Parametrization Quantization with Free Boundaries for Trimmed Quad Meshing

MAX LYON, RWTH Aachen University
MARCEL CAMPEN, Osnabrück University
DAVID BOMMES, University of Bern
LEIF KOBBELT, RWTH Aachen University

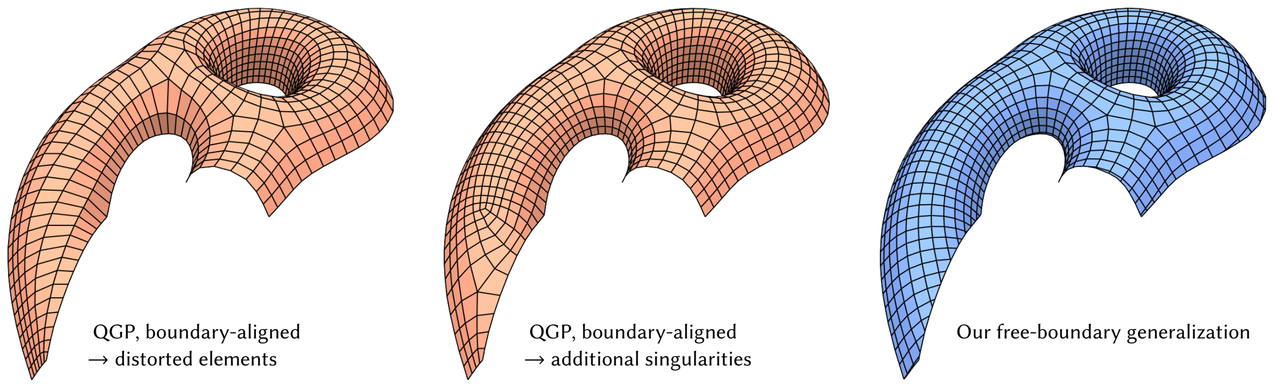


Fig. 1. When generating quad meshes for given surfaces, alignment of the mesh to the surface boundary may or may not be relevant, depending on the use case. Enforcing boundary alignment when this is not necessary, needlessly leads to lower mesh quality, e.g. distorted elements (left) or additional irregular vertices (center). *Trimmed quad meshing* with non-aligned boundaries (right), enabled by our free-boundary generalization of integer grid map quantization (QGP, [Campen et al. 2015]), avoids these issues and yields meshes of higher quality in such cases.

The generation of quad meshes based on surface parametrization techniques has proven to be a versatile approach. These techniques quantize an initial seamless parametrization so as to obtain an integer grid map implying a pure quad mesh. State-of-the-art methods following this approach have to assume that the surface to be meshed either has no boundary, or has a boundary which the resulting mesh is supposed to be aligned to. In a variety of applications this is not desirable and non-boundary-aligned meshes or grid-parametrizations are preferred. We thus present a technique to robustly generate integer grid maps which are either boundary-aligned, non-boundary-aligned, or partially boundary-aligned, just as required by different applications. We thereby generalize previous work to this broader setting. This enables the reliable generation of trimmed quad meshes with partial elements along the boundary, preferable in various scenarios, from tiled texturing over design and modeling to fabrication and architecture, due to fewer constraints and hence higher overall mesh quality and other benefits in terms of aesthetics and flexibility.

CCS Concepts: • **Computing methodologies → Computer graphics**; **Mesh models**; **Mesh geometry models**; *Shape modeling*.

Authors' addresses: Max Lyon, Visual Computing Institute, RWTH Aachen University; Marcel Campen, Osnabrück University; David Bommes, Computer Science, University of Bern; Leif Kobbelt, Visual Computing Institute, RWTH Aachen University.

0730-0301/2019/7-ART51 $15.00
https://doi.org/10.1145/3306346.3323019

Additional Key Words and Phrases: motorcycle graph, seamless parametrization, seamless texturing, trimmed NURBS

ACM Reference Format:
Max Lyon, Marcel Campen, David Bommes, and Leif Kobbelt. 2019. Parametrization Quantization with Free Boundaries for Trimmed Quad Meshing. *ACM Trans. Graph.* 38, 4, Article 51 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323019

1 INTRODUCTION

Integer grid maps have been introduced as a versatile tool for the generation of high quality quad meshes based on surface parametrization [Bommes et al. 2013a, 2009; Kälberer et al. 2007; Tong et al. 2006]. Focus has often been on surfaces without boundary, or surfaces with boundary where quad edges coincide with the boundary everywhere (*boundary-aligned quad meshes*).

For a variety of applications, e.g. in simulation, texturing, structural and architectural design (cf. Section 2), mesh or grid map alignment to the surface boundary is neither necessary nor beneficial – rather, it brings in needless distortion: the mesh could be of higher quality (in terms of structural regularity, element shape, element sizing, feature or curvature alignment) if it was not forced to align with the boundary.

Unfortunately, straightforward application of the state-of-the-art parametrization quantization algorithm for the robust generation of integer grid maps [Campen et al. 2015] in a scenario *without* or with only *partial*, selective boundary alignment leads to a number of critical issues:

- The method relies on a *motorcycle graph partition* of the surface into four-sided patches; without complete boundary alignment, patches with more or less sides emerge (cf. Fig. 3).
- The method relies on these partition's patches being simply-connected, disk-homeomorphic; without complete boundary alignment, they can be of complex topology (cf. Fig. 3).
- The method relies on a parametric separation test to prevent degeneracies; without complete boundary alignment, this test yields false results, causing the parametrization to degenerate or distort in an uncontrolled manner.

1.1 Contribution

We present a generalized quantization method for global seamless parametrization to generate integer grid maps *without* enforced boundary alignment. It is generic in the sense that it supports free boundaries, aligned boundaries, as well as mixed, selectively aligned boundaries.

On a technical level, the key innovation lies in generalizations or replacements of the core components of the quantized global parametrization (QGP) technique of [Campen et al. 2015]. Concretely, we propose the following:

(1) a domain partitioning algorithm that guarantees simply connected four-sided parametrically rectangular patches also in the presence of arbitrary boundaries; partial patches are completed through virtual extensions in order to streamline subsequent algorithmic stages. (cf. Section 4)
(2) a technique to guarantee the parametric separation of critical points (singularities, features) – crucial for non-degeneration of the resulting quantized map – also in the presence of surface boundaries and with respect to these, as well as the non-degeneration of boundaries themselves. (cf. Section 5)
(3) a replacement of the final parametrization computation by a robust constructive technique, drawing from ideas of [Myles et al. 2014] but reducing complexity and generalizing to settings with non-aligned boundaries. (cf. Section 6)

A more detailed, technical overview of these contributions is given after a recap of QGP in Section 3.

We show the correctness of our generalization to the free-boundary setting, and demonstrate the increased flexibility and consequent advantages of non-aligned parametrizations and meshes.

