Eurographics Symposium on Geometry Processing (2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

Domain Decomposition for Multiresolution Analysis

Ioana M. Boier-Martin

IBM T. J. Watson Research Center
Hawthorne, NY, USA

Abstract

This paper describes a method for converting an arbitrary mesh with irregular connectivity to a semi-regular multiresolution representation. A shape image encoding geometric and differential properties of the input model is computed. Standard image processing operations lead to an initial decomposition of the model that conforms to its salient features. A triangulation step performed on the resulting partition in image space, followed by resampling and multiresolution analysis in object space, complete the procedure. The conversion technique is automatic, takes into account surface properties for deriving a base domain, and is computationally efficient as the bulk of the processing is carried out in image space. Besides domain decomposition, our image-based approach to handling geometry may be used in the context of related applications, including model simplification, remeshing, and wireframe generation.

Keywords: *Subdivision surfaces, multiresolution, model segmentation, geometry images.*

1. Introduction

Representing surfaces through subdivision elegantly addresses many of the drawbacks of unstructured 3D shape representations. Subdivision offers a compact way to represent geometry with minimal connectivity information. It generalizes the classical spline patch approach to arbitrary topology, naturally accommodates multiple levels of detail, and produces meshes with almost regular structure, suitable for digital processing. When combined with multiresolution analysis, subdivision offers a powerful modeling tool, allowing for complex editing operations to be applied efficiently at different resolutions [41].

The set of tools for manipulating subdivision surfaces has grown steadily in recent years. The main obstacle to their widespread use is having to convert existing data to this format. We focus our attention on the conversion of irregular triangulated meshes to multiresolution subdivision hierarchies. The task of converting an arbitrary mesh to a semi-regular hierarchical representation can be viewed as consisting of three main steps: (a) finding a suitable parameterization domain for the input shape, preferably taking into account its salient features, (b) resampling the original data at dyadic positions over the newly found domain, and (c) applying multiresolution analysis to generate a hierarchy. This paper focuses on challenges posed by the first step: finding a base domain. We use established algorithms for resampling and multiresolution analysis.

We propose an image-based approach to domain decomposition. Recent work [16] has demonstrated the ability to cut and parameterize an entire mesh onto a single 2D image that captures the input geometry into a completely regular structure. Shape descriptors such as normals and curvatures can be encoded into similar images. We cast the complex task of partitioning an arbitrary 2-manifold input mesh into the simpler one of segmenting its shape image. Subsequently, we use the inverse mapping from the image to the mesh to obtain a partition of the latter. Based on this decomposition, we generate a coarse base domain over which we perform resampling and analysis.

Storing meshes into completely regular structures offers advantages in terms of efficiency for rendering and network transmission. Nevertheless, for certain applications like modeling and design, working with such a representation appears non-intuitive as the shape being modeled is distorted by flattening. For instance, it would be rather difficult to interactively locate and consistently update model fea-

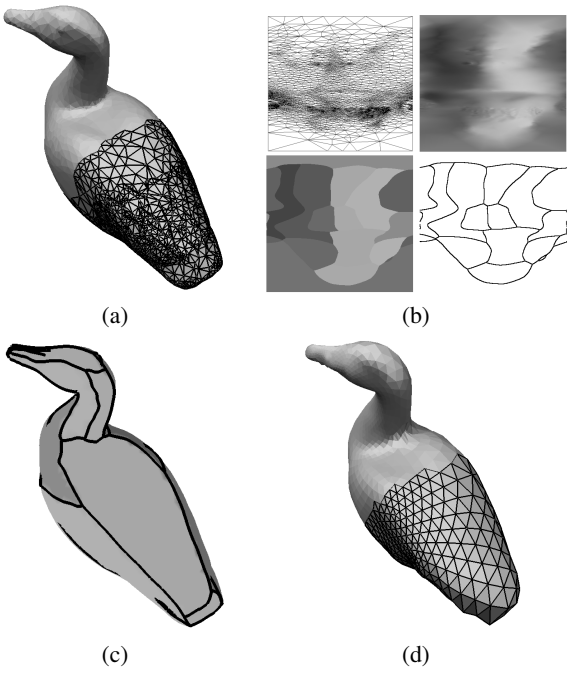


Figure 1: *(a) Irregularly triangulated input model. (b) Image processing for domain decomposition (rowwise from top left): planar parameterization of the input model; normal map; color quantized normal map; edges detected on the quantized map. (c) Color regions correspond to regions of the quantized image, black wireframe corresponds to curves found through edge detection. (d) Control mesh on level two of a Loop hierarchy obtained from the decomposition. See also color section.*

tures in geometry images [16]. In this context, semi-regular decomposition remains one of the best representation choices.

The main contributions of this work are:

- A fully automatic framework for converting irregular triangulated meshes to Loop [30] multiresolution hierarchies. The conversion process supports transferring attribute data (e.g., color) from the input mesh to the remeshed model.
- A novel and efficient image-based approach that takes into account input surface properties to generate a parameterization domain.
- A practical extension of geometry images [16] to applications beyond compression for network transmission.
- An elegant solution to other geometry processing problems, including face clustering, mesh simplification, and wireframe generation. We strongly believe that casting such computational tasks into image space provides new and efficient alternatives to existing algorithms.

Note on terminology: The the generic term *shape image* designates the 2D image used for model segmentation. We prefer this broad term as it emphasizes the applicability of our approach to other shape descriptors. The shape information in our examples consists of surface normals.

2. Background

2.1. Surface Representations

We process input represented by polygonal mesh discretizations of 2-manifold surfaces. Without loss of generality, we restrict our attention to triangle meshes. Figure 1 (a) illustrates an example of such a mesh corresponding to a duck model. The target output representation was proposed in various forms by [32, 35, 42]. Subdivision defines a smooth surface recursively, as the limit of a sequence of meshes. A finer mesh is obtained from a coarse mesh by applying a set of fixed refinement rules. Multiresolution surfaces extend subdivision surfaces by introducing *details* at each level. Each time a finer mesh is computed, detail offsets are added to the subdivided coarse mesh. A *semi-regular mesh* (i.e., a mesh with subdivision connectivity) can be converted to a multiresolution hierarchy by defining a restriction operator that generates vertices on a coarse level from those on a finer level. Multiresolution details are computed as differences between levels.

