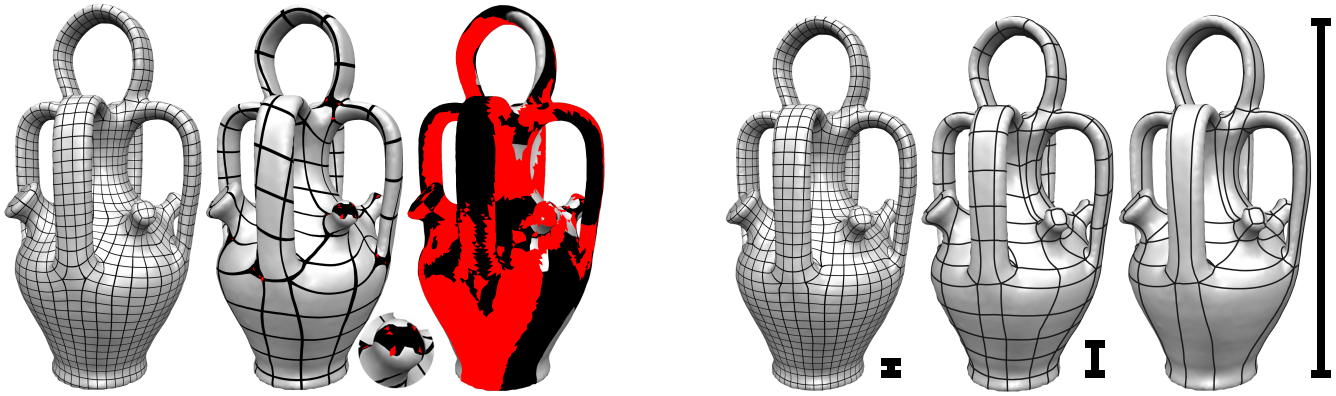
Integer-Grid Maps for Reliable Quad Meshing

David Bommes[1] Marcel Campen[2] Hans-Christian Ebke[2] Pierre Alliez[1] Leif Kobbelt[2]
[1]Inria Sophia Antipolis - Méditerranée [2]RWTH Aachen University



Greedy Rounding + Stiffening Our Reliable Approach

Figure 1: *(left) State-of-the-art parametrization based quad mesh generators, working with greedy rounding and stiffening, perform well if the target element sizing is chosen conservatively w.r.t. the distance of singularities but fail otherwise. Degeneracies in the map that prevent the iso-lines from stitching to a valid quad mesh – which mostly cannot be repaired locally – are highlighted in red. (right) Our novel reliable algorithm produces a valid output for any target sizing and thus in addition to ordinary quad-remeshing can be applied to coarse quad layout generation as well. The target edge length, indicated by bars, is identical for the left and the right triple.*

Abstract

Quadrilateral remeshing approaches based on global parametrization enable many desirable mesh properties. Two of the most important ones are (1) high regularity due to explicit control over irregular vertices and (2) smooth distribution of distortion achieved by convex variational formulations. Apart from these strengths, state-of-the-art techniques suffer from limited reliability on real-world input data, i.e. the determined map might have degeneracies like (local) non-injectivities and consequently often cannot be used directly to generate a quadrilateral mesh. In this paper we propose a novel convex Mixed-Integer Quadratic Programming (MIQP) formulation which ensures by construction that the resulting map is within the class of so called Integer-Grid Maps that are guaranteed to imply a quad mesh. In order to overcome the NP-hardness of MIQP and to be able to remesh typical input geometries in acceptable time we propose two additional problem specific optimizations: a complexity reduction algorithm and singularity separating conditions. While the former decouples the dimension of the MIQP search space from the input complexity of the triangle mesh and thus is able to dramatically speed up the computation without inducing inaccuracies, the latter improves the continuous relaxation, which is crucial for the success of modern MIQP optimizers. Our experiments show that the reliability of the resulting algorithm does not only annihilate the main drawback of parametrization based quad-remeshing but moreover enables the global search for high-quality coarse quad layouts - a difficult task solely tackled by greedy methodologies before.

CR Categories: I.3.5 [Computational Geometry and Object Modeling]: Hierarchy and geometric transformations

Keywords: remeshing, quadrangulation, parametrization, mixed-integer optimization

Links: DL PDF WEB DATA

1 Introduction

The automatic conversion of objects given as triangulated surface meshes into a quadrilateral representation (referred to as quad-remeshing) is a challenging task which has attracted lots of attention in the last years. In animation as well as in simulation quadrilateral meshes are very appealing due to their tensor-product nature. The edges in a quad mesh tend to form smooth curves on the surface that are very well suited to accurately represent feature curves and to naturally segment objects into structured parts. Apart from the well behaved edge flow of a single curve, intersections between pairs of curves are typically nearly orthogonal and exhibit good numerical properties in Finite-Element Methods.

Often quad-remeshing is a single building block embedded within a complex application pipeline like, e.g., an animation or simulation system. Consequently reliability of the involved algorithms is indispensable in order to be able to automatize as many sub steps as possible and thus achieve high productivity. As an extreme case, imagine an adaptive simulation where in each of the thousands of timesteps a remeshing has to be performed. Obviously it is not feasible to manually react on failure cases in the simulation and a

reliable automatization is absolutely essential. On the other hand in an interactive digital sculpting scenario a non-reliable dynamic remeshing algorithm would be equally problematic since it severely disrupts the workflow by keeping the designer away from her actual task. Apart from the above examples it is clear that the usage of a non-reliable algorithm presupposes expert knowledge on how to react in failure cases, a hurdle often strongly limiting the range of potential users.

Parametrization based algorithms are known to produce high quality quad meshes. Especially their abilities to (1) explicitely control the number and position of irregular vertices, (2) to smoothly distribute distortion by convex variational formulations and (3) to naturally preserve feature curves and boundaries make them a good choice for many practical applications. Unfortunately they lack reliability as illustrated in Fig. 1. The main source of non-reliable behavior are degeneracies in the constructed parametrization function, i.e. the Jacobian determinant of the piecewise linear map is locally less than or equal to zero. Geometrically a value less than zero characterizes triangles that flip their orientation in the domain while a zero value results from local non-injectivities where an edge or even a whole triangle is mapped to a single point. All such degeneracies destroy the essential property that the mapped grid of integer-isolines forms a quad mesh on the surface. In the worst case non-quad regions with an odd number of boundary edges are generated for which it is known that they cannot be quadrangulated individually by a postprocess without touching other parts of the quad mesh. While state-of-the-art algorithms [Bommes et al. 2009] typically succeed in the generation of finely tessellated quad meshes, especially the construction of coarse quadrangulations, where the edge length is equal to or larger than the distance between singularities, is delicate. For such instances it frequently happens that the energy of a degenerated map, e.g. where two nearby irregular vertices are mapped to the same integer location, is smaller than those of the closest non-degenerate one. This is the main motivation behind specialized algorithms for the construction of coarse quad layouts [Campen et al. 2012], instead of just driving a parametrization based quad mesh generator with a large target edge length. Note that our reliable parametrization based approach is the first one that offers a unified methodology for quad-remeshing on *all* desired scales and hence is indeed able to generate also coarse quad layouts.

The description is organized in the following way. In Section 2 we define the class of Integer-Grid Maps, which consists of all piecewise linear maps which non-degenerately map the grid of integer isolines into a quad mesh on the surface. Finding optimal maps from this class is computationally intractable, which is not surprising considering that quad meshes have to fulfill global topology constraints [Murdoch et al. 1997] and that the construction of coarse quad layouts [Tarini et al. 2011; Bommes et al. 2011; Campen et al. 2012] is well known to be a hard task, even for human designers. To enable the efficient generation of Integer-Grid Maps, in Section 3.1 we derive a novel *convex* Mixed-Integer Quadratic Programming (MIQP) formulation, which in principle can be optimized by modern branch-and-cut techniques like [IBM 2012; Gu et al. 2011]. Such solvers succeed if the search space is low-dimensional and if the optimal solution is not too far away from that of the *continuous relaxation*, i.e. the instance of Quadratic Programming (QP) where all integer conditions are neglected. The first aspect is addressed by an aggressive complexity reduction algorithm presented in Section 3.2, which intuitively can be understood as incremental mesh decimation in a carefully chosen flat parametric space. The number of remaining vertices is roughly equal to the number of singularities in the quad mesh and thus typically orders of magnitude smaller compared to the input mesh. The loss of accuracy is only marginal since solely flat parts are re-tessellated. The second important aspect of pushing the minimizer of the continuous relaxation towards the best integer solution is addressed in Section 3.3. The proposed singularity separation conditions exploit the fact that the distance between singularities in every integer solution can never be smaller than 1. The experiments in Section 4 show that the combination of complexity reduction and singularity separating conditions allows us to efficiently find good approximations of the original MIQP, which is impossible without them.

1.1 Contributions

In summary our main contributions are the following:

- A MIQP formulation where the minimizer in contrast to previous work is guaranteed to be an Integer-Grid Map, Sec. 3.1.
- A complexity reduction algorithm which accurately approximates the high-dim. problem by a low-dim. one, Sec. 3.2.
- Singularity separating conditions, which significantly improve the continuous relaxation of the MIQP, Sec. 3.3.

1.2 Related work

In the last years, quad mesh generation has been studied intensively such that we restrict the discussion to the most relevant aspects while referring the reader to a recent more complete survey [Bommes et al. 2012]. Previous work on quad remeshing shows a division between algorithms which come with guaranteed quality, and methods which aim at maximizing the quality. The guarantees of the first class lead to reliable algorithms that work every time, but often are limited to minimum guarantees, e.g. pure quad topology or convex elements, and do not provide strong guarantees on the overall quality. Most algorithms within this class are operator based, like Catmull Clark subdivision [Catmull and Clark 1998], the QMorph [Owen et al. 1999] algorithm, or quad mesh decimation techniques [Daniels et al. 2008; Daniels et al. 2009; Tarini et al. 2010].