2 RELATED WORK

Parametrization-based Quad Mesh Generation. The generation of semi-structured quadrilateral meshes through surface parametrization has proven to be a versatile approach. It supports automatic and interactive workflows as well as a multitude of local and global constraints concerning the resulting mesh geometry and connectivity. This field of integer grid mapping has been an active research topic for the past decade [Bommes et al. 2013a,b, 2009; Campen and Kobbelt 2014; Ebke et al. 2013, 2014, 2016; Kälberer et al. 2007; Kovacs et al. 2011; Liu et al. 2017; Marcias et al. 2013; Panozzo et al. 2014; Pietroni et al. 2011; Ray et al. 2010; Tong et al. 2006; Zhou et al. 2018]. It is based on finding a locally injective map from a given surface to the plane such that it pulls back the regular integer grid from the plane onto the surface, implying a quad mesh. To this end, the map needs to be seamless [Mandad and Campen 2019; Myles and Zorin 2012] and certain values (parametric positions of extraordinary vertices, translational components of transitions) need to be integers, defining an integer grid map [Bommes et al. 2013a].

The discrete optimization of the integer degrees of freedom of the problem (also referred to as *quantization* of the map) is one of the non-trivial challenges in this context. It has been addressed in a variety of ways. For instance, a continuous relaxed version of the problem can be optimized, followed by rounding/snapping to nearest integers, either atomically [Kälberer et al. 2007] or incrementally [Bommes et al. 2009; Nieser et al. 2011]. As this approach is non-robust in general (the resulting integers may not admit any non-degenerate integer grid map), advanced techniques have been proposed as well. It was shown that a generic branch-and-bound solver can be employed [Bommes et al. 2013a] for robustness. A special purpose optimization strategy tailored to the problem and guided by its geometric nature was subsequently proposed [Campen et al. 2015]. It was shown to be faster by several orders of magnitude in challenging cases. Unfortunately, this latter method does not support parametrizations with non-aligned boundary.

After the integer degrees of freedom have been settled, an integer grid map can be obtained via standard (non-convex) locally injective parametrization optimization or, more robustly, using the final parametrization construction step of [Myles et al. 2014]. However, this step likewise requires complete boundary alignment; non-aligned boundaries are not supported.

Our method is most closely related to the latter two works [Campen et al. 2015; Myles et al. 2014]; it can be viewed as a combination of their respective robustness benefits in the context of a generalization to the selective boundary alignment setting.

Other Quad Mesh Generation Paradigms. Our focus in this paper is on meshing using integer grid maps. A large number of other paradigms, with different properties, benefits, and limitations, have been proposed for the generation of quadrilateral meshes. An overview of the various approaches is given by a number of surveys [Armstrong et al. 2015; Bommes et al. 2013b; Campen 2017; Owen 1998; Shimada 2006].

Worth noting particularly for the non-boundary-aligned setting are the classes of methods based on local/periodic parametrization [Huang et al. 2018; Jakob et al. 2015; Ray et al. 2006; Schertler et al. 2017] or on Morse-Smale complexes [Dong et al. 2006; Fang et al. 2018; Huang et al. 2008; Ling et al. 2014; Zhang et al. 2010]. These naturally support the free-boundary setting – enforcing alignment actually requires extra measures. Our work provides an enrichment of the field by enabling integer grid maps to robustly and efficiently be used in that setting as well. Due to their different properties and specific benefits, additional use cases can be covered. For instance, any kind of non-local constraints – like global mesh connectivity constraints [Myles et al. 2010] and global holonomy prescription [Crane et al. 2010] – are inherently difficult to handle and not supported by current methods based on periodic parametrization or the Morse-Smale approach; precise control over extraordinary vertices likewise is an issue. A further benefit due to the integer grid map based approach is the immediate availability of a continuous, locally injective surface parametrization from the result, useful for applications beyond mesh generation. If other mesh generation techniques are used, some additional efforts may be necessary to this end.

Motorcycle Graphs. A central component of QGP as well as of our method is the parametric motorcycle graph, a variant of a construction proposed by [Eppstein and Erickson 1999]. We extend it in order to deal with free boundaries in a manner suitable for our purpose. This concept has also been used for quad mesh partitioning [Eppstein et al. 2008; Gunpinar et al. 2014], field-guided parametrization [Myles et al. 2014], quad layout construction [Razafindrazaka et al. 2015], and texturing [Schertler et al. 2018].

Applications of Non-Boundary-Aligned Grid Maps and Quad Meshes. One of the key application scenarios of quad meshes lies in the context of the Finite Element Method. Numerous extensions of this method to partial elements were proposed that facilitate dealing with and benefiting from interfaces and boundaries cutting through elements, e.g. X-FEM [Fries and Belytschko 2010; Nicolas et al. 1999], IGFEM [Kedi et al. 2015], CBF-FEM [Gu et al. 2011]. The texturing of objects (whether for virtual use or for physical fabrication) using a tiled texture or structure templates relies on grid parametrizations [Akleman et al. 2005]; alignment to the surface boundary can be unnecessary and would only needlessly incur distortion. Also in the context of architectural design, quad mesh structures without boundary alignment are of relevance, cf., e.g., [Pottmann et al. 2008; Zadravec et al. 2010]. For the reconstruction of surfaces on the basis of trimmed versions of NURBS [Farin and Hansford 2000], NURCCS, or other parametric representations, likewise a non-boundary-aligned regular or semi-regular grid-like surface parametrization builds the foundation.

3 OVERVIEW & RECAP

As our method is based on the high-level strategy of the quantized global parametrization (QGP) algorithm of Campen et al. [2015], in the following we summarize the this algorithm's key aspects; we refer to the original publication for further background. We then explain the issues that prevent it from handling the non-boundary-aligned setting and give an overview of our novel solutions.

3.1 Recap of QGP

Input to the method is a surface triangle mesh $\mathcal{M}$ with a (non-quantized) seamless parametrization $\mathcal{F}$. Seamlessness in this context refers to the parametrization being related by certain rigid transitions across a cutgraph on the surface. The method then proceeds in four main steps:

(1) Based on the input parametrization $\mathcal{F}$ a motorcycle graph [Eppstein et al. 2008] is constructed on $\mathcal{M}$ by simultaneously tracing iso-curves starting from all singular points of the parametrization. These traces form the arcs of a graph that partitions $\mathcal{M}$ into patches, cf. Fig. 2. In the case of the surface being closed or the parametrization being aligned to its boundary, the resulting partition is guaranteed to consist of four-sided disk-homeomorphic patches only. Parametrically, all these patches are rectangles.

