Available online at www.sciencedirect.com
ScienceDirect
Graphical Models 69 (2007) 1–18
Graphical Models
www.elsevier.com/locate/gmod

Manifold-based approach to semi-regular remeshing

Igor Guskov

University of Michigan, 4735 CSE Building, 2260 Hayward Street, Ann Arbor, MI 48109, USA

Received 25 June 2005; received in revised form 6 April 2006; accepted 2 May 2006
Available online 7 July 2006

Abstract

This paper describes a method for semi-regular remeshing of arbitrary shapes. The proposed approach is based on building a parameterization map which is smooth with respect to a differential structure built on the base domain. A global parametric energy functional is introduced and optimized in order to establish a globally smooth parameterization. The proposed approach avoids using meta-mesh construction during the parameterization and resampling stages which allows for an easier implementation. A simple extension of the method is proposed to improve the approximation properties of the resulting remesh.

Keywords: Semi-regular remeshing; Global parameterization

1. Introduction

Semi-regular surface representations offer a variety of attractive features including easy level-of-detail management, efficient data structures and processing algorithms, and therefore lower storage, transmission, and rendering costs. In the past ten years, a variety of remeshing techniques were proposed for transforming input irregular meshes into semi-regular form. Robust remeshing techniques typically rely on establishing a parameterization of the input shape onto a base domain [8,22,14,13,25]. Recently, several global parameterization methods have been introduced with the goal of building ''nice'' parameterization not only within each regular patch, but also across patch boundaries and corners. The ''niceness'' of parameterization is typically evaluated based on its smoothness or distortion metric [15,30,28].

The semi-regular remeshing approach presented in this paper also relies on building a global map from the vertices of the original input mesh onto a simple base domain. The smoothness of this map is evaluated by considering a specially constructed differential structure for the base mesh. We draw our inspiration from the recent work of Ying and Zorin [35], and fix an atlas of *open* charts that covers the base mesh. We use the resulting manifold structure for parameterization construction and for the final resampling step. The main advantage of the manifold approach is that all the mesh vertices use the same global criteria to choose their parametric values, which is in contrast to [14,15] where the global vertices are processed in a separate step.

The smoothness of the parameterization is specifically important for semi-regular surface meshing; it

E-mail address: guskov@eecs.umich.edu

1524-0703/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2006.05.001

is directly related to how well the resulting semi-regular mesh can be compressed. In practice, however, the parametric smoothness is not the only requirement. Other desirable properties include good approximation of the input surface and quality of output mesh elements (e.g. triangle area distribution and triangle roundness [18]). Construction of *anisotropic* surface meshes and its relation to shape approximation was explored in [6,1]. The parametric smoothness and the anisotropy of remesh triangles can conflict with each other. In this paper we propose a simple anisotropic modification to the globally smooth semi-regular remesher that produces better approximation of the original shape on coarser levels of the semi-regular hierarchy while still retaining the ability to efficiently compress the resulting semi-regular mesh.

Our algorithm is fully automatic and does not require user interaction. A few global parameters given to the input of the algorithm determine the number of base mesh vertices, the remeshing level, and the degree of anisotropy requested from the output semi-regular mesh. The remeshing algorithm then proceeds in an unattended fashion. This is an important feature that can allow its successful application in a general digital geometry processing pipeline.

1.1. Overview

Our remeshing algorithm consists of three stages: chartification, parameterization, and sampling. Chartification splits the input mesh into Voronoi tiles which are used to define the structure of the base mesh and to build the initial parameterization. The initial parameterization is then modified to minimize a global smoothness functional defined with respect to a differential structure of the base mesh. Finally, using the parameterization, we perform the resampling of the input mesh into a semi-regular representation.

We shall now describe an example of the remeshing process for the screwdriver model shown in Fig. 1. The remeshing starts with chartification which splits the input surface into *tiles* which are shown in Fig. 1(b). These tiles are in one-to-one correspondence to the vertices of the base mesh shown in Fig. 1(d). Not every tiling of the surface corresponds to a valid base mesh, and we ensure that the topology of the tiling corresponds to the dual of a valid triangular manifold mesh as described in Section 2. In the next step, we construct the initial parameterization within each tile (which is a map from the tile to the Voronoi region on the base mesh, defined in Section 3.3). In the next global parameterization stage we adjust the boundaries between tiles as well as the mapping within each tile. The parametric mapping within each tile can be interpreted as texture coordinates and used for visualization. Fig. 1(e) uses a colored striped image to visualize the parameterization within each tile. The global nature of our parameterization allows one to interpret the parameters within each tile as the

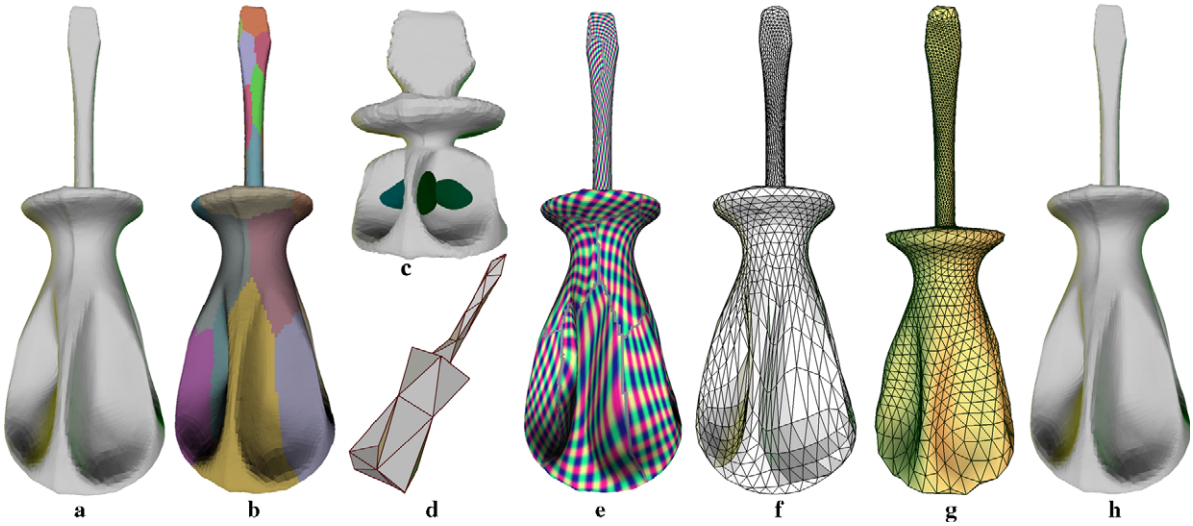


Fig. 1. Screwdriver remeshing overview: (a) input irregular mesh, (b) result of the chartification stage, (c) exaggerated mesh used to achieve anisotropic remeshing, (d) base mesh, (e) parameterization within each tile shown as a texture, (f) semi-regular remesh at level 2, (g) semi-regular remesh at level 3, (h) semi-regular remesh at level 5 (shaded). See text for the discussion of the remeshing process.

map from the original mesh onto the base mesh. The sampling stage uniformly refines the base mesh triangles to the desired level and inverts the mapping to construct the output remeshes at different levels. Figs. 1(f) and (g) show levels 2 and 3 of the remesh (note that most of the vertices are of valence six, with the exception of the small number of vertices corresponding to the base mesh vertices). At level 5, shown in Fig. 1(h) the shaded version is visually indistinguishable from the input irregular mesh.

The output semi-regular mesh consists of a number of large regular patches and is inherently hierarchical. It can be used in a level-of-detail rendering and multi-level computational applications. It can also be compressed with the semi-regular compression schemes [16]. The anisotropy of the output remesh is achieved by using the exaggerated version (see Fig. 1(c)) of the input mesh in the chartification and parameterization stages, as is described in Section 5. The sampling stage still uses the original input mesh.

The presented work is experimental in nature. Our goal was to design a remeshing method that performs well for the typical remeshing scenario, where a fine irregular mesh needs to be converted to a semi-regular mesh with a relatively small number of large patches. We also wanted to have a method that is easy to implement, hence our choice not to implement a meta-mesh construction. It is remarkable that even without the "hard" patch boundaries established by meta-mesh our method is able to handle many complex remeshing tasks. Table 1 summarizes the remeshing results for several models. More discussion on this table is presented in Section 6.

Table 1
Remeshing results summary table

Model	Or.Vs	Base Vs	Rel. Errors L_2, max	Time
Bunny	35K	149	2.2e-5, .157%	11 min
Horse	48K	180	3.8e-5, .33%	11 min
David	275K	509	4.9e-5, .50%	185 min
Feline	50K	256	2.2e-5, .49%	15 min
Holes3	6K	27	3.3e-5, .04%	1 min
Rockerarm	40K	85	2.2e-5, .18%	15 min
Skull	20K	67	4.6e-5, .25%	13 min

The second column shows the number of vertices in the original irregular mesh, the third column shows the number of base mesh vertices (also equal to the number of the chartification tiles). Column four lists the mean-square error between the original and remeshed model and the maximal error (both relative to the bounding box diagonal of the original model) measured with the Metro tool [5]. The last column shows the remeshing time.

1.2. Related work

Our remeshing algorithm follows a semi-regular remeshing pipeline established in [8], and consists of three stages: *chartification*, *parameterization*, and *sampling*.