The algorithms of the second class are typically based on variational formulations related to global parametrization and often lead to superior mesh quality in practice but do not provide any guarantees, not even the minimal requirement of generating a topological quad mesh. Approaches within this class are sometimes driven by a cross field [Ray et al. 2006; Kälberer et al. 2007; Bommes et al. 2009] or based on quad layouts [Dong et al. 2006; Tong et al. 2006; Bommes et al. 2010; Huang et al. 2008]. In any case, the problem of guaranteeing success of such a method is two fold, (1) determining integer positions of the singular vertices that admit a degeneracy free bijective parametrization and (2) finding such a map, which in addition optimizes the resulting mesh quality. The first issue is trivially solved for layout based approaches by assigning a positive length to each poly-chord [Daniels et al. 2009] in the quad layout, while it is a serious hurdle in cross-field based techniques, where apart from non-reliable greedy techniques no other solution exists.

The second issue has been tackled through many different strategies like, e.g., penalization of high distortion by stiffening [Bommes et al. 2009], singularity relocation techniques [Dong et al. 2006; Tarini et al. 2010], the inverse way of domain relaxation [Bommes et al. 2010] or varying the sizing field by curl correction [Ray et al. 2006]. Additionally to methods specifically designed for quad remeshing, there exists previous work on degeneracy-free parametrization [Floater and Hormann 2005; Sheffer et al. 2006; Hormann et al. 2007]. It is important to understand that singularities in the map, that are inevitable for quad remeshing, act like point constraints in texture mapping that require special treatment [Eckstein et al. 2001; Yu et al. 2012]. Such methods, developed for texture mapping, require a boundary and cannot be applied in

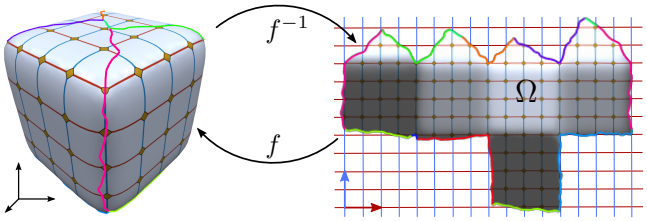


Figure 2: *The main idea of parametrization based quad mesh generation methods consists in mapping the canonical quad mesh formed by the 2D Cartesian grid onto the surface. First the mesh is parametrized, i.e. cut and flattened by a function f onto a 2D domain Ω. Then the intersection between the Cartesian Grid and the domain $\Omega \cap G$ is inversely mapped onto the surface, where a quad mesh is obtained by contouring. Special compatibility conditions are required along the colored cut-curves.*

our context of globally smooth parametrization with cone singularities [Springborn et al. 2008]. Therefore all of the above methods do not provide the required reliability or cannot be applied to our setting. In the area of quasiconformal maps, however, two recent approaches [Weber et al. 2012; Lipman 2012] are lifting the variational setting to a new level. While the nonlinear approach of [Weber et al. 2012] usually succeeds but cannot guarantee to find a bijective mapping, the powerful concept of "Bounded Distortion Mapping Spaces" [Lipman 2012] is closely related to one aspect of our setting. We hence provide a detailed comparison in Section 4.

Since none of the previous methods addresses the integer related problem (1), all of them fail in the context of coarse quad-remeshing (cf. Figure 1). This behavior motivated the design of conceptually different algorithms for quad layout generation based on, e.g., grid-preserving operators [Bommes et al. 2011], separatrix optimization [Tarini et al. 2011] or loop tracing [Campen et al. 2012]. Due to its reliability however, our global optimization algorithm is able to generate quad meshes within the complete range of sizes and even outperforms state-of-the-art greedy techniques.

From the quality and automatization point of view, cross-field based techniques seem to be the current choice for industrial practice. Such algorithms found its way into commercial software like the qRemesher of Pixologic's ZBrush or the re-topology tool of 3D Coat. Moreover, extensions to different input data like point clouds [Li et al. 2011] or range images [Pietroni et al. 2011] are available and thus a large application area is covered by them. Consequently, our overall goal is to improve the reliability of these practical variational approaches and eventually achieve high-quality and guarantees altogether. Stepping back to the root of the mathematical formulation, we identify the components that are missing for reliability and efficiency and thus are able to complement the formulation of such techniques [Kälberer et al. 2007; Bommes et al. 2009] by the missing ingredients.

2 Integer-Grid Maps

The main principle of parametrization based quad meshing algorithms is the mapping of the canonical quad mesh formed by the 2D Cartesian grid of integer iso-lines onto a surface embedded in 3D, see Figure 2 for an illustration. However, this map has to fulfill several requirements such that the image of the 2D integer-grid stitches to a valid quad mesh on the surface as discussed next.

In the following we restrict the discussion to piecewise linear maps given per triangle. More specifically, given a triangle mesh $\mathcal{M} = (V, E, T)$ composed of vertices, edges, and triangles, a map f is

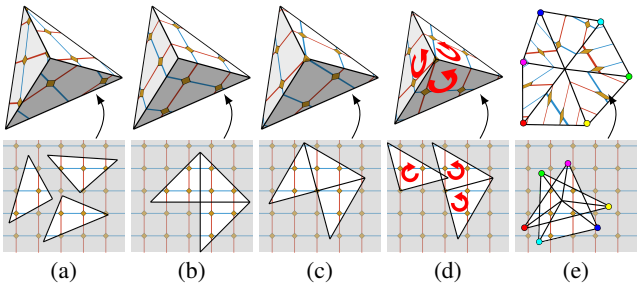


Figure 3: *Integer-Grid Map conditions: (a) For arbitrarily mapped triangles, the integer lines are discontinuous on the surface. (b) Compatibility results from transition functions of type (1), however, with a triangle at the singularity. (c) Moving the singularity to an integer position, i.e. fulfilling (2), results in a valid IGM leading to a pure quad mesh on the surface. (d) Invalid orientation consistency (3) results in non-quad cells. (e) A regular vertex is trapped in a valid but sub-optimal configuration. It is not possible to continuously unfold this configuration without temporarily degenerating one triangle.*

given as the union of all individual triangle maps specified by the images of their corresponding three vertices:

$$f_i : (u_i, v_i, w_i) \in \mathbb{R}^{2\times 3} \mapsto (p_i, q_i, r_i) \in \mathbb{R}^{3\times 3}$$

Note that following [Kälberer et al. 2007] each triangle is an individual chart and consequently a single vertex might have multiple copies.

The class of *Integer-Grid Maps* (IGMs) is defined to be the subset of all possible maps which additionally correctly stitch the grid of integer iso-lines to a valid quad mesh. The necessary and sufficient conditions are the following:

- *Transition Functions:* The transition function $g_{i\to j}$ from the chart of triangle t_i to the chart of a neighboring triangle t_j, identifying their common edge, has to be an integer-grid automorphism [Kälberer et al. 2007; Bommes et al. 2009] in $\mathbb{R}^2$ of the form
$$g_{i\to j}(\mathbf{a}) = R_{90}^{r_{ij}}\,\mathbf{a} + \mathbf{t}_{ij} \tag{1}$$
consisting of a $r_{ij} \in \{0, 1, 2, 3\}$ times $\pi/2$ ccw rotation and an integer translation $\mathbf{t}_{ij} \in \mathbb{Z}^2$.

- *Singular Points:* With the above transition functions it is possible to represent cone singularites with quarter-indices which are characterized by a nonzero angle defect[1] in the domain. Let S be the set of all singular vertices. Then in order to guarantee a pure quad mesh all singular vertices have to lie on integer locations in the domain, i.e.
$$\mathbf{f}^{-1}(s_i) \in \mathbb{Z}^2 \;\forall s_i \in S \tag{2}$$

- *Consistent Orientation:* All domain triangles $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ with $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^2$ should have a positive orientation, meaning that
$$det\,[\mathbf{v} - \mathbf{u}, \mathbf{w} - \mathbf{u}] > 0 \tag{3}$$

These conditions are illustrated graphically in Figure 3. For the remainder of this paper it is very important to keep in mind that an IGM is a bijective map between two piecewise linear 2-manifold

[1]The angle defect is defined in the usual way to be $2\pi - \sum_i \alpha_i$ for interior and $\pi - \sum_i \alpha_i$ for boundary vertices

meshes $f : \mathcal{M}_{\text{IGM}} \to \mathcal{M}$. Naively searching for the best IGM leads to an instance of Mixed-Integer Nonlinear Programming (MINLP) as discussed next.

2.1 Naive Search for Integer-Grid Maps

In parametrization based quad-remeshing a variational quality metric $E_q(f)$ is commonly chosen that penalizes undesired distortion of the resulting quad elements on the basis of the map f. Usually the unconstrained minimizer of $E_q(f)$ is not an IGM and to obtain a quad mesh we have to solve the following instance of MINLP:

$$\text{minimize} \quad E_q(f) \quad \text{s. t.} \quad (1),\,(2),\,(3) \tag{P1}$$

This naive problem formulation consists of $6|T| + 3|E|$ unknowns including at least $3|E|$ discrete variables. Due to (1) there are $2|E|$ many equality constraints that are nonlinear in r_{ij} and linear in $\mathbf{t}_{ij}$. Furthermore (3) generates $|T|$ many non-convex quadratic inequality constraints. Although the DOFs can be reduced along a spanning tree of dual edges without affecting the solution (cf. [Bommes et al. 2009]) the resulting number of unknowns is still in the order of $O(|V|)$ in both, the continuous and discrete variables.