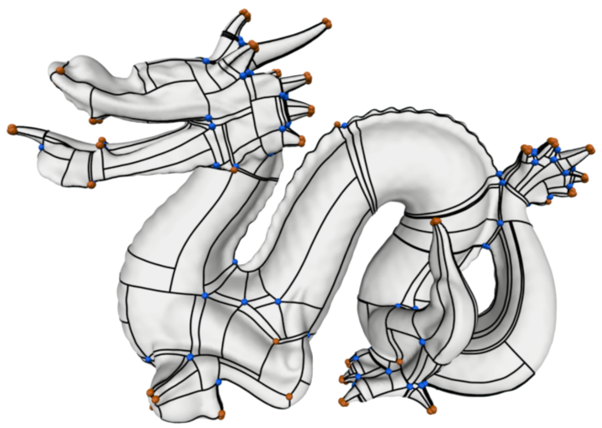


Fig. 2. Exemplary motorcycle graph on a closed surface. Arcs are depicted in black, extraordinary nodes in red (degree 3) and blue (degree 5).

(2) A quantization, i.e. an assignment of integer lengths to all motorcycle graph arcs, is constructed. It needs to be consistent (each patch needs to remain rectangular under these newly assigned lengths) while resembling the initial parametric arc lengths as closely as possible. It is constructed through an iterative greedy strategy that incrementally optimizes towards this goal while maintaining consistency throughout. The atomic operations that are used for consistency-preserving modification consist of equally increasing or decreasing the assigned lengths of each arc crossed by an elementary circuit of the dual graph of the partition.

(3) Besides consistency, another property is required of the quantization: certain critical points (singular points, feature points, boundary points) must be parametrically separated, i.e. their distance in terms of the newly assigned integer arc lengths must be greater than zero. Otherwise the quantization would imply degeneracies or inversions in the final integer grid map. Hence, after an initial phase where all arcs are assigned strictly positive integers, all further tentative modifications are only performed after passing a separation-preservation test concerning all involved critical points.

(4) Finally, a new parametrization of $\mathcal{M}$ is constructed, constraining parametric distances between pairs of critical points to the integer values given by the quantization (and one such point to the origin), yielding the final integer grid map.

3.2 Boundary Issues

The definition of quantization consistency, the definition of the dual graph, the separation-preservation test, as well as the final re-parametrization – these key ingredients of the above algorithm all rely on the partition's patches being simply connected four-sided *rectangular patches*:

Definition 3.1 (Rectangular Patch). A patch of a surface embedded graph, bounded by its arcs and/or the surface boundary, is called *rectangular* if it (modulo transitions across cuts) is mapped by the parametrization $\mathcal{F}$ bijectively onto an axis-aligned rectangular domain in the plane.

While with complete boundary-alignment the motorcycle graph is guaranteed to yield rectangular patches only [Eppstein et al. 2008], this fundamental assumption is violated when the parametrization is not aligned with the surface's boundary. In Figure 3a an exemplary motorcycle graph (with two nodes and eight arcs) on a surface that was parametrized without boundary alignment is depicted; the patches in this case are merely *sub-rectangular*:

Definition 3.2 (Sub-Rectangular Patch). A simply-connected patch is called *sub-rectangular* if it (modulo transitions) is mapped by the parametrization $\mathcal{F}$ bijectively onto a domain Ω in the plane, and there is an axis-aligned rectangle R such that $\Omega \subseteq R$ and $\partial_a \Omega \subseteq \partial R$, where $\partial_a \Omega$ is the part of the patch's boundary formed by graph arcs rather than the surface boundary.

The occurrence of such incomplete patches (and the related fact that the motorcycle graph forms multiple disconnected components, like in Figure 3a,) hinders the application of the QGP method; the dual graph is no longer well-defined, the consistency conditions cannot be formulated, and the separation test cannot be executed.

As depicted in Figures 3b and 3c, the situation can be even worse. Patches which are not four-sided, which are not simply connected, or which are more generally not rectangular or sub-rectangular can occur. These may arise even if the motorcycle graph does not even interact with the boundary, as shown in Figure 3b.

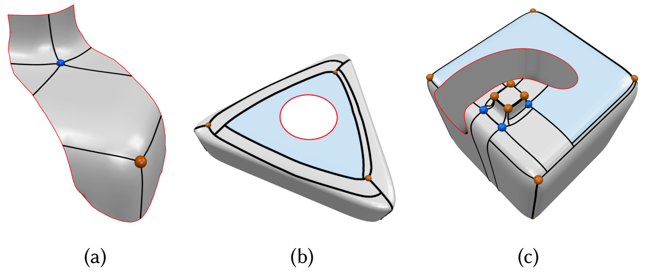
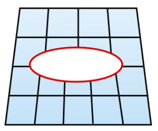


Fig. 3. Examples of input surfaces with free (non-aligned) boundary leading to motorcycle graphs (black arcs, red and blue nodes) with incomplete or otherwise complex and problematic patches (light blue).

3.2.1 Removing Boundaries? Observing these problems, a simple solution may come to mind: Why not simply fill all boundary loops using suitable surface patches to obtain a closed mesh, compute an integer grid map on the closed mesh, and remove the fillings in the end? While this indeed may be a viable option for certain simple local holes, this approach has many drawbacks. Not only is finding geometrically as well as topologically suitable patches a difficult task in general; closing all boundaries in such a way that the newly created geometry does not negatively affect and bias the parametrization result within the original surface is even more involved. Furthermore, the mere filling of a hole, regardless of chosen geometry, already reduces the degrees of freedom otherwise available in a mesh with boundary. The mesh in the inset for example could not be generated by a temporary hole filling approach due to the unequal number of quad strips emerging from the hole, top and bottom.



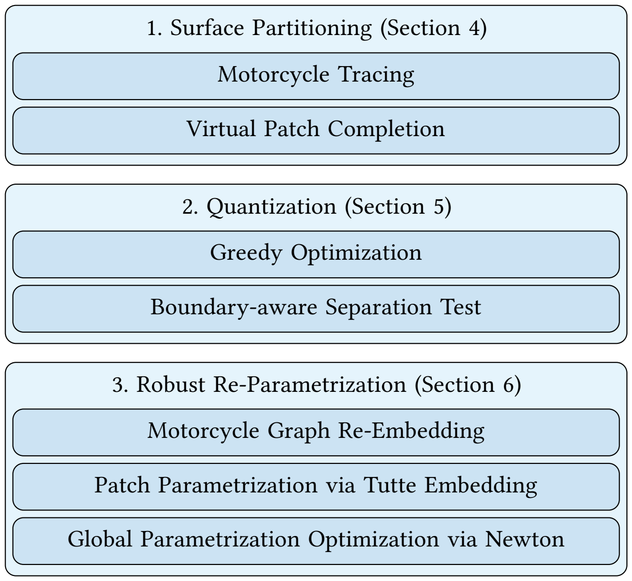

1. Surface Partitioning (Section 4)
Motorcycle Tracing
Virtual Patch Completion
2. Quantization (Section 5)
Greedy Optimization
Boundary-aware Separation Test
3. Robust Re-Parametrization (Section 6)
Motorcycle Graph Re-Embedding
Patch Parametrization via Tutte Embedding
Global Parametrization Optimization via Newton

Fig. 4. Overview of the three algorithmic stages of our method.

