Eurographics Symposium on Geometry Processing (2004)
R. Scopigno, D. Zorin (Editors)

Parameterization of Triangle Meshes over Quadrilateral Domains

Ioana Boier-Martin[1], Holly Rushmeier[1], and Jingyi Jin[2]

[1] IBM T. J. Watson Research Center, [2] Univ. of Illinois at Urbana-Champaign,

Abstract

We present a method for parameterizing irregularly triangulated input models over polyhedral domains with quadrilateral faces. A combination of center-based clustering techniques is used to generate a partition of the model into regions suitable for remeshing. Several issues are addressed: the size and shape of the regions, their positioning with respect to features of the input geometry, and the amount of distortion introduced by approximating each region with a coarse polygon. Region boundaries are used to define a coarse polygonal mesh which is quadrangulated to obtain a parameterization domain. Constraints can be optionally imposed to enforce a strict correspondence between input and output features. We use the parameterization for multiresolution Catmull-Clark remeshing and we illustrate two applications that take advantage of the resulting representation: interactive model editing and texture mapping.

1. Introduction

Polygon meshes are often used to represent three-dimensional models for rendering and computation purposes. Triangle meshes are common, but quadrilateral meshes are preferred by some classes of applications. Due to their tensor-product nature, quadrilaterals are favored for use in subdivision schemes and as parameterization domains for spline-based representations. They are used in Computer Graphics for surface design and animation [47] and in Engineering Analysis for finite element computations [39].

Meshes with regular or semi-regular (i.e., subdivision) connectivity have proven particularly useful for applications that require efficient processing of the input data, such as interactive modeling or compression for storage and transmission. Their parametric and hierarchical nature makes them ideal for editing and for encoding shapes at multiple resolutions. In particular, Catmull-Clark meshes [10] have been successfully included in commercial modelers such as Maya [1], 3D StudioMax [2], and Catia [3]. Unfortunately, few meshes generated in practice are semi-regular and the lack of reliable conversion tools has been a notable impediment to their widespread use.

In this paper, we propose a novel method for automatic generation of quadrilateral parameterization domains for arbitrary 2-manifold triangulations and for computing semi-regular representations over these domains. We exploit results from clustering analysis and optimal quantization theory to segment meshes into locally congruent patches suitable for quadrangulation.

1.1. Overview and Contributions

Given an input triangle mesh, our technique generates a multiresolution Catmull-Clark [10] subdivision hierarchy that approximates it. A coarse control mesh is first computed and serves as a domain over which the input model is parameterized. Multiresolution analysis is applied to populate the intermediate levels of the hierarchy.

Our main contribution is providing an **automatic** method for extracting a **high-quality quadrilateral** parameterization domain. The quality is achieved through predominantly **regular connectivity** at domain vertices and control over the **placement of faces** with respect to the input geometry. Our method operates **directly** on the 3D model, requiring no prior parameterization. This makes it **general** and readily applicable to complex meshes of **arbitrary topology**. The parameterization domain is used for **multiresolution remeshing**. We use a combination of clustering algorithms to partition the mesh into regions suitable for remeshing. We introduce a novel way of **sparsely sampling** the input model with sites around which the regions are built. **Constraints** may be optionally imposed to ensure a strict correspondence between new mesh edges and input elements. The final result is a hierarchy of progressively finer semi-regular approximations of the input model.

1.2. Related Work

Considerable literature exists on building high-quality parameterizations over triangulated base domains (see [28] for a recent example and references). In contrast, very little work has been done on deriving quadrilateral base complexes for arbitrary meshes. Krishnamurthy and Levoy [30] fit tensor-product

B-spline patches to irregular meshes. Guskov et al. [21] propose a scheme for building quad base domains as part of their hybrid mesh representation. Both methods require some degree of user intervention in defining the base domain. User-guided parameterization facilitates domain conformance to features. However, for parameterization of entire model databases, automatic schemes are desirable. Eck et al. [15] describe an automatic method for fitting B-splines to meshes of arbitrary topology. A quad base domain is generated indirectly from a triangulated one by simplification followed by pairing of neighboring triangles. In general, approaches based on mesh simplification suffer from several shortcomings: it is not clear how to determine when to stop the simplification, geometric error typically drives the process with little or no control over the resulting topology and connectivity (see Fig. 9), the simplified mesh is a triangle mesh for which a quadrilateral decomposition has to be found, and constraints are difficult to enforce [27]. Generating quadrilateral meshes by pairing of triangles (see also $4-8$ subdivision schemes [46]) pose additional problems: complete pairings may not always exist and finding ones that minimize distortion is expensive.

In [9] quad semi-regular meshes are obtained by projecting faces of octree cells onto the input surface. This tends to create base meshes with a large number of faces, even after cleanup. Also, there is no control over the placement of faces. Arbitrary meshes are typically parameterized by cutting and flattening. An extreme example is the parameterization of an entire mesh over a square [19]. Maintaining consistency across seams is difficult, especially if the model is to be modified (e.g., edited, compressed). In [20] conformal parameterizations of complex surfaces are computed without cutting. However, the resulting parameterizations are highly non-uniform and controlling them requires manual topology modification. Recent methods have targeted restricted classes of models. Mappings to either a sphere [40] or a plane [23] were used to recover quadrilateral meshes for genus 0 models with and without boundary, respectively. For the latter class, an interesting quad-dominant anisotropic remeshing method was described in [4]. Techniques for converting given models to quadrilateral meshes have also been proposed in the mesh generation community. Advancing front and packing are among the most common strategies [39, 7]). Typically, the resulting meshes have a large number of quads and are not suitable as parameterization domains.

Also related to our approach are mesh partitioning methods. The fuzzy clustering technique of [26] produces patches which are not homeomorphic to disks and cannot be directly used for parameterization. Other methods generate disk-like patches according to various criteria (e.g., [17, 43, 32, 44]), but no single method addresses the combination of patch shape, distrortion, number of neighbors, alignment to features, and constraints.

Surazhsky et al. [45] use centroidal Voronoi tessellations to generate dense isotropic triangulations. The centroid updates are performed in 2D and require computing local parameterizations of model regions. This is a remeshing approach which does not produce a parameterization domain for the input model.

2. Basic Concepts

2.1. Problem Definition

Our input representation is a 2-manifold triangle mesh M_I of arbitrary topology, possibly with boundaries. Feature curves along edges of M_I may be tagged as constraints. Such curves may be specified through user input or as a result of an automatic detection procedure.

Our target representation is a multiresolution subdivision surface defined by a control mesh hierarchy with L levels $M_{H_0}, \cdots, M_{H_{L-1}}$ such that:

1. M_{H_0} is a coarse quadrilateral mesh with a predominant number of regular control points (i.e., valence 4 in the interior and valence 2 on the boundary; we define the *valence* of a control point as the number of faces adjacent to it).
2. $M_{H_{L-1}}$ is a fine mesh such that its control points (or alternatively, their projections onto the limit surface of subdivision) are located on the input mesh M_I.
3. Each mesh M_{H_i} is obtained from the coarser mesh $M_{H_{i-1}}$ by applying the Catmull-Clark [10] subdivision rules, for $i = 1, \cdots, L-1$. To accurately capture the input shape, the positions of control points on each level may be perturbed from the locations computed by subdivision using multiresolution detail vectors.