Unfortunately MINLP problems are very hard to solve since they imply all difficulties from continuous as well as discrete optimization. Even by neglecting all integer constraints there is little hope of finding good solutions since due to (1) and (3) the continuous relaxation is within the still very difficult class of non-convex Nonlinear Programs. Figure 3 (e) gives an idea on how complicated the situation is. It shows a typical poor local minimum that we observed in optimization with non-convex consistent orientation constraints (3). It is clear that in the case of triangle meshes with thousands of vertices the naive search for IGMs is not feasible due to the huge discrete search space in combination with nonlinear and non-convex functions.

Modern mixed-integer optimizer codes are able to rapidly find good feasible solutions if the problem fulfills the following three requirements. (i) The objective function as well as all constraint functions are convex. (ii) The search space is low-dimensional. (iii) The optimum lies not too far away from the one of the continuous relaxation. Accordingly we show next how we successively simplify conditions (1), (2) and (3) in order to devise a practically feasible formulation, namely a low-dimensional convex Quadratic Program (convex objective function and linear constraints) with a well-behaved continuous relaxation.

3 Efficient Search for Integer-Grid Maps

One reason for the enormous dimension of discrete variables in problem (P1) is that potentially every vertex in the input triangle mesh can represent a singularity in the quad mesh. Therefore one appealing approach is to split the overall problem into one step which determines singular vertices and a second step which solves the parametrization problem with fixed singularities. Note that this approach, although usually not motivated from this perspective, is realized by all state-of-the-art parametrization based quad-remeshing algorithms, by using either a cross-field or a quad layout.

Compared to problem (P1), finding a cross-field with adequate singularities is a simpler task since angle based convex mixed-integer formulations [Ray et al. 2008] exist which can be tackled with fast greedy solvers [Bommes et al. 2009] in a reliable way.

Guiding problem (P1) with a cross-field and restricting to fixed singular vertices, all r_{ij} in the transition functions are determined, turning the non-linear constraints (1) into linear ones. The number of required independent $\mathbf{t}_{ij}$ is roughly $2|S|$ and thus also significantly reduced. Furthermore condition (2) is simplified since the set S is no longer dynamic but known a priori. The cross-field enables a powerful quadratic quality metric that on the one hand favors the alignment of quad elements along the cross directions and on the other hand tries to reproduce a specified element density [Kälberer et al. 2007; Bommes et al. 2009]

$$E(f) = \frac{1}{2}\sum_{t\in T} \mathcal{A}_t \|\nabla f_t - R_t H_t\|^2_{\text{Frobenius}} \tag{E1}$$

where $H_t = \operatorname{diag}(w_t^{-1}, h_t^{-1})$ is the locally desired anisotropic sizing, the columns of $R_t \in \mathbb{R}^{3\times 2}$ are the normalized u and v directions of the cross-field, and $\mathcal{A}_t$ is the triangle area.
In order to weight the importance of cross direction versus sizing fidelity and to penalize outliers by a higher-order norm (if desired), we generalize the quality metric to the following form

$$E^k_\alpha(f) = \frac{1}{2}\sum_{t\in T} \mathcal{A}_t \|R_t^T \nabla f_t - H_t\|^k_\alpha \tag{E2}$$

where $k \in 2\mathbb{Z}^+$, $\alpha \in [0,1]$ denotes the anisotropy factor, and

$$\left\| \begin{pmatrix} a & b \\ c & d \end{pmatrix} \right\|^k_\alpha = 2\alpha(a^k + d^k) + 2(1-\alpha)(b^k + c^k)$$

For α below 0.5 stretching along the cross-field directions is preferred over angular deviation. This is for example useful if cross-field aligned rectangular patches are desired as often seen in coarse quad layouts. Notice that despite its nonlinearity (E2) is still a convex functional and thus efficiently optimizable.

Up to here, the discussed simplifications are known and have been applied previously to achieve acceptable runtimes. Notice that for well behaved input data, i.e. a well chosen combination of geometry, cross-field R and sizing field H, the triangle orientation consistency constraints (3) can be neglected since the variational formulation already penalizes flipped triangles. However, for complicated real-world geometry automatic state-of-the-art algorithms frequently run into trouble due to the following two reasons. First of all singularities in an Integer-Grid Map behave similar to isolated point constraints in harmonic parametrizations. Hence, suboptimally chosen singularity locations are likely to induce flipped triangles, even for continuously relaxed $\mathbf{t}_{ij}$ (cf. [Yu et al. 2012; Lipman 2012]). Secondly, a greedy estimation of singularity integer locations (2) can intuitively be understood as snapping the singularities to their closest integer lattice point (cf. the NP-hard closest lattice problem formulated in [Springborn et al. 2008] and analyzed in [Micciancio 2006]). Obviously, if the relaxed minimizer of E_q positions several singular vertices close to the same integer lattice point these singularities are likely to snap onto each other and thus induce a degenerate map. While the first issue can be satisfactorily handled by the stiffening approach of [Bommes et al. 2009] or recent parametrization approaches that bound the quasi-conformal distortion [Lipman 2012; Weber et al. 2012], the second integer related one cannot. Moreover it seems to be very difficult if not impossible to design a greedy snapping strategy which guarantees a valid map while respecting the linear interdependency caused by the transition functions.

To overcome the integer related problem in the next sections we derive a convex formulation that is designed for modern branch-and-cut mixed-integer optimizers. In contrast to greedy rounding such solvers are able to scan the entire integer space for valid solutions.

The ultimate goal we want to achieve: Given any cross-field R and sizing field H, both living on an arbitrary triangle mesh $\mathcal{M}$, efficiently find the Integer-Grid Map $f : \mathcal{M}_{\text{IGM}} \to \mathcal{M}$ that optimizes the quality functional E^k_α.

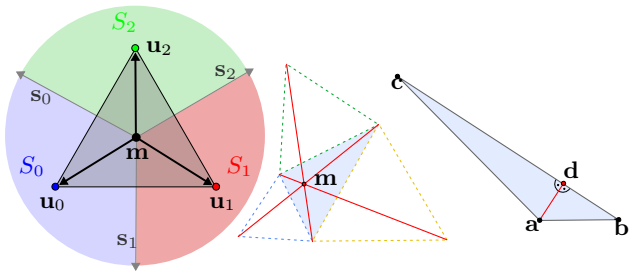


Figure 4: *(left) Given a* $\mathbf{m}$*-centered ccw-ordered trisector generated by* $\mathbf{s}_i$*, the triangle cannot flip its orientation if each vertex* $\mathbf{u}_i$ *remains strictly within sector* S_i *and the origin* $\mathbf{m}$ *is an interior point of the triangle. (middle) The Fermat point* $\mathbf{m}$ *is the intersection of the red lines, resulting from equilateral triangles, in blue, yellow and green, attached to each edge. (right) To guarantee a Fermat point inside the triangle, large angles above* 100° *are* virtually *split, meaning that* $\mathbf{d}= (1-\alpha)\mathbf{b} + \alpha\mathbf{c}$ *is a linear combination of the original degrees of freedom.*

3.1 Convex Consistent Orientation

The feasible set of the consistent orientation condition (3) is a non-convex region $\mathcal{N} \subset \mathbb{R}^6$. In our quest for IGMs, however, we need to find a convex approximation $\mathcal{C}$ which in addition is *conservative*, meaning that every point within $\mathcal{C}$ fulfills (3) and hence $\mathcal{C} \subset \mathcal{N}$. Based on the geometric interpretation of condition (3), i.e. the signed area of a mapped triangle is required to be positive, we derive such a conservative convex approximation.

Assume an (arbitrary) ccw-ordered tri-sector (S_0, S_1, S_2) as illustrated in Figure 4 (left). It is centered at $\mathbf{m} \in \mathbb{R}^2$ and generated by three ccw-ordered rays $\mathbf{m} + t \cdot \mathbf{s}_i$ with $t \in [0, \infty)$ that partition $\mathbb{R}^2$ into 3 sectors

$$S_i = \{\mathbf{u} \in \mathbb{R}^2 : (\mathbf{u} - \mathbf{m}) \cdot \mathbf{s}_i^\perp > 0 \ \wedge \ (\mathbf{u} - \mathbf{m}) \cdot \mathbf{s}_{i+1}^\perp < 0\}$$

with $()^\perp$ representing a ccw-rotation by 90 degrees and all index arithmetic modulo 3. Furthermore assume a triangle $T = (\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2)$ which is compatible to the trisector in a sense that $\mathbf{u}_i \in S_i$ and $\mathbf{m}$ is inside T. Then the area of T is guaranteed to be the sum of three positive areas belonging to the triangles $T_i = (\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{m})$ and thus is positive as well. Choosing an arbitrary interior point expressed in barycentric coordinates $\mathbf{m} = \sum_{i=0}^{2} \alpha_i \mathbf{u}_i$ with $\alpha_i > 0$, $\sum \alpha_i = 1$ as the trisector center and keeping $\mathbf{s}_i$ as well as α_i fixed results in a conservative and convex approximation of (3) in form of six linear inequality conditions $\{\mathbf{u}_i \in S_i\}$.

The previous discussion shows that each choice of $(\alpha, \mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2)$ defines a convex configuration space $\mathcal{C}$ of a triangle with strictly positive area. Geometrically such a choice is a trisector attached to a barycentric position in the triangle. Note that every specific trisector excludes a portion of valid triangle configurations since the convex set $\mathcal{C}$ is only a subset of the non-convex set $\mathcal{N}$. Imagine that for each triangle T we know a reference configuration $\bar{T} = (\bar{\mathbf{u}}_0, \bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2)$ which is likely to be close to the optimal shape. Accordingly the goal is to find a $\mathcal{C}$ which only excludes unlikely deformations of the reference $\bar{T}$. To enable maximal rotational freedom we choose a trisector origin $\mathbf{m}$ such that angles between $(\bar{\mathbf{u}}_i - \mathbf{m})$ and $(\bar{\mathbf{u}}_{i+1} - \mathbf{m})$ are 120° each such that the separation ray vectors $\mathbf{s}_i$ can be chosen as the corresponding bisectors. This point is know as the *first Fermat point* and can be constructed by first attaching an equilateral triangle at each edge of the reference triangle and then intersecting the three diagonals connecting both opposing vertices of each original edge as shown in Figure 4 (middle).