3.3 Outline

We deal with the issues that are due to non-aligned boundaries, and enable exploiting all degrees of freedom, in the following manner – ultimately resulting in the algorithm summarized in Figure 4.

Partitioning. In Section 4 we present a modification of the partition construction based on the motorcycle graph such that even in the presence of non-aligned boundaries all patches are guaranteed to be (at least) sub-rectangular. To this end, further arcs are added to the graph, generated by traces emanating from carefully chosen additional seed points. We then describe how virtual extensions can be added to sub-rectangular patches in order to make them virtually rectangular. This enables us to streamline the subsequent algorithmic stages, avoiding the need for numerous instances of special case handling for different types of incomplete patches.

Quantization. In Section 5 we introduce a parametric separation test suitable for the free boundary as well as the aligned boundary setting. Challenges that we need to address in this context are due to the virtual patch extensions and due to the additional need to prevent the collapse of free boundaries – issues that do not occur in the classical case of entirely aligned boundaries. With this generalized test available, thanks to our virtual patch completion strategy the core routine of QGP that incrementally constructs consistent quantizations can then be employed with only minor adjustments.

Re-Parametrization. In Section 6 we describe a robust procedure to obtain a global integer grid map in form of a locally injective parametrization, exactly adhering to the determined valid quantization. This is achieved by constructing guaranteed bijective discrete harmonic parametrizations for each individual patch of the motorcycle graph. These parametrizations are over rectangular domains with size given by the assigned integer arc lengths. By fixing the rectangle boundary map compatibly between adjacent patches, their union

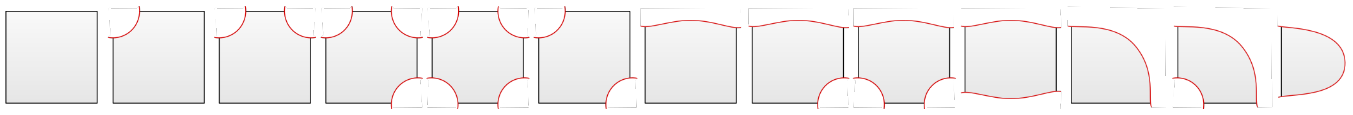


Fig. 5. The 13 configurations of patches (all of them sub-rectangular) that can occur in our free-boundary motorcycle graph. Arcs are depicted in black, boundary segments in red. The depicted shape of the red boundary segments is exemplary and can vary – to some extent, as long as it does not form any concave peaks in the interior.

is guaranteed to form a global integer grid map (which can serve as valid initialization for further distortion optimization). To apply this strategy in our setting we need to overcome two challenges: There may be (and commonly are) arcs of quantized length zero and patches of quantized area zero, and there are sub-rectangular patches at non-aligned boundaries. We re-embed the motorcycle graph such that zero-arcs have length zero and zero-patches have area zero on the surface; only in this way is a non-degenerate mapping possible. We describe an approach to efficiently support free boundaries in this re-embedding process; straightforward generalization of a previous zero-chain collapse technique [Myles et al. 2014] would lead to numerous special cases requiring separate treatment, so we instead break this global operation down into simple local elementary operators. Finally, the problem of sub-rectangular mapping can be reduced to that of rectangular mapping through virtual arc realization along the surface boundary.

4 PARTITIONING

The motorcycle graph $\mathcal{T}$ (cf. Figure 2) consists of *nodes* $n_i \in \mathcal{N}$ and *arcs* $a_i \in \mathcal{A}$. Every arc consists of two directed, opposite halfarcs $h_i \in \mathcal{H}$. Such a surface-embedded graph partitions the surface $\mathcal{M}$ into patches $p_i \in \mathcal{P}$. In QGP, it is constructed with nodes for every singular point of a seamless input parametrization $\mathcal{F}$. If it does not contain *any* singular point, a (regular) node is placed arbitrarily to avoid special cases. The arcs are created by tracing particles (motorcycles) on $\mathcal{M}$ from these seed nodes along iso-lines into all incident parametric directions. When a motorcycle hits the trace of another, it stops and an additional node (a T-joint) is created at the corresponding location on $\mathcal{M}$. In the end, the trace segments between nodes form the motorcycle graph's arcs.

Since motorcycle traces follow parametric iso-lines, by construction the parametric angle between two consecutive arcs at a node can only be $\pi/2$ or π. Thus, the motorcycle graph forms a non-conforming quadrilateral mesh, i.e. a T-mesh. We call halfarcs pointing into $\pi/2$-angles *corners* and the others *flat.* Since patches cannot contain singularities in their interior by construction, in a boundary-aligned setting there are exactly four corner halfarcs per patch. Thus, the halfarcs h_i of each patch p_i can consistently be assigned a parametric direction $d(h_i) \in \{(1,0),(0,1),(-1,0),(0,-1)\}$, such that every flat halfarc is followed by a halfarc with the same direction, and every corner halfarc is followed by a halfarc with a direction rotated counter-clockwise by $\pi/2$.

4.1 Free-Boundary Motorcycle Graph

As discussed in Section 3.2, in the presence of free boundaries the graph's patches may have a variety of complex shapes, cf. Figure 3. The goal of this section is to modify the motorcycle graph construction such that all generated patches are rectangular or sub-rectangular (cf. Definition 3.2).

Our key observation is that all complex patches can be turned into simple, sub-rectangular patches by splitting them into smaller ones. This can be achieved elegantly as part of the graph construction itself, by introducing additional seed points for motorcycles, leading to additional traces that split patches. The selection of additional seed points is based on local extrema of the input parametrization $\mathcal{F}$'s parameter functions u and v. As it is a chart-based parametrization with transitions across cuts, in the following we always assume working in local chart coordinates, with transitions implicitly applied without explicit mention.

Definition 4.1 (Peak Point). A point on the boundary curve $\partial\mathcal{M}$ in parametric space where u or v is a strict local extremum is called a u-peak or v-peak, respectively. If it is non-strictly extremal, it is only considered a peak if the value is lower/higher at least on one side; this avoids entire ranges of $\partial\mathcal{M}$ where u or v happen to be constant to be considered peak – only their end points.

Note that if the boundary is non-smooth (as in our piecewise linear mesh setting) a point can be both a u-peak and a v-peak.

We call a peak point *convex* or *concave* depending on whether the parametric image of the surface $\mathcal{M}$ is locally convex or concave at that point.

Definition 4.2 (Free-Boundary Motorcycle Graph). The free boundary motorcycle graph is defined and generated just as the (iso-line-based) motorcycle graph [Campen et al. 2015; Eppstein et al. 2008], with singular and feature points as seed points, except that additional seed points are taken into account to spawn motorcycles: the concave peak points. A u-peak spawns a motorcycle in $\pm u$-direction, a v-peak in $\pm v$-direction.