For conversion purposes, we regard a subdivision surface as a function over a domain. The most natural choice for a domain is the initial (coarsest level) mesh. Our method identifies such a domain for an arbitrary mesh and creates a multiresolution subdivision hierarchy on top of it.

2.2. Related work

Existing methods for converting arbitrary meshes to semi-regular representations typically fall into one of two categories. On one hand, there are techniques that involve some degree of manual adjustment (e.g., [25]) requiring the user to outline boundaries of surface patches corresponding to base mesh faces. A different approach is proposed in [29]. In this case, a generic semi-regular mesh provided by the user is fitted to the target irregular mesh. Fully automatic methods form the second category. In [11], a base domain is derived by computing a Voronoi tiling of the original mesh. Alternatively, a base domain is found by simplification of the original mesh, possibly with constraints (e.g., [27, 26, 18, 22]). Typically, such methods employ local error measures and the resulting partitions do not conform well to global salient features of the model. Several methods address semi-regular remeshing of certain classes of input surfaces. For instance, the method of Kobbelt et al. [24] applies to closed genus zero surfaces, while the approaches of Hormann et al. [21] and Alliez et al. [1] address remeshing of genus zero manifolds with boundary.

Image-based geometry encoding Geometry images can be automatically generated for an arbitrary mesh as described

in the innovative paper by Gu et al. [16]. Instead of viewing a semi-regular remeshing approach as a collection of abutting geometry images (as suggested in the original paper), a semi-regular decomposition of the 3D model is extracted from a single such image. By accommodating any cutting / flattening combination and removing the square boundary parameterization requirement, we generate images efficiently and with reduced distortion, taking advantage of state-of-the-art parameterization methods. Since there is no need for our application to compress the images, we do not have to worry about reconstruction artifacts.

Perhaps the closest to our technique is the method of Alliez et al. [2] which uses images to guide the remeshing process. The major distinctions lie in the fact that, in their case, multiple images are associated with a single model (i.e., a decomposition into charts is done using the method of [11] and not guided by the images), semi-regular remeshing is obtained by subdividing a uniform mesh in parameter space (thus ignoring model features), and the output is not recovered in multiresolution format.

Mesh Cutting and Parameterization The flattening of arbitrary meshes has been an extensively researched topic as it has numerous applications, including texture mapping, remeshing, and modeling. While many methods address the problem of flattening meshes with disc topology (e.g., [13, 20, 28, 9]), recent techniques for mesh cutting ([39, 16, 12]) allow parameterization of 2-manifolds of arbitrary genus onto planes. Our approach to domain decomposition is independent of the parameterization and may therefore be used in conjunction with any such scheme. In this paper we demonstrate our technique using several flattening scenarios: a cylindrical projection onto the polar plane proposed by Brechbühler et al. [7], the angle-based flattening approach of Sheffer and de Sturler [38], and the conformal mapping of Desbrun et al. [9].

Face clustering and feature extraction Many authors have addressed this problem [15, 23, 36, 28]. To the best of our knowledge, existing methods perform the partitioning directly on 3D mesh representations. This type of approach requires formulating planarity constraints (see, for example, [15]) that have to be checked as regions are generated. Additional measures are needed to control the shape and orientation of the clusters. In many cases the generated partitions are poorly aligned with geometric features of the underlying surface or tend to wrap around cylindrical shapes. Most importantly, they and do not lead to a corresponding mesh in an obvious fashion. Additional non-trivial processing must follow to recover a suitable triangulation [36].

Triangulations with constraints The issue of computing a planar triangulation of a set of points in the presence of vertex, edge, and hole constraints was studied by many authors. In our implementation we use a publicly available triangulation library [40].

3. Domain Decomposition

To compute an initial base domain for our multiresolution representation, we first identify regions of the input mesh that exhibit little or no shape variation and can therefore be parameterized over a few coarse polygonal elements with reduced distortion. The regions consist of connected sets of faces and completely partition the input model. This type of decomposition has numerous applications beyond parameterization and remeshing, to object recognition, shape perception, collision detection, and ray tracing.

We reduce the complexity of the partitioning task by casting it into the 2D space of a shape image corresponding to the model. Our algorithm for computing a base mesh for an arbitrary model consists of the following main steps:

1. Generate a shape image corresponding to the input mesh. Also compute an invertible mapping p between the mesh and the image.
2. Smooth and color quantize the shape image to identify regions of little or no shape variation.
3. Perform edge detection and linking on the quantized image to generate boundary curves between regions.
4. Simplify boundary curves and generate coarse polylines separating the shape regions.
5. Triangulate the polyline set using the polylines as constraints.
6. Using the inverse mapping p^{-1}, project the triangulation obtained in step 5 onto the input mesh to obtain a base mesh.
7. Cleanup and optimize the base mesh.

Each of these steps is described in detail next. A simple model is used to illustrate the main points (see Figure 5).

3.1. Image Creation

Following [16], a 2D image representing a 3D mesh can be obtained by cutting the mesh and flattening it onto a plane. Unlike in [16], where the boundaries of the flattened mesh are fixed onto a square, we accommodate various parameterization schemes, with and without fixed boundaries. Natural-boundary maps typically reduce distortion at the expense of an acceptable increase in computational cost.

Let M_1 denote the mesh resulting from cutting the input mesh M and D the domain of the parameterization of M_1 onto the plane. The parameterization p is an invertible piecewise linear map $p : M_1 \rightarrow D$ that associates domain coordinates (u, v) to each mesh vertex in M_1. We typically normalize the planar coordinates to the unit square and we compute the shape image over the rectangular region defined by the axis-aligned bounding box of the flattened mesh. Figure 5(c) shows a fixed boundary parameterization of the mushroom model from Figure 5(a). The corresponding shape image (i.e., normal map) generated by rendering the flat mesh $p(M_1)$ using the normalized normal vectors of

mesh M_1 as colors per vertex is shown in Figure 5(d). Similarly, the shape image corresponding to the face model in Figure 9 (a) was created from a parameterization with natural boundaries shown in Figure 9 (b). In the latter case, regions of the image outside $p(M_1)$ are marked as background (shown in black in Figure 9 (c)).