This construction is possible only if the reference triangle exhibits no angle larger than 120°. However, in all other cases we can simply virtually split the triangle along the altitude at its large angle as demonstrated in Figure 4 (right). After the split it is guaranteed that no angle is larger than 90°. Note that no explicit splitting is required, since the split point can be expressed in barycentric coordinates and thus is linear in the original vertices. In practice we virtually split all triangles exhibiting an angle larger than 100°, because for larger angles the Fermat point approaches a vertex which negatively affects the space of feasible transformations.

The convex subspace $\mathcal{C}$ resulting from the trisector positioned at Fermat's point contains a rich set of deformations around the reference shape. For example similarity transformations composed of (i) arbitrary translations, (ii) isotropic scalings within $(0, \infty)$ and (iii) rotations within $(-60^\circ, 60^\circ)$ are still possible.

Limited Precision Arithmetic: Due to limited precision floating point arithmetic it is not possible to handle constraints with strict inequalities. Accordingly we modify the above constraints to the following normalized form

$$\begin{aligned} \delta_i^0 \left((\mathbf{u}_i - \mathbf{m}) \cdot \frac{\mathbf{s}_i^\perp}{\|\mathbf{s}_i^\perp\|} - \epsilon \right) &\geq 0 \\ \delta_i^1 \left(-(\mathbf{u}_i - \mathbf{m}) \cdot \frac{\mathbf{s}_{i+1}^\perp}{\|\mathbf{s}_{i+1}^\perp\|} - \epsilon \right) &\geq 0 \end{aligned} \tag{4}$$

where 2ϵ is a lower bound on the distance between two vertices of the same triangle and δ_i^0 and δ_i^1 are normalization factors, normalizing the constraint value to 1 w.r.t. the reference shape. We choose ϵ as 1% of the smallest reference edge length, although substantially smaller values are also possible as long as the tolerance of the applied solver is chosen below 2ϵ.

Appropriate reference triangles result from the input cross- and sizing-field as discussed in more detail in Section 3.4. At this point we thus achieved a convex formulation which in theory is sufficient to find the best IGM within our convex subspace:

$$\text{minimize} \quad E_\alpha^k(f) \quad \text{s. t.} \quad (1),\ (2),\ (4) \tag{P2}$$

However, in practice the enormous time complexity of integer optimization does not allow the handling of instances with more than a few vertices as the experiments in Section 4 show. This is why we have to spend more effort in the next sections to obtain a practical problem formulation.

To understand the following two improvements it is helpful to briefly review the fundamentals of branch-and-cut solvers [Pardalos and Resende 2002]. Such a solver explores a tree of continuous problems until the best integer solution is found and proven to be optimal. The root of this tree, which is of exponential size in the discrete variables, is the original relaxed problem. At each branch heuristically an inequality is added that splits the search space into two disjunct parts with integer solutions on the newly created boundary. Depending on the problem characteristic such a search might only require the traversal of a linear subset of nodes or in the worst case visits the complete exponential tree. In any case we have to solve many continuous problems similar to the relaxation of (P2), leading to the first issue of node complexity which will be addressed by a complexity reduction algorithm in Section 3.2. The second issue arises if the minimizer of the initial continuous relaxation is far away from the optimal integer solution. In this

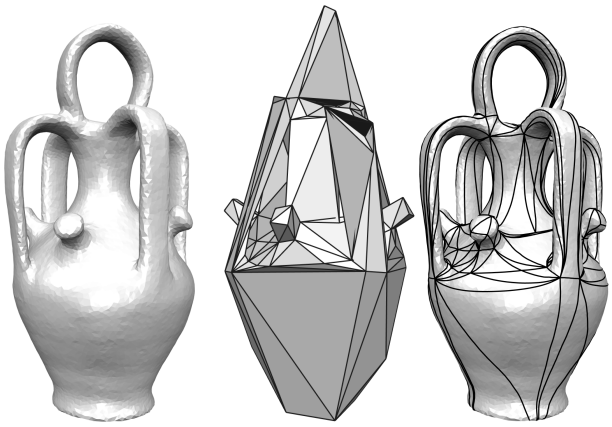
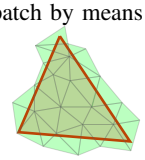


Figure 5: *Aggressively decimating the input mesh (left, 15k vertices) embedded in 3D is problematic, since the resulting surface (middle, 109 vertices) strongly changes as can be noticed at the upper handle. Moreover a straightforward mapping of the input cross-field to the decimated surface, which is required for the intended optimization, is not available. Instead we decimate the flat parts within a cone-metric parametrization, which preserves the original geometry (right, 109 vertices).*

case the branch heuristics frequently fail and let the solver dive into subtrees without any feasible solution. To prevent such inefficient tree traversal, in Section 3.3 we propose singularity separating conditions that strongly improve the initial continuous relaxation and thus significantly speed up the solution process.

3.2 Complexity Reduction

To reduce the node complexity of the branch-and-cut algorithm it is tempting to aggressively decimate the input mesh. Recall that due to (2) singularities of the cross-field imply integer conditions in the map and consequently cannot be removed from the mesh. In contrast this is possible for all regular vertices. The problem with such an extreme decimation pre-process is that the input surface itself would be heavily altered as illustrated in Figure 5 (middle). We cannot expect that the best quad mesh for the decimated surface is similar to the optimal one for the original surface. In order to nevertheless perform an extreme complexity reduction we can exploit the following observation: The solution of the continuous relaxation of (P2) is a globally smooth map $f_{\text{relaxed}} : \mathcal{M}_{\text{relaxed}} \to \mathcal{M}$ with $\mathcal{M}_{\text{relaxed}}$ being flat except from cone singularities at cross-field singularities. Accordingly the idea is to perform decimation in flat parametric space and thus being able to represent the original input surface by an extremely coarse mesh. After decimation, i.e. the map $\mathcal{M}_{\text{relaxed}} \to \mathcal{M}_-$, every new triangle overlaps with a set of triangles of $\mathcal{M}_{\text{relaxed}}$ in parameter space (see inset) and thus represents a surface patch by means of f_{relaxed} as shown in Figure 5 (right). Mathematically the decimation in parametric space corresponds to a change of function space and it is straightforward to formulate the original quality measure E^k_α in terms of the decimated mesh. Based on the known map $f : \mathcal{M}_- \to \mathcal{M}$ we search for the unknown IGM $g : \mathcal{M}_{int} \to \mathcal{M}_-$. The map g should minimize the energy of the combined function $f \circ g : \mathcal{M}_{int} \to \mathcal{M}$, which by the chain rule evaluates to the following expression



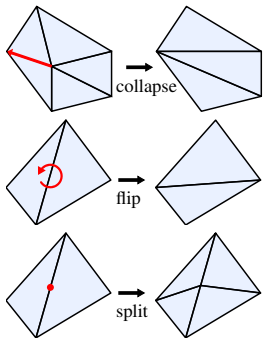
$$E^k_\alpha(f \circ g) = \frac{1}{2} \sum_{\substack{t \in T_{\mathcal{M}} \\ s \in T_{\mathcal{M}_-}}} \sum \mathcal{A}_{t \cap f(s)} \| R_t^T \nabla f_t \nabla g_s - H_t \|^k_\alpha \qquad \text{(E3)}$$

where the former sum is split into a double sum of piecewise linear parts, i.e. the intersection of both non-conforming triangular meshes. In case of an isotropic and globally constant sizing H_t we can moreover allow for a slight bias in the metric by observing that the first map roughly satisfies $R_t^T \nabla f_t \approx H_t$ leading to

$$E^k_\alpha(f \circ g) \approx \frac{1}{2} \sum_{s \in T_{\mathcal{M}_-}} \mathcal{A}_s \| \nabla g_s - I \|^k_\alpha \qquad \text{(E4)}$$

which avoids the necessity to intersect both triangle meshes.

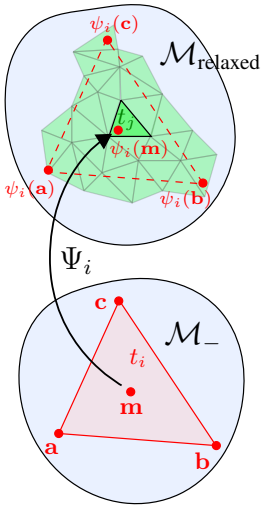
Operator based Mesh Decimation: Our mesh decimation in parametric space is a simple algorithm based on the three well known operations: halfedge collapse, edge flip and edge split (cf. [Botsch et al. 2010]) illustrated on the right. By disallowing all operations that move a cone singularity it is guaranteed that the decimation is an isometric re-tessellation of flat 2D space. Sorted by increasing edge length, we perform valid halfedge collapses until no more collapse is possible. We then flip edges that violate the Delaunay criterion [Botsch et al. 2010] until the intrinsic Delaunay triangulation of our current vertex set is reached. Due to the flips new halfedges might become collapsible and we iterate both steps until convergence. This simple strategy typically leads to the intrinsic Delaunay triangulation of the singular vertices, which is the coarsest mesh we can potentially reach. Due to the non-uniform distribution of singularities, the triangle quality of the decimated mesh is typically rather low and exhibits triangles with large aspect ratio. However, since we observed that good triangle quality is not critical for the solution of our mixed-integer problem, we favor low complexity over mesh quality and skip potential mesh improving strategies such as Delaunay refinement. More important in order to not exclude good integer solutions is that all crucial DOFs are available as discussed next.