The following proposition asserts that this modification is sufficient to guarantee sub-rectangular patches. In fact, only 13 types of

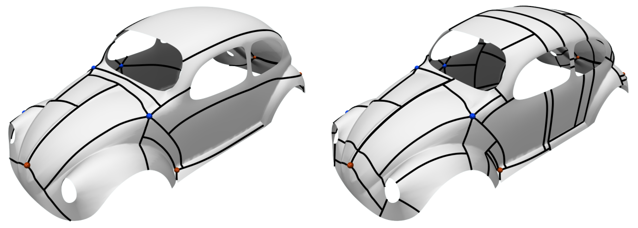


Fig. 6. Left: standard motorcycle graph. Right: our extended free-boundary motorcycle graph, with all patches being sub-rectangular. Notice that on the left, by contrast, patches are topologically and geometrically more complex.

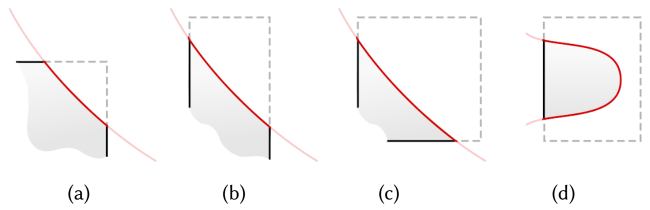


Fig. 7. Virtual arcs (dashed) are added along each red segment to complete sub-rectangular patches that are cut by the surface boundary (red). Four local situations, with $n \in \{1, 2, 3, 4\}$ cut off corners, can occur; they are treated by adding $n + 1$ virtual arcs and n virtual nodes.

patches (up to symmetry), as depicted in Figure 5, can occur in the free-boundary motorcycle graph. Key to this result is the observation that, due to tracing from concave peak points, no single patch can contain a piece of surface boundary bending outwards by more than $\pi/2$ parametrically.

Proposition 4.1 (Sub-Rectangular Patches Only). *Each patch of the free-boundary motorcycle graph is sub-rectangular.*

A proof is given in Appendix A.1. In Figure 6 the standard motorcycle graph and our free-boundary motorcycle graph are shown side-by-side on an example surface.

4.2 Virtual Patch Completion

In order to avoid the need for special case treatment for the various types of incomplete patches, we virtually complete them through the integration of *virtual nodes* and *virtual arcs*. Due to the property of all patches being sub-rectangular in the free-boundary motorcycle graph, each patch, regardless of type, can always be completed to a virtual rectangle. Note that the following construction is purely abstract, combinatorial; we do not associate geometry to virtual arcs.

Considering the different types of sub-rectangular patches depicted in Figure 5 one observes that a red segment (i.e. part of the surface boundary $\partial\mathcal{M}$) can cut off a piece of the underlying rectangle that contains either one, two, three, or four adjacent corners. For each red segment, this number $n \in \{1, 2, 3, 4\}$ can be deduced from the relative parametric orientation of the adjacent black segments, corresponding to arcs a and b (not necessarily distinct) that were traced from or into the surface boundary $\partial\mathcal{M}$ at neighboring locations.

We insert a sequence of $n + 1$ virtual arcs (with n intermediate virtual nodes) between a and b. The first and last one are assigned the parametric direction of a and b, respectively, and the direction of each other virtual arc in between is defined by rotating the previous virtual arc's direction by $\pi/2$. Figure 7 shows examples for all four relative orientations of a and b.

As this construction can be performed for every red segment of a sub-rectangular patch, this result follows immediately:

Proposition 4.2 ((Virtually) Rectangular Patches Only). *The free-boundary motorcycle graph extended by virtual arcs for each red segment consists of rectangular patches only.*

This virtual extension thus allows for streamlined processing in the subsequent steps in our pipeline, where we can rely on a partition graph structure consisting of (virtual) rectangles only. Figure 8 illustrates the result of our free-boundary motorcycle graph extended by virtual arcs on the surfaces from Figure 3.

5 QUANTIZATION

For the final global parametrization to be an integer grid map (as required in particular for quad mesh generation), certain surface points need to be mapped to integer (u, v)-coordinates $\in \mathbb{Z}^2$ in the plane. These points are 1) the singularities (corresponding to extraordinary vertices in the implied quad mesh) and 2) feature points (any kind of user-prescribed surface points where ultimately a quad mesh vertex shall be located). We collectively refer to these as *critical points* in the following.

Furthermore, one may want to require certain curves on the surface, for instance sharp feature curves or boundary curves, to be matched by quad mesh edges. To this end one of the two (u, v)-coordinates needs to be constant and integer along the curve. Such curves we refer to as *critical curves*.

Quantization in this context refers to the process of deciding *which* integers each critical entity shall be mapped to. As discussed in detail before [Bommes et al. 2013a; Campen et al. 2015], the challenge lies in making this decision in such a way that a corresponding non-degenerate integer grid map even exists. In particular, the quantization must ensure the parametric separation of critical entities. Otherwise, the map is forced to degenerate or be non-injective. Concretely, in the subsequent process of integer grid map construction, described in Section 6, any two non-separated nodes will (and have to) be collapsed onto a common point on the surface, violating the intention of two specific (distinct) surface points receiving integer parameters in the final integer grid map. A quantization is thus called *valid* if it separates all critical entities.

Campen et al. [2015] presented the QGP algorithm to efficiently compute such a valid quantization for the boundary-aligned case in the form $q : \mathcal{A} \rightarrow \mathbb{N}$, i.e. by assigning a non-negative integer

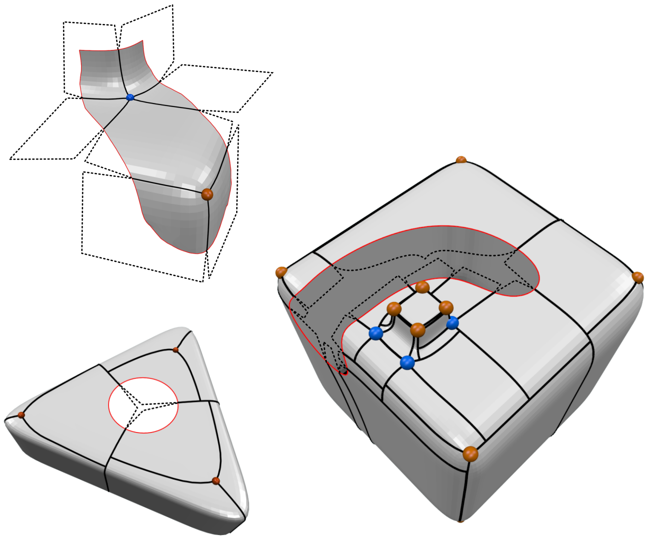


Fig. 8. The examples of Figure 3, however, with additional arcs due to peak point tracing and extended by virtual arcs (dotted).

length $q(a_i)$ to each arc a_i of the T-mesh $\mathcal{T}$ formed by the motorcycle graph. These integer values represent relative parametric differences rather than absolute parameters for the critical entities. In a first stage, all arcs a_i are assigned, in a consistent (rectangularity preserving) manner, a strictly positive integer length. This trivially implies separation of all critical entities, thus validity. In a second stage, these integer arc lengths are iteratively updated in a consistency-preserving manner so as to match the actual input parametric arc lengths as closely as possible. To this end, in this stage, zero arc lengths may be preferable. For each tentative update thus a *separation-preservation test* is performed to ensure the updated quantization remains valid.

