EUROGRAPHICS 2006 / E. Gröller and L. Szirmay-Kalos
(Guest Editors)

Volume 25 (2006), Number 3

A Robust Two-Step Procedure for Quad-Dominant Remeshing

Martin Marinov Leif Kobbelt

Computer Graphics Group, RWTH Aachen, Germany

Abstract

We propose a new technique for quad-dominant remeshing which separates the local regularity requirements from the global alignment requirements by working in two steps. In the first step, we apply a slight variant of variational shape approximation in order to segment the input mesh into patches which capture the global structure of the processed object. Then we compute an optimized quad-mesh for every patch by generating a finite set of candidate curves and applying a combinatorial optimization procedure. Since the optimization is performed independently for each patch, we can afford more complex operations while keeping the overall computation times at a reasonable level. Our quad-meshing technique is robust even for noisy meshes and meshes with isotropic or flat regions since it does not rely on the generation of curves by integration along estimated principal curvature directions. Instead we compute a conformal parametrization for each patch and generate the quad-mesh from curves with minimum bending energy in the 2D parameter domain. Mesh consistency between patches is guaranteed by simply using the same set of sample points along the common boundary curve. The resulting quad-meshes are of high-quality locally (shape of the quads) as well as globally (global alignment) which allows us to even generate fairly coarse quad-meshes that can be used as Catmull-Clark control meshes.

Categories and Subject Descriptors (according to ACM CCS):
I.3.5 [Computer Graphics]: Geometric algorithms, languages, and systems

1. Introduction

Remeshing algorithms are considered a fundamental component of the contemporary geometry processing pipeline. Traditionally, the focus of their development was mostly the quality of the mesh elements, i.e., shape and regularity. Indeed, various CAD and CAE applications would perform poorly, or even crash, if a bad mesh was fed to them as an input. Hence, improving solely the performance with respect to this *local* quality criterion was the predominant goal several years ago. To satisfy the requirements, compromises were made both with regards to the complexity of the output, i.e., dense remeshes were needed to provide enough degrees of freedom for the optimization, as well as with the alignment of the mesh faces to the structure of the input model, leading to the wide use of the term "unstructured" quad/triangle mesh.

Recently, the focus of the research in this area has shifted to a more challenging problematic: Is it possible to remesh the input surface producing sufficiently regular faces, and at the same time, preserve its structural characteristics? A crucial factor for this change was the observation that geometry processing applications can significantly reduce the production time and expense in several CAD domains, e.g, rapid prototyping, reverse engineering and conceptual design. By supplying CAD designers with tools to perform and test modifications without iterating the typical prototyping process, remeshing algorithms become employed to provide a starting point in the design of a CAD model. As a consequence, *global* alignment of the output remesh with the features of the processed model is now regarded as a quality criterion at least as important as the local shape of the elements. In addition, providing low complexity meshes which satisfy both criteria is a very important problem, since such meshes can be easily converted to subdivision or spline control meshes which are the representation of choice in many CAD systems.

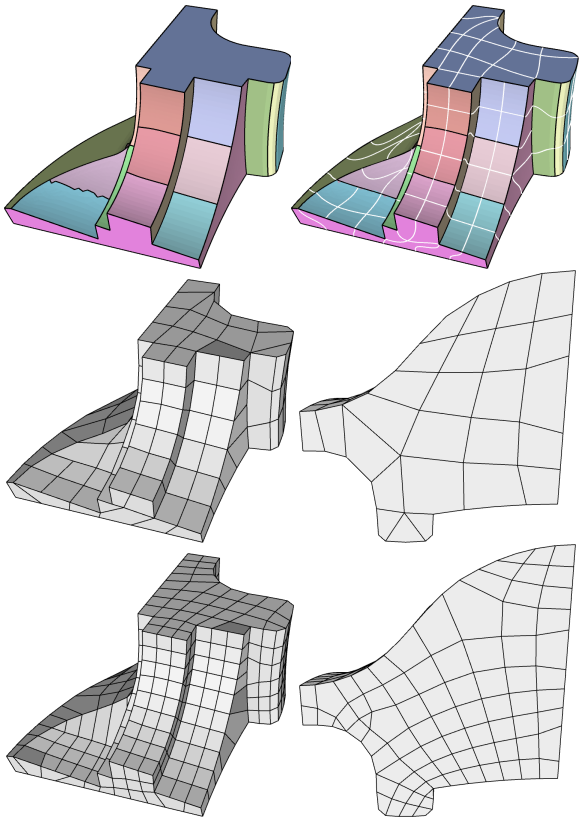

Figure 1: Top row: *Segmentation of the Fan model into* 30 *regions* (left) *and minimum bending energy cubics computed within them* (right). Middle and bottom row: *Remeshing with target edge length about* 10% *and* 5% *of the bounding box diagonal. Note the local regularity of the elements in the complex top and bottom segments achieved at these low resolutions.*

1.1. Contributions

We present a novel remeshing algorithm which separates the local regularity optimization from the global alignment requirements by splitting the process into two steps: In the first step we segment the input mesh M into a set of structure-preserving regions using a slight modification of the VSA method [CSAD04]. In the second step we generate a quad-dominant mesh inside a 2D parameter domain for every region. These meshes conform to a set of automatically defined constraints (vertices & edges) at the boundaries of the regions, which enables their seamless composition on the input 3D surface. The edges of the quad-dominant mesh are placed along smooth curves connecting pairs of constrained vertices, while additional internal vertices are defined at the intersection points of these smooth curves. We state explicitly at this point that the approximation efficiency of the output meshes is *not* a priority of our technique. Instead, we focus on generating meshes which are useful for CAD applications requiring conversion of scanned geometry to regular, structure-aligned quad-dominant representations, e.g., Catmull-Clark [CC78] subdivision surfaces. In particular, our algorithm has the following properties:

- It is able to generate consistent quad-dominant meshes at coarse, intermediate and dense resolutions, which are well suited as Catmull-Clark control meshes and for FEM computation (Fig. 1). Previous works, e.g., [ACSD*03, MK04], often fail to provide consistent remeshes at low resolutions which are required for meaningful subdivision surface modeling and approximation.
- The quality of the remeshing results can be controlled much more robustly: Since our remeshes conform to the input segmentation structure, features represented by the segmentation are naturally preserved by the final remesh as well (similar to the triangular remeshing techniques [BK01, AMD02]). Furthermore, the local element regularity is not affected by small-scale geometric features, which are intentionally ignored when specifying a certain region budget for the segmentation procedure.
- It does *not* depend on an input (curvature) tensor field estimated on the discrete input mesh, which is often of insufficient quality due to noise caused by the input geometry sampling process. Although the estimated tensor field can be filtered, this process often leads to considerable changes of its topology and hence, incorrect placement of singular points on the surface. In addition, large isotropic areas do not bear any meaningful (in the context of the anisotropic remeshing techniques) curvature information which requires various workarounds, e.g, Delaunay triangulations [ACSD*03] or geodesic line integration [MK04].
- Our algorithm has a simple formulation, which makes its practical implementation relatively easy, especially when compared to the significant engineering effort required for implementing previous quad-dominant remeshing techniques. In particular curves are no-longer integrated tangentially along a predefined vector field, but instead are simply computed as the minimum bending energy cubics connecting two points with clamped boundary conditions. This avoids a significant burden from an implementation point of view, e.g., dynamic line collision detection.

1.2. Related work

The number of published remeshing techniques is vast and a detailed survey on this topic [Owe98, AUGA05] is well beyond the scope of this paper. Therefore we focus mainly on works generating quad-dominant meshes or having some structure-preserving properties.

2D Quad remeshing: The pioneering techniques for quad-dominant remeshing were developed for FEM applications and work by transforming a triangle mesh to a quadrilateral

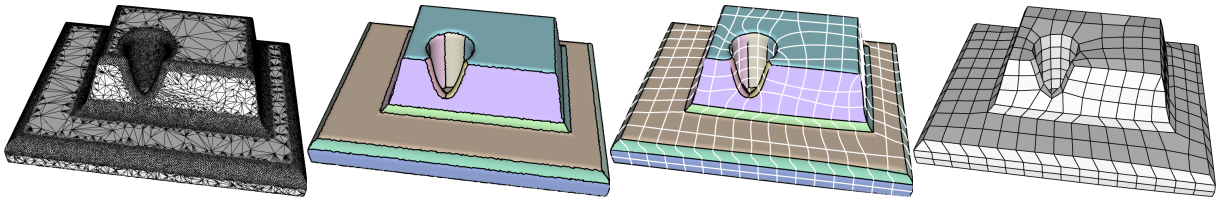

Figure 2: Left: *The input CAD mesh exhibits high irregularity and significant amount of noise.* Middle left: *The segmentation of the mesh into* 28 *regions highlights the important structural features of the model.* Middle right: *A set of computed minimum bending energy curves, connecting constrained samples at the region boundaries, is selected to form an as regular as possible quad-dominant mesh in the parameter domain of every region.* Right: *The quad-dominant meshes are composed together to form the final output mesh.*

one: In [BS91, OSCS98] an advancing front approach propagates triangle-to-quad transformations from the boundary to the interior of the mesh. Borouchaki and Frey [BF98] developed a method which adaptively fills a 2D polygon with a triangle mesh and then merges triangles to quads. Shimada et al. [SLI98] pack the contours of the input mesh with square cells and convert them to a quad-dominant mesh. Our method also employs a 2D quad-remeshing algorithm, however, instead of using a triangle-to-quad mesh conversion, we directly construct the elements by intersecting a set of curves in the parameter domain. This constructive approach allows us to operate more flexibly when optimizing the regularity of the output mesh.

Simplification: Garland and Heckbert showed in [HG99] that the QEM simplification [GH97] asymptotically aligns triangles with the anisotropy of the simplified model. Eck and Hoppe [EH96] merge triangles by solving a graph matching problem on a simplified coarse triangle mesh to form a quad-dominant B-spline patch layout used for approximation. Clustering of the input mesh faces can be used to generate either a mostly hexagonal mesh [BMRJ04] which is then converted to a quad-dominant one, or a triangular mesh [CSAD04] and later merge triangles to quads. In [MK05] a polygonal mesh simplification method is employed to generate structure-preserving coarse meshes which are then partitioned to quad-dominant ones. Similarly to these methods, our technique is able to provide structure-preserving remeshes at coarse resolutions, however, it is able to optimize the element regularity of the output as well.

Feature preservation: Alignment of the remesh with sharp features was specifically pursued in [BK01]. In [AMD02] sharp features are preserved by constraining edges to be aligned to them during the remeshing. By placing constraints on the boundaries between the regions, our algorithm preserves the important sharp features detected during the segmentation step.

Curve-based remeshing: Recently, Alliez et. al proposed a new method [ACSD*03] which integrates lines of principal curvature in the parameter domain of the input surface and intersects them to form an anisotropic quad-dominant remesh. This technique was extended in [MK04] to integrate the principal lines directly on the input mesh. Dong et al. [DKG05] replace the curvature tensor field with the gradient of a smooth harmonic scalar field defined on the surface. Although we do not align edges to predefined vector fields, our algorithm still relies on computing and intersecting curves to define the edges and vertices of the quad-dominant mesh.

Parameterization: Gu et al. [GGH02] generate a fully regular quad mesh for genus zero surfaces by unfolding the surface to a planar rectangle and resampling it by following a regular grid in the parameter domain. In [SWG*03] this process is extended by first segmenting the surface, and then resampling each patch separately. Finally the output remeshes are zippered to form a consistent mesh. In [KLS03] a globally smooth parameterization is defined on a base domain obtained through structure-preserving simplification and then it is used to produce a semi-regular triangular remesh. Ray et al. compute a periodic global parameterization [RLL*06] of the input surface aligned to the principal curvature vector fields and use it to produce an all quad remesh with T-joints. In [DBG*06] a globally smooth parameterization of the input mesh is computed using a quad domain obtained by spectral decomposition of the surface. Our approach also employs parameterization to sample the input mesh in the interior of each region, however, mesh faces are not aligned to the artificial boundary constraints typically used to constrain the parameter domain (rectangles or circles). Instead, we use [LPRM02] to compute a boundary-free parameterization and then map edges aligned to the features of the model to the parameter domain, which allows us to avoid the well-known reconstruction artifacts which usually emerge from mapping jagged patch boundaries in 3D to smooth boundary curves in the 2D domain.

2. Overview

Given an input triangle mesh M, we first perform a segmentation $S = \{R_0 \dots R_n\}$ of M using a slightly modified version of the VSA [CSAD04] method (Section 3, Fig. 2, left). For every region we compute a conformal map using [LPRM02].