Splitting Integer-critical Edges: Mapping from the continuous relaxation to the integer solution requires the singular vertices to snap onto the integer-grid $\mathbb{Z} \times \mathbb{Z}$. From this point of view a triangle with a large interior angle is problematic – it is likely that the best way to arrange its vertices in the integer-grid is on a line resulting in a degenerate triangle. To not exclude the possibility of arranging the vertices on a line, after decimation we split all *integer-critical edges*. An edge is integer-critical if it is incident to a triangle with three singular vertices where the angle opposing the edge is large. We choose a conservative angle threshold of 110° to capture all cases that might be problematic. Each split separates two singular vertices by a regular one and the number of integer-critical edges decreases monotonically. Thus, splitting integer-critical edges sorted by decreasing length is sufficient.

Maintaining the Map $f : \mathcal{M}_- \to \mathcal{M}$: The described decimation algorithm is conceptually simple. The only technically more involved aspect is related to the fact that we do not have a global embedding of $\mathcal{M}_{\text{relaxed}}$ into 2D. Consequently transition functions $g_{i \to j}$ between different charts have to be taken into account. The setting is comparable to [Pietroni et al. 2010; Khodakovsky et al. 2003] which potentially could be adapted to our setting. However, for our special case of a pure re-tessellation we propose a simpler

and more efficient solution. Considering that we know f_{relaxed} it is sufficient to track $f_{\text{deci}} : \mathcal{M}_- \to \mathcal{M}_{\text{relaxed}}$ while performing the decimation operations. Note that f_{deci} restricted to the chart of a single triangle $t_i \in \mathcal{M}_-$ is a simple linear map between two 2D tangent spaces. Accordingly, for each triangular chart t_i we allocate a function $\Psi_i : \mathbb{R}^2 \to T_{\mathcal{M}_{\text{relaxed}}} \times \mathbb{R}^2$ that maps from the tangent space of $t_i \in T_{\mathcal{M}_-}$ to the tangent space of $t_j \in T_{\mathcal{M}_{\text{relaxed}}}$. To avoid potential ambiguities due to global non-injectivities of a single-chart representation of f_{relaxed}, we require that the image of t_i's barycenter $\Psi_i(\mathbf{m}) = (t_j, \psi_i(\mathbf{m}))$ satisfies $\psi_i(\mathbf{m}) \in t_j$, i.e. it is inside the image triangle t_j. The situation is depicted on the right. Notice that expressing the green triangle patch in the tangent space of t_j in general involves transition functions.



In the beginning those maps are trivially initialized with the identity function $\Psi_i(\mathbf{u}) = (t_i, \mathbf{u})$. For all operations applied during decimation, we now state how to update the Ψ functions. To perform an edge collapse/flip/split operation in $\mathcal{M}_-$ it is convenient to first express the required neighborhood of triangles in a common coordinate chart by transforming their 2D coordinates with the corresponding transition functions g. In this case we update $\Psi_i^{\text{new}} = \Psi_i \circ g^{-1}$. If the geometry of the triangle changes such that the barycenter moves from $\mathbf{m}_o$ to $\mathbf{m}_n$, we map the update path via ψ_i to $\mathcal{M}_{\text{relaxed}}$. If the image of the path subsequently visits triangles $(t_{j_1} \dots t_{j_n})$, the map has to be updated by the corresponding transition functions $\eta = g_{j_{n-1} \to j_n} \circ \dots \circ g_{j_1 \to j_2}$ and consequently $\Psi_i^{\text{new}}(\mathbf{u}) = (t_{j_n}, \eta \circ \psi_i(\mathbf{u}))$. For all of our operations, the straight line between old and new barycenter is guaranteed to be within the local neighborhood of affected triangles and thus can be mapped unambiguously.

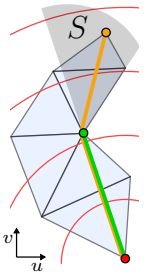
3.3 Singularity Separation

In this section we address the issue that the minimizer of the continuous relaxation might be arbitrarily far away from the best possible integer solution. The reason is that singular vertices might get arbitrarily close to each other within the continuous relaxation, while their distance in the integer-grid $\mathbb{Z} \times \mathbb{Z}$ of a valid IGM can never be less than 1. If we, for example, want to compute the coarse quad mesh of Figure 1 (rightmost), the minimizer of the continuous relaxation covers a total area of less than one (completely fits into one integer-grid cell), while the optimal integer solution shown requires 112 cells. Such a far-away continuous relaxation challenges modern branch-and-cut solvers, since they are very likely to spend much time searching solutions in infeasible sub-trees until enough information about the problem is collected to succeed. Instead of letting the solver algebraically search for separation conditions, it is much more efficient to provide geometrically derived ones right from the start. In optimization literature such conditions that do not cut away good solutions but improve the continuous relaxation are called cuts, explaining the second term in "branch-and-cut".

Based on the insights of the above example, our strategy is as follows. We search for pairs of singularities with a $\infty-$norm distance of less than 1 and add a linear condition that spreads it to at least 1. For each singular vertex s_i, we grow a geodesic disc G_i of radius $\sqrt{2}$ within the mesh representing the decimated continuous relaxation $\mathcal{M}_-$. All singularities s_j within the geodesic disc G_i with $i < j$ are candidates for a singularity separation condition. For each such pair (s_i, s_j) we trace the shortest geodesic path P connecting them. If P contains another singularity s_k with $k \notin \{i, j\}$ we reject the candidate, since it is redundant due to shorter candidate paths (s_i, s_k) and (s_k, s_j). Otherwise, we express the difference vector in a common chart $(\Delta u, \Delta v)^T = (u_i, v_i)^T - \eta(u_j, v_j)$, where η is the combined transition function along the path as before. Separation is performed along the u axis if $|\Delta u| \geq |\Delta v|$ leading to the condition $|\Delta u| \geq 1$ and otherwise along the v axis with an analog condition. Linearization of this condition is done by fixing the sign found in $\mathcal{M}_-$, leading to

$$\begin{aligned} \left\{\text{sgn}\,(\Delta u)|_{\mathcal{M}_-}\right\} \cdot \Delta u \geq 1 \\ \text{or} \quad \left\{\text{sgn}\,(\Delta v)|_{\mathcal{M}_-}\right\} \cdot \Delta v \geq 1 \end{aligned} \tag{5}$$

for u and v separators respectively. Notice that these conditions depend linearly on translational variables introduced by the transition functions of η. On the right side an example is shown with geodesic iso-contours in red, a valid separation path in green and an invalid one in orange. The orange path is not a straight line, since its singularity lies in the shadow region S of the green singularity, which can be induced either by a non-convex boundary point or a cone singularity with a total angle larger than 2π. Notice that typically many such paths exist, which can be skipped.



To compute geodesic distances we use the Fast Marching method [Kimmel and Sethian 1998] driven by the circle update [Novotni and Klein 2002], which is exact in the obstacle-free planar case – the only interesting one for us due to rejection of all paths that are not straight connections between singularities.

It is clear that the above separation constraints do not induce an infeasible problem since a global scaling of the whole mesh by a large positive factor is sufficient to separate all singularities and thus to satisfy all conditions. Moreover it is very unlikely that good solutions are lost since the orientation field energy strongly increases if two singularities swap their spatial ordering.

Re-computing the continuous relaxation with all generated singularity separation conditions effectively avoids the clustering of singularities. However, in rare cases the separation conditions might induce new pairs of close singularities which are then separated by iterating the above approach. Due to anisotropic stretching induced by the separators edges may become integer-critical. Similarly to 3.2 we remove them through edge splits.

3.4 Algorithm and Implementation Details

Based on the described components, our algorithm can be understood as a series of bijective maps between meshes. As depicted in Figure 6 it is composed of the following steps:

1. Globally smooth parametrization ($\mathcal{M} \to \mathcal{M}_{\text{relaxed}}$)
2. Decimation in parametric space ($\mathcal{M}_{\text{relaxed}} \to \mathcal{M}_-$)
3. Addition of Singularity Separators ($\mathcal{M}_- \to \mathcal{M}_-^{\text{sep}}$)
4. Optimization of MIQP ($\mathcal{M}_-^{\text{sep}} \to \mathcal{M}_{\text{int}}$)
5. (optional) Refinement of map ($\mathcal{M}_{\text{int}} \to \mathcal{M}_{\text{IGM}}$)
6. Quad mesh extraction ($\mathcal{M}_{\text{IGM}} \to \mathcal{Q}$)

In the following we describe additional aspects and implementation details of our algorithm that has been implemented in C++ as a plu-

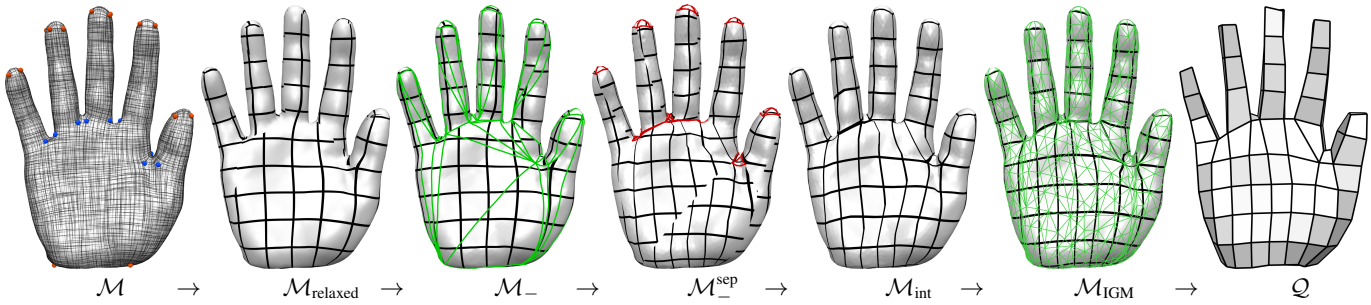


Figure 6: *Algorithm overview from left to right: Starting with a cross-field on a hand model $\mathcal{M}$, a series of maps is constructed to generate the quad mesh $\mathcal{Q}$. Since intermediate meshes cannot be embedded in 3D, the image of the integer-grid on $\mathcal{M}$ is shown for visualization instead. At first the relaxation of the mixed-integer problem leads to $\mathcal{M}_{relaxed}$, which is subsequently decimated within its flat parts into $\mathcal{M}_{-}$, where the new tessellation is overlaid in green. Due to violated integer conditions the grid does not conform everywhere. Singularity separators, shown in red on $\mathcal{M}_{-}^{sep}$, are added to make it more similar to the unknown IGM $\mathcal{M}_{int}$. Then $\mathcal{M}_{int}$ is computed by branch-and-cut on the low-dimensional and improved problem. Optional refinement of $\mathcal{M}_{int}$ adds DOFs to get smoother isolines in $\mathcal{M}_{IGM}$, which is eventually contoured to the resulting quad mesh $\mathcal{Q}$.*

gin for the geometry processing framework OpenFlipper [Möbius and Kobbelt 2012].