In the *chartification* step, a vertex-based discrete Voronoi diagram [17] is built based on a set of seeds by flooding, and topological consistency is enforced using a strategy similar to the Ruppert's algorithm [27,20]. We also use a re-seeding procedure of the Lloyd's algorithm [24,29], and iterate it until the stopping criteria are met. Our chartification procedure is similar to the approach used by Boier-Martin et al. [3] to generate layouts for quad remeshing (our procedure is however simpler since we are not interested in creating quad patches). Another inspiration for our chartification method comes from recent papers of Labelle and Shewchuk [20] and Cohen-Steiner et al. [6]. The main advantage of Voronoi-based approaches combined with Lloyd's algorithm lies in their ability to perform global optimization of the Voronoi region arrangement and uniform seed positioning on the input surface. The discrete version of these methods is easy to implement and is intuitive for the user.

Our *parameterization* procedure works under the assumption that the original mesh is much finer than the base mesh produced in the chartification stage (which is true for most semi-regular remeshing scenarios, see Fig. 2 for an example of input and output meshes). We therefore avoid constructing the meta-mesh as in [30,14,21]. Each vertex of the input mesh stores its Voronoi tile index and a parametric 2D value within its tile. No additional patch boundary information is stored, which makes the implementation of our technique simpler than the meta-mesh-based approaches.

Our parameterization and resampling stages rely on the differential structure for the base mesh that is very similar to the one used by [35] in the surface modeling context. This is another reason why we are able to avoid the construction of the meta-mesh like structures: our *sampling* procedure uses the triangulation within each chart and then blends the possibly different sample positions for each output semi-regular vertex (note that a chart is larger than the Voronoi tile, and multiple charts cover the same regions of the base mesh). The notion of the manifold has been used in differential geometry for a long time [33,10]. Grimm and Hughes paper [12] introduced a geometric modeling application of

Fig. 2. Part of David's surface before and after remeshing. Note that the input mesh is very irregular. A single patch of the remeshed surface is highlighted in the result. See also Fig. 14.

manifolds. The manifold approach has also been used for parameterizing simple surfaces [11]. This paper shows the applicability of manifold-based approaches to complex geometries. Note that while the paper of Khodakovsky et al. [15] uses the notion of overlapping charts, their charts do not form *an open atlas*.

Our approach is also close to [30,19] in that the base mesh only plays an intermediary role; and the smoothness of our parameterization is evaluated on a different domain, built with our differential structure construction. The method of Schreiner et al. [30] can achieve similar results, however it is harder to implement due to the need for meta-mesh construction.

1.2.1. Contribution

The main contribution of this paper is to introduce a manifold-based approach to semi-regular parameterization and resampling that does not require a meta-mesh construction and is simple to implement.

1.2.2. Restrictions

- Our method is specifically targeted at the semi-regular mesh construction, and will not work for morphing applications as in [30,19] which typically require an elaborate meta-mesh construction.
- Our method assumes that the input mesh is a valid triangular mesh with topological noise removed [34].
- While our method can handle meshes with boundaries, sharp creases are not preserved in the current implementation.

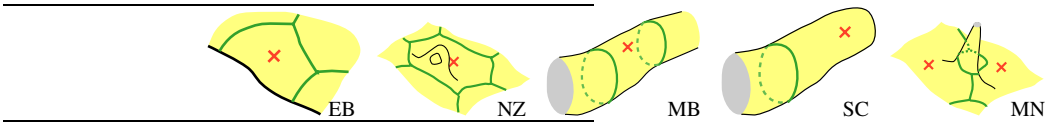
1.2.3. Organization of the paper

The following sections describe the details of our approach. Section 2 covers our chartification procedure, Section 3 introduces differential structure on the base mesh and its application in the resampling process. Section 4 details our parameterization procedure, the initial parameter assignment, and the global parametric energy functional. Section 5 introduces the anisotropic modifications. The results are presented in Section 6.

2. Chartification

The goal of the first stage of the remeshing process is to construct the parametric domain (base mesh) to be used in later stages. The base mesh connectivity corresponds to the coarsest level of the output semi-regular mesh, thus a careful choice of the base mesh is important. Our chartification approach combines the ideas of several previously known approaches; we shortly describe it for completeness.

The paper of Klein et al. [17] discusses the advantages of vertex-based Voronoi tilings on triangulated surfaces, in particular, the automatic satisfaction of the condition that no more than three tiles can meet at any internal point of the surface, and no more than two can meet on a boundary of the surface. The tile-growing algorithm of [17] proceeds by employing the multi-source Dijkstra algorithm on the edge graph of the mesh using Euclidean distance between vertices as the edge cost. The algorithm checks for the topological conditions and stops when one of the conditions is not satisfied or when all the vertices are added. Our flooding approach differs only in that we do not check the topology conditions until *after* all the vertices have been added. The tiles that do not satisfy the topological conditions are then refined by adding more seeds near the offending sites.



We use the following topological conditions for our collection of Voronoi tiles:

- *Encrouched boundaries* (EB). A Voronoi tile that touches a surface boundary has to have its seed on that boundary. If this condition fails we introduce a new seed on the boundary of the surface within the current tile.
- *Nonzero genus* (NZ). A Voronoi tile needs to have zero genus. If this condition fails we introduce a new seed at a vertex adjacent to the seed vertex of the tile.
- *Multiple tile boundary components* (MB). The boundary of a Voronoi tile can only consist of a single connected component. If this condition fails we introduce a new seed at a vertex adjacent to the seed vertex of the tile.
- *Single curve tile boundaries* (SC). A Voronoi tile that is adjacent to other tiles should not have more than one neighbor along each tile boundary curve. If this condition fails we introduce a new seed at a vertex adjacent to the tile boundary curve.
- *Multiple neighbor instances* (MN). Any pair of Voronoi tiles cannot touch along more than a single tile boundary curve. If this condition fails, we introduce a new seed at a vertex adjacent to the shorter among the offending tile boundary curves.

Note that each of the topological fixes described above may not actually get rid of the offending condition right away. For example, if a tile is of genus five, introducing an additional tile will not make the resulting tiles to be of genus zero. However, it will refine the tiling in the correct neighborhood. Therefore, when sufficiently many iterations of these corrections are applied (together with seed repositioning), we will obtain a topologically correct tiling.

We use the $L_{2,1}$ proxies introduced in [6] to identify tiles with large area and high curvature. Thus, an additional condition is checked for every tile: we compute the area-weighted average proxy normal $\bar{n}$ within each tile (see [6] for details) and check whether $\int (n-\bar{n})^2 \, d\sigma < \epsilon^2$ is satisfied. If this normal error is above the threshold ϵ specified by the user, we insert another seed next to the original tile seed.

We initialize our chartification process with randomly distributed set of seed vertices. We then iterate the flooding, topological and error checking, and seed repositioning until convergence is achieved. The seed repositioning step moves the seed of each tile to the vertex closest to the 3D centroid of the tile.

In our experiments, the above process always converged, in a sense that the tile assignment of each vertex stays the same and no new tiles are introduced. Note that when the input mesh has topological noise in the form of small tunnels, the topology enforcement procedure will produce very small tiles with possibly only a single vertex per tile in the tunnel region. Therefore, even though the chartification refinement process will always stop (the refinement of the tiling cannot be finer than the tiling that has one original vertex per tile), the resulting tiling will not be useful for our parameterization procedure described in the next section, as it assumes having sufficient number of vertices in every tile. Hence, as we stated earlier, our method in its current implementation does not handle meshes with topological noise.

After chartification is completed, we construct the triangular base mesh by creating a base vertex for each tile, and by connecting the base vertex pairs that correspond to adjacent tiles. In effect, this is similar to the construction of the Delaunay

triangulation from the corresponding Voronoi decomposition. Once the base mesh is constructed, we are ready to proceed to the surface parameterization stage discussed in the following sections.

In order to avoid confusion, we clarify our naming convention for describing surface regions:

- A *tile* is a surface region produced by the chartification procedure described in this section. The insides of the tile regions do not overlap, but tiles share their boundaries. If two tiles share a curve, the corresponding base vertices are connected by an edge. Three tiles sharing a corner point correspond to a triangle in the base mesh.
- A *chart* is an open region on the surface and the collection of charts forms the atlas on the surface. We shall introduce charts formally in the next section.
- A *patch* is the surface region that maps to a triangle of the base mesh. It directly corresponds to the regular connectivity patch of the output semi-regular mesh.

3. Manifold-based resampling approach

In this section, we give the details of what we mean by the manifold-based parameterization and describe the resampling process. The details of how the parameterization is actually computed are delayed until the next section.

Leaving the details out, the overall remeshing process starts with the tiles produced by the chartification procedure, and constructs a base mesh and the initial parametric mapping from the original surface to that base mesh as described in Section 4.1. Then the global parameterization adjustment happens that improves the properties of this mapping, as described in Section 4.3. The base mesh is then recursively refined by performing one-to-four splits of each triangle, and the remesh samples are produced that approximate the original surface.

The manifold parameterization consists of a number of maps from the original mesh regions to the one-rings of vertices in the base mesh (base vertices). The base mesh itself is not a Euclidean space hence in order to use off-the-shelf optimization and resampling routines, each of these base one-rings is unwrapped onto a planar region. The map from the original surface to these planar regions are actually stored by the manifold parameterization. The inverse of each of these maps is easy to find, and each output semi-regular sample is collected as a weighted average of several (usually three) inverses of individual chart mappings as described in Section 3.4.