Next, we approximate the regions' boundaries with a network of smooth 3D curves. On every boundary curve we pick equidistantly a set of sample points which are then mapped to the parameter domain of the regions adjacent to it (Section 4). From this point on, we treat every region R_i separately: First, for every pair of sample points in the parameter domain Ω_i of R_i, a minimum bending energy cubic curve is computed, so that it is orthogonal to the tangent of the boundary curves at the sample points (Section 5). Then, from the computed set of cubics inside Ω_i, a subset L is selected through combinatorial optimization minimizing an energy functional $E(L)$ (Section 6, Fig. 2, right). $E(L)$ evaluates for each configuration of elements (polygons) defined by the intersections of the curves in L a shape quality score which penalizes non-regular elements, i.e., low (near-zero) values correspond to "nice" quads like squares, rectangles and parallelograms, while triangles, pentagons and distorted quads are penalized by higher values. The distribution of the final mesh samples is further optimized through relaxation in the parameter domain before mapping them back to the input mesh (Section 7).

3. Mesh segmentation

Global alignment, i.e., alignment of the elements along the structural features of the input model, is often an underrated factor when evaluating the quality of a remeshing algorithm. However, it is an important property for many applications which employ the remesh as a source for degrees of freedom for modeling scanned objects, e.g., conceptual design and rapid prototyping, since the mesh structure determines the support of the deformation basis functions. In order to ensure this property, the first step of our algorithm is to segment the input mesh M into a set of regions $S = \{R_0 \dots R_n\}$ so that every region R_i bears a semantic and geometrical meaning, e.g., R_i represents a feature or a (part of a) structural component of the surface represented by M. Therefore, it makes sense to generate a remesh for every R_i separately — this way we can afford to perform global mesh optimization within the boundaries of the corresponding structural entity.

Moreover, by ensuring that the surface geometry represented by a given R_i is relatively flat, we can safely map the corresponding cluster of faces to the plane and remesh it there by computing an element structure independent from the 3D geometry. This is motivated by the fact that controlling the shape of a set of elements is significantly more efficient, robust and simple in the 2D domain than in 3D, e.g., testing a general 3D polygon for convexity is not a robust process without first mapping it to a plane.

To obtain such a segmentation automatically is indeed a challenging problem, but the recently published VSA method [CSAD04] performs quite close to our requirements: It generates a set of regions which align to the structure of the input model and at the same time represent relatively flat geometrical features. In addition, when using the $L^{2,1}$ error metric, the region boundaries are correctly aligned locally to the sharp features of the input model. Hence, by forcing the quad-remeshing for each region to conform to its boundaries (Section 4), our algorithm preserves sharp features without any additional effort.

Since the VSA algorithm is a global relaxation method, initialized by spreading heuristically a set of seeds on the input surface, it can easily get stuck at a local minimum. The solution often can be further improved by performing region *teleporting,* which helps the algorithm to escape such situations. However, teleporting might lead to worse results as well, or more generally can be made more efficient if we carefully monitor its performance. Therefore, we modified slightly this aspect of the VSA algorithm: First, we track the behavior of the global quality of the segmentation. Each teleporting is first performed tentatively and accepted only if we indeed detect an improvement after a few more relaxation steps. Otherwise the region teleporting is rejected and only relaxation steps are further iterated until a new local minimum is found.

4. Remeshing constraints setup

Although we process every region R_i separately in its parameter domain Ω_i, we have to glue back in 3D the remeshes of each R_i to form a consistent polygonal mesh approximating the input geometry. To resolve this dependency we follow a classic strategy: we setup a set of constraints on the boundaries between each two adjacent regions R_i and R_j, map these constraints to the corresponding parameter domains Ω_i and Ω_j and then generate quad dominant remeshes Q_i, Q_j for R_i and R_j separately so that they conform to the predefined boundary constraints. The boundary constraints are a sequence of vertices (and edges connecting them), smoothly approximating the shape of the edge paths defining the boundaries of the given region R_i.

Segmentation connectivity: Our first step is to identify the topology of the segmentation S. The vertices of S (a.k.a. *anchor* vertices) are the subset A of vertices of M, such that for every $a_i \in A$ there are at least three (two if a_i is boundary vertex of M) adjacent faces belonging to different regions. The arcs of S separate adjacent regions and are identified as the *unique* path of edges $E^{i,j}_{k,l}$ on M which divides two adjacent regions R_i and R_j and connects two anchor points a_k and a_l (this index notation is required since there might be more than one edge path separating two adjacent regions). As all such paths $E^{i,j}_{k,l}$ represent the complete topology between the regions R_i and R_j, they are the natural location to define the consistency constraints for our quad-remeshing procedure inside R_i and R_j.

Constraints computation: Typically, an edge path dividing two regions on the input mesh has poor smoothness properties — either due to noise, coarse mesh resolution or local mismatch of the segmentation procedure. Therefore, if we

are to pick the set of constrained vertices directly on this jagged edge path, without considering the smoothness of the resulting sample sequence, we might end up with a rather poor set of constraints, which in turn will compromise the quality of the quad remeshing in the interior of the region. Among the many possible ways to resolve this issue we choose one that fits most naturally to our framework: Given an edge path $E_{k,l}^{i,j}$ along the set of vertices $\{v_0 \dots v_m\}$, where $v_0 = v_k$ and $v_m = v_l$, we approximate it in least squares sense with a 3D cubic curve $C \equiv C_{k,l}^{i,j}$ by minimizing the integral L^2 metric:

$$\sum_{i=0}^{m-1} \int_{s_i}^{s_{i+1}} \left\| \frac{s_{i+1}-t}{s_{i+1}-s_i} v_i + \frac{t-s_i}{s_{i+1}-s_i} v_{i+1} - C(t) \right\|^2 dt, \quad (1)$$

under the constraints $C(s_0) = v_0$ and $C(s_m) = v_m$. Here $\{s_0 \dots s_m\}$ is the arc-length parameterization of the vertices $\{v_0 \dots v_m\}$, i.e., $s_i = \sum_{j=0}^{i-1} \|v_j - v_{j+1}\|$. Although C does not lie on the input mesh M, for practical purposes it is sufficiently close to it. In case there is a hard requirement for all samples to lie on the input mesh, a cubic curve can be always replaced with a cubic spline curve and one can interleave the fitting step with a projection technique like in [HP04] to get a curve lying precisely on M.