(optional) Refinement: The refinement of step 5 can be seen as an inverse of the decimation. The coarse mesh $\mathcal{M}_{-}$ provides enough DOFs to accurately determine the integer part of the desired IGM, i.e. the connectivity of the quad mesh. However, its extreme coarseness sometimes leads to unpleasant distortions of the iso-lines which can be avoided by a denser mesh with more DOFs. For the runtime the refinement is not critical as long as we only optimize continuous DOFs while keeping the integers fixed. Our refinement simply splits edges in $\mathcal{M}_{\text{int}}$ at their midpoint, prioritized by decreasing edge length. The algorithm terminates if all edges are shorter than a user-prescribed tolerance with a default value of 1, where the number of triangle mesh vertices roughly coincides with that of the quad mesh. We then optimize the map while keeping the $\mathbf{t}_{ij}$ integer DOFs of the transition functions fixed. This step is optional and can be easily left out or replaced by a more sophisticated adaptive up-sampling, which only adds new DOFs in distorted regions.

Optimization of (P2) type problems is done in steps 1, 3, 4 and 5, which require some more explanation. Instead of redundantly allocating variables in each triangle, vertices are merged along a dual spanning-tree. In steps 1, 3 and 4, an arbitrary singularity is additionally fixed to (0,0) in order to obtain a unique minimum. Both steps are common practice and details can be found in [Bommes et al. 2009]. For conditions (4) we need 2D reference triangles to determine appropriate trisectors. In step 1 they are chosen based on the input cross- and sizing-field, i.e. the reference triangles minimize $\|R_t^T \nabla f_t - H_t\|_\alpha^2$. Since the subsequent steps solely map between 2D spaces, reference triangles for 3, 4 and 5 are automatically available through the latest configuration. This means that trisectors are greedily re-initialized in every step, which in spirit is similar to local global approaches [Bouaziz et al. 2012]. It is worth noting that only the low-dimensional optimization of step 5 implies integer conditions.

Features and Boundaries: Preservation of features and boundaries can be integrated in a straightforward way. We require an explicit representation of feature edges in the input mesh and add linear conditions, ensuring that such edges are mapped onto integer-grid lines (cf. [Bommes et al. 2009]). The feature graph (also containing all boundaries) consists of feature lines, that are potentially chains of mesh edges, which are either closed or end at feature vertices. During the decimation, it is sufficient to treat feature vertices like singularities and forbid their movement. If we additionally forbid all halfedge-collapses between vertices lying on different feature lines, the rest of the algorithm can proceed as before.

Numerical Solver: In all of our examples, we apply the interior point method IPOPT [Wächter and Biegler 2006] for continuous optimization problems and the branch-and-cut algorithm of CPLEX [IBM 2012] for problems containing integer variables. CPLEX is restricted to quadratic programs with $k = 2$ and we leave higher order optimizations for the future. Alternatively CPLEX could also be used for the continuous optimization but IPOPT is faster if the constant Hessian and constant Jacobian of the constraints are exploited. Apart from this option we use the default settings, which probably leaves some potential for performance tuning.

Lazy Constraints: In steps 1 and 5, the conditions (4) for consistent orientation are typically violated only for a small fraction of triangles. To benefit from this we implemented a lazy constraint evaluation. The idea is to start with an empty set of type (4) constraints and only if constraints are violated we add them and iterate both steps until a valid solution is found. To significantly reduce the number of outer iterations, we conservatively add all (normalized) constraints with a function value below 0.5. In all of our examples the lazy mechanism improved the performance. For example, the runtime of the buddha model of Figure 10 decreased from 9.3 to 4 minutes while the result is identical.

4 Results

We performed different experiments and comparisons in order to evaluate our algorithm. The algorithm requires a cross-field, a sizing field and parameters k-norm and anisotropy α as input. Instead of tuning these parameters for each example, we decided to apply "default" settings by choosing $\alpha = 0.1$ for quad layout generation and $\alpha = 0.5$ else and $k = 2$ in all of our examples. Furthermore, except from one example we restrict to uniform sizing fields, i.e. a constant target edge length. The reason for these choices is that we do not intend to show the best possible solutions by tuning the inputs, but instead validate reliability and quality within an automated environment. Unless stated otherwise cross-fields have been generated with [Bommes et al. 2009].

Importance of Decimation and Singularity Separation: In order to demonstrate the importance of our decimation and singularity separation steps we performed an experiment with the BOTIJO model of Figure 1 (rightmost). To be able to find any solution without our optimizations, we reduced the mesh from 15k to 3k vertices (by keeping the singularities). In the first test we used our decimation strategy without addition of singularity separators and not a single solution could be found within 8h. In the second test we conversely skipped the decimation step but added singularity separators, leading to a first low-quality solution after 20 minutes. However, 8h were still not sufficient to find a solution comparable to Figure 1 (rightmost), which has been generated by our full algorithm with decimation and singularity separation within 17s. Disabling the split of integer-critical edges behaves similarly to the first experiment without singularity separators and the solver spends hours of computation without finding any feasible solution.

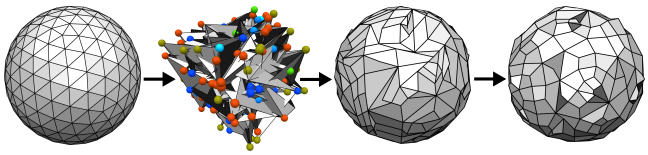


Figure 7: *Random Example: The 254 vertices of a sphere are randomly positioned inside a unit-cube, where a random cross-field is constructed, resulting in 194 singularities with valences between 1 and 9. Despite the complicated setting, our reliable algorithm finds an IGM within 60s, leading to a valid quad mesh. After generation, the quad mesh is mapped back onto the sphere to better inspect the result. A few smoothing iterations uncover the correctly reproduced inhomogeneous distribution of singularities.*

Random Data: A very good way to validate reliability is to perform experiments with random data, as e.g. those in Figure 7. Starting with a sphere, we position all of its vertices at random positions within the unit-cube and generate a random cross-field as input for our algorithm. Solely for visualization reasons the resulting quad mesh is mapped back onto the input sphere and smoothed. The random field leads to extreme valences between 1 and 9, all handled by our algorithm without any problems. There is one mild condition for the input cross-field, i.e. no singularity index should be greater or equal to one, corresponding to a quad mesh valence of 0 or less and obviously impossible. In a cross-field, that is not artificially distorted, the angle between neighboring crosses is never larger than 45°, which gives a hint why random cross-fields still contain valid parametrization in the induced linearized space. However, theoretically there is no guarantees for that since counter-examples with artificial extreme distortions can be constructed easily. The situation is comparable to [Lipman 2012], where a local/global approach for resetting the cross-field is proposed to escape such constellations. In our setting, we never observed such a case, although we performed a large number of random examples. This is probably because the cross-field energy enables a good estimate of triangle rotations.

Coarse Quad Layouts: For the generation of coarse quad layouts, we compare against Dual Loops Meshing [Campen et al. 2012] w.r.t. identical models and cross-fields. We choose a target edge length of 20% of the largest bounding box edge. The comparison results are shown in the following table and Figure 8 respectively.

Model	Singularities	Patches		
		Dual Loops	Our Method	Enhanced CF
BOTIJO	72	221	175	112*
BLOCK	48	76	76	-
GUY	40	168	55	-
ELK	52	86	62	58
ROCKERARM	30	115	74	66
FERTILITY	48	98	117	85

*see Figure 1

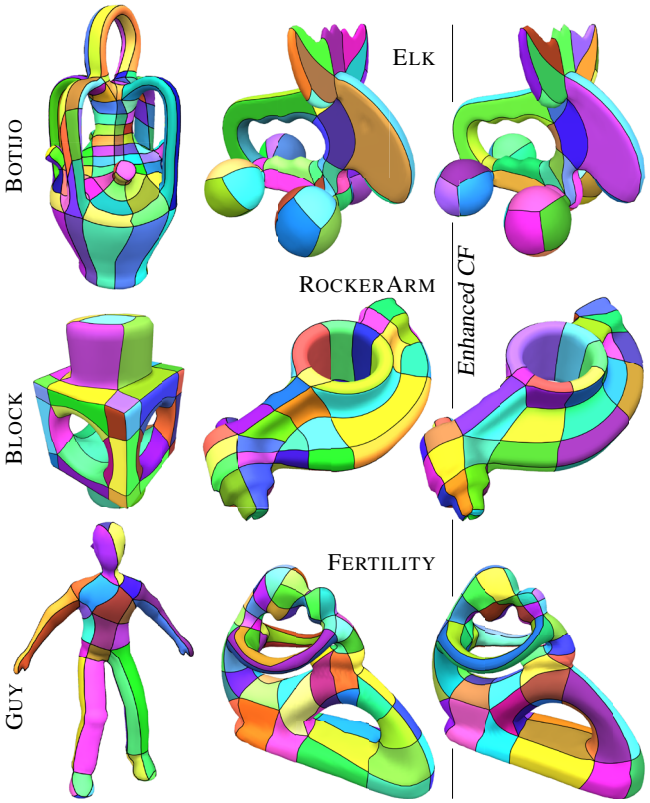


Figure 8: *Coarse quad layouts generated with our approach.*

The global branch-and-cut optimization strategy leads to quad layouts with fewer patches compared to the greedy strategy of [Campen et al. 2012], while still being well aligned to the given cross-field. Comparing many results, we found that there is another characteristic difference between both algorithms. The Dual Loops algorithm is not affected much by the precise location of irregular vertices, since it usually searches for curves far away from them. Our algorithm, however, searches for the map with minimal energy, where the position of singularities has a stronger influence. Therefore, we provide additional experiments with enhanced cross-fields, where some singularities are manually moved to geometrically more meaningful positions. The results show that for some models like BOTIJO slight improvements of the cross-field lead to significant improvements of the quad layout.