3.1. Notation

We employ the usual notation for triangular meshes. The two meshes that participate in the remeshing process are the original mesh $\mathscr{M}$ with the sets of vertices $\mathscr{V}_{\mathscr{M}}$, edges $\mathscr{E}_{\mathscr{M}}$, and faces $\mathscr{F}_{\mathscr{M}}$, and the base mesh $\mathscr{B}$ with its sets of primitives denoted similarly as $\mathscr{V}_{\mathscr{B}}, \mathscr{E}_{\mathscr{B}}, \mathscr{F}_{\mathscr{B}}$.

The parameterization of $\mathscr{M}$ onto $\mathscr{B}$ is a bijective mapping $\bar{u} : \mathscr{M} \to \mathscr{B}$. One usually specifies the mapping $\bar{u}$ on the set of original vertices and extends it to the whole of $\mathscr{M}$ as a piecewise-linear mapping. Then, given a set of semi-regular samples from $\mathscr{B}$ one finds the corresponding points on the original surface $\mathscr{M}$ by inverting $\bar{u}$. The establishment of an exact piecewise linear mapping often requires the introduction of a *meta-mesh* that splits the original mesh by curves corresponding to the base edges. Such construction is absolutely necessary in the case when the original and the base meshes are of comparable size and mesh density (for instance, in a morphing application [21,26]). In a typical semi-regular remeshing scenario, however, the goal is to build large patches of regular connectivity. Hence, the original mesh is typically much finer than the base mesh. It should therefore be possible to avoid the meta-mesh construction.

We shall use a manifold atlas together with partition of unity approach to make the remeshing work without a meta-mesh. We do not use a global bijective parameterization defined on the whole of the original mesh; rather, we use the restriction of the parametric map to the set of original mesh *vertices*, and denote it as $u \stackrel{\text{def}}{=} \bar{u}|_{\mathscr{V}_{\mathscr{M}}}$. In this section, it will be useful to think of u as the discretization of a continuous bijective map from the original surface mesh onto the base mesh. We will delay the description of the particular method by which the mapping $u : \mathscr{V}_{\mathscr{M}} \to \mathscr{B}$ is computed until Section 4. At this point we introduce a differential structure on the base mesh, and show how we can combine it with the mapping u to perform the semi-regular resampling of the surface. While this resampling step follows the computation of parametric mapping u in the actual algorithm, the resampling stage description should give a clearer picture of the role the parameterization plays.

3.2. Differential structure for the base mesh

The differential structure of the base mesh is introduced in order to allow characterization of the smoothness of parameterization maps. It is very similar to the structure for quad meshes presented in [35]. The maps R_b introduced in this section will relate the base mesh with a collection of planar charts. Combining these with the parameterization map u, we obtain a collection of charts for the input surface. We start by defining the differential structure itself (that is, a collection of neighborhoods with associated mappings to planar regions). See Fig. 3 for the illustration of involved mappings.

We consider a collection of charts $\{\Omega_b\}_{b\in\mathscr{V}_\mathscr{B}}$, where each chart Ω_b is an open subset of the base mesh $\mathscr{B}$ consisting of all the primitives that are adjacent to a base vertex b; that is, $\Omega_b \stackrel{\text{def}}{=} \text{star}(b) \subset \mathscr{B}$. We define a fixed injective mapping R_b from each Ω_b to the 2D plane, using a generalization of the polar map that allows more control over the distribution of remeshed vertices near patch corners (using the parameter h defined below).

Consider an internal base vertex b (the boundary case is similar and is presented afterwards). Let the valence of b be equal to n. Using orientability of $\mathscr{B}$ we can index the base triangles belonging to Ω_b from 0 to $n-1$ in the counter-clockwise fashion, denoting them as $\tau_0,\ldots,\tau_{n-1}$. Our implementation and the discussion in this section do not deal with non-orientable surface, although such an extension should be possible.

Consider a point $s \in \Omega_b$. There will be at least one triangle $\tau_{m(s)} = (b_1, b_2, b_3)$ such that s belongs to its closure: $s \in \bar{\tau}_{m(s)}$ (here $m(s)$ is the index of the triangle with respect to the vertex s). Without loss of generality, assume that $b = b_1$. There is a vector of barycentric coordinates $\gamma(s)$ such that $s = \sum_{k=1}^{3} \gamma_k(s) b_k$. A part of the map construction is more convenient to describe in the complex plane, hence we introduce $z(s) \stackrel{\text{def}}{=} \sum_{k=1}^{3} \gamma_k(s) K_k$, where $K_1 = 0, K_2 = 1, K_3 = (1+\sqrt{3}\mathrm{i})/2$ are the corners of a canonical equilateral triangle (Fig. 4).

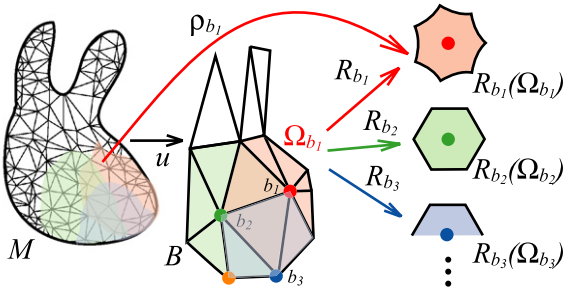


Fig. 3. Mappings and sets used in the parameterization. Map u sends vertices of the original mesh $\mathscr{M}$ onto the base mesh $\mathscr{B}$. Maps R_{b_k} are fixed for each base vertex b_k and map one-rings $\Omega_{b_k} \subset \mathscr{B}$ onto the planar region $R_{b_k}(\Omega_{b_k})$. The mapping u is to be found during the parameterization process for every vertex of the original mesh, with the goal to improve properties of maps $\rho_{b_k} = R_{b_k} \circ u$.

The polar map is often used for mapping the neighborhood of a vertex onto a plane (see for instance [22]). In order to control the sampling pattern near patch corners we introduce an additional rescaling after the application of the polar map that changes the character of the singularity at the origin (see Fig. 5). We define a map (Fig. 6)

$$\kappa_\alpha^h(z) \stackrel{\text{def}}{=} z^\alpha / |z|^{(\alpha-1)h}, \quad \alpha = 6/n.$$

For $h = 0$, the usual polar map results, for $h = 1$ the map actually preserves the magnitude of the input complex number, only rescaling the angular component. Now, accounting for a rotation that depends on the index $m(s)$ of the containing base triangle and is necessary to have a globally consistent definition, we obtain the mapping

$$r_b(s) \stackrel{\text{def}}{=} \kappa_{6/n}^h(z(s)) \mathrm{e}^{\mathrm{i}\frac{2\pi m(s)}{n}}.$$

Note that this particular definition works well with the default complex power function implemented in the C++ standard library. We can now define the final mapping of one-ring of the base vertex b to a 2D region surrounding the origin as follows:

$$R_b(s) \stackrel{\text{def}}{=} (\text{Re}(r_b(s)), \text{Im}(r_b(s))).$$

For the boundary vertex b, the above construction works by replacing n to be $2*(\text{val}(b)-1)$ where $\text{val}(b)$ is the valence of the base vertex b. The planar region for the boundary case will be fully contained in the upper half of the 2D plane. This is similar to the approach used in the MAPS paper [22].

3.3. Base mesh as a metric space

Sometimes it will be convenient to treat the base mesh as a metric space. In order to do that we specify a unit equilateral triangle for each base face and we glue adjacent pairs of such triangles along common edges and vertices. The distance $\text{dist}(s_1, s_2)$ between any two points $s_1, s_2 \in \mathscr{B}$ is the length of the shortest (piecewise linear) path. For two neighboring base vertices $b_1, b_2 \in \mathscr{V}_\mathscr{B}$ connected by an

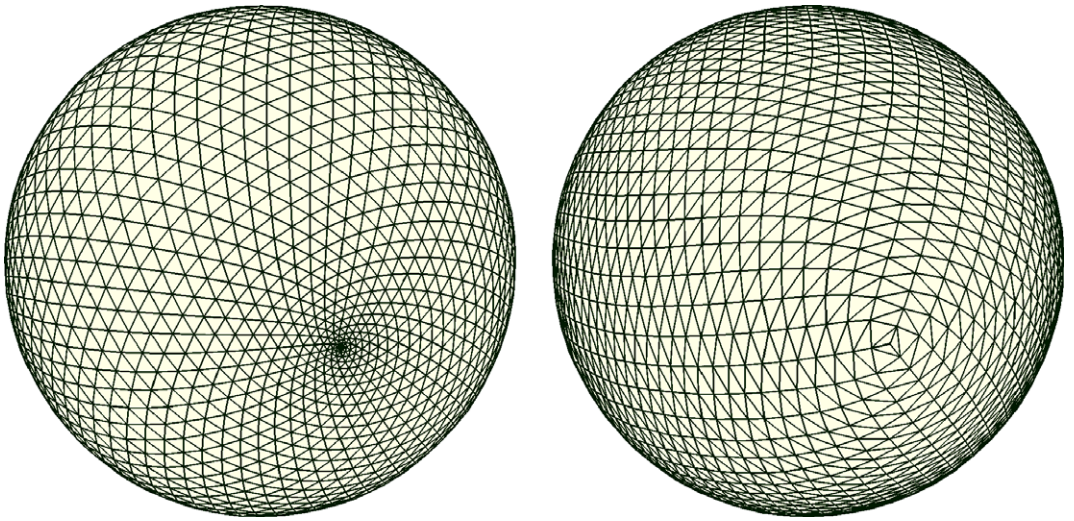


Fig. 4. Two sphere remeshings with different values of h parameter used in the differential structure mappings. Left: $h = 0$ results in a typical polar map picture; right: $h = 1$ results in a better triangle area distribution (but distorts the triangle angles).