2.2. Method Overview

Our method converts the input to the target representation in several steps (see Fig. 1):

1. **Normal-based clustering:** An initial classification of input mesh faces into approximately flat regions / clusters is obtained through center-based clustering of normals. The shape of these regions is analyzed. If all satisfy a shape criterion (see implementation details in section 6), the algorithm proceeds to step 4 to extract a coarse mesh (this is the case in Fig. 10).
2. **Cluster refinement:** In most cases, the initial partition is too coarse and not all regions have a shape suitable for remeshing. We use this partition to guide the placement of samples onto the model and we compute a constrained *spatial* tessellation. The latter leads to a decomposition of the mesh into smaller, better shaped clusters.
3. **Cluster cleanup:** At this point most clusters are suitable for mesh extraction. Due to sparse sampling (see Fig 9) in the previous stage, some may still fail to satisfy all our shape constraints. Such clusters are split using *local* normal and spatial decompositions.
4. **Coarse mesh extraction:** Cluster boundaries are used to create a polygonal mesh with edges along boundaries and faces approximating the clusters. Short edges are selectively collapsed to produce a cleaner mesh.
5. **Quadrangulation:** The coarse polygonal mesh obtained in the previous step is converted to a quadrilateral mesh which is used as a base domain over which the input mesh is parameterized.
6. **Resampling and analysis:** To obtain the final semi-regular representation, we adopt a traditional approach for resampling at dyadic positions in parameter space followed by multiresolution analysis.

2.3. Data Clustering

Classification is an essential tool for data exploration and analysis. *Clustering* (i.e., unsupervised classification) groups items into distinct clusters based on similarity and proximity metrics [25]. *Center-based clustering* methods stand out for two important reasons: low complexity and a clearly defined objective function being minimized. They proceed iteratively, alternating two steps: (a) computing a Voronoi partition of the data around a set of *centers* (i.e., a region is formed around each center such that data in that region is closer to its center than to any of the other centers); and (b) updating the centers to better represent the newly formed regions, until some convergence criterion is satisfied.

A class of center-based clustering methods known as *Centroidal Voronoi Tessellations* (CVT) has emerged as offering many advantages over ordinary Voronoi tessellations [13]. We focus on their restriction to surfaces embedded in Euclidean space, the so-called *Constrained Centroidal Voronoi Tessellations* (CCVT) [14] to create decompositions suitable for remeshing.

Given a bounded domain $\Omega \subset R^n$ and a set of K sample points $\{c_i\}_{i=1}^K \in \Omega$, the *Voronoi partition* or *tessellation* $\mathcal{V}$ induced by $\{c_i\}_{i=1}^K$ on Ω is defined as:

$$\mathcal{V} = \{V_i = \{x \in \Omega : |x - c_i| < |x - c_j|, j = 1, \cdots, K, j \neq i\}, \\ i = 1, \cdots, K\}$$

The points $\{c_i\}_{i=1}^K$ are referred to as *generators* of the corresponding Voronoi regions V_i.

Definition 1. A *centroidal Voronoi tessellation* (CVT) is a Voronoi tessellation in which the centroids (i.e., centers of mass) of the regions serve as their generators.

CVTs are closely related to statistical clustering algorithms. For discrete data sets, CVTs can be identified with K-means clustering methods [13]. This concept can be adapted to produce CVTs over arbitrary surfaces, by restricting $\Omega = S \subset R^n$ to be a compact continuous surface and letting $\{c_i\}_{i=1}^K \in S$ be a set of sample points on it [14].

Definition 2. A *constrained centroidal Voronoi tessellation* (CCVT) is a Voronoi tessellation for which the *constrained mass centroid* c_i^* of each region V_i serves as its generator. The constrained mass centroid of a region $V_i \in S$ is defined as:

$$c_i^* = argmin_{c \in S} F_i(c), \text{ where } F_i(c) = \int_{V_i} \rho(x)|x - c|^2 dx$$

where $\rho(x)$ is a given density function over Ω. For the purpose of surface decomposition, we note several key properties (see [13, 14, 37] for details):

(P1) For $\Omega \subset R^2$, the average number of Voronoi edges per Voronoi region does not exceed six.
(P2) For $\Omega \subset R^2$, CVT Voronoi regions become locally congruent hexagons as the number of generators increases.
(P3) The computation of the CCVT is based on Euclidean distances between points, which makes it considerably more efficient in practice than the geodesic distance-based CVT.
(P4) The vector defined by the center of mass of a region and its constrained centroid c_i^* is normal to S at c_i^* (i.e., c_i^* is the projection of the center of mass onto the surface along normal direction).

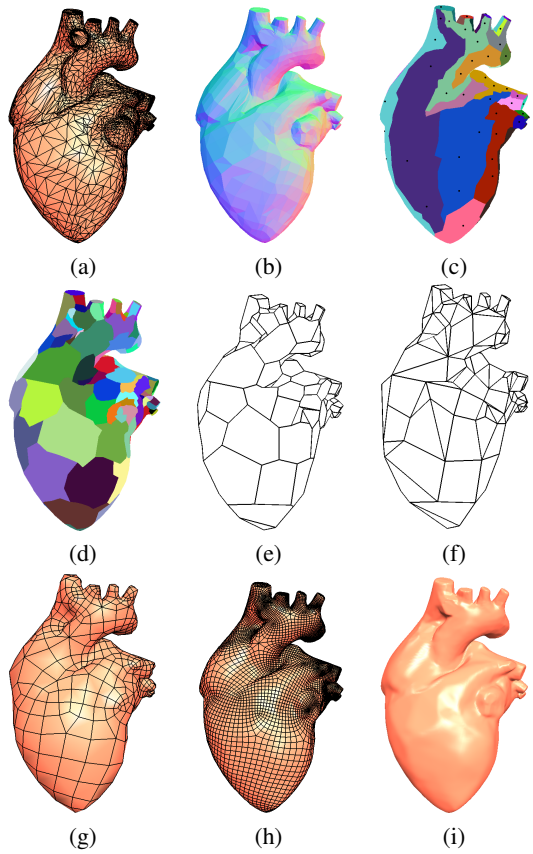
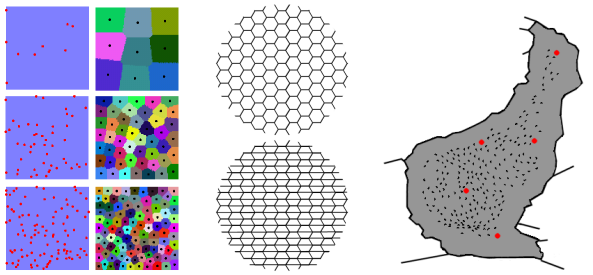


Figure 1: *Main steps: (a) Input. (b) Variation of normals across the model. (c) Normal-based clustering and initial placement of generators (black dots). (d) Final partition after spatial-based clustering (using the generators from (c)) and cluster refinement. (e) Coarse polyhedral approximation extracted from the partition in (d). (f) Quadrangulated coarse mesh. (g)-(h) Coarsest and finest levels of a Catmull-Clark hierarchy with 3 levels obtained after resampling and multiresolution analysis. (i) Finest level smooth shaded.*

(P2) stems from *Gersho's conjecture* [18], a more general result in quantization theory. It states that, as the number of generators increases, the regions of an optimal quantizer are locally congruent polytopes in R^n. For planar domains, the conjecture was proven and the optimal polytope is the regular hexagon.