Comparison to Greedy Rounding: We compare our algorithm, which we call Reliable Mixed-Integer Quadrangulation (RMIQ), against QuadCover (QC) [Kälberer et al. 2007] and Mixed-Integer Quadrangulation (MIQ) [Bommes et al. 2009] and its variant with stiffening (MIQ Stiff). The first experiment is shown in Figure 1. The results of another representative model, i.e. the ROCKERARM,

Algorithm	# Degenerate Faces / Run Time (s)							
	$h = 10^{-4}$		$h = .025$		$h = .05$		$h = .2$	
QUADCOVER	8	2.5	51	2.5	82	2.6	2047	2.6
MIQ	8	3.9	18	3.8	42	4.2	568	4.0
MIQ STIFF	0	12.3	0	111.3	1	166.6	541	155.6
RMIQ 1S	0	12.0	0	12.0	0	11.9	0	11.6

Table 1

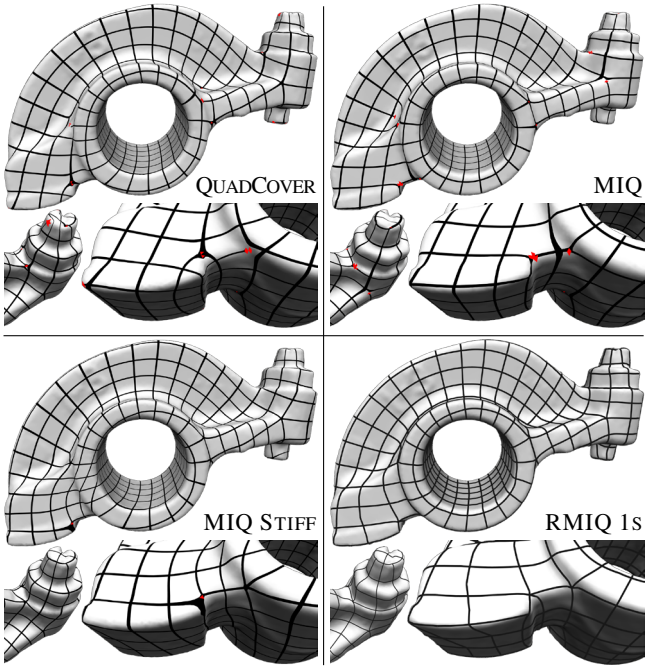


Figure 9: *The shown parametrizations are generated with $h = .05$. Degeneracies in the map are highlighted in red.*

are shown in Figure 9 and Table 1. While QuadCover and MIQ fail to generate a valid mapping even for the smallest target edge length of 10^{-4}, MIQ with stiffening succeeds up to a size where the integer-rounding aspects become dominant. Notice that the improved robustness of stiffening comes at the price of a large increase in runtime. Our method, however, efficiently produces valid parametrizations for arbitrarily large target edge lengths. The runtime, which is less than 12 seconds, is dominated by the first relaxed solution including all vertices of the input triangle mesh (35k for the ROCKERARM), since the branch-and-cut algorithm found solutions close to the optimum already within 1 second. It is also worth to mention, that the solution quality is superior to greedy rounding, leading to better results. Interestingly, the solver gets faster in finding the optimal solution with increasing target edge length. The reason is that more and more separator constraints are generated such that the feasible integer space close to the optimal solution gets smaller.

Figure 10 shows additional results where the largest is the buddha model with 200k faces, efficiently computed within 4 minutes due to the lazy constraint mechanism. Sharp features of the mechanical models are preserved as explained in Section 3.4.

Non-uniform sizing is not the main topic of this paper and we refer the reader for details to [Kovacs et al. 2010; Zhang et al. 2010]. In our opinion sizing belongs to the cross-field generation step be-

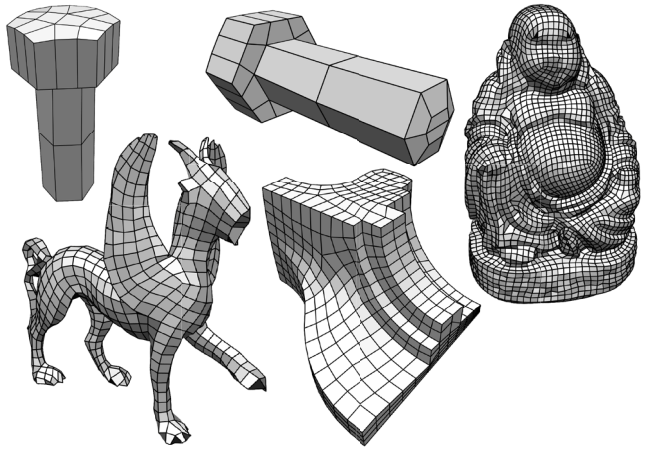


Figure 10: *Additional result showing feature handling and two more complex models.*

cause orientation and sizing cannot be chosen independently in a quad mesh. However, as a proof of concept Figure 11 shows a quad mesh of the BOTIJO model generated with a linearly changing sizing function and the cross-field as used before. Since the cross-field singularities are not synchronized with the sizing function, undesired distortions in the transition regions are unavoidable. However, it is worth mentioning that even for such inconsistent input data our approach reliably finds the solution which is closest to the desired orientation and sizing.

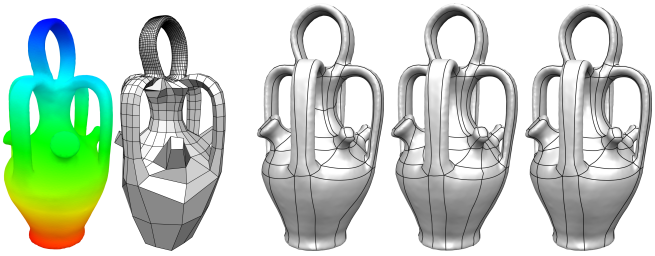


Figure 11: *(a) A quadmesh resulting from a linearly varying sizing field. (b)-(d): Different variants of Lipman's constraints applied to the example of Figure Figure 1. The ∞-norm variant unnecessarily subdivides the long and thin patch, resulting in 124 quads. Similarly the 1-norm refines in a different location, also leading to 124 quads. The octagon version is the union of both previous spaces and thus able to generate a coarser mesh with 112 quads, as our approach does. The overall runtime was 71s, 58s, 227s, 45s and 17s for ∞-norm, 1-norm, octagon, our constraints and our lazy constraints respectively.*

Comparison to Bounded Distortion Mapping Spaces: Recently Lipman proposed the powerful concept of bounded distortion mapping spaces [Lipman 2012] that is related to one aspect of our algorithm, namely the orientation consistency constraints. Consequently we compare both in order to clarify similarities and differences. In a nutshell, the setting of [Lipman 2012] is the following: Given a 2D affine map with a linear part of $A = \left(\begin{smallmatrix} a+c & d-b \\ d+b & a-c \end{smallmatrix}\right)$, orientation flips are prevented and the conformal distortion is bounded to C by the convex cone condition $\|(c,d)^T\|_2 \leq a\frac{C-1}{C+1}$ with $a > 0$. A conservative linear version results from exchanging $\|\cdot\|_2$ with $\sqrt{2}\|\cdot\|_\infty$. The first obvious difference is that we do not explicitly

bound conformal distortion, since we neither want to exclude rectangles with strong anisotropy nor want to preserve triangle quality. Both constraint types depend on a reference configuration, chosen for linearization. Our constraints depend explicitly on the shape, while those of Lipman constrain the mapping function instead. We performed experiments with the above constraints and found that the quadratic cone-constraints are significantly slower within the branch-and-cut algorithm, increasing the runtime to an impractical level. The ∞-norm linearization on the other hand leads to a runtime comparable to ours, however, it cuts away solutions that are relevant for our integer-grid maps. Analyzing the constraint for the map $A = \mathrm{diag}(w, h)$ shows that the ∞-norm restricts anisotropy to $w/h \leq (\sqrt{2}+1)/(\sqrt{2}-1) \approx 5.8$ and the same for h/w, even for $C = \infty$. To overcome this limitation, we investigated different linearizations, namely a 1-norm and an octagon approximation as shown on the side. The green disc is a slice of the cone, corresponding to a specific a value. The approximation with an n-gon requires n linear inequality conditions resulting in a tradeoff between accuracy and runtime, that is not available for our constraints. In practice, however, except for the limited aspect ratio of the ∞-norm case, we observed only slightly differences in the overall behavior of our constraints, the 1-norm and the octagon version and conclude that the corresponding feasible regions, although different, all are rich enough for our purpose. Figure 11 (b)-(d) shows an experiment comparing the different alternatives on a complicated example. Thus we could potentially exchange our orientation consistency constraints by the 1-norm version of [Lipman 2012]. The main reason to introduce our novel set of constraints is that in contrast to [Lipman 2012], our constraints generalize in a straightforward way to the volumetric 3D case as outlined Section 5 and thus will be important for future work, e.g. in the area of hexahedral meshing.