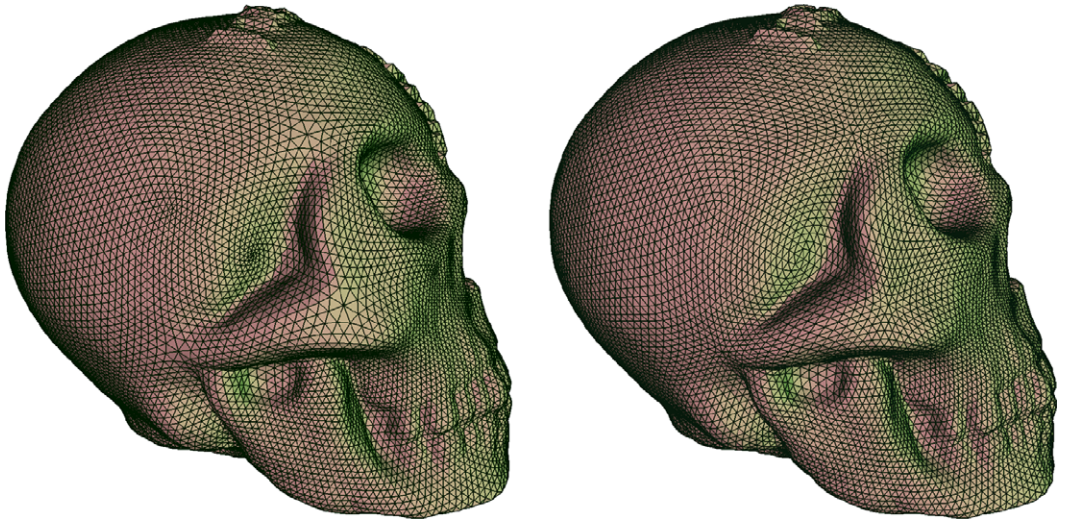


Fig. 5. Skull remesh with $h = 0$ (left) and $h = 1$ (right). Setting $h = 0$ results in a typical polar map picture; $h = 1$ results in a better triangle area distribution (but distorts the triangle angles).

edge we have $\mathrm{dist}(b_1,b_2) = 1$, and for any base face $t = (b_1, b_2, b_3) \in \mathscr{F}_{\mathscr{B}}$ its barycenter $c(t)$ is a constant distance away from its vertices $\mathrm{dist}(c(t), b_k) = \sqrt{3}/3$ for $k = 1, 2, 3$. We will not actually be using any paths longer than 1 in this paper.

3.4. Resampling step of our algorithm

We can now describe the semi-regular resampling procedure used in our algorithm (assuming that $u : \mathscr{V}_{\mathscr{M}} \to \mathscr{B}$ is given).

Using the parameterization map together with the differential structure mapping, we determine the planar points corresponding to the vertices within the chart Ω_b, that is, $\rho_b(v) \stackrel{\mathrm{def}}{=} R_b(u(v))$ (please refer to Fig. 3). We shall extend this 2D mapping in a piecewise linear fashion to the set of triangles D_b whose vertices are all within $u^{-1}(\Omega_b)$. Formally, $D_b \stackrel{\mathrm{def}}{=} \{t \in \mathscr{F}_{\mathscr{M}} : t = (v_1, v_2, v_3), v_k \in u^{-1}(\Omega_b), \quad k = 1, 2, 3\}$. Given a set of planar samples within the image of ρ_b it is then possible to determine the corresponding points on the original mesh. Formally, if

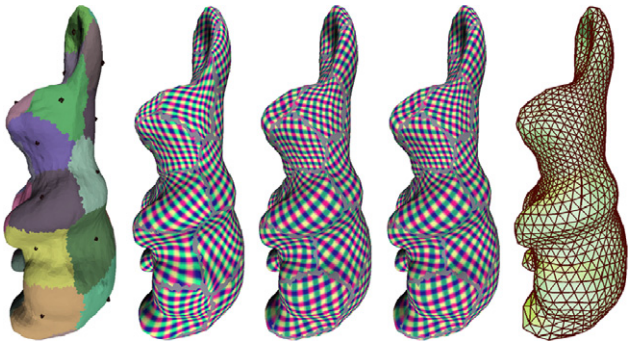


Fig. 6. An example of the parameterization process for the *Rabbit* model (left to right): Voronoi tiles, result of initialization within each tile, result of edge-based optimization stage, result of vertex-based optimization stage, the third level of the remeshed surface. All the textures visualize the values of $\rho_b(v) = R_b(u(v))$ stored with every vertex. Note that only very small change happens during the vertex-based optimization.

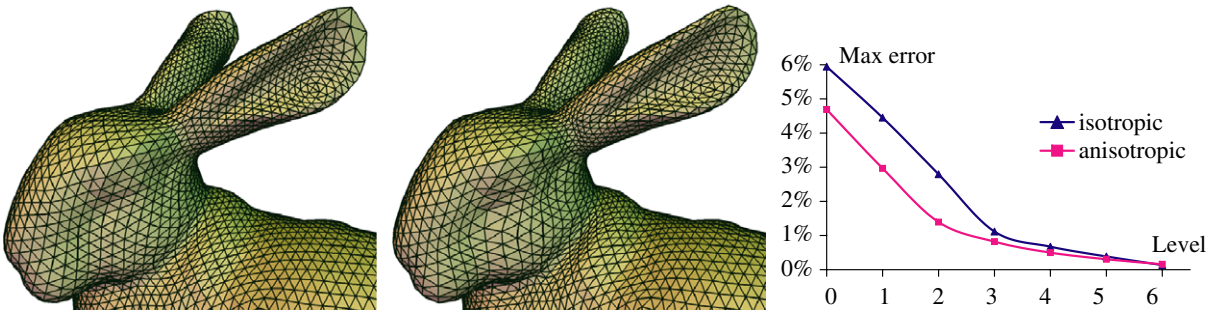


Fig. 7. Comparison of isotropic and anisotropic parameterization for the same base mesh and initial tiling. On the right, the plot of maximal relative error measured with Metro tool [5].

a sample s is such that $R_b(s) \in \rho_b(D_b)$ for some base vertex b, then the point $p(s) \in \mathscr{M}$ can be found by using the map $\rho_b^{-1} \circ R_b$. However, there may be multiple base vertices b that can be used for the above mapping; this subset of base vertices can be defined as $\mathscr{V}^s_{\mathscr{B}} \stackrel{\text{def}}{=} \{b \in \mathscr{V}_{\mathscr{B}} : R_b(s) \in \rho_b(D_b)\}$. If the original mesh is sufficiently fine, the results of all these remappings will be close (they coincide on the vertices of the original mesh by construction). We use the blending weights of the sample s in order to recombine these multiple possibilities; namely, we determine the remeshed position $p(s)$ corresponding to the sample s via

$$p(s) \stackrel{\text{def}}{=} \frac{1}{\sum_{b \in \mathscr{V}^s_{\mathscr{B}}} w_b(s)} \sum_{b \in \mathscr{V}^s_{\mathscr{B}}} w_b(s) \rho_b^{-1}(R_b(s)).$$

The blending weights for a point $s \in \mathscr{B}$ are defined as follows: if s lies within base triangle (b_1,b_2,b_3) then the weights $w_{b_k}(s) = \beta(\text{dist}(s,b_k))$, and the weights are zero for all other base vertices. The function $\beta(r)$ is defined as a smooth cubic spline $(0 < \Delta_0 < \Delta_1 < 1)$:

$$\beta(r) \stackrel{\text{def}}{=} \begin{cases} 1, & r \leqslant \Delta_0, \\ B(\frac{\Delta_1 - r}{\Delta_1 - \Delta_0}), & \Delta_0 < r \leqslant \Delta_1, \\ 0, & r > \Delta_1. \end{cases} \quad B(t) = t^2(3 - 2t),$$

We use $\Delta_0 = 0.4$ and $\Delta_1 = 0.75$ for the examples in this paper.

The resulting resampled 3D point $p(s)$ produced by blending does not lie exactly on $\mathscr{M}$ but will be very close to it (see our remeshing results in Section 6). In fact, the error committed by a blending process will be zero on the vertices of the original mesh, and inside of an input mesh triangle this error is proportional to the disparity of the functions $\rho_b^{-1} \circ R_b$ (which is bounded by the triangle size of the input mesh) and the curvature of the surface.

Using the described approach, given any set of semi-regular samples on $\mathscr{B}$ we can produce the corresponding samples of the output semi-regular surface. The following section will explain how we construct the parametric value for each vertex of the input mesh.

4. Parameterization

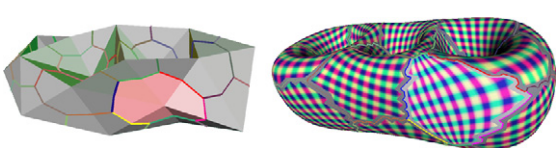
We build the parameterization map in two steps. First, we construct the initial mapping $u_{(0)}$ from $\mathscr{M}$ to $\mathscr{B}$ using the correspondence between the Voronoi decomposition of the original map and the base mesh. This mapping will only be smooth within each Voronoi tile. The global parametric smoothness functional will then be introduced in Section 4.2 and the map $u_{(0)}$ will be used as the initial guess to the optimization procedure.