The computation of the inverse map p^{-1} involves a point location procedure. Given a point (u_0, v_0) in the plane of the parameterization, we search for the parametric triangle T_D containing it. If such a triangle is found, the barycentric coordinates α and β of (u_0, v_0) with respect to T_D are computed and used to recover the pre-image of (u_0, v_0) in 3D: $p^{-1}(u_0, v_0) = (1-\alpha-\beta)a + \alpha b + \beta c$, where a, b, and c denote the vertices of the mesh triangle T_{M_1} corresponding to T_D. To avoid resampling problems along the parameterization boundary, an additional map $\pi : \partial M_1 \rightarrow \partial D$ is also maintained. A boundary point (u_1, v_1) for which a triangle containing it cannot be found is first mapped to the closest point on ∂D and then onto M_1 via π^{-1}.

3.2. Color Quantization for Region Identification

Mesh smoothness is typically the main criterion by which faces are grouped into regions. Although triangle meshes are piecewise linear surfaces and therefore not C^1 continuous, a number of authors have defined and studied smoothness in a discrete sense. In this context, smooth meshes are characterized by low discrete curvature, where the latter is approximately computed from vertex normal estimates (see [34] for a survey). Once computed, such differential quantities are encoded as colors in the shape map. Thus, region finding on a 3D mesh can be formulated as a color quantization / segmentation process applied to the corresponding shape image. While any type of quantization produces a partition of the input mesh, accurate color classification is important to ensure a good decomposition. We explain our choice in the remainder of this section. A Gaussian filter is typically applied to the shape image before quantization to attenuate any artifacts due to discretization. Figure 5 (e) shows the result of color quantization applied to the normal map in Figure 5 (d). The corresponding partition of the original model is shown in Figure 5 (g).

K-means clustering is a popular clustering algorithm with applications not only to image processing, but to data mining in general [37]. It iteratively refines the clusters defined by K centers while attempting to minimize the total mean square quantization error. Given K centers $C = \{c_1, \cdots, c_K\}$ and N data points $X = \{x_1, \cdots, x_N\}$, the performance function to be optimized during classification is defined as:

$$\mathcal{Q}(\{x_i\}_{i=1}^{N}, \{c_k\}_{k=1}^{K}) = \sum_{k=1}^{K} \sum_{x \in S_k} \| x - c_k \|^2$$

where S_k is the subset of X containing points that are closest to c_k than to any other centers in C (i.e., $\{S_k, k = 1, \cdots, K\}$ is the Voronoi partition of X given by the K centers).

For segmentation of color images depicting scenes with complex spatial arrangements, K-means clustering is typically used to obtain an initial segmentation and is followed by more sophisticated schemes to account for local intensity variations and spatial constraints [8]. For the segmentation of shape images corresponding to a single object, we have found the K-means partition adequate without the need for additional processing. We proceed as follows:

Initialize the centers of the K clusters.
Repeat
    Assign each pixel to a cluster such that $\mathcal{Q}$ is minimized (i.e., compute the Voronoi partition of the pixels with respect to the current centers)
    Update the centers of the clusters
until (*no pixels have changed clusters*)

The choice of K and the initial centers depends on the semantics of the shape image. For normal maps, we sample the space of all normals along a small number of directions (e.g., the vectors originating at the center of the unit cube and pointing to its vertices and face centers).

The partition of the input model obtained through quantization may be used in a variety of applications, including texture mapping and bounding volume computation for spatial queries.

3.3. Edge Detection and Wireframe Generation

For base mesh extraction, we further process the result of quantization to identify a coarse polyhedral domain with facets covering the identified regions and with edges along the boundaries between them. To achieve this, we use another common image processing operation: edge detection.

Figure 5 (f) illustrates the result of edge detection applied to the quantized image in Figure 5 (e). The goal is to find the *edge curves* delineating the color clusters. These edges are computed efficiently in a single pass over the image using a marching algorithm. We view the image as a grid G, with grid points located at pixel centers. A two-step approach is used:

For each grid cell in G do
    Find edge segments inside cell
Link segments and form edge curves

In the first step we use a *marching-squares*-type algorithm [31]. Intersection points of the edge curves with grid cells are found by testing for color changes along the four edges of a cell. If a color change is detected a curve point is inserted in the middle of the edge. Curve points are then connected by segments, following a routine marching approach. In the second step, curve segments from the previous step

are linked together by matching their endpoints to form edge curves as shown in Figure 5 (f).

At this point, a wireframe representation of the model can be computed by projecting the edge curves onto the input mesh using the inverse mapping p^{-1}. An example is shown in Figure 5 (h). This kind of representation could serve, for instance, as a minimal representation to be displayed on resource-constrained devices such as cell phones and wristwatches.

Figure 2: *Steps of the Douglas-Peucker polyline simplification algorithm. (a) The farthest point from the shortcut (filled circle) is computed. (b) The polyline is refined to meet the tolerance criterion. The process is repeated for each of the two segments. (c) The resulting simplified polyline is shown bold.*

3.4. Curve Simplification for Polyline Detection

To generate a triangulation that takes into account the boundary curves between regions, we approximate each curve segment with a coarse polyline. The edges of these polylines form a subset of the base domain edges in the plane.

Given a polygonal chain, the *curve simplification* problem is to compute a polyline of reduced complexity that approximates the original chain according to some predefined error criterion. Curve simplification is useful in a number of fields and has been studied by numerous authors. As near-linear time algorithms for simplification are rather involved, simple heuristics have been proposed [10, 19]. Such heuristics are particularly attractive in our case, as the curves resulted from segmentation contain a relatively small number of points. We implemented a variant of the well-known Douglas-Peucker method [10] as it is efficient and produces solutions similar to subjective simplifications a human user would perform. The basic idea is illustrated in Figure 2. Given a tolerance ε, an initial solution is formed by the shortcut between the first and last points of the curve (Figure 2 (a)). If all vertices of the input chain are within ε distance from the shortcut, the algorithm terminates with the shortcut as the solution. Otherwise, the farthest point from the shortcut is found and the shortcut is refined into two segments passing through this point (Figure 2 (b)). The process is repeated recursively for each segment. The final solution is shown as a bold line in Figure 2 (c).