We now sample the 3D curve C by picking r equidistant points $\{c_1 \dots c_r\}$ on it. These points $\{c_0 = v_0, c_1 \dots c_r, c_{r+1} = v_m\}$ constitute the set of constraints (3D vertex positions) which will be interpolated by the quad-remeshes computed independently for the regions R_i and R_j in order to join them seamlessly when composing the final output mesh.

Constraints mapping: Evaluating element shape quality in 2D is a significantly less complicated problem than in 3D. Since our region quad-remeshing algorithm employs a global optimization strategy for the shape quality of *all* generated elements, it pays off from both the efficiency and robustness point of view to initially map the region's geometry to the plane. Therefore, using LSCM [LPRM02], we compute a parameterization domain Ω_i and a conformal parameterization $\chi_i : \Omega_i \rightarrow R_i$ for each region R_i. Note that the unfolding distortion is typically low due to the properties of the segmentation algorithm (Section 3).

Since the 3D positions $\{c_1 \dots c_r\}$ sampled from a 3D cubic curves $C_{k,l}^{i,j}$ approximating an edge boundary $E_{i,j}^{k,l}$ of R_i do *not* lie in general on M (by construction only c_0 and c_{r+1} are points on M), we need to define their parameterization in the parameter domains Ω_i in order to force the quad remeshing of R_i to interpolate them. To avoid compromising the smoothness of our constrained edge sequence $\{\overline{c_0c_1} \dots \overline{c_rc_{r+1}}\}$ due to the distortion associated with the parameterization, we compute a smooth planar cubic curve $\tilde{C}$ approximating each incident edge path, this time in the parameter domain Ω_i. We use exactly the same energy functional (1), replacing the 3D positions v_i with their 2D parameter values $\tilde{v}_i$. Again, we impose the same interpolation constraints $\tilde{C}(s_0) = \tilde{v}_0$ and $\tilde{C}(s_m) = \tilde{v}_m$, but most importantly, we *keep* the same arc-length *parameterization* $\{s_0 \dots s_m\}$, which we used when solving (1) for the 3D curve $C_{k,l}^{i,j}$. Hence, the curve $\tilde{C}$ is a smooth preimage of C inside the parameter domain Ω_i. Let $\{t_0 = 0, t_1 \dots t_r, t_{r+1} = s_m\}$ be the parameter values of the constrained points $\{c_0 = v_0, c_1 \dots c_r, c_{r+1} = v_m\}$. Then, by construction, the points $\{\tilde{c}_0 = \tilde{C}(t_0) = \tilde{v}_0, \tilde{c}_1 = \tilde{C}(t_1) \dots \tilde{c}_r = \tilde{C}(t_r), \tilde{c}_{r+1} = \tilde{C}(t_{r+1}) = \tilde{v}_m\}$ are their parameter values in Ω_i. Note that although some of the points $\{\tilde{c}_1 \dots \tilde{c}_r\}$ might lie outside the boundaries of the Ω_i, we *never* need to evaluate the parameterization χ_i at $\{\tilde{c}_0 \dots \tilde{c}_{r+1}\}$ — we already know the corresponding 3D positions $\{c_0 \dots c_{r+1}\}$.

5. Curves network computation

Curve-based remeshing is a powerful paradigm for generating quad-dominant meshes, already employed in several previous methods: [ACSD*03, MK04, DKG05]. In these works, the curve network is computed by integration along some estimated (or predefined) vector field on the surface. Especially for the anisotropic remeshing techniques, this integration procedure is quite complicated: One needs to continuously test if the currently computed curve is going to collide with itself (or another, already integrated curve) or to approach a singular point on the vector field, where special treatment is required. Moreover, due to the nature of the forward integration methods, the trajectory of the curve cannot be constrained to more than one point on the surface. Hence, in our algorithm, we use a much simpler and more robust approach to define the curve network: We simply connect pairs of boundary points (with tangents prescribed at them) by a minimum bending energy cubic curve.

More precisely, given a segmentation region R, its parameter domain Ω and the boundary constraints (the parametric cubic curves $\{\tilde{C}_0 \dots \tilde{C}_n\}$ and the sample points on them $\{\tilde{c}_0 \dots \tilde{c}_m\}$), we compute $O(m^2)$ minimum bending energy cubic curves connecting each pair of boundary points $(\tilde{c}_i, \tilde{c}_j)$. A boundary point $\tilde{c}_i$ is considered for connection iff the internal angle $\alpha_i = (\tilde{c}_{i-1}, \tilde{c}_i, \tilde{c}_{i+1})$ is larger than some user defined threshold ψ ($\psi > \pi/2$ for all our examples). This is motivated by the simple observation that if we are to split α_i by an edge (along a curve), then at least one of the resulting angles will be too small and this will lead to poorly shaped adjacent elements. A natural choice for prescribing a tangent at $\tilde{c}_i$ is the normal of the corresponding boundary curve $\tilde{C}$ at the same parameter value. However, if α_i is too large, that is if $\tilde{c}_i$ is a concave boundary point, then we actually allow two tangents (and two simultaneous connections) for the same boundary point, since in this case $\tilde{c}_i$ can accommodate two adjacent (interior) elements with sufficiently large angles. These two tangents are chosen so that they trisect the angle α_i (Fig. 3).

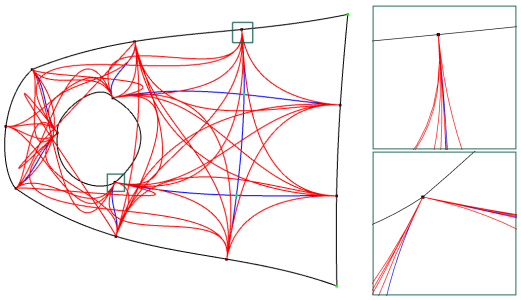

Figure 3: Left: *Computed lines for a region on the Rocker arm model. Note the concave constrained points on the internal (hole) contour. Since the incident angle is too large, connections interpolating two different tangents are allowed.* Right: *Zoom-in view of a convex* (top) *and a concave* (bottom) *constrained point.*

