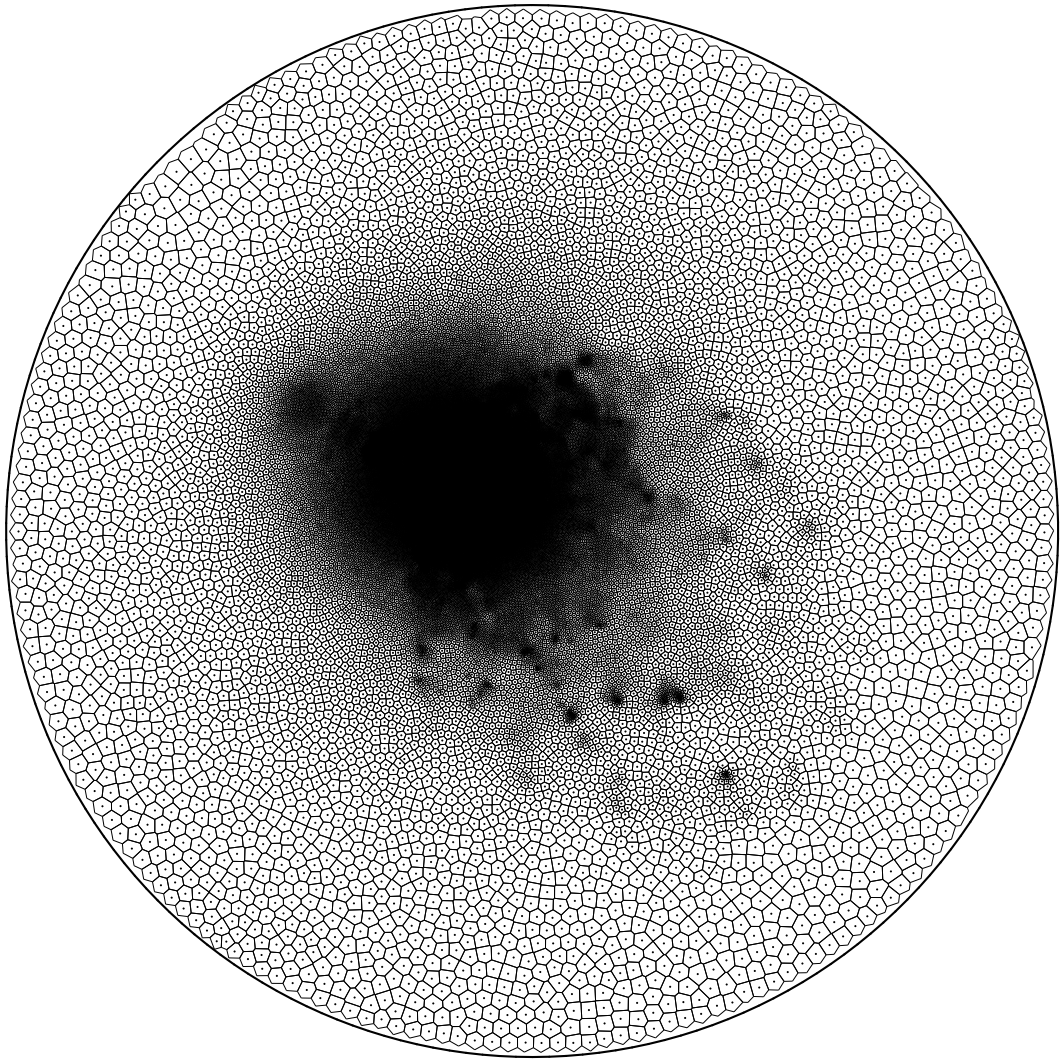

Fig. 13. Weighted centroidal Voronoi diagram built with 100 iterations of Lloyd relaxation.

Figure 17 illustrates uniform remeshing of the genus-1 *rotor* model with 10k vertices. The initial mesh is cut and parameterized on a disk-like domain. The cut graph (highlighted) is composed of several seaming backbones on which curve sampling technique is applied, as done for ordinary features. Although the pairing of seaming backbones and samples allow a perfect stitching of the remeshed model, we believe that the seams generated by univariate sampling of the cut graph limit the proposed remeshing technique to the class of low-genus surfaces.