We keep the endpoints of the edge curves fixed and we simplify the curves according to the method just described. The result is a connected set of coarse polylines approximating the boundaries of the shape regions. Depending on the tolerance parameter, coarser or finer approximations are possible (simplified polylines are shown in black in Figures 5(i) and (j)).

We have adapted the Douglas-Peucker algorithm to our problem. The edge curves represent boundaries of relatively flat regions of the input mesh. Since they are detected on the flattened mesh, they appear distorted. To avoid propagating the distortion to the simplified curves, we always perform the selection of the farthest point from the shortcut in 3D:

For each edge curve $C = \{c_0, \cdots, c_n\}$ do
  Compute 3D curve: $C_{3D} = \{p^{-1}(c_0), \cdots, p^{-1}(c_n)\}$
  $Q \leftarrow \{(c_0, c_n)\}$
  $S \leftarrow \emptyset$ (the simplified output curve)
  While (Q is not empty) do
    remove shortcut segment (c_i, c_j) from Q
    find $P_{3D} = p^{-1}(c_k) \in C_{3D}$ farthest
      from $(p^{-1}(c_i), p^{-1}(c_j))$
    if $(dist(P_{3D}, (p^{-1}(c_i), p^{-1}(c_j)) > \varepsilon)$ then
      $Q = Q \cup \{(c_i, c_k), (c_k, c_j)\}$
    else $S = S \cup (c_i, c_k)$

In addition to avoiding distortion, 3D simplification is also needed for properly stitching meshes cut before flattening. The cut curves may project into different shape planar curves (e.g., labeled curves in Figure 7 (b) also shown separately in Figure 3 (a) and (b)) that would not be simplified consistently using a planar version of the algorithm. Working in 3D allows us to elegantly solve this problem.

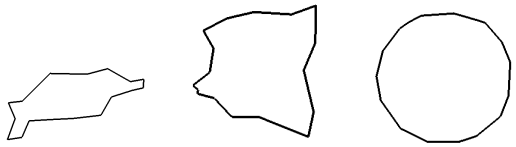


Figure 3: *Our modified curve simplification algorithm ensures that two cut curves in the plane corresponding to the same 3D curve are consistently simplified ((a) and (b)). Simplified 3D curve is shown in (c). Cut curves correspond to those marked in Figure 7 (b).*

Since edge curves are simplified independently, it is possible that the simplified planar curves intersect at points other than their endpoints. In cartography this is known as the *map simplification problem* and has received some attention in the graphics community (e.g., [33]). We implemented a simple check for each curve being simplified. If S is a shortcut for the curve C such that S satisfies the tolerance criterion but intersects at least one of the already simplified curves, we further refine S until no intersections occur.

3.5. Base Mesh Creation

The next step is to use the polylines separating regions of the shape image to compute a constrained triangulation T_M. To obtain a triangulation, we use an off-the-shelf 2D Delaunay triangulator [40]. Figures 5 (i) and (j) show two constrained triangulations generated for the mushroom model, corresponding to two different sets of simplified curves. The constraints are shown as solid lines and the edges introduced by the triangulation with dashes.

A base mesh B_M is computed from the planar triangulation T_M: $B_M = p^{-1}(T_M)$. Two different base meshes for the mushroom model are shown in Figures 5 (k) and (l). A multiresolution Loop representation with six levels is shown in Figure 5 (b). Note that, as the input model is coarsely tessellated, the multiresolution representation captures the tessellation edges as multiresolution detail.

If the original mesh was cut before flattening, the base mesh has to be stitched accordingly. Our solution is to include the curves along the cut into the set of curve segments defining the wireframe and to consistently process them as explained in the previous section.

3.6. Base Mesh Optimization

Figure 4 (a) illustrates a constrained triangulation of a given mesh region. The quality of the triangulation can be improved by inserting Steiner points inside the region. We use a heuristic that places additional points at approximate centers of maximal balls. For a mesh region enclosed by a set of 3D curves, we iteratively find vertices within the region with the largest minimum distance to the boundary curves (in 3D) and to previously computed Steiner points. We then project these points into the parameter domain and we use them as constraints during triangulation (see Figure 4 (b)). Steiner points are inserted one-by-one, until the largest minimum distance falls below a threshold.

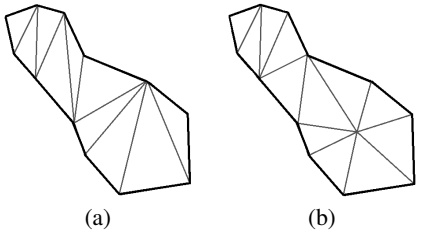


Figure 4: *Base mesh optimization: (a) constrained triangulation; (b) improved triangulation with one additional Steiner point.*

This type of optimization is aimed at improving the quality of the triangulation. Further optimization of the base mesh takes place during multiresolution analysis, as described in the next section.