To compute the minimum bending energy cubic curve $b_{i,j}$ connecting the points $(\tilde{c}_i, \tilde{c}_j)$ with the corresponding tangents $(\vec{q}_i, \vec{q}_j)$, we leverage its Bèzier form:

$$b_{i,j}(t) = \tilde{c}_i B_0^3(t) + (\tilde{c}_i + \lambda\vec{q}_i) B_1^3(t) + (\tilde{c}_j + \mu\vec{q}_j) B_2^3(t) + \tilde{c}_j B_3^3(t)\,.$$

Minimizing the bi-variate non-linear functional $E(\lambda,\mu) = \int_0^1 \kappa_{i,j}^2(t)dt$, where $\kappa_{i,j}$ is the curvature of $b_{i,j}$, under the constraints $\lambda,\mu \geq 0$ is a simple problem that can be solved efficiently by a standard solver, e.g., [SKW85]. Avoiding self-intersecting curves is easily achieved by conservatively constraining the minimization procedure to ignore a search direction whenever the segments $[\tilde{c}_i,\, \tilde{c}_i + \lambda\vec{q}_i]$ and $[\tilde{c}_j,\, \tilde{c}_j + \mu\vec{q}_j]$ of the control polygon of $b_{i,j}$ intersect. During the minimization, we do not test if $b_{i,j}$ is completely in the interior of (the constrained boundary loops of) R. Instead, once the minimization is completed, we simply discard the curve if it intersects some of the boundary loops defined by the constrained points $\{\tilde{c}_0 \ldots \tilde{c}_m\}$. This is usually not a problem since due to the choice of the tangents most of the computed curves lie inside R and hence we have sufficiently many curves to select for our final remeshing.

6. Curve selection algorithm

The shape of the generated elements is the criterion typically used to rate the quality of a remeshing technique. Indeed, the mesh faces' shape affects crucially the performance of various applications such as subdivision surfaces modeling and FEM. Therefore, the curve selection algorithm described in this section aims at finding such a subset L of all computed candidate curves internal to the region R, so that the quad-dominated remesh Q generated by intersecting the curves in L minimizes a shape quality functional $E(L)$ (Fig 4).

Meshing: The meshing procedure by itself is very simple: we first compute the intersections P of all curves in L. Each curve $b_{i,j} \in L$ is split into the edges $[\tilde{c}_i, p_0]$, $[p_0, p_1] \ldots [p_{n-1}, p_n]$, $[p_n, \tilde{c}_j]$ where $\{p_0..p_n\} \subset P$ are the intersection points incident with $b_{i,j}$ sorted according to their parameter value with respect to $b_{i,j}$. Once the edges for all curves in L are defined, we use them together with the edges along the boundary curves $\{\tilde{C}_0 \ldots \tilde{C}_n\}$ (which do not depend on L) to create the mesh faces. After splitting all concave faces, the curves $L \bigcup \{\tilde{C}_0 \ldots \tilde{C}_n\}$ are transformed to a set of mesh elements Q, which we use to evaluate the energy $E(L) = \sum_{f \in Q} E(f)$. Notice that we allow for T-joints in our quad meshes if the incident angle is $\approx \pi$.

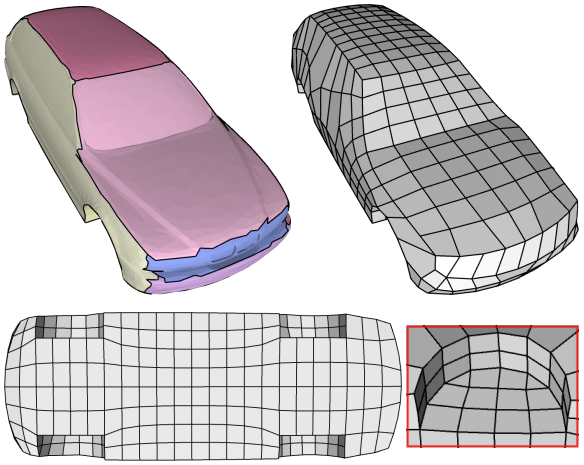

Figure 4: Top row: *Segmentation and output mesh of the Car model.* Bottom row: *View of the bottom part and the wheelhouse. Whenever possible, our selection algorithm produces a very regular mesh, consisting of well shaped quad elements.*

Element shape measure: Various factors such as orthogonality, parallelism and aspect ratio can affect the objective notion of "quality of an element". Hence, our shape evaluation function accounts for all of these factors. There are also various criteria which can be used to evaluate the shape of non-quad elements, e.g., triangles and pentagons. Since we are interested in quad-dominant remeshing, our evaluation measure is intolerant to such elements, especially if their shape deviates strongly from isotropic (i.e., nearly equal angles and sides). Higher valence elements are accepted in our procedure if their excess vertices can be considered as T-joints, otherwise such faces are penalized by high values. Configurations including flipped faces are rejected immediately.

More precisely, given an element with valence n with internal angles $\{\gamma_0 \ldots \gamma_{n-1}\}$ and side lengths $\{l_0 \ldots l_{n-1}\}$, we rate it according to the following formulas: In case $n = 4$ let us define the following four energy factors:

- Orthogonality:

$$E_\perp = \sum_{i=0}^{3} |\gamma_i - \pi/2|$$

- Parallelism:

$$E_{\parallel} = \sum_{i=0}^{3} |\gamma_i + \gamma_{(i+1)\%4} - \pi|$$

- Deviation from rhombus/square:

$$E_{\square} = \prod_{i=0}^{3} \left(2 - \frac{l_i}{\max_{j=0..3}\{l_j\}}\right)$$

- Deviation from parallelogram/rectangle:

$$E_{\boxminus} = \left(2 - \frac{\min\{l_0,l_2\}}{\max\{l_0,l_2\}}\right)\left(2 - \frac{\min\{l_1,l_3\}}{\max\{l_1,l_3\}}\right)$$

The final shape quality measure for a quad element combines all of these factors:

$$E_q = \left[(1+E_{\perp})\cdot(1+E_{\parallel})\cdot E_{\square}\cdot E_{\boxminus}\right] - 1\,. \qquad (2)$$

According to (2), a perfect square will have a score of 0, where a highly distorted quad with small angles and unequal sides can have a score higher than 1000. While these values do not have any intuitive meaning, they provide a way for us to map efficiently the shape quality of quad elements to a numerical value. Note that, due to the length ratio measure $E_{\square}$, squares/rhombuses are more preferred than rectangles/parallelograms. From the standard quad shapes the trapezoid receives the lowest score.