Figure 18 illustrates uniform remeshing of the *pig* model with 15k vertices. This model has 7 boundaries and generates an extreme range of area distortion in parameter space. Notice that the longest boundary has been automatically chosen for parameterization (foot boundary is highlighted). Due to cell clipping using constrained edges, the Lloyd relaxation is guaranteed to correctly

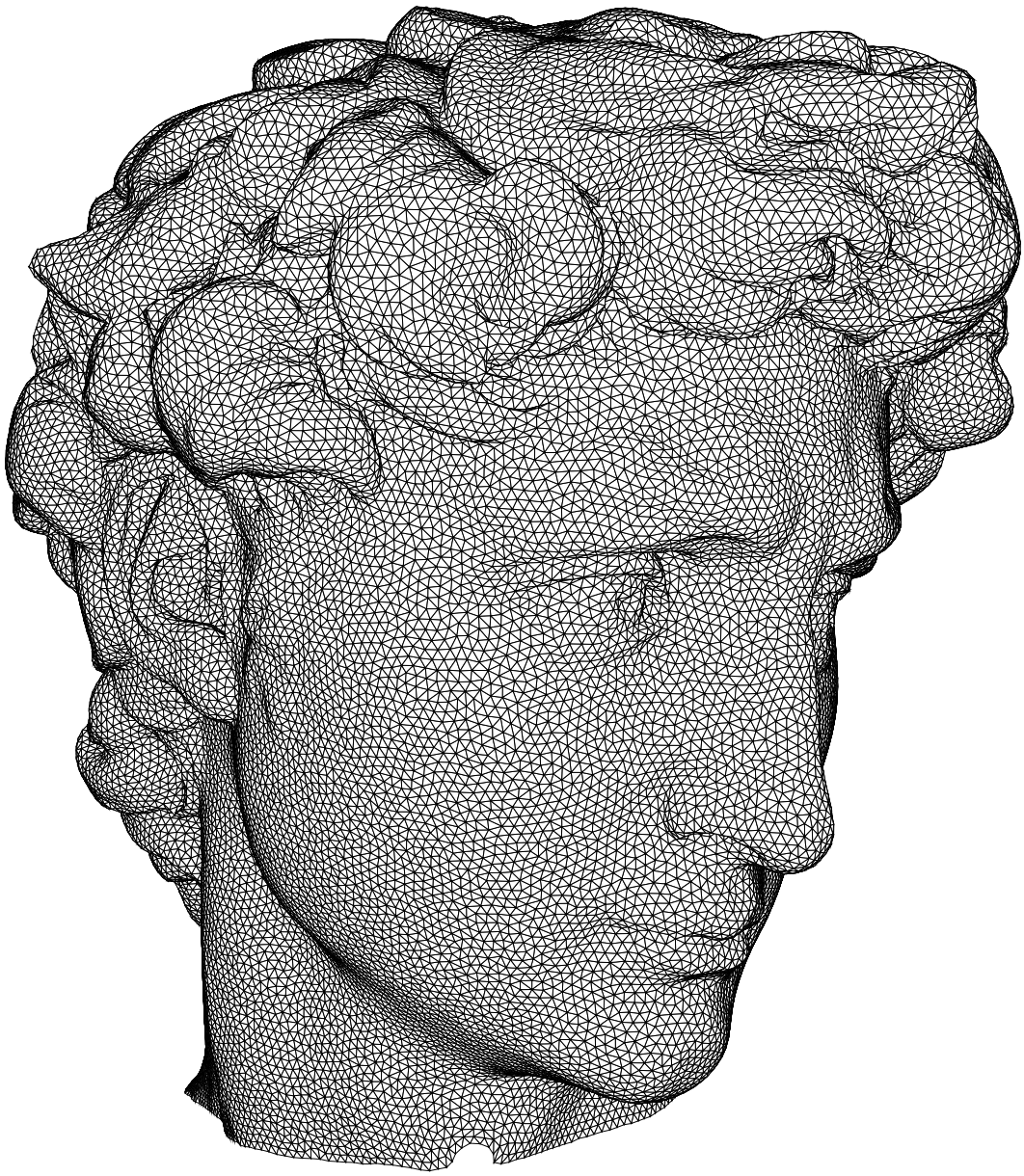

Fig. 14. Uniform remeshing with 50k vertices.

handle models with several boundaries. The density function computed in parameter space correctly compensates the area distortion due to flattening. Figure 19 illustrates four curvature-adapted remeshings of the *pig* model with exactly 30k vertices and $\gamma = 3$ set in the transfer function. The only difference between the models lies into the mesh gradation progressively smoothed with respectively 0, 3, 10, and 100 iterations of low-pass filtering over the density function. A few closeups of the ear shows the effect of smoothing the density function over the final mesh gradation. This result illustrates the key concept demonstrated in this paper, *i.e.*, the user can act on some parameters until the density function fits his desire in terms of sampling distribution and gradation on the surface and the features.