Limitations: Since we search for the IGM which best reproduces a given sizing- and cross-field, it is not surprising that the resulting quality strongly depends on them. For example, while searching for coarse quad layouts, sub-optimally placed singularities might lead to an energetically optimal solution, which is not the one a human designer has in mind. Another limitation stems from the fact that, although we strongly reduce the problem complexity, the branch-and-cut algorithm still has an exponential runtime component within the discrete unknowns. Consequently, the algorithm is likely to extremely slow down for meshes with thousands of singularities. In practice it seems to be hard to precisely estimate the feasible range, since the overall time strongly varies due to many unpredictable criteria that support or handicap the success of the branch-and-cut strategy.

5 Conclusion and Outlook

We presented a reliable approach to quad remeshing that unifies the generation of fine-scale quad meshes and coarse quad layouts. We believe that our method will be an important building block for many future approaches, that build on top of its reliability. A unique feature is that due to our aggressive yet accurate complexity reduction, we are able to perform a global optimization approach instead of the commonly used greedy strategy. Interesting directions for future work include the generation of sizing fields, a degree of freedom that we did not exhaust yet and the interleaved optimization of singularities, which in our setting are kept fixed.

The Volumetric Case: Another very promising direction is the extension of our reliable approach to the volumetric case. Once a (valid) 3D cross-field is available, volumetric integer-grid parametrization approaches like [Nieser et al. 2011; Li et al. 2012] follow a structure similar to the surface case. Our paper is designed in a way, where the extension of each step to 3D seems to be possible. For each tetrahedron, the orientation consistency constraint can be linearized analogously to Section 3.1 by choosing an interior point and generating a separating plane through it for each edge of the tetrahedron. A similar way of decimation, separator generation between singular lines and solution of the quadratic integer problem seems to be possible as well.

Acknowledgements

This project was funded by by the European Research Council (ERC Starting Grant 'Robust Geometry Processing', Grant agreement 257474); the DFG Cluster of Excellence on Ultra-high Speed Mobile Information and Communication (UMIC), German Research Foundation grant DFG EXC 89, `http://www.umic.rwth-aachen.de`; and the Aachen Institute for Advanced Study in Computational Engineering Science (AICES). Models have been obtained from the *AIM@SHAPE* repository and the Image-based 3D Models Archive, Telécom Paris. The GUY model was initially created using Cosmic Blobs© by Dassault Systemes Solidworks Corp. Additionally we would like to thank Jan Möbius for the geometry processing framework `http://www.OpenFlipper.org` and the reviewers for their competent and helpful comments.

References

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, 1–10.

BOMMES, D., VOSSEMER, T., AND KOBBELT, L. 2010. Quadrangular parameterization for reverse engineering. In *Proc. on Mathematical Methods for Curves and Surfaces*, Springer-Verlag, Berlin, Heidelberg, MMCS'08, 55–69.

BOMMES, D., LEMPFER, T., AND KOBBELT, L. 2011. Global structure optimization of quadrilateral meshes. *Comput. Graph. Forum 30*, 2, 375–384.

BOMMES, D., LÉVY, B., PIETRONI, N., PUPPO, E., SILVA, C., TARINI, M., AND ZORIN, D. 2012. State of the art in quad meshing. In *Eurographics STARS*.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LÉVY, B. 2010. *Polygon Mesh Processing*. AK Peters.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. *Comp. Graph. Forum 31*, 5 (Aug.), 1657–1667.

CAMPEN, M., BOMMES, D., AND KOBBELT, L. 2012. Dual loops meshing: Quality quad layouts on manifolds. *ACM Trans. Graph. 31*, 4.

CATMULL, E., AND CLARK, J. 1998. Seminal graphics. ACM, New York, NY, USA, ch. Recursively generated B-spline surfaces on arbitrary topological meshes, 183–188.

DANIELS, J., SILVA, C. T., SHEPHERD, J., AND COHEN, E. 2008. Quadrilateral mesh simplification. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, ACM, New York, NY, USA, 1–9.

DANIELS, II, J., SILVA, C. T., AND COHEN, E. 2009. Localized quadrilateral coarsening. In *SGP '09: Proceedings of the Symposium on Geometry Processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1437–1444.

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. C. 2006. Spectral surface quadrangulation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, 1057–1066.

ECKSTEIN, I., SURAZHSKY, V., AND GOTSMAN, C. 2001. Texture mapping with hard constraints. *Comput. Graph. Forum 20*, 3.

FLOATER, M. S., AND HORMANN, K. 2005. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, N. A. Dodgson, M. S. Floater, and M. A. Sabin, Eds., Mathematics and Visualization. Springer, Berlin, Heidelberg, 157–186.

GU, Z., ROTHBERG, E., AND BIXBY, R., 2011. Gurobi optimizer 4.5: `http://www.gurobi.com`.

HORMANN, K., LÉVY, B., AND SHEFFER, A. 2007. Mesh parameterization: theory and practice. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, 1.

HUANG, J., ZHANG, M., MA, J., LIU, X., KOBBELT, L., AND BAO, H. 2008. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph. 27*, 5, 1–9.

IBM, 2012. Ilog cplex optimizer 12: `http://www.ibm.com`.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3 (Sept.), 375–384.

KHODAKOVSKY, A., LITKE, N., AND SCHRÖDER, P. 2003. Globally smooth parameterizations with low distortion. In *ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, SIGGRAPH '03, 350–357.

KIMMEL, R., AND SETHIAN, J. A. 1998. Computing geodesic paths on manifolds. In *Proc. Natl. Acad. Sci. USA*, 8431–8435.

KOVACS, D., MYLES, A., AND ZORIN, D. 2010. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, ACM, New York, NY, USA, SPM '10, 137–146.

LI, E., LÉVY, B., ZHANG, X., CHE, W., DONG, W., AND PAUL, J.-C. 2011. Meshless quadrangulation by global parametrization. *Computer and Graphics*.

LI, Y., LIU, Y., XU, W., WANG, W., AND GUO, B. 2012. All-hex meshing using singularity-restricted field. *ACM Trans. Graph. 31*, 6 (Nov.), 177:1–177:11.

LIPMAN, Y. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31*, 4 (July), 108:1–108:13.

MICCIANCIO, D. 2006. The hardness of the closest vector problem with preprocessing. *IEEE Trans. Inf. Theor. 47*, 3 (Sept.), 1212–1215.

MÖBIUS, J., AND KOBBELT, L. 2012. Openflipper: An open source geometry processing and rendering framework. In *Curves and Surfaces*, vol. 6920 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 488–500.

MURDOCH, P., BENZLEY, S., BLACKER, T., AND MITCHELL, S. 1997. The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes. *Finite Element in Analysis and Design 28*, 2 (December), 137–149.

NIESER, M., REITEBUCH, U., AND POLTHIER, K. 2011. Cubecover- parameterization of 3d volumes. *Comput. Graph. Forum 30*, 5, 1397–1406.

NOVOTNI, M., AND KLEIN, R. 2002. Computing geodesic distances on triangular meshes. In *The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002 (WSCG'2002)*.

OWEN, S. J., STATEN, M. L., CANANN, S. A., AND SAIGAL, S. 1999. Q-morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering 44*, 9, 1317–1340.

PARDALOS, P., AND RESENDE, M. 2002. *Handbook of applied optimization*. Oxford University Press, Incorporated.

PIETRONI, N., TARINI, M., AND CIGNONI, P. 2010. Almost isometric mesh parameterization through abstract domains. *IEEE Transaction on Visualization and Computer Graphics 16*, 4 (July/August), 621–635.

PIETRONI, N., TARINI, M., SORKINE, O., AND ZORIN, D. 2011. Global parametrization of range image sets. *ACM Transactions on Graphics, Proceedings of SIGGRAPH Asia 2011 30*, 6.

RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph. 25*, 4, 1460–1485.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27*, 2, 1–13.

SHEFFER, A., PRAUN, E., AND ROSE, K. 2006. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2* (January), 105–171.

SPRINGBORN, B., SCHRÖDER, P., AND PINKALL, U. 2008. Conformal equivalence of triangle meshes. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, 1–11.

TARINI, M., PIETRONI, N., CIGNONI, P., PANOZZO, D., AND PUPPO, E. 2010. Practical quad mesh simplification. *Computer Graphics Forum (Special Issue of Eurographics 2010 Conference) 29*, 2, 407–418.

TARINI, M., PUPPO, E., PANOZZO, D., PIETRONI, N., AND CIGNONI, P. 2011. Simple quad domains for field aligned mesh parametrization. *ACM Transactions on Graphics, Proceedings of SIGGRAPH Asia 2011 30*, 6.

TONG, Y., ALLIEZ, P., COHEN-STEINER, D., AND DESBRUN, M. 2006. Designing quadrangulations with discrete harmonic forms. In *Proc. of SGP*, EG Association, Aire-la-Ville, Switzerland, SGP '06, 201–210.

WÄCHTER, A., AND BIEGLER, L. T. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program. 106* (May), 25–57.

WEBER, O., MYLES, A., AND ZORIN, D. 2012. Computing extremal quasiconformal maps. *Comp. Graph. Forum 31*, 5 (Aug.), 1679–1689.

YU, H., LEE, T.-Y., YEH, I.-C., YANG, X., LI, W., AND ZHANG, J. J. 2012. An rbf-based reparameterization method for constrained texture mapping. *IEEE Transactions on Visualization and Computer Graphics 18*, 7 (July), 1115–1124.

ZHANG, M., HUANG, J., LIU, X., AND BAO, H. 2010. A wave-based anisotropic quadrangulation method. *ACM Trans. Graph. 29* (July), 118:1–118:8.