4.1. Initial parameter assignment

The chartification algorithm of Section 2 builds the Voronoi decomposition $\{T_b^{\mathscr{M}}\}$ of the original mesh $\mathscr{M}$ whose dual complex is the base mesh $\mathscr{B}$. We construct the initial parameterization by considering the corresponding Voronoi decomposition $\{T_b^{\mathscr{B}}\}$ of the base mesh $\mathscr{B}$ and mapping the boundaries of each tile $T_b^{\mathscr{M}}$ onto the corresponding boundaries of $T_b^{\mathscr{B}}$.

Let $T_b^{\mathscr{B}}$ denote the Voronoi tile of a base vertex b. Note that there is a unique corresponding Voronoi tile $T_b^{\mathscr{M}}$ on the original mesh. The boundary of the tile $T_b^{\mathscr{M}}$ is a loop of surface curves, each curve γ separating $T_b^{\mathscr{M}}$ and its neighboring tile $T_{b'}^{\mathscr{M}}$. The base mesh will contain an edge $e=(b,b')$; assuming that (b,b') is not on the boundary, there will be two base faces $t_1=(b,b',b_1)$ and $t_2=(b',b,b_2)$. The boundary between two *base mesh* Voronoi tiles $T_b^{\mathscr{B}}$ and $T_{b'}^{\mathscr{B}}$ is a two-segment path Γ starting at the barycenter $c(t_1)$ of t_1, crossing the edge e at its midpoint $c(e)$, and proceeding to $c(t_2)$. We stipulate that $u_{(0)}$ maps the curve γ of the original mesh bijectively onto the path Γ, so that $u_{(0)}(\gamma)=\Gamma$. The endpoints of γ map onto $c(t_1)$ and $c(t_2)$ and the inside of γ maps according to its arclength. This establishes the boundary conditions for $u_{(0)}$ within each $T_b^{\mathscr{M}}$.

In order to initialize the parameterization on the vertices of $\mathscr{M}$ we represent the parameterization within each tile $T_b^{\mathscr{M}}$ as the planar values $\rho_b(v)=R_b(u_{(0)})$, and solve the mean-value parameterization (MVP) [9] with the boundary conditions given on the boundary curves of each tile. The picture below highlights the corresponding regions on the base and the input meshes.



4.2. Global parametric energy functional

In order to build a useful semi-regular remesh (for instance, a remesh that is amenable to semi-regular mesh compression techniques [16]), it is important for the underlying parameterization to be smooth [15]. A proper parameterization should also be bijective, so that its inverse is a well-defined and smooth function. The definition of smoothness in the global sense is not trivial. On the other hand, there exist very good methods for creating smooth parameterization of mesh patches onto planar regions. We take advantage of these methods by using the differential structure of the base mesh defined above.

For a single base vertex b, we can consider the mapping $R_b \circ u : u^{-1}(\Omega_b) \to \mathbf{R}^2$. We postulate that for a good parameterization u its values in the plane will satisfy the mean-value parameterization scheme of Floater [9]:

$$R_b(u(v)) = \sum_{v' \in \omega_1(v)} a_{vv'} R_b(u(v')), \tag{1}$$

where $a_{vv'}$ are MVP coefficients and $\Omega_1(v)$ is the one-ring of neighbors of v. Note that we can only enforce such a constraint for an internal vertex v of the chart $u^{-1}(\Omega_b)$, whose neighbors all belong to $u^{-1}(\Omega_b)$ as well.

The charts $u^{-1}(\Omega_b)$ cover the vertex set $\mathscr{V}_{\mathscr{M}}$ of the original mesh. If the original mesh is sufficiently fine, each vertex will belong to multiple charts. Therefore, there will be multiple constraints of the type (1) that a parametric value at a vertex should satisfy. We find that using a weighted combination of squared residuals of these constraints works well in practice. That is, we define a global functional

$$J(u) =^{\text{def}} \sum_{v \in \mathscr{V}_{\mathscr{M}}} \sum_{b:\omega_1(v) \cup \{v\} \subset u^{-1}(\Omega_b)} \sigma(v) w_b(u(v)) \times \left(R_b(u(v)) - \sum_{v' \in \omega_1(v)} a_{vv'} R_b(u(v')) \right)^2,$$

where $\sigma(v)$ is the area associated with the vertex v. Note that the dependence is on the parameterization function $u : \mathscr{V}_{\mathscr{M}} \to \mathscr{B}$. Also, note that this is not a quadratic functional due to the presence of u in the weighting terms.

The above energy functional depends on the parametric values of each vertex. It is minimized in a piecewise fashion as described in the following section. In our experiments we observed a good

optimization performance, so that the produced parameterization smoothes out across tile boundaries and results in a bijective mapping from the input surface onto the base mesh. Currently, we do not have the theoretical justification on the convergence of the optimization procedure, or the properties of the resulting mapping.

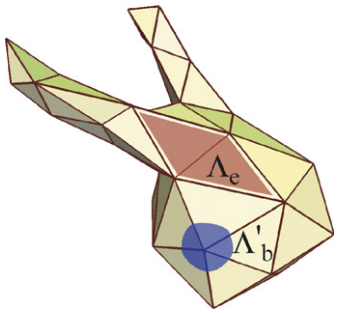
4.3. Optimization strategy

The functional $J(u)$ defined in the preceding section has as its argument the parameterization function $u : \mathscr{V}_{\mathscr{M}} \to \mathscr{B}$. The range of this function is not a Euclidean space but a base mesh manifold. In order to be able to apply standard optimization methods which work in Euclidean space, we minimize $J(u)$ in a piecewise fashion. Every optimization step will choose a region Λ of the base mesh that can be smoothly[1] mapped onto the plane. The set $u^{-1}(\Lambda)$ of vertices of $\mathscr{M}$ that currently have their parametric values within Λ will be collected, and all its vertices will get some 2D value based on their current parameterization value. The parametric energy functional $J(u)$ is then re-expressed in terms of these 2D values. At this stage, a standard optimization procedure is invoked to produce the locally optimal values for each vertex. These resulting 2D values are then mapped back onto the base mesh $\mathscr{B}$, and the optimization proceeds to the next chosen region. One could also implement the optimization of the functional varying the parametric values in all the charts at once, however due to the overlapping nature of our atlas, the resulting optimization would have to include constraints. In the piecewise optimization, we can proceed in the unconstrained fashion.

One natural choice for the collection of regions is given by the atlas of base vertex charts $\{\Omega_b\}$. However, in such an optimization one has to account for the interaction between the parameterization in a chart and all the overlapping charts at once, and the number of these charts depends on the valence of the vertex. Additionally, the remapping expressions between charts become very complex. We find that a different collection consisting of two types of smaller regions results in a simpler expression for remapping and works well in practice. We define two types of regions:

- "Base-edge regions": $\{\Lambda_e\}_{e\in\mathscr{E}_{\mathscr{B}}}$, where Λ_e is formed by two base triangles adjacent to a base edge e (for a boundary edge e only a single adjacent base triangle is used).
- "Base-vertex regions": $\{\Lambda'_b\}_{b\in\mathscr{V}_{\mathscr{B}}}$, where $\Lambda'_b \stackrel{\text{def}}{=} \{s : \operatorname{dist}(s,b) < \Delta_0\}$.

The constant Δ_0 in the above definition is the same as the one used in the definition of blending weights $w_b(s)$. The optimization within the region $u^{-1}(\Lambda_b)$ becomes a simple quadratic optimization problem since a single weight $w_b(s)$ is equal to 1 while all the other $w_{b'}$ for $b' \neq b$ are zeroes. The parameterization values are mapped to 2D values by using $\rho_b(v) = R_b(u(v))$, and the optimization within $u^{-1}(\Lambda_b)$ becomes equivalent to solving for a mean-value parameterization. Note, that only the vertices that are contained in $u^{-1}(\Lambda_b)$ together with all of their neighboring vertices are free to change the parameterization during this step. The remaining vertices of $u^{-1}(\Lambda_b)$ are not allowed to change their parameterization and play the role of boundary conditions.



Example regions Λ_e and Λ'_b.

The optimization step for the "edge-region" $u^{-1}(\Lambda_e)$ is less trivial. First of all, the parametric values within Λ_e are mapped onto the plane by using a hinge map on two adjacent equilateral triangles (which correspond to two base faces sharing e). Formally, we define a piecewise linear mapping $\xi{:}\Lambda_e \to \mathbf{R}^2$ as follows: let $e = (b_1,b_2)$, and (b_1,b_2,b_3) and (b_2,b_1,b_4) be two adjacent base faces. Define ξ so that $\xi(b_1) = (-1/2, 0)$, $\xi(b_2) = (1/2, 0)$, $\xi(b_3) = (0, \sqrt{3}/2)$, $\xi(b_1) = (0, -\sqrt{3}/2)$. We remap the values of $u(v)$ using ξ, denoting the result as $y(v) \stackrel{\text{def}}{=} \xi(u(v))$. We can now redefine J in terms of y-values and compute the corresponding gradients to be used in the optimization procedure.

Let $F(y) \stackrel{\text{def}}{=} J(\xi^{-1} \circ y)$ and $\zeta_k(y) = R_{b_k}(\xi^{-1}(y))$. The expressions for $\zeta_k(y)$ will have the following form:

[1] With respect to the differential structure defined by $\{R_b\}$.

$$\zeta_k(y) = \kappa^h_{6/val(b_k)}(A^e_k(y_1 + \mathrm{i}y_2) + B^e_k), \quad k = 1,2,3,4.$$

Here A^e_k and B^e_k are complex constants. The expression for the functional to optimize is then

$$F(y) = \sum_{v\in\check{\Lambda}_e}\sum_{k=1}^{4}\sigma(v)w_{b_k}(\xi^{-1}(y(v))) \times\left(\zeta_k(y(v)) - \sum_{v'\in\omega_1(v)} a_{vv'}\zeta_k(y(v'))\right)^2; \qquad (2)$$

where the outer summation is over the set $\check{\Lambda}_e$ of vertices of the region Λ_e whose neighbors also belong to Λ_e. The remaining vertices form the set of boundary condition vertices $\Lambda_e \setminus \check{\Lambda}_e$, the parameterization values of which do not change during the current optimization step. We compute the analytic expression for the gradient of $F(y)$ and use it in the optimization library [36]. Upon convergence, we reassign the parametric values for every vertex (in this step, the tile assignment of a vertex may change).