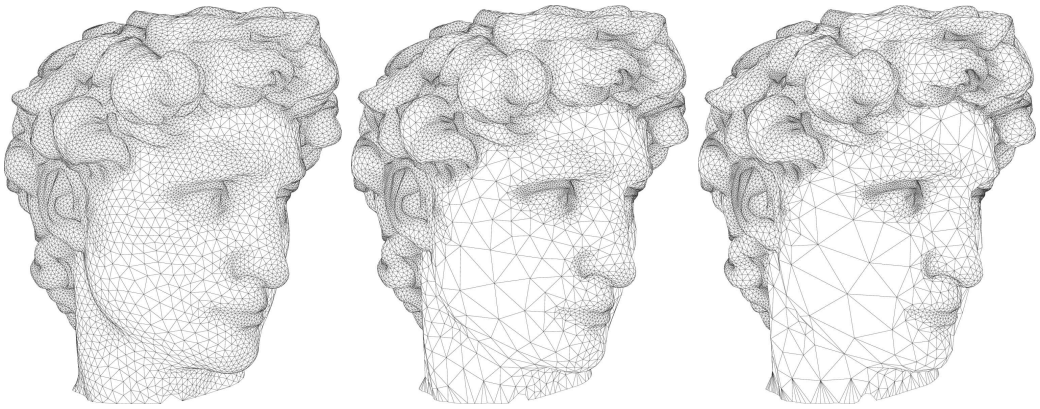

Fig. 15. Curvature-adapted remeshing using 30k vertices with $\gamma = 1$, $\gamma = 2$ and $\gamma = 2.5$ respectively.

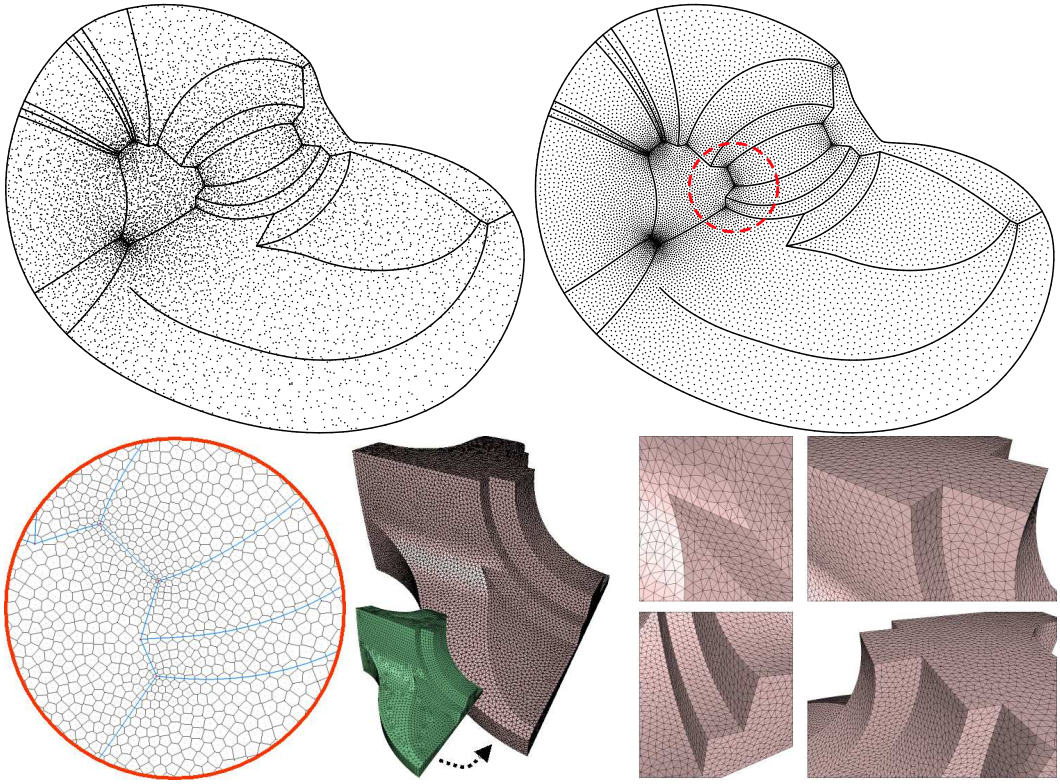

Fig. 16. Uniform remeshing of the fandisk model with 10k vertices. Top row: Initial sampling computed by direct error diffusion, and sampling obtained after 20 iterations of Lloyd relaxation. Bottom row: closeup over the Voronoi diagram after Lloyd convergence, global view of the remeshed model, and several closeups nearby the features.