The scoring function is quite simple in the case of triangles/pentagons. We simply sum the angle deviation $E_a = \sum_{i=0}^{n-1} |\gamma_i - \beta|$, where $\beta = \pi/3$ if $n = 3$ and $\beta = 3\pi/5$ if $n = 5$, and multiply it by some user defined weights $w_{\triangle}$ and w_{PT}. Setting these weights high will make our meshing procedure prefer distorted quads over isometric triangles or pentagons.

Finally, for some applications, T-joints might present a useful compromise between the element's shape and its valence. Hence, given an element with valence $n > 4$ with k T-joints ($k < n-2$) we rate its shape in the same way we would rate an element with valence $n-k$. However, the resulting energy is augmented by $k \cdot w_T$, where w_T is a user-defined weight. Hence, setting w_T to a large number will force the procedure to avoid T-joints, while $w_T = 0$ makes elements with T-joints indistinguishable from the corresponding elements with valence $n-k$.

6.1. Selection procedure

For clarity of the presentation, we count from here on the concave boundary points with two tangents as two different points with common position but different tangents. Our task is to select a subset L of curves from the $O(m^2)$ candidates which leads to the best quality score $E(L)$. Based on the same argument by which we defined the number of tangents per constrained vertex $\tilde{c}_i$ in Section 5, we conclude that in the set L there should be at most one curve starting at each boundary vertex (which we have to select from the $m-1$ candidates for each vertex). Otherwise, the internal angle at $\tilde{c}_i$ will be split into several small angles, leading to poorly shaped adjacent elements. This allows us to formulate the mesh optimization as a classical (incomplete) graph matching problem, i.e., finding a set of edges without common vertices that minimize some cost functional.

However, our setup is particular difficult because we cannot assign constant weights to each connecting curve separately. Instead the functional $E(\cdot)$ sums over all *faces* $\in Q$, which in turn depend on several curves each. Hence there is little hope that a polynomial time algorithm exists for this task. Notice that the mere enumeration of all graph matchings in a complete graph has already $O(n!)$ complexity in the number of vertices.

Overview: Since we cannot hope to find a practical algorithm converging to the global minimum of $E(\cdot)$, we propose a strategy which aims at finding at least a *good local* minimum. Typically, such problems are solved by first finding an initial solution (phase I of our algorithm) and then improving it by "descending" in the direction of the largest decrease of the target functional until a local minimum is found (phase II). In addition, we use a third step (phase III), which attempts to escape a local minimum by evaluating a set of configurations (not necessarily better than the existing one), and choosing the one that leads to the lowest energy.

Phase I: To find an initial solution, we select "greedily" a subset of all curves using a simple heuristic dependent only on the end points of the curve (Fig. 5). Curves are selected until all constrained points are saturated, i.e., every boundary point is connected, or until all available curves are processed.

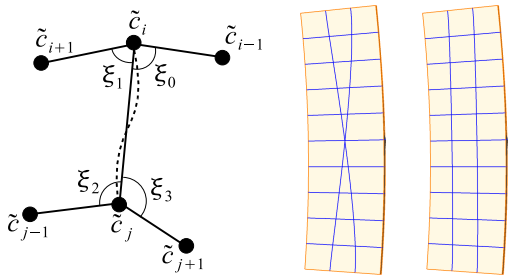

Figure 5: Left: *evaluating the angle deviation* $\sum_{i=0}^{3} |\xi_i - \pi/2|$ *scores how orthogonal the line* $b_{i,j}$ *is to the constrained edges at its end points.* Middle: *Due to the absence of any other internal information, i.e, an element structure, this simple heuristic is used to generate an initial solution for our selection algorithm. However it can often lead to a sub-optimal solution.* Right: *Our "steepest descent" minimization technique easily corrects the meshing by swapping the two wrong lines.*

Phase II: Once a solution is available, we optimize it by performing a "steepest descent" minimization, iteratively swapping (Fig. 6) or reconnecting (Fig. 7) some of the curves in

L iff by this the energy $E(L)$ decreases. At each iteration we evaluate all possible steps, compute the energy change for each candidate and choose the one that yields the largest decrease of $E(L)$. This phase concludes when no "downhill" moves are possible anymore, i.e., we have converged to a local minimum.

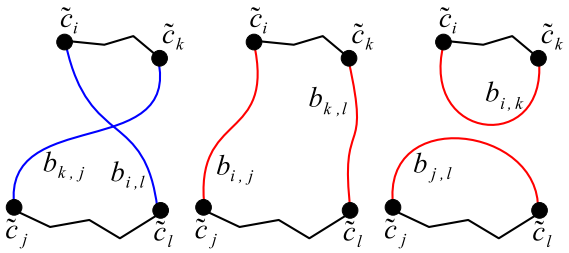

Figure 6: Swapping curves: *the pair of curves* $(b_{k,j}, b_{i,l}) \subset L$ (left) *can be replaced by the pair* $(b_{i,j}, b_{k,l})$ (middle) *or the pair* $(b_{j,l}, b_{i,k})$ (right).

Phase III: The third and final phase attempts to escape from the current local minimum by first removing all curves from L which have an intersection point incident with a small angle ($< \pi/4$). After all such "bad" curves are removed, we iteratively saturate the unconnected boundary vertices by adding at each iteration the curve which yields the smallest energy value. After each addition, phase II is again executed in order to optimize the new set of curves, potentially leading to a better configuration. This process is terminated as in phase I — whenever all boundary points are saturated or all available curves are processed. Since adding a curve can actually lead to an increment of $E(L)$ (even after optimization), a backup of the best configuration found up to the moment is kept at all times and is restored at the end the procedure if needed. This might lead to several unconnected points, which potentially generates T-joints or obtuse angles in the final mesh.

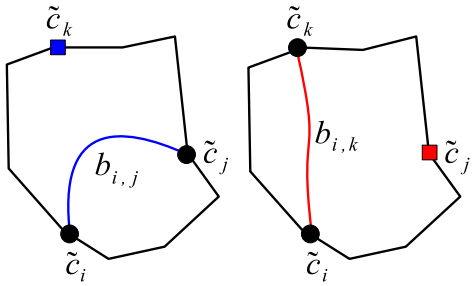

Figure 7: Reconnecting curve: *The curve* $b_{i,j} \in L$ (left) *is replaced by the* $b_{i,k}$ (right)*, connecting* $\tilde{c}_i$ *to the currently unlinked vertex* $\tilde{c}_k$.