Our key idea is to segment a given 2-manifold into regions of reduced normal variation (i.e., almost flat) and to perform CCVT starting from generators placed in these regions. This allows us to exploit (P1) and (P2) over the model to obtain a predominantly hexagonal partition from which a quad-dominant mesh can be extracted (see Fig. 2a, b). Furthermore, (P3) and (P4) allow us to do so efficiently using Euclidean distance computations and observing that the CCVT of an almost flat region reduces to a CVT in Euclidean space. Thus, we can detect regions of reduced normal variation and we can compute centroidal partitions without having to perform expensive centroid projections onto the mesh.

The remaining definitions pertain to quadrangulations of spatial polygons (see [35]).



(a) (b) (c)

Figure 2: *(a) CVT partitions of a planar mesh starting with randomly placed generators. Left: initial generators (red points). Right: mesh partition and final positions of the generators (black points). Top to bottom: 9, 49, and 100 generators. (b) A quad mesh with regular connectivity is obtained by quadrangulating a regular hexagonal mesh. (c) Region sampling cf. Algorithm 2; red points indicate selected samples.*

Definition 3. A *conformal decomposition* of a polygon is a partition of its enclosed area into strictly convex quadrilaterals (i.e., with all interior angles less than π), such that any two quads that share more than one point share exactly one edge. If the quadrilaterals are all convex, but not strictly convex, we call the decomposition *quasi-conformal*.

Definition 4. A *perfect decomposition* is a (quasi-)conformal decomposition with no internal vertices.

Definition 5. A *partial (quasi-)conformal decomposition* is a (quasi-)conformal decomposition in which all but one face are quadrilaterals.

3. Center-Based Parameterization

In this section we describe the process of extracting a coarse polyhedral approximation for a given input mesh. First, normal-based and spatial clustering methods are combined to produce a global partition of the input mesh. The resulting regions are further refined to satisfy shape and flatness requirements using intra-region clustering.

For clustering, we associate a data item (e.g., normal or location information) with each triangular face of the input mesh. We compute centroidal decompositions based on the data items using discrete versions of the methods defined in section 2.3 in a MacQueen-type approach [34]:

Algorithm 1 (center-based clustering of mesh faces):

Given an input mesh M with F faces,
per-face data items $\{d_f\}_{f=1}^{F}$,
and an initial set of K generator items $\{c_i\}_{i=1}^{K}$:
Repeat
  For each face f of M do
    1. Find c_{i^*} among $\{c_i\}_{i=1}^{K}$ closest to d_f
    2. $j = Label(f)$, $Label(f) = i^*$
    3. $c_{i^*} \leftarrow (|C_{i^*}|c_{i^*} + d_f)/(|C_{i^*}|+1)$,
       $c_j \leftarrow (|C_j|c_j - d_f)/(|C_j|-1)$,
until (*convergence*)
($|C|$ denotes the number of faces of cluster C; the label of each face is the index of the cluster to which it belongs)

A mesh partition is suitable for remeshing if it satisfies the following requirements:

(R1) Each region is homeomorphic to a disk.
(R2) The closed piecewise linear curve defining the boundary of each region can be approximated within some tolerance by a convex polygon.
(R3) For each region there exists a direction $\overrightarrow{h}$ in space such that the corresponding geometry defines a height-field along $\overrightarrow{h}$ and can be approximated within some tolerance by a plane perpendicular to $\overrightarrow{h}$.

(R1) is needed if mesh faces are to correspond to regions and (R2) ensures that they are well-shaped. (R3) is a stricter requirement than necessary. It would be possible for a face to subtend geometry which is not a height field over that face. A mapping-dependent distortion is likely to be introduced when the mesh geometry is parameterized onto the face. To reduce the amount of distortion and to avoid resampling problems, we enforce (R3) (see also our texture-mapping requirements in section 6).

3.1. Normal-Based Clustering

Since we would like coarse mesh faces to correspond to relatively flat regions of the input mesh, we begin with a first pass which identifies flat regions. Face normals are computed and Algorithm 1 is applied with $d_f = normal(f)$, for all input faces f. The initial generators are chosen to be a small fixed subset of the set of all possible unit normal vectors (those pointing from the origin to the vertices, mid-edges, and face centers of a cube centered at the origin). To reduce the influence of geometry discretization, the normals are smoothed prior to classification. The result is a partition $\mathcal{P}_{\mathcal{N}}$ of the input mesh into regions of reduced normal variation. This process is illustrated in Fig. 1b. The variation of normals over the heart model is shown using a linear mapping of normal vector components to RGB color. The resulting mesh decomposition is shown in Fig. 1c.

For smooth, relatively simple shapes, $\mathcal{P}_{\mathcal{N}}$ has nice regions which are suitable for remeshing (Fig. 10). However, this cannot be guaranteed. Since only a small number of regions are generated at this step, we can quickly check if all of them can be approximated by convex polygons. If this is not the case, we use the resulting decomposition for further refinement, as described next. In the interest of clarity, we defer the description of the shape check to section 6.

3.2. Cluster Refinement

In contrast to the normal-based partitioning scheme just described, spatial CCVT-type decompositions of the mesh are guaranteed to produce almost circular regions centered around their generators. The main issue to be addressed is choosing the number of generators and their initial locations.

Random sampling. A common strategy to select initial generators is to place random samples over the mesh. Two things must be specified: the number of samples and their distribution. For our problem, it is difficult to estimate in advance how many generators to use: too many lead to base meshes with many faces, whereas too few fail to properly represent the input. Given a number of samples, their placement can be controlled using Monte Carlo methods or an error diffusion strategy [5], both of which can be computationally expensive for arbitrary models.

Region-based sampling. As an alternative, we use $\mathcal{P}_\mathcal{N}$ to estimate both the number of generators needed and their initial locations. Since regions of $\mathcal{P}_\mathcal{N}$ correspond to relatively flat model parts, the idea is to produce a refined decomposition in which clusters are centered on such flat spots. Ideally, we want to place samples along the medial axis of each region. In practice, we use the following heuristic method to avoid medial axis computation:

Algorithm 2 (region sampling):

Given a mesh region R with F_R faces:
1. Identify the set of boundary faces B
2. Uniformly distribute a set S_R of N random samples on $R \setminus B$
3. Select sample $s_1 \in S_R$ with the largest minimum distance to B
4. for $i = 2, \cdots, N$ do
5. Select $s_i \in S_R$ farthest from $s_1, \cdots, s_{i-1}$
6. Retain N_0 of the N selected samples

We exclude faces on the boundary of the region to keep the samples in its interior (Fig. 2c). Similar to [26], we consider the function defined by the minimum distance of a sample from previously selected samples. The value of N_0 is chosen to maximize the first derivative of this function (see implementation details in section 6). The set of sample points used as generators for the entire model is the union of the region samples.