6 Implementation Details

Special care has been taken to obtain a robust triangulation algorithm along with an efficient implementation of Lloyd's algorithm. This requires a few

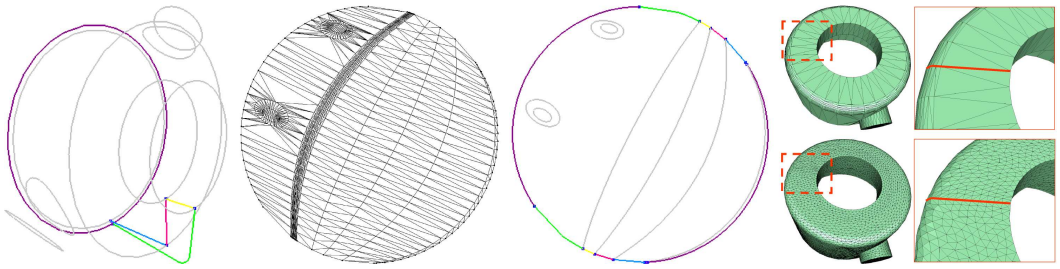

Fig. 17. Uniform remeshing of the genus-1 rotor model with 10k vertices. From left to the right: cut graph (colored) and feature skeleton (light grey); mesh parameterization on a disk-like domain, and associated parameterized feature skeleton (boundary backbones with identical colors are associated pairwise). Far right, top to bottom: original, and remeshed model. Two closeups over the cut graph illustrate the univariate sampling performed along seaming backbones.

implementation details to be described separately for the sake of clarity.

To prevent any robustness issue for meshing, we parameterize the constrained Delaunay triangulation of CGAL [5] with a filtered kernel so that all the predicates use exact arithmetic when required, and ordinary double floating point numbers otherwise.

We wanted the Lloyd algorithm to be output-sensitive, which is not the case if one computes the centroid of every cell from the set of intersected faces of the original model. Therefore we propose the following heuristic for a major speed improvement: the corrected density function is only evaluated *over the new samples* and linearly interpolated on the new triangles in parameter space. Surprisingly, and despite the drastic simplification performed, it turns out to behave extremely well on all our experiments if some special care is taken. In case of re-sampling or over-sampling there is no significant change with the input-sensitive version of the algorithm. In the case of under-sampling there are two main drawbacks: some details in the density function may be missed in-between two samples distributed on the model, and the Lloyd algorithm would badly converge if the density function is varying too rapidly on parameter space, since the corrected density function is reevaluated after each sample displacement due to Lloyd relaxation. To alleviate those drawbacks, we advocate for correlating the level of under-sampling with the low-pass filtering applied over the density function. This has the effect to *diffuse* the details described by the density function, making the influence of a curved detail wider on parameter space and therefore harder to miss. Note that this is in essence similar to some multiresolution methods where it makes no sense to try approximating some very fine details when the mesh complexity (*i.e.*, the vertex budget) is low. Such a simplification requires fast face localization by walking [16] on the original mesh and computation of corrected density from barycentric coordinates. For additional speed improvement, we also ex-