5. Anisotropic extension

Both theory and practice of surface meshing indicate that in order to achieve optimal shape approximation one needs to change the aspect ratio and orientation of approximating elements according to the curvature tensor of the surface [4,32]. This can be done by either using a directly computed curvature [1], or by solving an optimization problem [6]. In the mesh generation community, anisotropic meshes are often built by the modification of the metric on the underlying space [31]. A similar prewarping idea has also been used for adaptive isosurface extraction [2].

For the semi-regular remeshing, both the chartification and the parameterization components of the algorithm should be modified to account for the surface curvature. Additional difficulty is created by the potential misalignment of sharp surface features and the regularly sampled patches of the output surface.

We propose an approach that is very simple and that extends the idea of modifying the surface metric. Given an original mesh $\mathscr{M}$ we create its exaggerated version $\hat{\mathscr{M}}$ and use it in both the chartification and parameterization steps. The final resampling step is then performed on the original surface. In order to do so, we need to have a smooth correspondence between the original and the exaggerated meshes. Let $\chi : \mathscr{M} \to \hat{\mathscr{M}}$ be such a map, then given the parameterization $\hat{u}$ of $\hat{\mathscr{M}}$, the corresponding parameterization of the original mesh can be written as $u = \hat{u} \circ \chi$. The resampling step can use this parameterization without any problem. In fact, our exaggerated mesh will be of the same connectivity as the original mesh; only the vertex positions will be modified. Hence, the map χ is a trivial vertex-to-vertex correspondence.

Note that all the algorithms described in the preceding sections can work seamlessly on either the original or the exaggerated meshes without any modifications. For instance, the chartification stage relies on the edge lengths of the underlying mesh in the flooding step, and uses the surface normals in the proxy error computation. It is easy to substitute edge lengths and normals of the exaggerated mesh instead of the original ones. Similarly, the parameterization step can be modified by using exaggerated positions for the computation of the coefficients of mean-value parameterization.

In order to produce the exaggerated version of the input mesh, we employ a well-known approach that extrapolates from a smoothed version of the model through the original to produce an enhanced model with exaggerated high curvature features. Let $a \geqslant 0$ be the anisotropy parameter specified by user, with the anisotropy increasing as a grows. We use a mean-curvature flow algorithm similar to [7] that can produce smoother versions of the input model by evolving the surface based on the computed curvature. Let $\tilde{\mathscr{M}}_a$ be the result for time $t = a$. We define the exaggerated mesh $\hat{\mathscr{M}}_a \stackrel{\text{def}}{=} (1+a)\mathscr{M} - a\tilde{\mathscr{M}}_a$. We do not claim that this exaggeration results in an optimal anisotropic behavior, however this choice is simple to implement and it does improve the approximation properties of the resulting semi-regular remesh.

Thus, the larger values of a will result in more iterations of the smoothing algorithm, as well as increased extrapolation extent. Note that the vertex-to-vertex mapping from $\mathscr{M}$ to its smoothed version $\tilde{\mathscr{M}}_a$ is a smooth function because the curvature flow we use preserves the surface sampling. The exaggerated mesh is therefore also in a smooth correspondence to the original mesh. Note however that as a increases, the actual quality of exaggerating map deteriorates (this is a mild variant of the instability of the backward curvature flow).

6. Results

We have experimented with a variety of models using our remeshing approach. Table 1 summarizes

the results for several models. The timing was measured on 3GHz Pentium 4 CPU. The main bottleneck is currently in running the non-linear optimization on the edge-based patch regions. Most of our results were obtained after a single pass of optimization over all the base-edge regions followed by a single pass for every base-vertex region. We measured all the errors (L_2 and maximum relative to the bounding box diagonal of the original model) using the Metro tool [5].

We compared the performance of our anisotropically remeshed models with the semi-regular mesh compression package of [16]. The plots in Fig. 12 show the rate-distortion curves for four remeshed models. The horizontal axes show the size of archives in thousands of bytes, the vertical axis measures the PSNR $= 20\log_{10}(\text{BBoxDiag}/L^2\text{-error})$ [16] computed from the mean-square error returned by Metro tool. Higher values of PSNR mean smaller errors. The comparisons are with the Globally Smooth Parameterization remesher of [15], Normal remesher of [14], and MAPS remesher of [22].

We see that the performance of our algorithm is comparable with GSP remesher of Khodakovsky et al. [15]. The main benefit of our approach is that the remesh is constructed fully automatically, without user intervention needed to constrain some base vertices to stay on extended features.

The anisotropic modification improves the approximation quality of the resulting semi-regular remesh, especially on coarser level as is illustrated by the data in Figs. 8 and 9. The plots show the comparison of the PSNR distortion metric for the isotropic and anisotropic versions of the remesher. We see that for the same polygon count the anisotropic versions have much smaller error. These results should be useful for rendering applications where the number of triangles rather than the size of compressed archive is important for level-of-detail management. At the same time, the output semi-regular mesh can still be efficiently compressed as shown above.

The approximation improvement is due to the fact that the base mesh vertices are better adapted to the underlying shape, and that the parameterization does not pull the samples away from extended features. Our experiments show that it is not enough to only use the anisotropic chartification without modifying the parameterization weights (using exaggerated mesh for computing the parameterization effectively changes the coefficients used for the mean-value parameterization, as described in Section 5). Fig. 10 shows the comparison of using the anisotropic chartification followed by a compatible anisotropic parameterization versus using the same initial tile placement followed by the isotropic parametrization using the coefficients of the original mesh. We see that the isotropic relaxation step pulls the samples away from the extended feature, and results in an undersampled area on a high curvature part of the surface. This affects the quality of the resulting remesh. A similar example is shown for the bunny model (see Fig. 7); again the comparison shows that the anisotropic modification allows to stretch the triangles along the edge of the bunny ear. The error comparison plot shows a much better

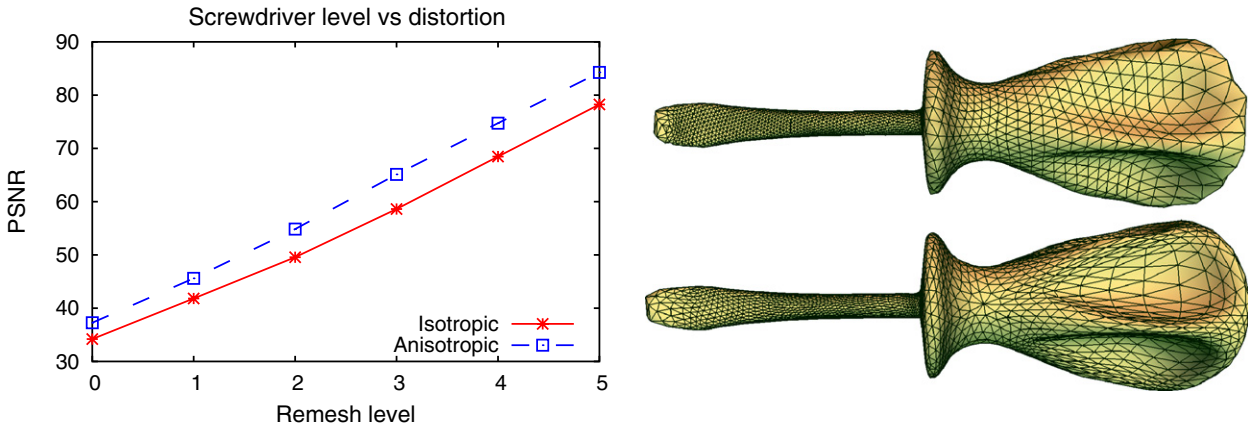


Fig. 8. Screwdriver remesh approximation. Left: distortion plot; right: isotropic(top) and anisotropic(bottom) remesh at level 3 with the same polygon count. Note that the isotropic remesh will compress to a smaller size archive. Both remeshes were obtained without user intervention.

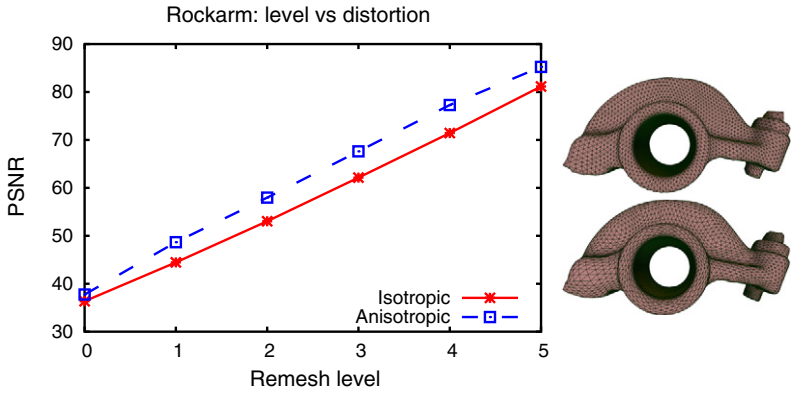


Fig. 9. Rockerarm remesh approximation. Left: distortion plot; right: isotropic(top) and anisotropic(bottom) remesh at level 3 with the same polygon count. Both remeshes were obtained without user intervention.