Spatial partitioning. After generators have been placed on the mesh according to Algorithm 2, the initial partition $\mathcal{P}_\mathcal{N}$ is discarded (i.e., we use the global normal-based partition only for automatic placement of generators). A new partition $\mathcal{P}_\mathcal{S}$ is computed with Algorithm 1, this time based on spatial information. We classify mesh faces according to their proximity (measured using Euclidean distance) to generators. We use $d_f = barycenter(f)$, for all input faces f.

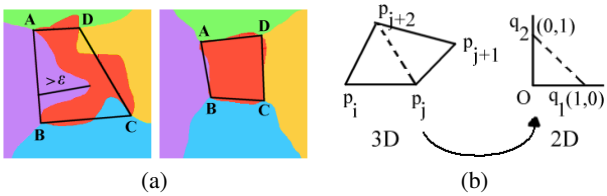


Figure 3: *(a) Shape test: the boundary of the red region is approximated with a polygon with vertices at points where 3 or more clusters meet. Left: ABCD is not a good approximation for the red region. Right: ABCD is an acceptable approximation as it passes both the convexity and distance-to-region tests. (b) Mapping of a quad corner to a right triangle.*

3.3. Cluster Cleanup

By construction, clusters obtained after the refinement stage are isotropic, almost circular in shape. If the generators are sufficiently dense, (R1) is guaranteed to be satisfied by all clusters [6]. However, since we use sparse generator sets, some clusters may not conform to (R1) (see Fig. 9)a. The purpose of the cleanup phase is to enforce (R1), (R2), and (R3) on all clusters. For this, we treat the geometry of each region of $\mathcal{P}_\mathcal{S}$ individually. The height-field condition (R3) is checked first and if violated, the region is split using normal-based clustering. If any of the resulting subregions does not satisfy (R1) or (R2), the subregion is further split using spatial clustering. The normal-based split guarantees that the resulting regions are height fields. Subsequent spatial decompositions ensure that each height field is decomposed into regions with disk-topology and of approximately circular shape. Fig. 1d shows the final decomposition of the heart model.

3.4. Coarse Mesh Extraction

Having found a partition of the input model that satisfies all of our requirements, the algorithm proceeds to generate a coarse mesh corresponding to it. Mesh vertices are placed at the points where three or more regions meet (empty space counts as a region when boundaries are present in the input mesh). Boundaries between regions define the edges of the coarse mesh. Edges due to discretization errors during clustering are collapsed (Fig. 10c). The result of this step is illustrated in Fig. 1e and Fig. 10d.

4. Resampling and Multiresolution Analysis

Having found a coarse polygon mesh M_B that approximates the input model, we build a Catmull-Clark multiresolution representation by parameterizing the model over M_B.

4.1. Polygonal Mesh Quadrangulation

The faces of M_B correspond to regions that satisfy requirements (R1)-(R3). Also, according to (P2), many faces are hexagonal. Brute-force application of a Catmull-Clark subdivision step leads to a quadrilateral mesh with a high number of irregular (valence 6) vertices. Such meshes are not desirable in practice. Instead, we aim to find a minimum perfect conformal decomposition by splitting faces into quads along face diagonals. For a regular hexagonal mesh, this strategy leads to a quadrilateral mesh with regular connectivity (Fig. 2b).

In general, the coarse mesh has faces other than hexagonal. A polygon admits a conformal decomposition if and only if it has an even number of vertices, but the problem of finding a minimum such decomposition for a polygon mesh is NP-hard [35]. However, since the faces of M_B are convex polygons by construction, we can always find a partial quasi-conformal decomposition as follows: split each face with $2k$ edges into $k-1$ quads; split each face with $2k+1$ edges into $k-1$ quads and 1 triangle. We apply the decomposition recursively: first we find the quad with the lowest shape number (see section 6) that shares an edge with the face being quadrangulated, then we repeat the process for the remaining portion of the face. The decomposition is also perfect, as no Steiner points are introduced.

This leads to a mesh consisting of predominantly quadrilateral faces (since the starting mesh was predominantly hexagonal) and a small number of triangular faces. One step of subdivision leads to a quadrilateral base mesh with a predominant number of regular vertices. Let M_{H_0} denote the quad mesh after one step of subdivision.

4.2. Resampling

M_{H_0} becomes a parameterization domain for the input model. To generate a remeshed version, we resample its surface at

dyadic parametric positions. Our resampling procedure follows a normal-based / closest-point approach [31]. By construction, input mesh regions are height-fields over the corresponding faces of M_B and the vertices of M_B lie on the input mesh. This makes our resampling procedure more robust than general resampling scenarios, in which a mesh is resampled over an arbitrarily simplified domain.

To produce a hierarchy with subdivision connectivity, we subdivide M_{H_0} to the desired number of levels L and compute data on each level by resampling. We use Catmull-Clark subdivision combined with a progressive resampling strategy, which gives us the chance to optimize intermediate meshes before proceeding with resampling at finer resolutions:

Algorithm 3 (progressive resampling):

Given an input mesh M_I and a coarse mesh M_{H_0}:
1. $M'_{H_0} = project(M_{H_0}, M_I)$
2. $M''_{H_0} = regularize(M'_{H_0})$
3. for $l = 0, \cdots, L-2$ do
    4. $M_{H_{l+1}} = subdivideOnce(M''_{H_l})$
    5. $M'_{H_{l+1}} = project(M_{H_{l+1}}, M_I)$
    6. $M''_{H_{l+1}} = regularize(M'_{H_{l+1}})$
7. $M_{H_{L-1}} = project(M''_{H_{L-1}})$
8. Perform multiresolution analysis starting from $M_{H_{L-1}}$, updating $M_{H_{l-2}}, \cdots, M_{H_0}$ and computing details

subdivideOnce() accomplishes the refinement of the current mesh with one step of Catmull-Clark subdivision. The actual resampling is done in *project()* by iterating over the vertices of the current mesh and projecting them onto the input mesh. The placement of vertices after projection is optimized through a small number of Laplacian smoothing steps in *regularize()*. To prevent vertices from straying too far from the input mesh during regularization, we adjust their positions using only the tangential component of the Laplacian [36]. The final projection of step 7 ensures that the control points of the finest control mesh (or alternatively, their limit positions) lie on the input mesh.

4.3. Multiresolution Analysis

The applicability of multiresolution analysis to processing of meshes was established by [33, 48, 41]. The basic idea is that, at different resolutions, multiresolution details capture different spatial frequencies of a 3D model. Our method uses a traditional iterative approach for multiscale analysis ([47]). Starting with the finest level mesh $M_{H_{L-1}}$ we alternate two steps: (a) a *restriction operation* in which control points of $M_{H_{l-1}}$ are computed from those of M_{H_l}; and (b) a *detail computation* step in which details on level l are computed as the difference between control point positions of M_{H_l} and the positions obtained by subdividing $M_{H_{l-1}}$. The details are 3D vectors expressed in local frames on each level.