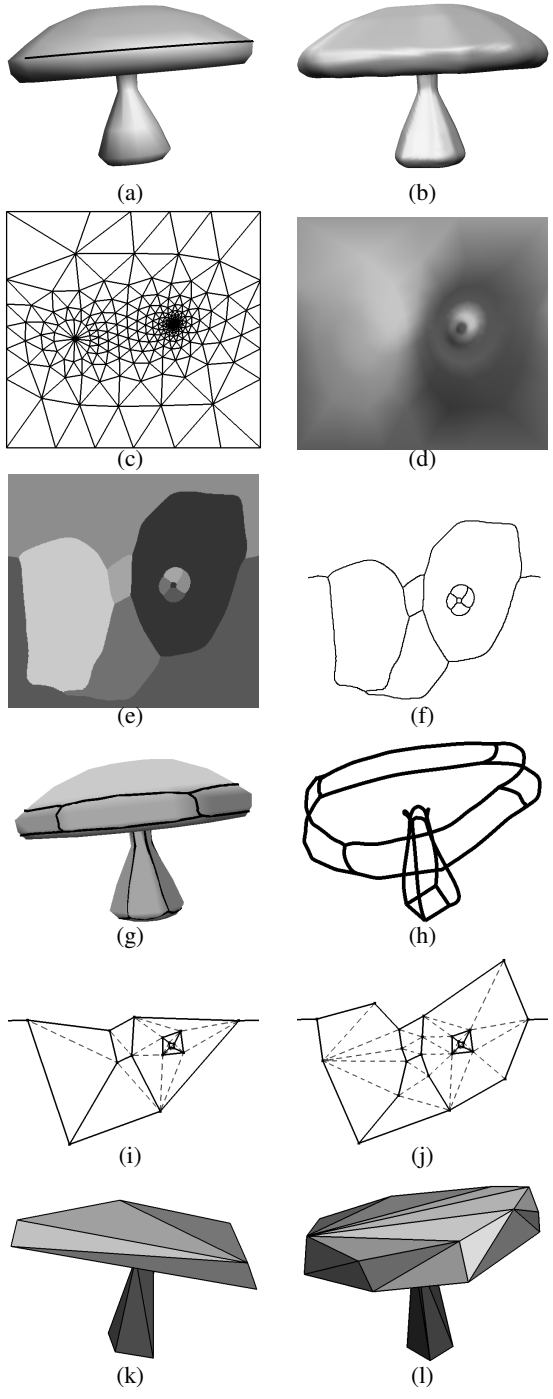


Figure 5: *(a) Irregular input mesh. (b) Multiresolution Loop result. (c) Planar parameterization (model was cut along the black curve shown in (a). (d) Shape image. (e) Quantized shape image. (f) Edge curves. (g) Decomposition corresponding to (e). (h) Wireframe corresponding to the curves in (f). (i), (j) two different degrees of curve simplification. Triangulation with constraints along the polylines (solid lines). (k), (l) Base domains corresponding to (i), (j), respectively. See also color section.*

4. Resampling and Analysis

As a primary application of the domain decomposition method presented, we describe the creation of a multiresolution hierarchy $\mathcal{H}$. Given a coarse polyhedral base domain K^0, $\mathcal{H}$ is built in two steps:

1. Compute data on the finest level of $\mathcal{H}$ by resampling the input mesh over the base domain K^0.
2. Fill in coarser levels of the hierarchy by multiresolution analysis.

Resampling Given a target number of levels L, $L-1$ steps of Loop subdivision are applied to the base domain K^0. This generates a smooth domain K^{L-1} with the same connectivity as the target control mesh H^{L-1} (see Figures 7 (e) and 10 (a)). For each domain vertex v_K its position v_K^{lim} and normal on the limit surface are computed. A sample in H^{L-1} is generated by intersecting the directed line through v_K^{lim} along the normal direction with the original surface. In either scenario, if multiple intersection points occur, we retain the closest one. If no intersection points are found (this may happen near boundaries of open meshes), we find the closest original mesh face to the position being resampled and we intersect the ray with the plane containing that face. More accurately, boundary curves of the original mesh could be projected onto the resampled multiresolution mesh which could then be trimmed to these curves following a method similar to the one proposed in [6]. We have yet to pursue this solution.

Note that other alternatives to resampling by normal shooting could be employed. For example, when the mesh being flattened is the input mesh, the resampling can be performed in the parametric domain. As discussed in the next section, this may not always be the case (e.g., if, for practical reasons, simplified versions of the input models are used to recover a base domain).

Multiresolution analysis The basic idea of multiresolution analysis is to decompose a function or signal into a smooth background and a collection of details. At a given scale, the original function is approximated by ignoring all details too small to be discerned at that scale. In a multiresolution mesh hierarchy, data on a coarse level H^{j-1} can be computed from data values on the finer level H^j by applying a restriction operator. Multiresolution details are computed at each level as the difference between the original positions on that level and the positions obtained by subdivision from the coarser level. The details are encoded with respect to local frames.

Fitting is used to further adjust the positions of control mesh vertices on coarser levels. The fitting procedure minimizes in least-squares sense the difference between values of the smooth surface evaluated at vertex positions on the finest level $L-1$ of the hierarchy and values obtained after applying $L-j-1$ steps of subdivision to the data on level j, as described in [4]:

$$\min_p \sum_{w \in V^{L-1}} ||[p]_w^{L-1} - [S^{L-j-1} p^j]_w||^2 \qquad (1)$$

where the minimum is computed over all possible choices of control points p^j, V^{L-1} is the set of vertices on the finest-level of the hierarchy, S^{L-j-1} is the subdivision matrix for $L-j-1$ subdivision steps, and $[]_w$ means that the resulting smooth surface is evaluated at parameter values corresponding to vertices w of the control mesh.

Resampling appearance attributes In addition to resampling geometry, we can also extract appearance attributes from the original model. An example is shown in Figure 11 (d): color information from the head of the Bastet cat is transferred onto the corresponding multiresolution representation. In this case, color data is associated only with vertices of the finest level mesh. We have not yet investigated the issue of restricting appearance data to coarser levels of the hierarchy with minimum impact on visual quality.

5. Discussion and Results

Performance statistics The following table illustrates the performance of our method. The measurements have been recorded on a Pentium IV 2.20GHz machine with 1.5GB of RAM. Angle-based flattening [38] was used to parameterize the face model. The other models were mapped onto the plane using the method of [7]. Domain decomposition measurements include image processing times (typical image resolution was 400×400 pixels or less), curve simplification, and triangulation. The base mesh extraction time includes computing a 3D mesh corresponding to the computed triangulation, optimization, and stitching.