7. Post-processing

Since samples in the interior of a given region R are generated only at the intersections of the selected cubic curves $b_{i,j}$, their location might be suboptimal. Therefore once the selection process is complete, we relax the mesh structure in the parameter domain by Laplacian smoothing. In some cases, this simple procedure improves the distribution of the mesh samples quite significantly. Once the smoothing is complete, we evaluate the parameterization of the intersection points and find the corresponding 3D positions.

Note that due to the method we used in Section 4 to map the boundary curves and points to the parameter space, they (usually) do not lie entirely inside Ω. Hence it is possible (although highly unlikely) that an intersection point cannot be located in Ω, i.e., in the parameterization of a face belonging to R. There are several possible ways to resolve this issue. We simply represent such points using barycentric coordinates with respect to the closest face of R in Ω and then evaluate the coordinates in 3D. This linear extrapolation is sufficient, since in practice the boundary curves lie close to the parameter domain of R and therefore the extrapolation error is small.

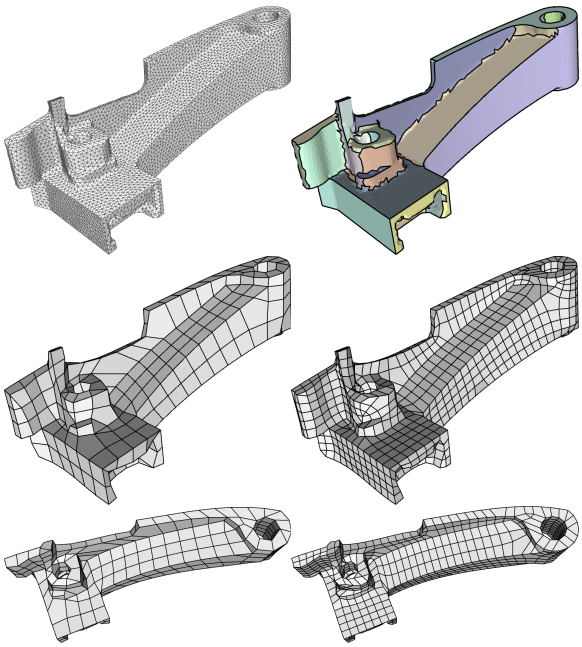

Figure 9: Top row: *The Alpha model and its segmentation.* Middle and bottom row: *Output meshes at two resolutions.*

8. Results

We tested our algorithm on several models, mostly mechanical objects and parts, which are difficult to process by previous techniques. Our two-step remeshing method has two

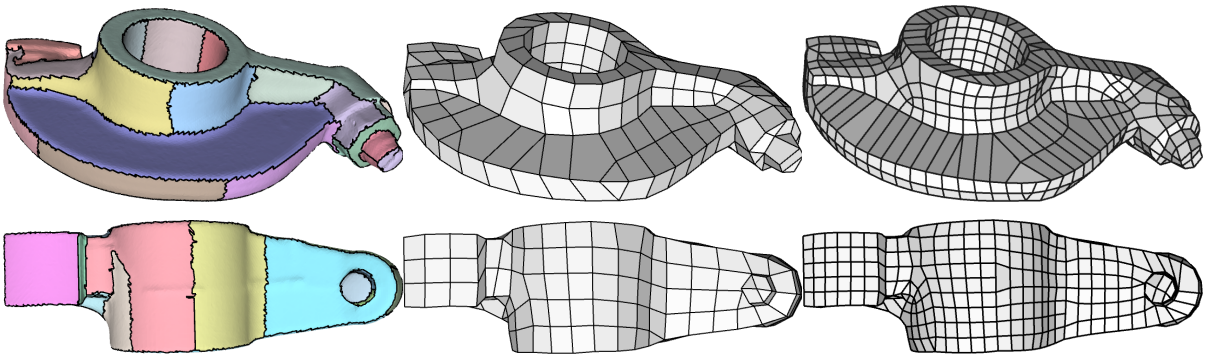

Figure 8: *Segmentation and output meshes of the Rocker arm model at two resolutions.*

Model	input	$\|S\|$	output	regularity	max val.	time	δ	$w_\triangle / w_{PT} / w_T$
	F/V		F/V	F%/V%	F/V	mm:ss		
Fan (Fig. 1)	12K/6.5K	30	185 / 193	94% / 76%	5 / 6	00:18	10%	500 / 500 / 500
—	—	30	484 / 501	92% / 87%	6 / 6	01:30	5%	100 / 100 / 100
Drill-hole(Fig. 2)	57K/28K	28	684 / 685	96% / 95%	5 / 6	02:27	5%	40 / 40 / 100
Rocker arm (Fig. 8)	80K/40K	41	431 / 416	84% / 86%	6 / 5	01:21	6%	20 / 70 / 70
—	—	41	1096 / 1081	90% / 92%	5 / 6	06:31	3%	30 / 30 / 100
Car (Fig. 4)	5K/2.5K	28	546 / 559	92% / 90%	6 / 5	07:20	4%	40 / 40 / 40
Alpha (Fig. 9)	18K/9K	51	544 / 537	86% / 83%	6 / 6	05:37	5%	40 / 40 / 100
—	—	51	1428 / 1425	92% / 90%	5 / 6	34:49	2.5%	40 / 40 / 100
Feline (Fig. 10)	100K/50K	92	571 / 554	64% / 70%	5 / 7	02:04	5%	40 / 40 / 100
—	—	106	1756 / 1749	76% / 84%	6 / 7	13:45	2%	20 / 20 / 100

Table 1: *Remeshing statistics for various models: The regularity numbers show the percentage of all faces/vertices which have valence* 4. *Valence* 6 *faces are pentagons with one T-joint, or quads with two T-joints.* δ *is the target edge distance between samples along the region boundaries and is given as percentage of the bounding box diagonal of the input mesh. All timings are taken on* 2.8*GHz Pentium IV PC.*

properties which make it especially useful for such cases: Due to the $L^{2,1}$ metric, all important sharp features are identified and preserved by the remeshing because boundary constraints are placed at the sharp features dividing adjacent regions on the surface. Furthermore, quality quad-dominant meshes for (nearly) flat areas are produced, since our method does not rely on anisotropic curvature tensor information to generate the quad elements.