5. Constrained Parameterization

By design, our method builds a coarse mesh with faces corresponding to relatively flat regions of the input model. However, subsequent processing steps (regularization, in particular) slightly perturb the alignment to features. In many instances, this is not an issue as perturbations are small and multiresolution details can be used to approximate features. However, for some models or applications it is desirable to guarantee an exact correspondence between input and output features. This is useful for reproducing sharp features or ensuring that resulting mesh edges follow designated paths.

Our remeshing framework supports constraints in the form of closed curves along edges of the input model. They can be user-defined or the result of automatic detection [24]. Edges along feature curves are tagged and receive special handling in various stages of the remeshing procedure, as outlined next.

Clustering. Clusters are clipped to feature curves. In the cleanup stage, regions that are intersected by feature curves are first split into subregions. Starting with faces along a curve, we distinguish between faces inside the curve and faces outside it. For each of the two types, we run a region growing algorithm inside the current cluster which yields a partition of the cluster into regions that are completely inside or completely outside a feature curve. These regions may undergo further refinement, as described in section 3.3.

Polygonal mesh extraction. Coarse mesh edges corresponding to cluster boundaries along constraint curves are tagged. We also tag vertices along tagged edges so that during simplification, tagged vertices on untagged short edges being collapsed stay fixed. Fig. 10b-d illustrates the steps of extracting a polygonal mesh with tags (shown in red) and its constrained simplification.

Resampling. Data at subdivided locations along tagged coarse mesh edges is resampled from the corresponding feature curves by arc length parameterization. During regularization, tagged control points remain fixed.

Fig. 6 and 10 show a number of remeshing results with constraints. For open meshes, boundaries are automatically detected and treated as constraints. Resampling boundaries from the original boundary curves gives better quality approximations and is more robust than normal shooting approaches which require a virtual extension of boundary faces to infinity [31].

6. Implementation and Results

Discussion Several parameterization examples are illustrated in Fig. 5-8 and 10-12. Fig. 6 and 10 show results for meshes with and without boundaries, as well as with interior sharp features. Constraint curves are shown in red. Among the models illustrated, those of Fig. 10 are the only ones that passed the shape test after normal-based clustering. All other models were subject to the full clustering pipeline.

Basic geometric shapes are included for comparison with “expected" subjective partitions a human would perform. The woman preparing beer in Fig. 5 and the oil lamp in Fig. 11 offer examples at the opposite spectrum of complexity: they have higher genus and a number of challenging geometric features, such as high-curvature areas and thin walls which require a good alignment between the model and the coarse parameterization domain. We have also included the two models of Fig. 7 and Fig. 8 as they have been previously used in remeshing literature. Note how the base mesh for the bunny follows

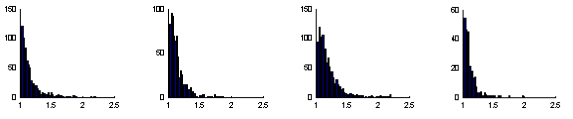
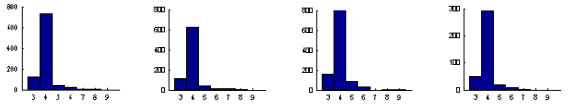
	heart	bunny	woman	pharaoh
Size:				
# F	7,412	71,040	112,478	315,462
# CF	173	155	219	71
Quality:				
HD	0.01	0.02	0.008	0.008
R_V	0.9994	0.9981	0.9991	0.9990
$\mathcal{H}_S$				
$\mathcal{H}_V$				

Table 1: *Quality statistics: # F = number of input faces; # CF = number of coarse faces extracted with our method (before quadrangulation); HD = Hausdorff distance between input mesh and M_{H_3} (4th subdivision level); $R_V = V_{M_{H_3}}/V_I$ = ratio of output mesh volume (M_{H_3}) to the input mesh volume; $\mathcal{H}_S$ = histogram of face shape numbers for the quad base mesh M_{H_0}; $\mathcal{H}_V$ = histogram of vertex valences for M_{H_0}.*

its features. Although our spatial decomposition is isotropic in nature, anisotropic elements are generated (especially along tubular features such as ears and limbs), due to intra-region normal-based clustering. Our method also produces nearly regular patches with few extraordinary vertices even in difficult regions, such as around the tail of the cow (for comparison see the remeshing of the same model in [40]).

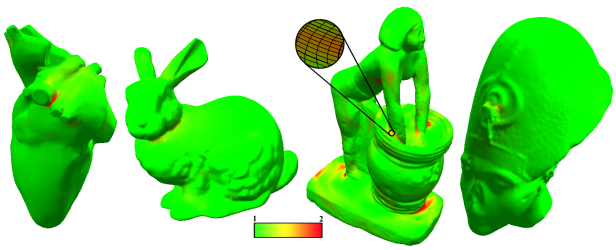


Figure 4: *Color-coded visualization of the face shape numbers across several models (green corresponds to square shapes; gradation to red quantifies deviation from square).*

Performance statistics We evaluate the quality of our output meshes using four different measures:

Hausdorff distance: provides a numerical estimate of the maximum distance between two meshes as the largest of two directed max-min distances from each mesh to the other. We use Metro [11] to report the results as a percentage of the mesh bounding box diagonal.

Volume ratio: quantifies the change in volume after remeshing (closed meshes only). It is defined as the ratio between the volume of the finest-level Catmull-Clark control mesh and that of the original.

Face shape distribution: characterizes the deviation of remeshed faces from a square. We plot a histogram of face shape measures for the faces of the base mesh. With the notations of Fig. 3b, we quantify the deviation of a 3D quadrilateral mesh face $(p_i, p_j, p_{j+1}, p_{j+2})$ from a square by considering the affine mapping of the triangle (p_i, p_j, p_{j+2}) to the right triangle with unit-length legs (O, q_1, q_2). As pointed out in [22, 43], the singular values of the Jacobian of this map characterize the local distortion between the right triangle and its 3D counterpart. We use the ratio of the singular values, i.e., the condition number of the Jacobian, as our shape measure. Simple calculations give $\mathcal{K}(J) = (|p_j - p_i|^2 + |p_{j+2} - p_i|^2)/(2A)$, where A is the area of (p_i, p_j, p_{j+2}). We define the *shape number* $\mathcal{S}$ of a quadrilateral face as the average of the four condition numbers at its vertices, normalized so that the shape number of a square equals 1.

Valence distribution: characterizes the regularity of the base mesh.

Table 1 shows results for remeshed models with 4 levels of subdivision. The histograms confirm the quality of the base meshes, with a majority of well-shaped, almost square faces (shape numbers close to 1), a high percentage of regular vertices, and no highly irregular ones (valence 10 or higher). Face shape variation is also illustrated in Fig. 4.

In contrast, Fig. 9 illustrates the problems that would be encountered if a naive simplify-and-pair approach were used instead of our method. We used the QSlim software [16] to generate simplified meshes for some of the same objects we remeshed with our method (tubular shape in Fig. 9a and the pipes and chess model in Fig. 10). We set a target number of faces equal to double the number of base faces retrieved by our approach (assuming that subsequent face pairing would be applied to produce a quad mesh). In addition to poor face shape and connectivity quality, the lack of topological guarantees leads to non-manifold results and features such as boundaries are not preserved.