Model	face	cat	duck	bunny	Venus
Number of faces:					
Input	1,686	14,569	5,004	69,666	100,000
Simplified input	N/A	1,705	N/A	1,756	1,718
Output base mesh	173	142	116	262	78
Processing times (seconds):					
Parameterization	10.496	8.692	35.354	14.387	9.339
Domain decomp.	7.171	6.970	13.520	9.073	7.371
Base mesh extrac.	0.691	0.371	1.021	0.591	0.230
Total	18.358	16.033	49.895	24.051	16.943

For some of the larger models we used simplified versions to recover the base domain. Simplification was done offline using the method of Gueziec [17] and took under 5 seconds in all cases. While many existing parameterization techniques can be carefully tuned to handle large models,

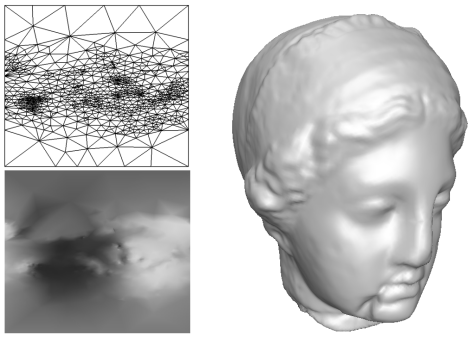


Figure 6: *(a) Parameterization (top) and shape map (bottom) for the Venus model. (b) Loop multiresolution hierarchy (control mesh on level five is shown). Input model courtesy of Cyberware, Inc.*

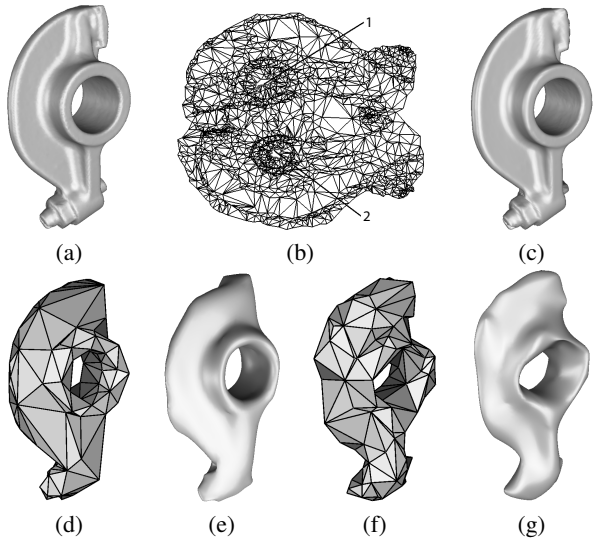


Figure 7: *(a) Rocker arm (model courtesy of Cyberware, Inc.). (b) Planar parameterization (after [9]) - two corresponding cut curves are labeled 1 and 2. (c) Loop multiresolution remesh (control mesh on level 4 is shown). (d) Base mesh. (e) Smooth domain for resampling. (f), (g) Base mesh and domain obtained by mesh simplification.*

using simplified versions is often a more practical choice. We typically apply domain decomposition to lower resolution models (still reflecting the general shape of the original, see Figure 11 (b)). Then we perform resampling with respect to the high-resolution input data. This technique produces very good results efficiently. In effect, the base mesh is being conformed to the general shape of the input and not to the high-frequency information typically captured by the fine tessellation.

Resampling was performed on the high resolution data, with running times on the order of several minutes, depending on the finest level used for subdivision.

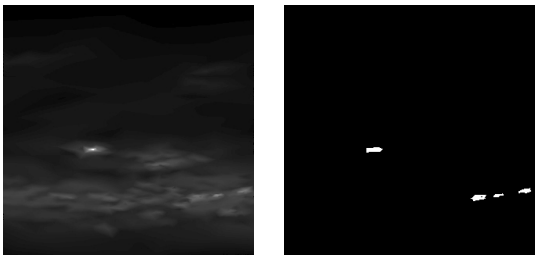
Remeshing quality By construction, the domain faces correspond to relatively flat regions of the input mesh. Hence, we expect the resampling to produce good results. For a given subdivision level, we compute the approximation error using the metric proposed in [14] which measures the average squared distance between the remeshed and the original models. For example, the approximation error for the bunny, Venus, and rocker arm models is less than 0.0015% of the input bounding box diagonal (0.0013%, $0.3e-4\%$, and $0.27e-5\%$, respectively) using five levels of subdivision.

Figure 7 illustrates comparatively the base meshes and the corresponding smooth domains obtained with our method (in (d) and (e)) and using a tolerance guided simplification method [17], respectively (in (f) and (g)) with a similar base face count.

Choice of parameters The current implementation depends on several parameters. The default values for some of the most important ones are briefly mentioned. The default parameterization is angle-based flattening [38]. The number K of color clusters is set to 14, corresponding to the 14 directions defined by the origin and the face centers and vertices of the unit cube. The curve simplification tolerance ε is a percentage of the model bounding box diagonal. For some of the examples these values were altered for illustration purposes. Thus, only 6 color clusters were used for remeshing the mushroom in Figure 5. Three different parameterization methods are shown. The effects of varying ε are reflected in Figures 5 (i) and (j).

Limitations of the approach The main limitation of our approach derives from the fact that it requires flattening the input model. It is well-known that, in general, planar parameterizations of arbitrary surfaces cannot be simultaneously conformal and authalic. Since most existing methods aim for conformality (a notable exception is [9]), certain portions of the input model are likely to suffer from severe area distortion. Figure 10 (a) shows a smooth base domain for the bunny model obtained with our method. Note the undersampling of the bunny's left ear due to severe compression of its surface area in parameter space. Our solution to this problem is briefly described next.