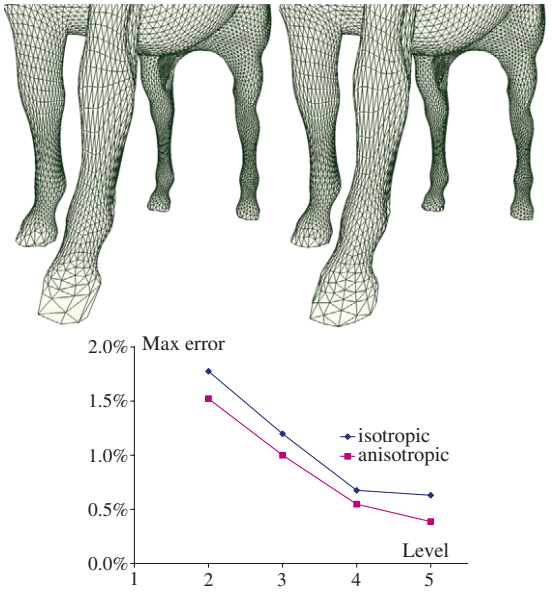


Fig. 10. Two meshes produced from the same anisotropic tiling. Top: using the isotropic parameterization; Bottom: using anisotropic parameterization. The plot of maximal error measured by Metro tool is also shown.

approximation for the coarser levels of the semi-regular remesh.

This effect of pulling the sampling away from extended features is a well-known problem in surface remeshing. In order to produce acceptable results during parametric relaxation in [14,15] it is necessary for the user to intervene and fix the global vertices at the ends of the extended features such as legs of the horse and horns of the feline models. Our approach achieves the same effect without any user intervention.

Fig. 11 shows a failure of our algorithm to fully capture the sharp features of the Fandisk model. Note that while the parameterization is trying to align along creases, it is not enough to produce a good quality remesh. The most troublesome places are the corners of the model where three conflicting directions meet. It should be possible to extend boundary handling functionality of our method to be able to incorporate strong constraints.

We have explored the influence of the differential structure mapping parameter h on the resulting semi-regular meshes. Fig. 13 shows the distribution of the roundness and the areas of output mesh triangles for $h = 0$ and $h = 1$ for the skull model (see Fig. 5). The case of $h = 0$ uses the conformal map to unwrap the one-ring of each base vertex onto a plane. By preserving the angles during the mapping the conformal polar map can introduce smaller triangles in the neighborhoods of vertices of valence below six, and bigger triangles near extraordinary vertices of large valence. This is confirmed by the area histogram. The value of parameter $h = 1$ on the other hand makes for a good distribution of triangle areas, but may sacrifice their roundness in order to accommodate extraordinary vertices into the mesh. Based on what is needed by a particular application the user may choose different settings of h. In our examples, we defaulted to $h = 1$ as it makes for more uniform-looking remeshes (Fig. 14).

7. Conclusions and future work

We have introduced a manifold-based method for semi-regular remeshing, as well as a simple extension for controlling the anisotropy of the output meshes. Our method treats all the vertices in the

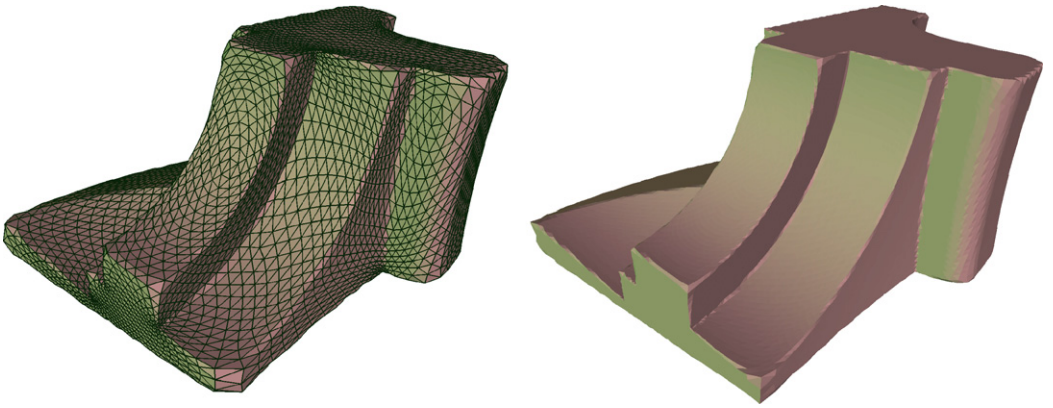


Fig. 11. An example showing a failure mode of our method: the sharp features cannot be handled by our parameterization adjustment. Left: level 4 of the anisotropic remesh; right: level 5 of the anisotropic remesh shaded.

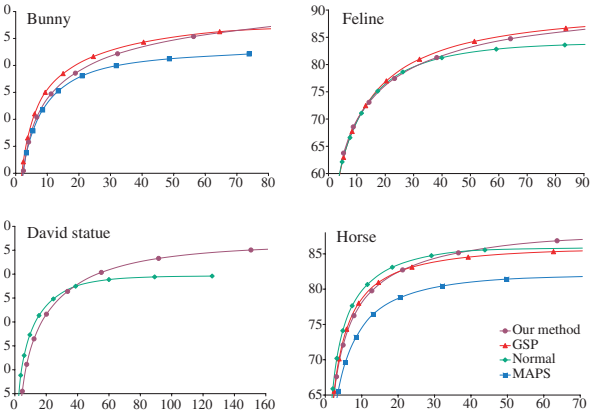


Fig. 12. Comparison of compression results. All the remeshes were compressed with Butterfly transform of the PGC package. See text for further explanations.

input mesh as equally important and the produced parameterization is truly global. The method is easy to implement and require almost no user intervention except for the choice of the base domain complexity.

There remains a lot of directions for future research. First of all, the proposed global parametric energy functional is very simple; it remains to be seen whether one can modify it in such a way that the produced parametric mapping is provably a bijection. The current optimizaion procedure uses an off-the-shelf implementation of the non-quadratic optimization, and should be made much faster using a multigrid-like approach. It is also important to analyze which exaggeration technique would give the best control over the anisotropic properties of the produced parameterization.

Acknowledgments

This work was supported in part by NSF CAREER award (CCR-0133554), and by the NSF ERC-RMS (award EEC-9529125). Datasets are courtesy Cyberware, Headus, The Scripps Research Institute, University of Washington, and the Digital Michelangelo project at Stanford University [23].

Appendix A. Gradient of the functional

The functional $F(y)$ that is optimized in each edge-based diamond shaped region can be written as follows:

$$F(\mathbf{y}) = \sum_{v \in \breve{\Lambda}_e} \sum_{k=1}^{4} \sigma(v) E_v^k(\mathbf{y}),$$

where

$$E_v^k(\mathbf{y}) = W_v^k(y(v)) \left(\zeta_k(y(v)) - \sum_{v' \in \omega_1(v)} a_{vv'} \zeta_k(y(v')) \right)^2;$$

and $W_v^k(y) = \beta(|y - \xi(b_k)|)$. Note that positions $\xi(b_k)$ are the corners of the diamond in the y-plane, defined in Section 4.3. Also, note that $\mathbf{y} = (y_m(v))_{v \in \breve{\Lambda}_e}$ can be treated as a $2N$-dimensional vector (if the region $\breve{\Lambda}_e$ contains N vertices). In order to take the gradient, it is enough to compute the partial derivative $\partial E_v^k / \partial y^m(\bar{v})$ for each $\bar{v} \in \breve{\Lambda}_e$ and $m = 1, 2$:

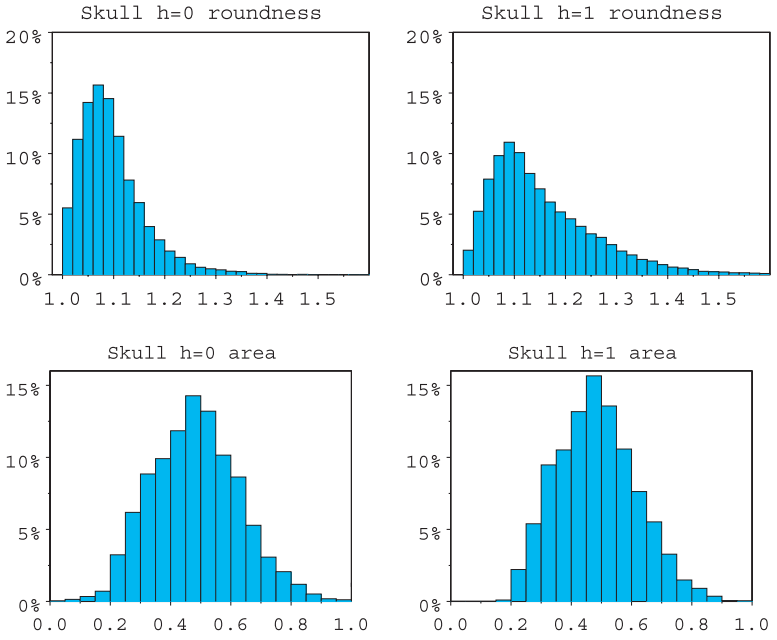


Fig. 13. The roundness and the triangle area distributions shown for two remeshes from Fig. 5. It is clear that while the left remesh ($h = 0$) contains some very small triangles, it also has better roundness properties due to its conformal properties.