Regarding time complexity, Algorithm 1 runs linearly through the input mesh faces and, for each face, through the set of generators. Algorithm 2 is also linear in the number of faces in each region. Algorithm 3 is the bottleneck of the computation, as it requires repeated resampling of the input mesh over the various levels of the new hierarchy. To speed it up, we use a uniform grid approach as in [11] which gives reasonable results. For larger models, additional speedup of the pipeline is obtained by using two different versions of the input model: a medium resolution for base mesh extraction (i.e., to recover coarse shape), and a full resolution for resampling (i.e., to recover high-frequency details). In our tests, running times for the extraction of quadrilateral parameterization domains (i.e., steps 1-5) ranged from 6 to 40 seconds. Typical resampling times ran between a few seconds and 25 minutes for 4 levels of subdivision. The measurements were performed on a Pentium 4 3GHz PC with 2GB of RAM.

Implementation details Our method has several parameters which can be set with default values. We briefly describe our choices. We use a simple check to decide whether a region obtained by clustering satisfies (R2). We apply this check follow-

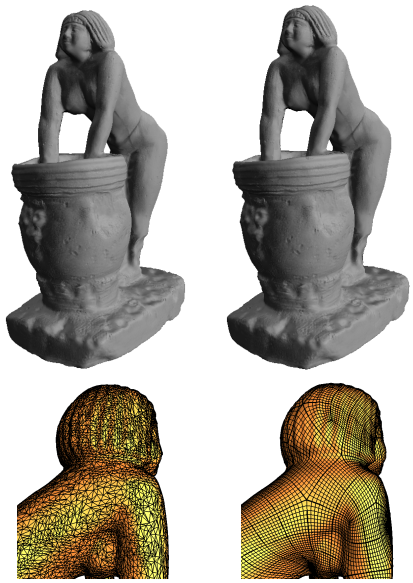


Figure 5: *Statuette of a woman preparing beer. Top: input model (left) and Catmull-Clark remesh (right) rendered with detailed normals. Bottom: detail of the two meshes (thick lines indicate coarse patches).*

ing normal-based clustering, to regions that are relatively flat, so it makes sense to measure the convexity of their boundaries. A coarse polygon is defined for each cluster with vertices at the confluence of three or more clusters (A, B, C, D in Fig. 3a). A polygon is considered convex if each of its interior angles is at most π. If a polygon is found to be convex, we then consider how well it approximates the boundary of the region. For this, we measure the maximum distance between points along the boundary curves and the corresponding coarse lines approximating them. For example, in Fig. 3a left, the distance between the curve segment (AB) along the boundary of the red region to the line segment (AB) is significant. In this case, $(ABCD)$ is not an acceptable approximation for the red region. In the right image the edges of $(ABCD)$ are within an acceptable tolerance from the corresponding curves and the approximation is acceptable. Since clusters are groups of faces, boundaries between them may appear jagged (a face equidistant from two generators is arbitrarily categorized in one of the corresponding clusters). We define the approximation tolerance as a constant times the average edge length in the cluster and we ignore such jaggedness in shape testing.

To check (R3), we compute the average normal of a region $\overrightarrow{h}$ and we test for the deviation of face normals from it. We also compute a plane perpendicular to $\overrightarrow{h}$ that passes through the center of the bounding box of the region. The region passes the test if the maximum distance from region vertices to this plane is within some tolerance (5% of the bounding box diagonal). In practice, we observed this latter test to be rarely needed, due to the way regions are created (normal-based clustering first).

We use a fixed number of Laplacian smoothing iterations on the normal field prior to clustering to attenuate noise. The default number in our implementation is 10, however this value should be increased for very noisy data. A simple test checking

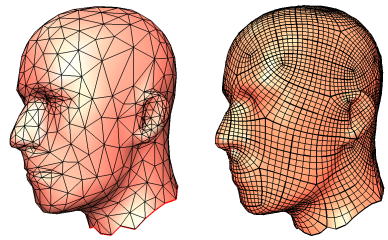


Figure 6: *Parameterization of a mesh with boundary. Left: input model. Right: Catmull-Clark remesh (thick lines indicate coarse patches).*

if any faces have chnaged clusters during an iteration is used as a *stopping criterion* in center-based clustering. This is combined with a limit on the number of iterations to deal with cases when a face on the border between two clusters flips back and forth between them. We used a limit of 50 iterations for all our tests. Typical number of iterations to convergence for our models were less than 10. For *random sampling* according to Algorithm 2, we followed the following strategy: uniformly distribute [38] a large number of samples over the region (we use $F_R/2$ samples) and of these select $N = F_R/4$ using a farthest-first approach; N_0 of the N samples are kept, where N_0 is the index of the sample for which the largest jump in the maxmin distance is observed [26]. After *coarse mesh extraction* edges are selectively simplified. We identify edges that are almost collinear with an adjacent edge and can be collapsed without causing degeneracy of the adjacent faces. Fig 10 (c) and (d) show a coarse mesh before and after such simplification.

Applications *Scanning/editing.* 3D scanning systems can produce highly detailed geometric models of existing objects. Most systems produce point clouds or triangle meshes derived from point clouds. While these are convenient for viewing, operating directly on the detailed point sets or triangles is cumbersome when modifications are required. Conversion to multiresolution semi-regular representations provides access to editing and compression tools not available for the input mesh. In Fig. 12 we illustrate a high-resolution scan of a pharaoh's head for which different restorations of the nose have to be studied. We use a free-form variational editing tool (similar to [29]) to reconstruct the nose interactively. The user can smoothly deform the shape while preserving the Catmull-Clark connectivity. Many possibilities can be generated and evaluated quickly.

Texture mapping. Many models also have associated color and normal information (in some cases, at higher resolution than the geometry [42]). Colors and detailed normals can be conveniently stored in texture maps. Because the faces of our parameterization domain correspond to height-field regions of the object, they can be readily used as partitions for texture mapping. We define an orthogonal camera for computing texture coordinates for each partition by generating a view direction equal to the average surface normal and a bounding box of the region aligned with the view vector. For scanned models for which images are captured together with geometry, it is important to be able to recompute textures from new viewpoints [8] without flattening the model. This has been an important motivation for enforcing (R3). To avoid seams when texture maps are down-

sized, we expand the area covered by each map [43]. By design, each of the regions represented by the base mesh polygons is nearly square, and the individual texture maps pack efficiently into a single map. In Fig. 5 and 12, we use color and normal texture maps. The normal maps provide the appearance of detail when coarser meshes are used for rendering [12]. Because our remeshing follows surface features, they can be used to visualize the model with little distortion even when a very coarse level mesh is used in rendering.