Layered parameterization A *zoom lens* approach allows us to process severely distorted regions in the parameterization separately. We use a stretch map (similar to [36, 3]) to identify severely distorted regions in the shape image. Figure 8 (a) shows the stretch image corresponding to the parameterization of the bunny model. Regions in this image which suffer from severe distortion are isolated through simple image



(a) (b)

Figure 8: *(a) Stretch map for the bunny model. (b) Severely distorted areas are segmented using contrast enhancement and thresholding. This image is used a mask in the zoom lens approach (see Figure 10(b)).*

processing operations: contrast enhancement and thresholding. The result is a black-and-white image in which clusters of white pixels correspond to highly compressed mesh regions (see Figure 8). We use this image as a mask to extract the corresponding submeshes and we process them separately, as illustrated in Figures 10 (b) and (c). The domain decomposition method is first applied to the submesh corresponding to the marked region (e.g., the left ear of the bunny): the submesh is flattened, its shape image is computed, color quantized and polyline constraints are detected. Second, the decomposition method is applied to the rest of the mesh. The two sets of polylines are merged and a triangulation is performed as previously described. Figure 10 (b) illustrates the two shape images corresponding to the left ear (right image) and to the rest of the bunny (left image). The ear polyline set is shown in 3D in Figure 10 (c) for illustration purposes. The resulting base mesh is shown in the color section. The final Loop multiresolution mesh is shown in Figure 10 (d).

6. Conclusions and Future Work

In this paper we present an image-based approach to domain decomposition and its application to multiresolution semi-regular mesh extraction. Our approach takes advantage of the information stored in a shape image corresponding to the model to find regions of relatively constant shape on top of which base domain faces are built. The extraction of a base mesh is very efficient, as most computations are carried out in image space through simple image processing operations.

The multiresolution meshes generated by our method can be used by various algorithms designed to operate on this representation, such as interactive editing and boolean operations [4, 5]. Variations of our technique are useful for other applications that require mesh partitioning and feature extraction. For example, by varying K in the quantization phase, different partitions of the model can be obtained for use in methods involving spatial queries, from ray tracing to collision detection. Similarly, the wireframe generated can be used as a standalone minimal representation of the model or for further processing.

We envision a number of enhancements:

1. The method is applicable to the generation of quadrilateral meshes and Catmull-Clark multiresolution hierarchies. In fact, in the current implementation we can easily generate a quad base domain by splitting each base triangle into three quads. However, this technique is not very efficient as it triples the number of base mesh faces and it introduces a large number of valence 3 vertices. Instead, we are investigating quadrangulations with constraints which will likely yield superior quad meshes.
2. Different decompositions can be achieved by storing different kinds of shape information into the shape image. For instance, segmentation based on model curvature or other scalar / vector field defined over the mesh may be useful.
3. We also plan to address the problem of exactly aligning mesh edges with sharp features. This functionality can be added to our method by tagging sharp edges in the planar parameterization and using them along with the rest of the constraints during triangulation.

Acknowledgements

I wish to thank the anonymous reviewers for their competent and constructive observations. I am grateful to Fausto Bernardini and Denis Zorin for their valuable suggestions and to Gabriel Taubin for his help with the computation of stretch maps.

References

1. P. Alliez, D. Cohen-Steiner, O. Devilliers, B. Levy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM TOG. Special issue for SIGGRAPH conference*, 2003.
2. P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. *ACM TOG. Special issue for SIGGRAPH conference*, 21(3):347–354, 2002.
3. L. Balmelli, G. Taubin, and F. Bernardini. Space-optimized texture maps. In *Proceedings of Eurographics'02*, 2002.
4. H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *Proceedings of SIGGRAPH 01*, pages 185–194, August 2001.
5. H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM TOG. Special issue for SIGGRAPH conference*, 21(3):312–321, 2002.
6. H. Biermann, I. Martin, D. Zorin, and F. Bernardini. Sharp features on multiresolution subdivision surfaces. *Graphical Models*, 64(2):61–77, 2002.
7. C. Brechbühler, G. Gerig, and O. Kübler. Parametrization of closed surfaces for 3-D shape description. *CVIU*, 61(2):154–170, 1995.

8. M. M. Chang, M. I. Sezan, and A. M. Tekalp. Adaptive bayesian segmentation of color images. *Journal of Electronic Imaging*, 3:404–414, 1994.

9. M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. In *Eurographics conference proceedings*, pages 209–218, 2002.

10. D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

11. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH 95*, pages 173–182, 1995.

12. J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. In *SOCG 2002*, 2002. 244–253.

13. M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

14. M. Garland and P. S. Heckbert. Surface simplication using quadric error metrics. In *Proceedings of ACM SIGGRAPH 97*, pages 209–216, August 1997.

15. M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of ACM Symp. on Interactive 3D Graphics*, 2001.

16. X. Gu, S. Gortler, and H. Hoppe. Geometry images. *ACM TOG. Special issue for SIGGRAPH conference*, 21(3):355–35, 2002.

17. A. Guéziec. Locally toleranced surface simplification. *IEEE TVCG*, 5(2), 1999.

18. I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *Proceedings of SIGGRAPH 00*, pages 95–102, July 2000.

19. J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. In *Proc. of the 5th International Symposium on Spatial Data Handling*, volume 1, pages 134–143, 1992.

20. K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. pages 153–162, 2000.

21. K. Hormann, U. Labsik, and G. Greiner. Remeshing triangulated surfaces with optimal parameterizations. *Computer-Aided Design*, 33(11):779–788, 2001.

22. T. Kanai. Meshtoss: Converting subdivision surfaces from dense meshes. In *Proc. Modeling and Visualization 2001*, pages 325–332, 2001.

23. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, 1995.

24. L. P. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink wrapping approach to remeshing polygonal surface. In *Proceedings of Eurographics'99)*, volume 18(3), pages 119–130, 1999.

25. V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of SIGGRAPH 96*, pages 313–324, 1996.

26. A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proceedings of SIGGRAPH 00*, pages 85–94, July 2000.

27. A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. In *Proceedings of SIGGRAPH 98*, pages 95–104, July 1998.

28. B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM TOG. Special issue for SIGGRAPH conference*, 21(3), 2002.

29. N. Litke, A. Levin, and P. Schröder. Fitting subdivision surfaces. In *Proceedings of IEEE Visualization 2001*, pages 319–324, October 2001.

30. C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

31. W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings of SIGGRAPH 87*, pages 163–169, 1987.

32. M. Lounsbery, T. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM TOG*, 16(1):34–73, January 1997.

33. N. Mustafa, E. Koutsofios, S. Krishnan, and S. Venkatasubramanian. Hardware assisted view-dependent planar map simplification. In *Proc. of the 17th ACM Symposium on Computational Geometry*, 2001.