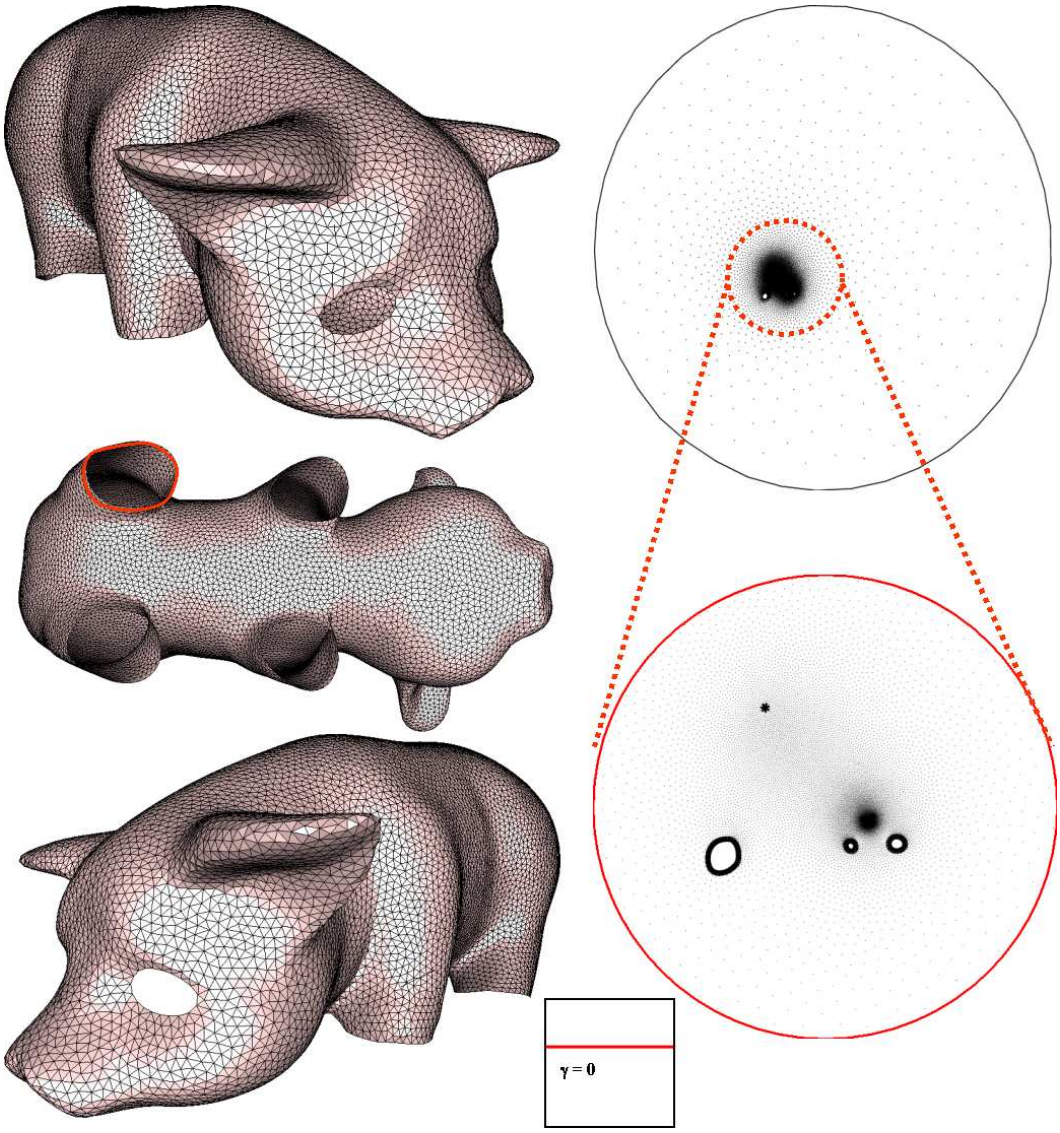

Fig. 18. Uniform remeshing of the pig model with 15k vertices. The model has been parameterized from the longest boundary and undergoes a high range of area distortion due to flattening.

ploit the temporal coherence of the Lloyd algorithm by keeping track of the last localized face as a seed for the localization performed during the next Lloyd iteration. As a last significant speed improvement, we also exploit the fact that the compacity of the cells improves while Lloyd algorithm converges by tagging some samples if all their incident Delaunay edges are so-called "de Gabriel" (see [50], page 255). In the latter case the intersection between the Voronoi cells and the density function reduces to a set of quadrilaterals on which computing the centroid becomes trivial. In our experiments the proportion of tagged samples quickly increases from 6% to more than 80% for meshes with sufficiently smooth gradation. The speed improvements described above play a crucial role for the optimization process, since it makes the algorithm