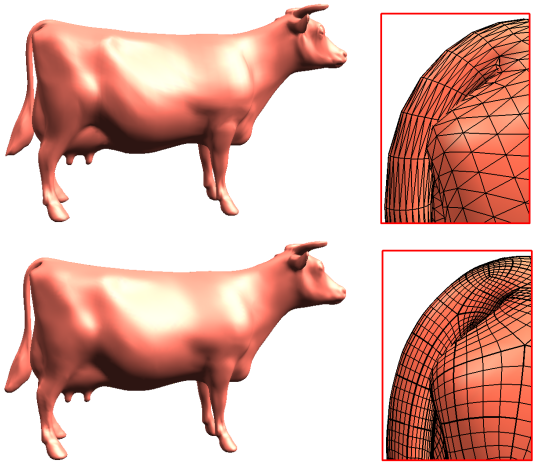


Figure 7: *Top: input model and detail of the complex geometry around the tail; bottom: Catmull-Clark result and tail detail (control mesh shown on 3rd subdivision level; coarse patches are shown with thick lines)*

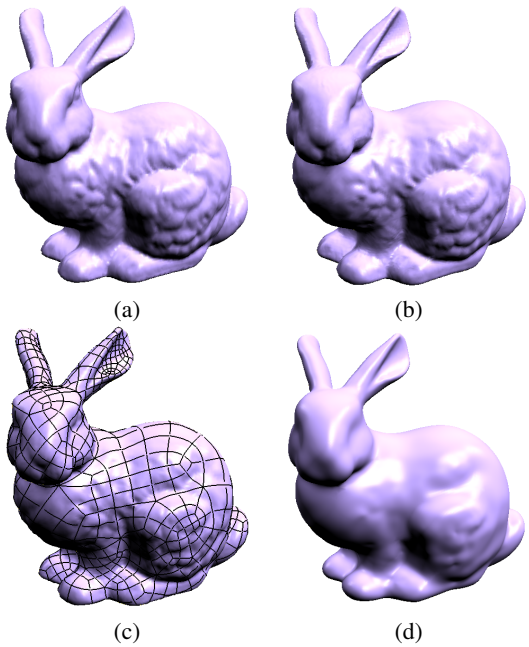


Figure 8: *(a) Input model. (b) Catmull-Clark result. (c) Model with patch outlines superimposed. (d) Model obtained after removing high-frequency details from the multiresolution hierarchy.*

Limitations of the approach Spatial center-based clustering leads to isotropic decompositions of the input geometry. When combined with normal-based partitioning, some anisotropy is introduced. However, our method does not provide support for systematically orienting clusters according to a direction field defined over the surface. This would be a natural next step and we plan to explore it in the future. Also, it would be useful to handle additional types of constraints, such as points and arbitrary curves that are not aligned with mesh edges.

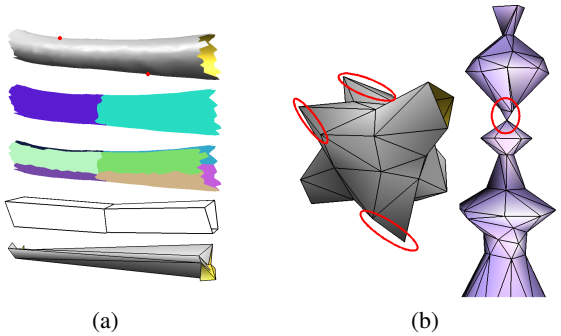


Figure 9: *(a) Region generation on an undersampled portion of geometry (from top to bottom): input geometry with 2 samples (shown in red); isotropic clustering produces a partition which does not satisfy (R1)–(R3); intra-region split after cluster cleanup; base domain (8 faces) extracted from the resulting partition; domain with 16 triangle faces obtained by QEM simplification shown for comparison. (b) Additional QEM-simplified meshes of the pipes and chess models in Fig. 10 suffer from topological degeneracy (red outlines) and are not suitable as parameterization domains.*

7. Conclusions and Future Work

In this paper we introduced a novel method for computing high-quality parameterizations of triangulated manifolds over quadrilateral domains. The creation of the base domain is performed through a combination of clustering methods which control the *shape* and *flatness* of clusters. The result is a smooth parameterization with nicely-shaped patches and low distortion. Our method also offers a completely automated way to convert complex triangle meshes to Catmull-Clark multiresolution representations for subsequent processing (e.g., editing, texture mapping, compression). Such conversions are particularly useful in the context of libraries of models or parts which require quick editing, filtering, or other forms of geometric processing impractical to perform on the original data.

This work opens interesting possibilities for further exploration. Building anisotropic domains according to given direction fields is a natural next step. Another challenge is to use our constrained parameterization approach to generate maps between different models of similar shape with correspondence between features.

Acknowledgements

This work was performed as part of a collaboration with Dassault Systèmes. We would like to thank David Bonner and his team for their feedback and support. We are grateful to Fausto Bernardini for valuable discussions and to Richard Giantisco

for the video editing work. The models of the pharaoh head, the woman preparing beer, and the oil lamp are courtesy of the Egyptian Center for Documentation of Cultural and Natural Heritage. The bunny model is courtesy of the Stanford Computer Graphics Lab.