34. S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 34(2):211–262, 2002.

35. K. Pulli and M. Lounsbery. Hierarchical editing and rendering of subdivision surfaces. Technical Report UW-CSE-97-04-07, Dept. of CS&E, University of Washington, Seattle, WA, 1997.

36. P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 00*, pages 409–416, 2001.

37. S. Z. Selim and M. A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Trans. on PAMI*, 6, 1984.

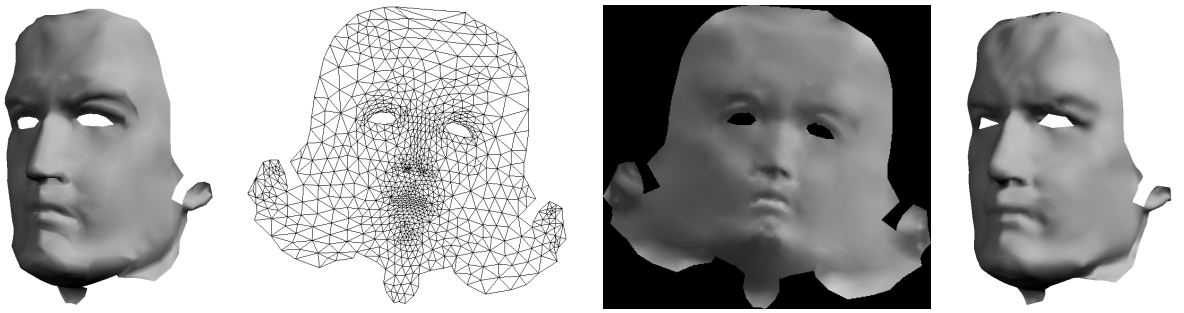
38. A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17(3):326–337, 2001.

39. A. Sheffer and J. Hart. Seamster: Inconspicuous low-distortion texture seam layout. In *IEEE Visualization*, pages 291–298, 2002.

40. J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Comp. Geom.*, volume 1148, pages 203–222. 1996.

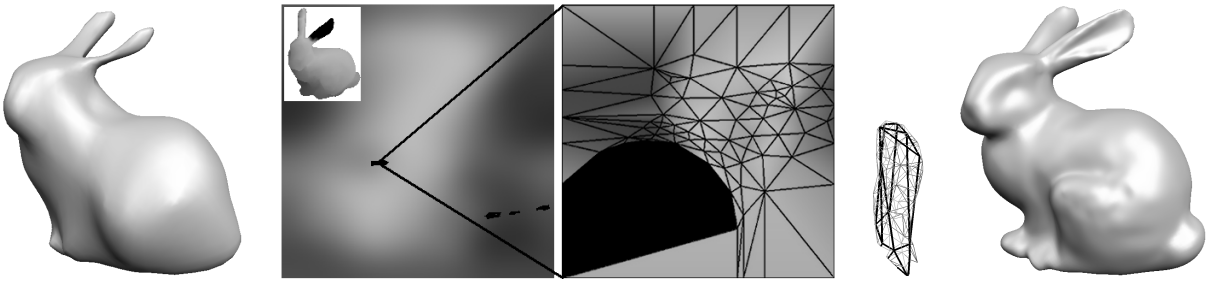
41. D. Zorin, P. Schröder, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. SIGGRAPH'00 Course Notes, 2000.

42. D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proceedings of SIGGRAPH 97*, pages 259–268, August 1997.



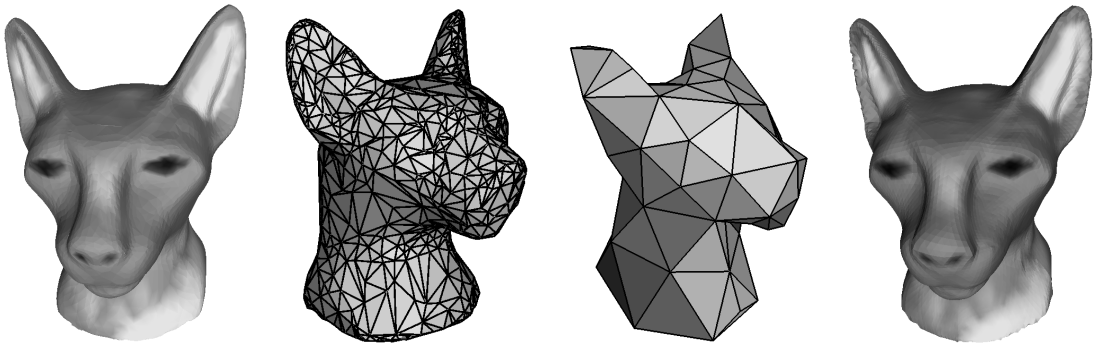
(a) (b) (c) (d)

Figure 9: *(a) Input face model. (b) Planar parameterization using angle-based flattening. (c) Normal map with holes. (d) Loop multiresolution hierarchy (control mesh on level three is shown). See also color section.*



(a) (b) (c) (d)

Figure 10: *(a) Smooth base domain generated for the bunny model. Note the undersampled left ear due to area distortion. (b) Zoom lens approach: area distorted region is masked out of the bunny's normal map (left) and the corresponding geometry (in black on the left inset) is processed separately: the right image shows the parameterization corresponding to the ear geometry overlayed on top of its normal map. (c) Simplified wireframe recovered for the ear is shown with thick black lines. Constraints along the corresponding parametric segments are added to the set of constraints for the rest of the bunny. (d) Multiresolution Loop reconstruction with 5 levels (details of the original mesh are sampled on the third level). Input model courtesy of Stanford University. See also color section.*



(a) (b) (c) (d)

Figure 11: *(a) Scanned model of the Bastet cat. (b) Simplified model used as input for domain decomposition. (c) Base mesh extracted with our method. (d) Loop multiresolution hierarchy with four levels. Color attribute is resampled on the finest level. See also color section.*