Thanks to our constructions described in Section 4, yielding a purely rectangular T-mesh also in the free-boundary setting, we are able to adapt the integer assignment technique of QGP. This requires two adjustments which we discuss in the following: an assignment of suitable target lengths to virtual arcs (Section 5.1), and a modification and generalization of the separation test (Section 5.2), as it natively would yield false negatives and false positives in the free-boundary setting, causing invalid or low-quality quantizations.

5.1 Virtual Arc Target Length

The quantization minimizes the differences between the assigned integer length q and the target length of an arc. For a regular arc the target length is defined as its parametric length in the input parametrization $\mathcal{F}$. Since virtual arcs are abstract, their target length needs to be defined differently.

We exploit the sub-rectangularity of all patches, and define target lengths for virtual arcs based on their length in the parametric rectangular completion. In this way, no sizing bias is introduced by the virtual patch extensions. Note that the rectangular completion is fully determined only in the case depicted in Figure 7a. In the other cases there are degrees of freedom. We consider as natural completion the smallest of all completing rectangles. In Figure 7 slightly larger rectangles are depicted for these cases for clarity, as the length of some of the arcs can be zero in the natural completion, depending on the shape of the red segments.

5.2 Separation Test

By construction of the motorcycle graph, each critical point coincides with a node of the T-mesh, and each critical curve coincides with an arc. We refer to such nodes and arcs as critical in the following.

If $q(a) > 0$ for each arc a, critical nodes and arcs are trivially separated with respect to the assigned integer parameters. To enable quantizations of high quality (not every single patch of the T-mesh should imply one or more quads in the integer grid map), however, also $q(a) = 0$ is permitted; in this case separation is not trivial but must be checked for explicitly.

Two nodes are not separated iff there exists a weakly-u- or weakly-v-monotone arc path of (assigned) length $(0, 0)$ between them, as discussed by [Campen et al. 2015]. This property – that is due to the parametrically convex (rectangular) nature of the surface partition by the T-mesh – enables a relatively simple and efficient separation test using a local graph search in $\mathcal{T}$, starting from critical nodes. Essentially, one needs to search only along those arcs that keep one of the two dimensions of the accumulated integer (u, v)-distance from the start node at zero, and which do not unnecessarily bring the other further away from zero. If any critical node or any point on a critical arc is reached with an accumulated distance of $(0, 0)$, non-separation is reported (and the tentative quantization update that caused this is rejected). If no such node or point is reached from any critical node, separation is certified.

As due to our modifications the T-mesh has only (virtually) rectangular patches in the free-boundary setting as well, thus enjoys the same key property, the same principle can be applied.

A challenge, however, is posed by the potential non-alignedness of (parts of) the surface boundary in our case. One generally needs to ensure that

- critical entities are separated from the surface boundary,
- different parts of the surface boundary are separated mutually,
- entire boundary loops do not parametrically degenerate to a point, i.e. each boundary loop contains two points that are separated.

If the entire surface boundary is an aligned boundary, all of the above is easily ensured simply by treating the boundary curves as critical curves, thus the arcs along the boundary as critical arcs, and boundary corners as critical nodes – as done in QGP. In our more general setting, a different treatment is required at non-aligned boundaries as there are no arcs directly aligned with the boundary.

5.2.1 Free-Boundary Separation. First of all, we treat the virtual arcs as critical. Intuitively, while we have illustrated the abstract virtual arcs as lying outside the surface in Figure 8, we are free to (preliminarily) realize them directly along the boundary curve (cf. Section 6.1.1). They hence serve as a suitable proxy for our purpose of separation testing with respect to the boundary.

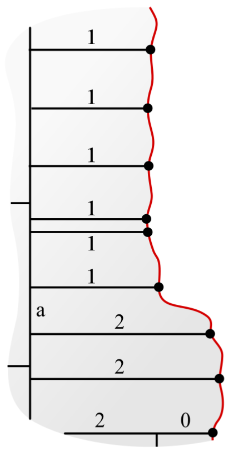
Unfortunately, this very simple modification is insufficient. The separation test described above, based on path searching in the motorcycle graph starting from critical nodes, tests for separation among critical nodes as well as between critical nodes and critical arcs – not among critical arcs. Hence, two parts of the boundary could be non-separated in a given quantization without the separation test noticing this.

One might expect other critical arcs (feature arcs, aligned-boundary arcs) to have the same issue. However, (sequences of) feature or boundary arcs have incident critical nodes (feature or boundary corner nodes) in [Campen et al. 2015]. In this case, if two arcs are non-separated, so is one of their incident nodes (due to parametric straightness of the arcs) – which is covered by the separation test.

We could easily solve our issue at non-aligned boundaries in an analogous manner: treat all boundary nodes, i.e. nodes incident to virtual arcs, as critical. This leads to the following definition:

Definition 5.1 (Strong Separation). A quantization is called *strongly separating* if it passes the separation test with boundary nodes and virtual arcs considered critical entities.

This simple solution, however, is unnecessarily strict. The strong separation test is conservative – in the sense that it precludes certain quantizations (relevant for high quality results) that admit valid integer grid maps. The issue is as follows: boundary nodes which are not critical per se (i.e. not prescribed feature nodes) do not represent specific isolated points on the surface which are to be parametrized in a specific (integer) way. Instead, they are part of a curve (the surface boundary) and are free to move (to be relocated, cf. Section 6.1) *along* this curve. Thus separation *along* the boundary is not required – but inherently imposed by the notion of strong separation. This is illustrated in the inset: for strong separation, of the vertical arcs, only arc *a* may be quantized to zero; seven of the eight patches depicted need to be quantized to a height of at least 1 each (i.e. a total height of 7) to pairwise separate the depicted boundary nodes – no matter how parametrically small the depicted surface region and how beneficial a zero-assignment thus would be.



We thus do not consider boundary nodes critical (unless they are prescribed feature nodes), but we still add them to the set of starting sources for the separation test's search process. If a search that started from a non-critical boundary node source, however, reaches a point on the boundary (a boundary node or virtual arc) with distance $(0, 0)$, this is *not* to be considered a non-separated configuration *if* there is one path of zero arcs along the surface boundary between this point and the source. In the above inset (where virtual arcs are not depicted), all patches could thus be quantized to zero height (if desired) as long as their virtual arcs are quantized to zero as well.