References

[1] In *http://www.alias.com*. 1
[2] In *http://www.discreet.com*. 1
[3] In *http://www.catia.ibm.com*. 1
[4] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM TOG SIGGRAPH*, pages 485–493, 2003. 2
[5] P. Alliez, É. Colin de Verdière, O. Devillers, and M. Isenburg. Isotropic surface remeshing. In *Proc. SMI03*, pages 49–58, 2003. 4
[6] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Proc. SIGGRAPH 98*, pages 415–421, 1998. 5
[7] M. W. Bern and D. Eppstein. Quadrilateral meshing by circle packing. *Int. J. Comp. Geom. Appl.*, 10(4):347–360, 2000. 2
[8] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002. 8
[9] P.-T. Bremer, S. Porumbescu, B. Hamann, and K. I. Joy. Automatic semi-regular mesh construction from adaptive distance fields. In *Proc. Curves and Surfaces 02*, 2002. 2
[10] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *CAD*, 10(6):350–355, 1978. 1, 2
[11] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified ssurfaces. *CG Forum*, 17(2):167–174, 1998. 7
[12] J. Cohen, M. Olano, and D. Manocha. Appearance-preserving simplification. In *Proc. SIGGRAPH 98*, pages 115–122, 1998. 9
[13] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41:637–676, 1999. 3
[14] Q. Du, M. D. Gunzburger, and L. Ju. Constrained centroidal Voronoi tessellations for surfaces. *SIAM J. Sci. Comput.*, 24(5):1488–1506, 2003. 3
[15] M. Eck and H. Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Proc. SIGGRAPH 96*, pages 325–334, 1996. 2
[16] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. ACM SIGGRAPH 97*, pages 209–216, 1997. 7
[17] M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proc. of ACM Symp. on Interactive 3D Graphics*, 2001. 2
[18] A. Gersho. Asymptotically optimal block quantization. *IEEE Trans. Inf. Theory*, 25(4):373–380, 1978. 3
[19] X. Gu, S. Gortler, and H. Hoppe. Geometry images. *ACM TOG SIGGRAPH*, 21(3):355–361, 2002. 2
[20] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proc. SGP03*, pages 127–137, 2003. 2
[21] I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens. Hybrid meshes: Multiresolution using regular and irregular refinement. In *Proc. Symp. Comp. Geom.*, pages 264–272, 2002. 2
[22] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In *Curve and Surface Design*, pages 153–162. 2000. 7
[23] K. Hormann and G. Greiner. Quadrilateral remeshing. In *Proc. Vision, Modeling, and Viz 2000*, pages 153–162, 2000. 2
[24] A. Hubeli and M. Gross. Multiresolution feature extraction from unstructured meshes. In *Proc/ IEEE Viz'01*, pages 287–294, 2001. 6
[25] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999. 3
[26] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM TOG SIGGRAPH*, pages 954–961, 2003. 2, 5, 8
[27] Y. Kho and M. Garland. User-guided simplification. In *Proc. ACM I3D Symp*, pages 123 – 126, 2003. 2
[28] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM TOG SIGGRAPH*, pages 350–357, 2003. 1
[29] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proc. of SIGGRAPH 98*, pages 105–114, July 1998. 8
[30] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proc. SIGGRAPH 96*, pages 313–324, 1996. 1
[31] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proc. SIGGRAPH 00*, pages 85–94, July 2000. 6
[32] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM TOG SIGGRAPH*, pages 362–371, 2002. 2
[33] M. Lounsbery, T. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM TOG*, 16(1):34–73, 1997. 6
[34] J. MacQueeen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math Statistics and Prob.*, pages 281–297, 1967. 4
[35] M. Müller-Hannemann and K. Weihe. Minimum strictly convex quadrangulations of convex polygons. In *Symp. Comp. Geom.*, pages 193–202, 1997. 3, 5
[36] Y. Ohtake, A. G. Belyaev, and I. A. Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization. In *Proc. GMP*, pages 229–237, 2000. 6
[37] A. Okabe, B. Boots, K. Sugihara, and S. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley, 2000. 3
[38] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM TOG*, 21(4):807–832, 2002. 8
[39] S. Owen, M. Staten, S. Canann, and S. Saigal. A survey of unstructured mesh generation technology. In *Proc. 7th Int. Meshing Roundtable*, pages 239–267, 1998. 1, 2
[40] E. Praun and H. Hoppe. Spherical parameterization and remeshing. *ACM TOG SIGGRAPH*, pages 340–348, 2003. 2, 7
[41] K. Pulli and M. Lounsbery. Hierarchical editing and rendering of subdivision surfaces. Technical Report UW-CSE-97-04-07, Dept. of CS&E, Univ. of Washington, Seattle, WA, 1997. 6
[42] C. Rocchini, P. Cignoni, C. Montani, and R. Scopigno. Acquiring, stitching and blending appearance attributes on 3D models. *The Visual Computer*, 18(3):186–204, 2002. 8
[43] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proc. SIGGRAPH 00*, pages 409–416, 2001. 2, 7, 9
[44] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proc. SGP 03*, pages 146–274, 2003. 2
[45] V. Surazhsky, P. Alliez, and C. Gotsman. Isotropic remeshing of surfaces: a local parameterization approach. In *Proc. 12th Intl. Meshing Roundtable*, 2003. 2
[46] L. Velho. Quadrilateral meshing using 4-8 clustering. In *Proc. CILANCE'00*, pages 61–64, 2000. 2
[47] D. Zorin, P. Schröder, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animation. SIGGRAPH'00 Course Notes, 2000. 1, 6
[48] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proc. SIGGRAPH 97*, pages 259–268, August 1997. 6

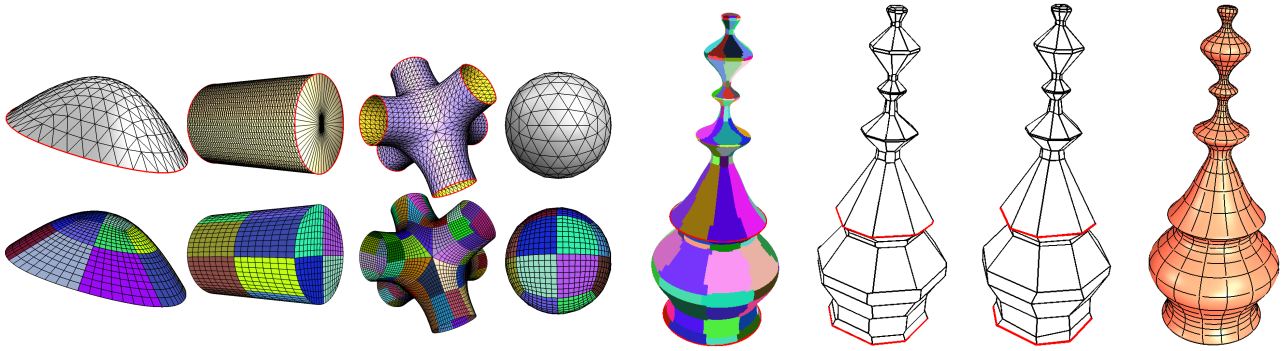


Figure 10: *(a) Quadrilateral parameterization of simple shapes. (b)-(e) Parameterization of chess piece over domain extracted after normal-based clustering. Constraint curves and edges are shown in red.*

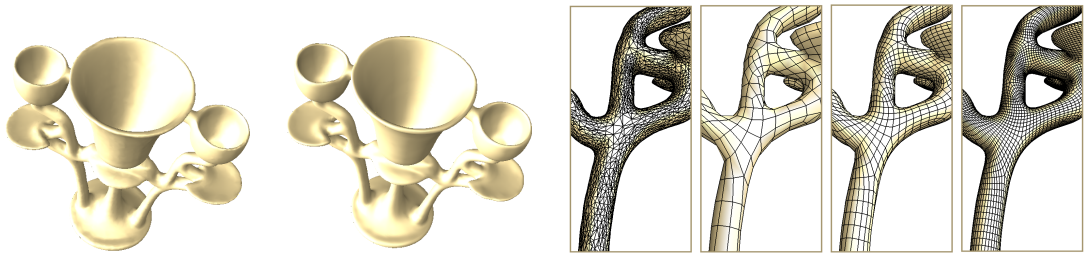


Figure 11: *Oil lamp. From left to right: input model; Catmull-Clark hierarchy with 3 levels; zoom-in sequence illustrating a detail of the mesh before and after remeshing.*

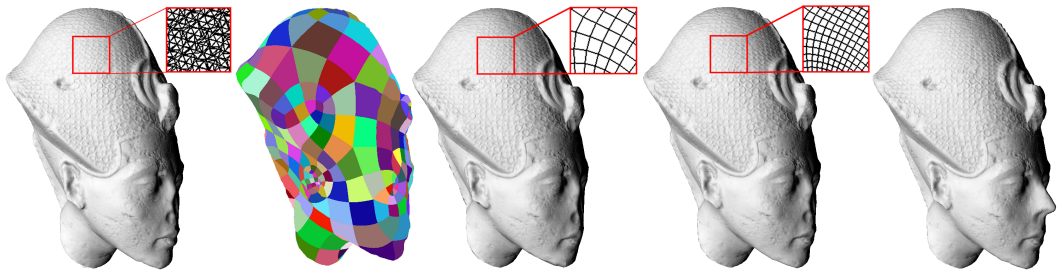


Figure 12: *Left to right: Triangulated pharaoh model. Quadrilateral patches corresponding to a parameterization domain generated with our method. Control meshes on the 3rd and 4th levels of a Catmull-Clark hierarchy built over this domain. Restoration of the broken nose through multiresolution editing of the semi-regular model. Shaded images are rendered with detailed normals from the input data set.*