Fig. 14. David's statue remesh (level 4).

$$\partial E_v^k/\partial y^m(\bar{v}) = \begin{cases} \partial W_v^k/\partial y^m|_{y(\bar{v})} & \\ \quad \times \Big(\zeta_k(y(v)) - \sum\limits_{v' \in \omega_1(v)} a_{vv'} \zeta_k(y(v')).\Big)^2 & \\ \quad + 2W_v^k(y(v))\langle \partial\zeta_k/\partial y^m|_{y(v)} \zeta_k(y(v)) & \\ \quad - \sum\limits_{v' \in \omega_1(v)} a_{vv'} \zeta_k(y(v'))\rangle & \text{if } \bar{v} = v; \\ -2a_{v\bar{v}} W_v^k(y(v))\langle \partial\zeta_k/\partial y^m|_{y(v)} \zeta_k(y(v)) & \\ \quad - \sum\limits_{v' \in \omega_1(v)} a_{vv'} \zeta_k(y(v'))\rangle & \text{if } \bar{v} \neq v, \bar{v} \in \omega_1(v); \\ 0 & \text{otherwise.} \end{cases}$$

Now, denoting $s_k = \xi(b_k)$, we can write

$$\frac{\partial W_v^k}{\partial y^m} = \frac{\partial \beta(|y - s_k|)}{\partial y^m} = \frac{y^m - s_k^m}{|y - s_k|} \beta'(|y - s_k|);$$

here $\beta(r)$ is a spline blending function previously defined. Its derivative for completeness is given as

$$\beta'(r) = \begin{cases} 0 & \text{for } r < \Delta_0 \text{ or } r > \Delta_1; \\ 6\frac{(r-\Delta_0)(r-\Delta_1)}{(\Delta_1-\Delta_0)^3} & \text{for } \Delta_0 \leqslant r \leqslant \Delta_1. \end{cases}$$

In order to compute the partial derivatives of the function $\zeta_k(y)$ we use the chain rule on the following representation:

$$\zeta_k(y) = H_k(v_k(y^1 + \mathrm{i}y^2)).$$

One of these functions is analytic and given by $v_k(z) = (A_k + B_k z)^{\alpha_k}$ where $\alpha_k = 6/\mathrm{val}(b_k)$. Its (complex) derivative is equal to $v_k'(z) = A_k\alpha_k(A_k z + B_k)^{\alpha_k - 1}$. The constants for the four corners of the diamonds are given as follows:

$$A_1 = 1, \quad B_1 = 1/2,$$

$$A_2 = -1, \quad B_2 = 1/2,$$

$$A_3 = \mathrm{i}, \quad B_3 = \sqrt{3}/2,$$

$$A_4 = -\mathrm{i}, \quad B_4 = -\sqrt{3}/2.$$

The function $H_k(z) = (z_1|z|^{-q_k}, z_2|z|^{-q_k})$, where $q = (\alpha_k - 1)h/\alpha_k$, is used to control the behavior of

parameterization near global vertices. Its partial derivatives can be easily computed as follows:

$$\frac{\partial H_k^1}{\partial z_1} = |z|^{-q_k}\left(1 - q_k \frac{z_1^2}{|z|^2}\right),$$

$$\frac{\partial H_k^1}{\partial z_2} = \frac{\partial H_k^2}{\partial z_1} = -q_k z_1 z_2 |z|^{-2-q_k},$$

$$\frac{\partial H_k^2}{\partial z_2} = |z|^{-q_k}\left(1 - q_k \frac{z_2^2}{|z|^2}\right).$$

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, M. Desbrun, Anisotropic polygonal remeshing, ACM Trans. Graph. 22 (3) (2003) 485–493.

[2] L. Balmelli, C.J. Morris, G. Taubin, F. Bernardini, Volume warping for adaptive isosurface extraction, in: VIS '02: Proceedings of the Conference on Visualization '02, 2002, pp. 467–474.

[3] I. Boier-Martin, H. Rushmeier, J. Jin, Parameterization of triangle meshes over quadrilateral domains, in: SGP '04: Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry Processing, 2004, pp. 193–203.

[4] H. Borouchaki, P. Frey, Adaptive triangular-quadrilateral mesh generation, Intl. J. Numer. Methods Eng. 41 (1998) 915–934.

[5] P. Cignoni, C. Rocchini, R. Scopigno, Metro: Measuring error on simplified surfaces, Computer Graphics Forum 17 (2) (1998) 167–174.

[6] D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, ACM Trans. Graph. 23 (3) (2004) 905–914.

[7] M. Desbrun, M. Meyer, P. Schröder, A.H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, Proceedings of SIGGRAPH (1999) 317–324.

[8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle, Multiresolution analysis of arbitrary meshes, Proceedings of SIGGRAPH (1995) 173–182.

[9] M.S. Floater, Mean value coordinates, Comput. Aided Geom. Des. 20 (1) (2003) 19–27.

[10] S. Gallot, D. Hulin, J. Lafontaine, Riemannian Geometry, Springer-Verlag, 1993.

[11] C.M. Grimm, Simple manifolds for surface modeling and parameterization (figures 3 and 6), in: SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02), IEEE Computer Society, 2002, p. 277.

[12] C.M. Grimm, J.F. Hughes, Modeling surfaces of arbitrary topology using manifolds, in: SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, 1995, pp. 359–368.

[13] X. Gu, S.J. Gortler, H. Hoppe, Geometry images, in: SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, 2002, pp. 355–361.

[14] I. Guskov, K. Vidimče, W. Sweldens, P. Schröder, Normal meshes, Proceedings of SIGGRAPH (2000) 95–102.

[15] A. Khodakovsky, N. Litke, P. Schröder, Globally smooth parameterizations with low distortion, Proceedings of SIGGRAPH (2003) 350–357.

[16] A. Khodakovsky, P. Schröder, W. Sweldens, Progressive geometry compression, Proceedings of SIGGRAPH (2000) 271–278.

[17] A. Klein, A. Certain, T. DeRose, T. Duchamp, W. Stuetzle, Vertex-based Delaunay triangulation of meshes of arbitrary topological type, Tech. Rep., University of Washington, 1997.

[18] L. Kobbelt, S. Campagna, H.-P. Seidel, A general framework for mesh decimation, in: Proceedings of the Graphics Interface Conference '98, 1998.

[19] V. Kraevoy, A. Sheffer, Cross-parameterization and compatible remeshing of 3D models, ACM Trans. Graph. 23 (3) (2004) 861–869.

[20] F. Labelle, J.R. Shewchuk, Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation, in: SCG '03: Proceedings of the Nineteenth Annual Symposium on Computational Geometry, 2003, pp. 191–200.

[21] A.W.F. Lee, D. Dobkin, W. Sweldens, P. Schröder, Multiresolution mesh morphing, in: SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, 1999, pp. 343–350.

[22] A.W.F. Lee, W. Sweldens, P. Schröder, L. Cowsar, D. Dobkin, Maps: Multiresolution adaptive parameterization of surfaces, Proceedings of SIGGRAPH (1998) 95–104.

[23] M. Levoy, The Digital Michelangelo project, in: Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling (Ottawa, October 1999).

[24] S. Lloyd, Least square quantization in PCM, IEEE Trans. Inform. Theory 28 (1982).

[25] E. Praun, H. Hoppe, Spherical parametrization and remeshing, ACM Trans. Graph. 22 (3) (2003) 340–349.

[26] E. Praun, W. Sweldens, P. Schröder, Consistent mesh parameterizations, in: SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 2001, pp. 179–184.

[27] J. Ruppert, A Delaunay refinement algorithm for quality 2-dimensional mesh generation, in: SODA '93: Selected Papers from the Fourth Annual ACM SIAM Symposium on Discrete Algorithms, Academic Press, Inc., 1995, pp. 548–585.

[28] P.V. Sander, S.J. Gortler, J. Snyder, H. Hoppe, Signal-specialized parametrization, in: EGRW '02: Proceedings of the 13th Eurographics Workshop on Rendering, 2002, pp. 87–98.

[29] P.V. Sander, Z.J. Wood, S.J. Gortler, J. Snyder, H. Hoppe, Multi-chart geometry images, in: SGP '03: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, Eurographics Association, 2003, pp. 146–155.

[30] J. Schreiner, A. Asirvatham, E. Praun, H. Hoppe, Inter-surface mapping, ACM Trans. Graph. 23 (3) (2004) 870–877.

[31] K. Shimada, Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles, in: 6th Intl. Meshing Roundtable, 1996, pp. 63–74.

[32] R.B. Simpson, Anisotropic mesh transformations and optimal error control, in: Proceedings of the Third ARO Workshop on Adaptive Methods for Partial Differential Equations (New York, NY, USA, 1994), Elsevier North-Holland, Inc., 1994, pp. 183–198.

[33] M. Spivak, A Comprehensive Introduction to Differential Geometry, second ed., Publish or Perish, 1979. In 5 volumes.
[34] Z. Wood, H. Hoppe, M. Desbrun, P. Schröder, Removing excess topology from isosurfaces, ACM Trans. Graph. 23 (2) (2004) 190–208.
[35] L. Ying, D. Zorin, A simple manifold-based construction of surfaces of arbitrary smoothness, ACM Trans. Graph. 23 (3) (2004) 271–275.
[36] C. Zhu, R.H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, ACM Trans. Math. Softw. 23 (4) (1997) 550–560.