Note that we must specifically require *one* path of zero-arcs along the boundary. If there are two, an entire boundary loop is formed by zero-arcs. This would imply parametric degeneration of an entire surface boundary component to a point, which must be prevented.

Implementation Note. In QGP it suffices to start the separation test's graph search in forward direction only. This is because all critical points have traces in all parametric directions. Here we have additional source nodes (boundary nodes) for which this is not the case in general. One thus needs to include search paths whose first edge is not in forward direction, but pointing sideways, left or right.

6 ROBUST RE-PARAMETRIZATION

A given valid quantization implies all integer degrees of freedom of an integer grid map, i.e. the positions of the singularities and the transition functions across seams (between parameter charts). In order to compute the final integral parametrization Campen et al. [2015] suggest solving an accordingly constrained optimization problem. Due to non-convexity, even though it is guaranteed that a non-degenerate locally injective (flip-free) parametrization adhering to the prescribed integers exists for the input surface, it is not guaranteed that the solver will find such a solution. In fact, it may even be impossible to find a solution for the given triangulation without refinement.

In order to guarantee obtaining the desired integer grid map, we therefore instead make use of a different approach. We exploit the surface partition into disk-topology patches described by the embedded T-mesh, and construct an initial global integer grid map out of individual per-patch parametrizations (Section 6.2). This is a common approach, used before for conforming [Dong et al. 2006; Khodakovsky et al. 2003] as well as non-conforming quadrilateral partitions [Myles et al. 2014]. This initial map can then serve as valid starting point for further (local injectivity preserving and integer constrained) optimization.

However, this strategy only works if for each T-mesh face, a bijective map between their surface patch and their desired parameter rectangle exists. To this end, if a patch has been quantized to a parametric rectangle of area zero, the patch needs to have zero area on the surface as well – otherwise, the map will necessarily be non-injective. Therefore, we are going to show in Section 6.1 how all patches prescribed to a parametric area of zero by the quantization can be re-embedded onto zero area regions on the input surface.

6.1 T-Mesh Re-Embedding

By construction, an initial embedding of regular arcs onto the input surface is given by the traced motorcycle paths. The fact that with this embedding triangles can span more than one patch complicates the computation of individual patch parametrizations that are consistent with their neighbors. One approach to resolve this is the integration of the arcs into the input mesh by splitting elements [Myles et al. 2014]. Downsides of this approach are the increase in mesh complexity and the potentially bad triangle shapes.

We therefore prefer a method that modifies the input mesh only when necessary. Instead of arc integration through mesh splitting, we propose to snap all nodes and arcs onto nearby vertices and edges, respectively (cf. Figure 9a→b). Only if there are not enough vertices or edges is the mesh split to admit a one-to-one mapping between T-mesh and triangle mesh elements.

6.1.1 Virtual Arc Embedding. Virtual nodes, for which so far no embedding existed, are distributed onto boundary vertices between the two adjacent boundary nodes, and virtual arcs are embedded onto the boundary edges in between (cf. Figure 9a→b). Intuitively, this corresponds to effectively stretching the sub-rectangular patch onto its completing rectangle in parameter space. This embedding is only preliminary; it simplifies subsequent algorithmic steps. During final optimization (cf. Section 6.2) this sub-optimal initial embedding is optimized, as illustrated in Figure 10.

While all arcs being embedded onto edges of the triangle mesh simplifies the computation of the individual patch parametrizations one problem may still remain. The parametric domain of a patch as defined by the quantization may be degenerate, i.e. the patch may need to be mapped onto a parametric line or point. If the patch itself is not also embedded as a line or point on the triangle mesh, respectively, the map is bound to degenerate. We therefore further adjust the T-mesh embedding such that all these *zero-patches* are re-embedded onto lines or points. We make use of the following definitions.

Definition 6.1 (Border Arc/Node). An arc or a node (virtual or non-virtual) is *border* if it is embedded in the surface boundary.

Definition 6.2 (Collapsible Arc). A zero-arc a is *collapsible* in direction from its incident node n_0 to its incident node n_1 if n_0 is non-critical, and either a is border or n_0 is non-border.

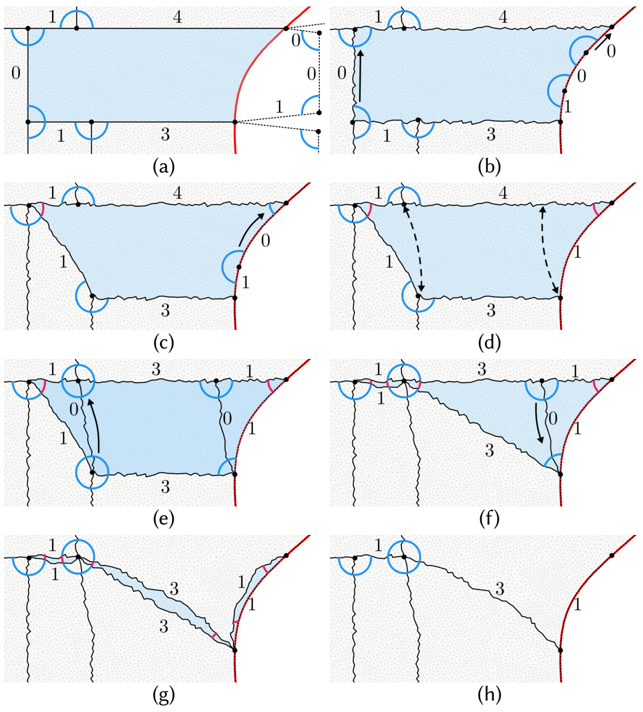


Fig. 9. Re-embedding of the Free-Boundary Motorcycle Graph. Numbers correspond to quantized arc lengths. Blue circular arcs indicate patch corners, red circular arcs double corners (two corners collapsed). (a): The initial embedding as traced during construction. (b): Nodes and arcs are snapped onto vertices and edges, virtual nodes and arcs are mapped onto the boundary. (c)-(h): The zero-patch (blue) is re-embedded onto a line via iterative zero-arc collapses (c,e,f,g), arc insertions (d,e) and simple zero-patch collapses (g,h).

The re-embedding is performed by simply applying the following three local operators until none can be applied anymore (cf. Figure 9 for an example).

(1) *Zero-arc collapse.* Let a be an arc collapsible from n_0 to n_1. Its collapse changes the T-mesh embedding in the following way (cf. Figure 9b→c): n_0 is embedded onto n_1, pulling its incident arcs with it, i.e. their embedding path is adjusted such that they connect to n_0 at its new position. Arc a is embedded onto a single point (coincident with the nodes n_0 and n_1). If n_0 now lies at a critical point, it is subsequently considered critical. Notice that the restriction to collapsible arcs ensures that critical nodes are not moved, and that border nodes are only moved along the border.