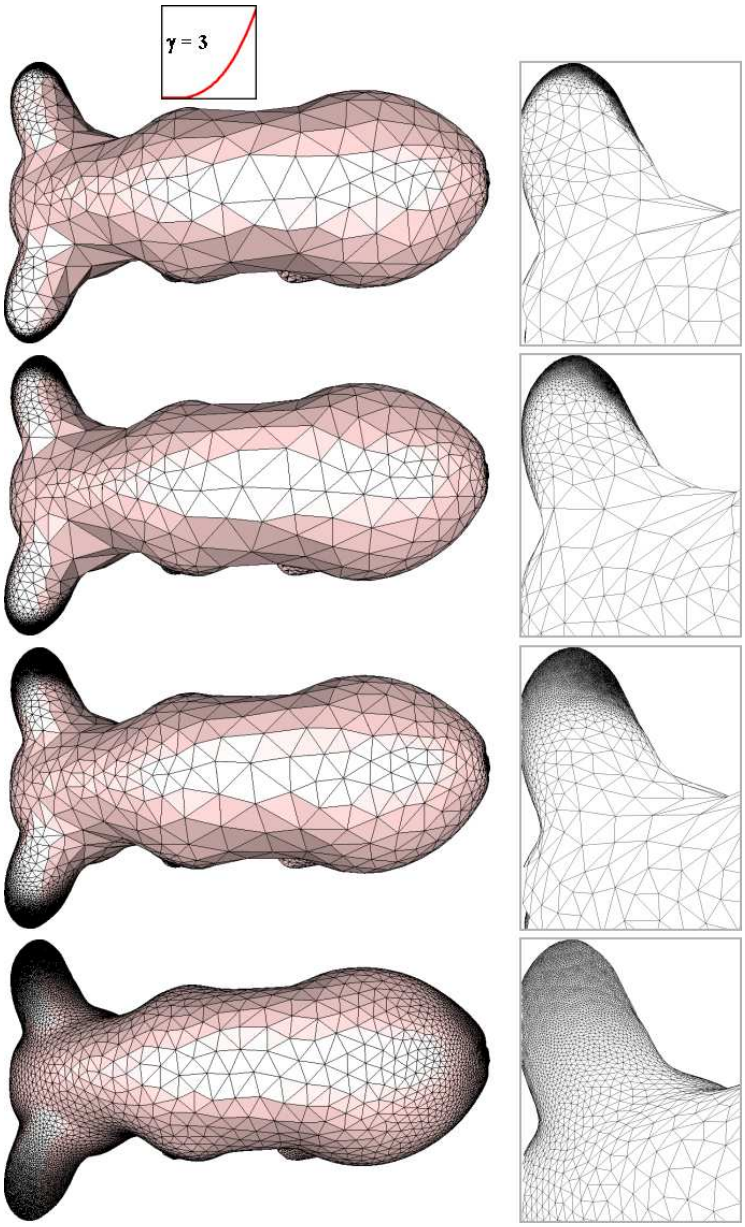

Fig. 19. Bottom: curvature-adapted remeshing with 30k vertices. The gradation is more and more smoothed by specifying 0, 3, 10 and 100 iterations for low-pass filtering the density function.

output-sensitive and globally divides the running time in our experiments by two orders of magnitude.

7 Conclusions and Future Work

We have presented a new technique for isotropic remeshing of triangle meshes. Our approach consists of specifying a density function over the surface to be

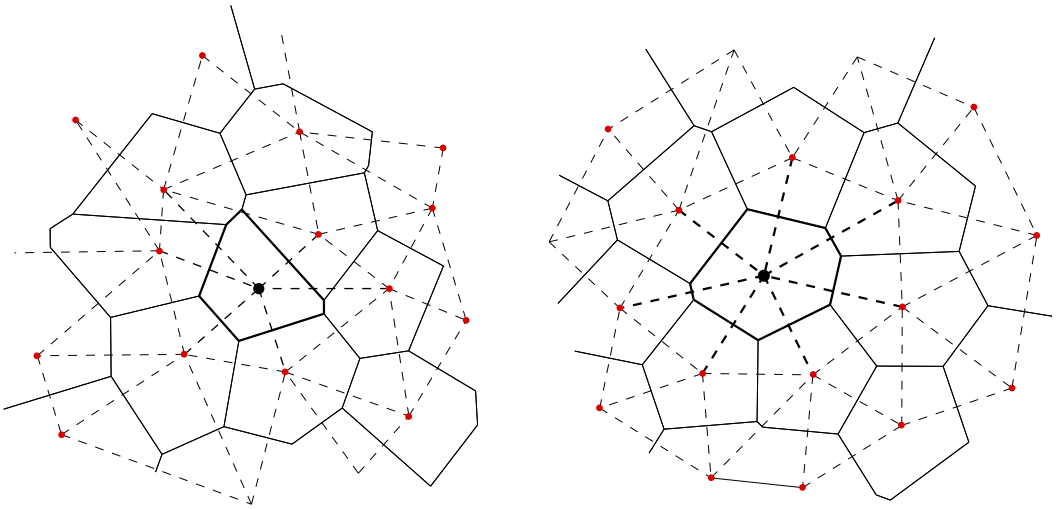

Fig. 20. Left: ordinary cell. Right: all the edges are "de Gabriel", greatly simplifying the computation of the centroid since each intersected polygon is a quadrilateral.

repartitioned between a set of samples so that each of them covers the same amount of density on a neighborhood as compact as possible. This task is performed using two algorithms: the first one performs a fast mesh resampling by generalizing the core principle of error diffusion over the original mesh triangles. The second tool is a Lloyd relaxation used to build a weighted centroidal Voronoi diagram in a conformal parameter space. The latter, commonly used for optimal clustering [17], tends to optimally distribute the density function between all the samples both on the feature skeleton and on smooth parts of the models. The weak part of the algorithm is the need for cutting closed or genus > 0 surfaces. Such a cut generates a seaming backbone on which *curve sampling* is applied instead of *surface sampling*, which is not consistent if the cut graph does not coincide with a set of feature edges. Such an approximation becomes unacceptable for high genus surfaces, which would require a complex cut graph. Some recent works [54] remove the need for global parameterization and hence simply remove the cutting stage from the algorithm.

As future work we plan to investigate the conditions over the density function to guarantee the convergence of Lloyd's algorithm. Being able to remesh volume meshes with error diffusion and 3D centroidal Voronoi diagram is also of some interest.

Acknowledgements

The work reported here was supported in part by the ECG project of the EU No IST-2000-26473, and by an ARC TéléGéo grant from INRIA. Spe-

cial thanks to Mariette Yvinec, Sylvain Pion and Lutz Kettner for help with CGAL. The David's head mesh is courtesy of Marc Levoy and the Digital Michelangelo Project, and Kai Hormann provided us with the Darmstadt model.