Statistics about output complexities, performance and input parameters are given in Table 1. We set the weights affecting the penalty of non-quad elements, e.g., triangles and pentagons, according to the following strategy: For coarse output meshes, higher weights are used to force the algorithm to generate as regular as possible meshes. For denser output meshes, more flexibility is provided by setting lower values, otherwise some distorted quads might appear.

Limitations: Due to the high complexity of the combinatorial optimization, the computation of dense quad-meshes for large regions can take up to several minutes (see Table 1). This dependence on the *output* mesh complexity results

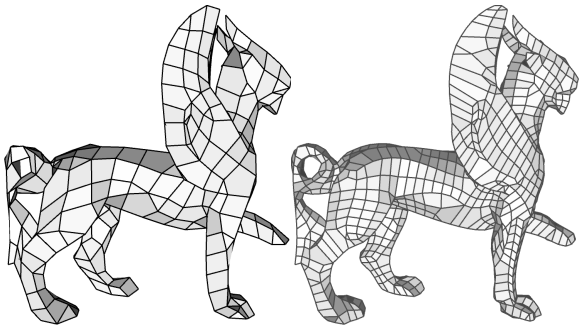

Figure 10: *Output meshes of the Feline model at two resolutions. The results for this organic shape do not exhibit the same regularity as the remeshes of the CAD examples, however, they are still suitable as a starting point for a subdivision control mesh design.*

from the fact that the number of possible curve network configurations increases drastically with the number of candidate curves. One way to reduce the computation time would be to start with a finer initial segmentation, yielding more segments with smaller complexity each. This however could lead to an inferior quality of the final output.

On the implementation level there is still some room for improvement, e.g., whenever a change in the set of selected lines L is made, we rebuild the connectivity of the whole processed region, i.e., edge and face creation. The remeshing energy is evaluated from scratch as well. A more sophisticated implementation could just modify the affected (split or merged) faces and update the energy accordingly.

Another issue is that the user can only control the output resolution along the region boundaries and not in the interior of the segments. This could be addressed in the future by taking additional constraints in the interior of the regions into account or by splitting regions based on a 2D compactness criterion rather than a flatness criterion alone.

9. Future work

An interesting route for further research is the improvement of our curves selection strategy. For instance, genetic algorithms often prove their effectiveness when employed to find a nearly global minimum in a large discrete space such as the one described in Section 6.1. Another possible option are the "Branch and Bound" methods which could use some geometrical heuristic to partition the problem and then find optimal solutions for the resulting, lower complexity subproblems.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. *ACM Trans. Graph.* (2003), 485–493.

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive geometry remeshing. *ACM Trans. Graph. 21(3)* (2002), 347–354.

[AUGA05] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: *Recent Advances in Remeshing of Surfaces*. Technical report, AIM@SHAPE Network of Excellence, 2005.

[BF98] BOROUCHAKI H., FREY P. J.: Adaptive triangular-quadrilateral mesh generation. *Numerical Methods in Engineering 41* (1998), 915–934.

[BK01] BOTSCH M., KOBBELT L.: Resampling feature and blend regions in polygonal meshes for surface anti-aliasing. *Computer Graphics Forum 20*, 3 (2001), 402–410.

[BMRJ04] BOIER-MARTIN I., RUSHMEIER H., JIN J.: Parameterization of triangle meshes over quadrilateral domains. In *Proc. of the Symposium on Geometry Processing* (2004).

[BS91] BLACKER T., STEPHENSON M.: Paving: A new approach to automated quadrilateral mesh generation. *Numerical Methods in Engineering 32* (1991), 811–847.

[CC78] CATMULL E., CLARK J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Geometric Design 10*, 6 (Sep 1978), 350–355.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 905–914.

[DBG*06] DONG S., BREMER P.-T., GARLAND M., PASCUCCI V., HART J. C.: Spectral surface quadrangulation. *ACM Trans. Graph.* (2006), to appear.

[DKG05] DONG S., KIRCHER S., GARLAND M.: Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer-Aided Geometric Design 22*, 5 (2005), 392–423.

[EH96] ECK M., HOPPE H.: Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Proc. of SIGGRAPH 96* (1996).

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *ACM Trans. Graph. 21(3)* (2002), pp. 355–361.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proc. of SIGGRAPH 97* (Aug. 1997), pp. 209–216.

[HG99] HECKBERT P. S., GARLAND M.: Optimal triangulation and quadric-based surface simplification. *Computational Geometry: Theory and Applications 14*, 1-3 (Nov. 1999), 49–65.

[HP04] HOFER M., POTTMANN H.: Energy-minimizing splines in manifolds. *ACM Trans. Graph. 23*, 3 (2004), 284–293.

[KLS03] KHODAKOVSKY A., LITKE N., SCHRÖDER P.: Globally smooth parameterizations with low distortion. *ACM Trans. Graph. 22*, 3 (July 2003), 350–357.

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (July 2002), 362–371.

[MK04] MARINOV M., KOBBELT L.: Direct anisotropic quad-dominant remeshing. In *Pacific Graphics* (2004), pp. 207–216.

[MK05] MARINOV M., KOBBELT L.: Automatic generation of structure preserving multiresolution models. *Computer Graphics Forum 24*, 3 (2005), 479–486.

[OSCS98] OWEN S. J., STATEN M. L., CANANN S. A., SAIGAL S.: Advancing front quadrilateral meshing using triangle transformations. In *Proc. of the 7th Int. Meshing Roundtable* (1998), pp. 409–428.

[Owe98] OWEN S. J.: A survey of unstructured mesh generation technology. In *Proc. of the 7th Int. Meshing Roundtable* (1998), pp. 239–267.

[RLL*06] RAY N., LI W. C., LÉVY B., SHEFFER A., ALLIEZ P.: Periodic global parameterization. In *ACM Trans. Graph. (to appear)* (2006).

[SKW85] SCHNABEL R. B., KOONTZ J. E., WEISS B. E.: A modular system of algorithms for unconstrained minimization. *ACM Trans. Math. Software 11* (1985), 419–440.

[SLI98] SHIMADA K., LIAO J.-H., ITOH T.: Quadrilateral meshing with directionality control by packing square cells. In *Proc. of the 7th Int. Meshing Roundtable* (1998), pp. 61–75.

[SWG*03] SANDER P. V., WOOD Z. J., GORTLER S. J., SNYDER J., HOPPE H.: Multi-chart geometry images. In *Proc. of the Symposium on Geometry Processing* (2003), pp. 146–155.