References

[1] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. *ACM Transactions on Graphics*, 21(3):347–354, 2002. SIGGRAPH conference proceedings.

[2] M. Bern and D. Eppstein. Mesh Generation and Optimal Triangulation. *Computing in Euclidean Geometry, edited by D.-Z. Du and F. K. Hwang*, pages 23–90, 1992.

[3] Marshall Bern and Paul Plassmann. Mesh generation. In J. Sack and J. Urrutia, editors, *Chapter 6 in Handbook of Comp. Geometry*. Elsevier Science, 1999.

[4] Jean-Daniel Boissonnat and Frédéric Cazals. Coarse-to-fine Surface Simplification with Geometric Guarantees. In *Eurographics proceedings*, pages 490–499, 2001.

[5] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in CGAL. *Comput. Geom. Theory Appl.*, 22:5–19, 2002.

[6] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998.

[7] H. Borouchaki, P. L. George, F. Hecht, P. Laug, and E. Saltel. Delaunay Mesh Generation Governed by Metric Specifications. *Finite Elements in Analysis and Design*, 25:61–83, 1997.

[8] Houman Borouchaki. Geometric Surface Mesh. In *Int. Conf. on Integrated and Manufacturing in Mechanical Engineering*, pages 343–350, 1998.

[9] Houman Borouchaki, Frédéric Hecht, and Pascal J. Frey. Mesh Gradation Control. In *Proceedings of 6th International Meshing Roundtable*, pages 131–141, 1997.

[10] Mario Botsch and Leif Kobbelt. Resampling Feature and Blend Regions in Polygonal Meshes for Surface Anti-Aliasing. In *Eurographics proceedings*, pages 402–410, sep 2001.

[11] Mario Botsch, Christian Rössl, and Leif Kobbelt. Feature Sensitive Sampling for Interactive Remeshing. In *Vision, Modeling and Visualization proceedings*, pages 129–136, 2000.

[12] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 274–280, 1993.

[13] Éric Colin de Verdière and Francis Lazarus. Optimal system of loops on an orientable surface. In *Proceedings of 43rd Annu. IEEE Sympos. Foundations of Computer Science*, November 2002.

[14] H. L. de Cougny and M. S. Shephard. Surface Meshing Using Vertex Insertion. In *Proceedings of 5th International Meshing Roundtable*, pages 243–256, 1996.

[15] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Proceedings of Eurographics*, pages 209–218, 2002.

[16] Olivier Devillers, Sylvain Pion, and Monique Teillaud. Walking in a triangulation. *Internat. J. Found. Comput. Sci.*, 13:181–199, 2002.

[17] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tesselations: Applications and Algorithms. *SIAM review*, 41(4):637–676, 1999.

[18] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In *Proceedings of SIGGRAPH*, pages 173–182, 1995.

[19] D. Eppstein and M. Gopi. Single-strip triangulation of manifolds with arbitrary topology. In *Proc. Eurographics*, 2004. To appear.

[20] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. In *Proceedings of the 18th Annual ACM Symposium on Computational Geometry*, pages 244–253, 2002.

[21] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. On the design of CGAL a computational geometry algorithms library. *Softw. – Pract. Exp.*, 30(11):1167–1202, 2000.

[22] R.W. Floyd and L. Steinberg. An adaptive algorithm for spatial grey scale. In *Proc. of the Society for Information Display*, volume 17, pages 75–77, 1976.

[23] Pascal J. Frey. About Surface Remeshing. In *Proceedings of the 9th Int. Meshing Roundtable*, pages 123–136, 2000.

[24] K. Fujimura. On cutting a torus. In *Proc. Int. Conf. on Shape Modeling and Applications (SMA'99)*, pages 10–14. IEEE Computer Society, 1997.

[25] Michael Garland and Paul Heckbert. Simplifying Surfaces with Color and Texture using Quadric Error Metrics. In *IEEE Visualization Conference Proceedings*, pages 263–269, 1998.

[26] P. L. George and H. Borouchaki, editors. *Delaunay Triangulation and Meshing Application to Finite Elements*. Hermes, Paris, 1998.

[27] X. Gu, S. Gortler, and H. Hoppe. Geometry Images. In *Proceedings of SIGGRAPH*, pages 355–361, 2002.

[28] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution Signal Processing for Meshes. In *Proceedings of SIGGRAPH*, pages 325–334, 1999.

[29] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. Normal Meshes. In *Proceedings of SIGGRAPH*, pages 95–102, 2000.

[30] Igor Guskov and Zoë Wood. Topological noise removal. In *Proceedings, of Graphics Interface*, pages 19–26, 2001.

[31] Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *IEEE Visualization Proceedings*, pages 59–66, 1999.

[32] K. Hormann, U. Labsik, and G. Greiner. Remeshing Triangulated Surfaces with Optimal Parameterizations. *Computer-Aided Design*, 33:779–788, 2001.

[33] A. Hubeli and M. Gross. Multiresolution feature extraction from unstructured meshes. In *IEEE Visualization conf. proceedings*, pages 287–294, 2001.

[34] Pierre-Marc Jodoin and Victor Ostromoukhov. Error-Diffusion with Blue-Noise Properties for Midtones. *SPIE*, 4663:293–301, 2002.

[35] E. Kartasheva. Reduction of h-genus polyhedrons topology. *Int. Journal of Shape Modeling*, 5(2):179–194, 1999.

[36] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Comput. Geom. Theory Appl.*, 13:65–90, 1999.

[37] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive Geometry Compression. In *Proceedings of SIGGRAPH*, pages 271–278, 2000.

[38] L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Computer Graphics Forum, Eurographics '99 issue*, 18:119–130, 1999.

[39] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust. Computing a Canonical Polygonal Schema of an Orientable Triangulated Surface. In *Proceedings of 17th Annu. ACM Sympos. Comput. Geom.*, pages 80–89, 2001.

[40] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *Proceedings of SIGGRAPH*, pages 95–104, 1998.

[41] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least Squares Conformal Maps for Automatic Texture Atlas Generation. In *Proceedings of SIGGRAPH*, pages 362–371, 2002.

[42] P. Lindstrom and G. Turk. Fast and Memory Efficient Polygonal Simplification. In *IEEE Visualization Proceedings*, pages 279–286, 1998.

[43] A. Lingas. Voronoi diagrams with barriers and the shortest diagonal problem. *Inform. Process. Lett.*, 32:191–198, 1989.

[44] S. Lloyd. Least square quantization in PCM. *IEEE Trans. Inform. Theory*, 28:129–137, 1982.

[45] G.H. Meisters. Polygons have ears. *American Mathematical Monthly*, pages 648–651, jun/july 1975.

[46] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, 2002. VisMath.

[47] Victor Ostromoukhov. A Simple and Efficient Error-Diffusion Algorithm. In *Proceedings of SIGGRAPH*, pages 567–572, 2001.

[48] J-C. Léon P. Véron. Static Polyhedron Simplification using Error Measurements. *Computer-Aided Design*, 29(4):287–298, 1997.

[49] Ulrich Pinkall and Konrad Polthier. Computing Discrete Minimal Surfaces and Conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[50] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[51] A. Rassineux, P. Villon, J-M. Savignat, and O. Stab. Surface Remeshing by Local Hermite Diffuse Interpolation. *International Journal for numerical methods in Engineering*, 49:31–49, 2000.

[52] C. Rocchini and P. Cignoni. Generating random points in a tetrahedron. *Journal of Graphics tools, A.K. Peters Ltd*, 5(4):9–12, 2000.

[53] P. Sander, S. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Eurographics Workshop on Rendering 2002*, 2002.

[54] Vitaly Surazhsky, Pierre Alliez, and Craig Gotsman. Isotropic remeshing of surfaces: a local parameterization approach. In *Proceedings of 12th International Meshing Roundtable*, 2003.

[55] Wim Sweldens and Peter Schröder, editors. *Digital Geometry Processing*. Course Notes. ACM SIGGRAPH, 2001.

[56] Joseph R. Tristano, Steven J. Owen, and Scott A. Canann. Advancing Front Surface Mesh Generation in Parametric Space Using a Riemannian Surface Definition. In *Proceedings of 7th International Meshing Roundtable*, pages 429–445, 1998.

[57] Greg Turk. *Generating random points in triangles. Graphics Gems (A.S. Glassner)*. Academic Press, 1990.

[58] Greg Turk. Re-Tiling Polygonal Surfaces. In *Proceedings of SIGGRAPH*, pages 55–64, 1992.

[59] R. Ulichney. A review of halftoning techniques. *SPIE*, pages 378–391, 2000.

[60] Jens Vorsatz, Christian Rössl, Leif Kobbelt, and Hans-Peter Seidel. Feature Sensitive Remeshing. In *Eurographics proceedings*, pages 393–401, sep 2001.

[61] Kouki Watanabe and Alexander Belyaev. Detection of Salient Curvature Features on Polygonal Surfaces. *Computer Graphics Forum. Eurographics 2001*, 20(3):385–392, 2001.

[62] Jianhua Wu and Leif Kobbelt. Fast Mesh Decimation by Multiple-Choice Techniques. In *Vision, Modeling and Visualization Proceedings*, 2002. To appear.