(2) *Non-simple zero-patch split.* A zero-patch is non-simple if more than two non-zero arcs are involved, i.e. if there are flat arcs, corresponding to T-joints along the patch's non-zero sides. The T-joints are extended through the patch to the corresponding point on the opposite side by inserting a new zero-arc (cf. Figure 9d→e). This means, if there is a node on the opposite side which has the same quantized distance to either end of the zero-patch, the new arc is connected to this node; otherwise a new node is inserted by splitting an arc at the corresponding location (and marked as critical, if the split arc is critical). The inserted zero-arcs can then be collapsed by operator (1). Note that this operation splits a non-simple zero-patch into several simple zero-patches.

(3) *Simple zero-patch collapse.* A zero-patch is simple if exactly two non-zero arcs are involved. A simple zero-patch is easily collapsed (after its zero-arcs have been collapsed) by replacing the embedding of one non-zero arc with the embedding of the other one (cf. Figure 9g→h). Note that a zero-patch without any non-zero arc (i.e. one that is supposed to be embedded onto a single point rather than a curve) is already handled by the zero-arc collapse.

Operator Implementation. In operators (1) and (2) edge paths to (re-)embed arcs are determined simply using Dijkstra's shortest path algorithm between the respective two vertices on the triangle mesh. The graph search is restricted to not intersect (cross or touch) other arcs in order to preserve the topology of the embedded T-mesh. While this restriction may preclude the existence of any edge path between two vertices to be connected (if the triangulation of the region between the other arcs is not 3-connected), this is easily resolved by refinement with a few edge splits.

If a border arc is collapsed from border node n_0 to border node n_1, the other border arc b incident to n_0 is re-embedded onto the joint edge paths of b and a, such that the surface boundary remains covered by arcs. For the same reason, when a simple zero-patch is collapsed where one of the two non-zero arcs is border, we change the embedding of the non-border arc. With this strategy it is ensured that the re-embedded T-mesh still covers the entire surface: Since border nodes are only collapsed *along* the boundary (not into the surface interior), the T-mesh boundary remains coincident with the surface (i.e. triangle mesh) boundary.

Proposition 6.1. *Given a T-mesh with a quantization satisfying the free-boundary separation test (Section 5.2.1), it can be re-embedded such that each zero-arc is embedded onto a point.*

A proof is given in Appendix A.3. As, in contrast to the zero-arc collapse, there is no restriction for the applicability of the zero-patch operators (other than the zero-arcs being collapsed already), it follows readily:

Corollary 6.3. *All zero-patches of a quantization satisfying the free-boundary separation test (Section 5.2.1) can be collapsed.*

We are thus able to obtain a re-embedding of the T-mesh such that 1) the entire surface remains covered, 2) all zero-arcs are embedded onto points, and 3) all zero-patches are embedded onto curves (or points). The integer grid map initialization discussed in the following can thus be applied.

A conceptually very similar re-embedding strategy was employed by [Myles et al. 2014]. Key differences of our approach are:

- We refine the underlying triangle mesh only if necessary, instead of generally.
- Discrete Dijkstra's algorithm for arc re-routing, instead of continuous path tracing.
- Local operators concerning individual zero-patches, instead of entire zero-chains.
- Support for non-aligned boundaries, instead of requiring complete alignment.

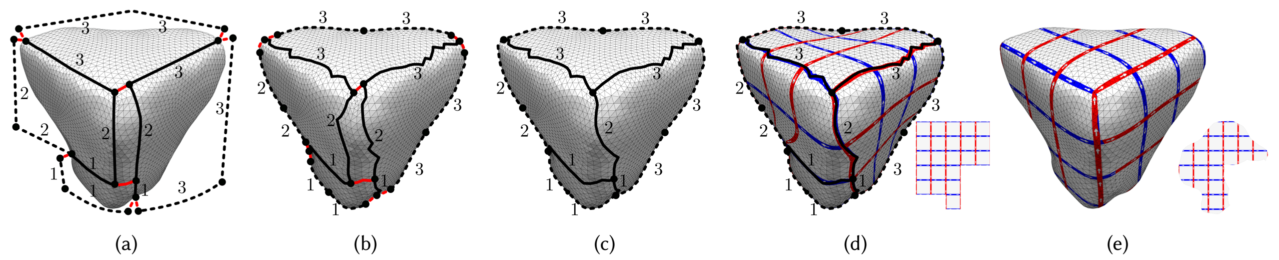


Fig. 10. Re-parametrization overview. (a) T-mesh arcs (non-virtual and virtual) with their assigned quantized lengths; red indicates zero lengths. (b) T-mesh embedded onto edges of the input mesh. (c) T-mesh re-embedded by collapsing zero-arcs (red). (d) Each patch mapped onto axis-aligned rectangular domains for an initial, seamless, guaranteed locally-injective, integer-grid parametrization. (e) During final optimization patches can transform into arbitrary shapes in parameter space, reducing the distortion of the initial embedding.

6.2 Re-Parametrization

The desired integer grid map can now be computed from the re-embedded T-mesh with assigned arc lengths: A bijective flip-free parametrization for an individual patch is easily computed as a convex combination map [Tutte 1963], constraining its boundary vertices onto the parametric axis-aligned rectangle prescribed by the quantization. By fixing one of the patch's nodes to the origin (or any integer position), all other nodes get mapped onto integer positions, ensuring that all singular vertices, feature vertices, and feature lines are mapped onto integer points and lines, respectively.

The union of all patch parametrizations then forms an integer grid map since the implied transition functions across triangle edges are grid automorphisms: at triangle mesh edges on the boundary between two patches, both edge images are on an integer grid line (thus only rotations about multiples of $\pi/2$ are possibly involved) and have the same length (due to the corresponding arc being mapped onto the same assigned quantized length in both patches).

This initial parametrization can then be optimized with one of the various strategies proposed in recent years, e.g. [Rabinovich et al. 2017; Shtengel et al. 2017]. Note that during this optimization most of the vertices which were assigned to fixed positions for the per-patch parametrizations can now freely move in the parametrization domain. In particular this means that all non-aligned free boundaries that were preliminarily mapped onto rectangular shapes in the individual patch parametrizations are now free to assume any shape beneficial to yield low distortion energy (cf. Figure 10).

Table 1. Statistics. From left to right: Model and figure number, number of triangles, number of singular vertices, number of arcs in the T-Mesh, timings for surface partitioning (t_p), quantization (t_q), T-mesh re-embedding (t_r), and final parametrization optimization (t_o). All times in seconds.

Mesh	#T	#S	#A	t_p	t_q	t_r	t_o
Beetle (11)	38726	9	299	0.0318	0.0433	0.0624	7.94
Hand (11)	3000	36	344	0.0052	0.0306	0.0291	1.83
Face (11)	62467	24	432	0.0474	0.0784	0.1491	15.9
Simple (11)	1444	4	99	0.0024	0.0063	0.0072	0.11
Train Station (12 left)	2546	8	243	0.0050	0.0335	0.0109	0.94
Train Station (12 center)	2546	8	245	0.0049	0.0383	0.0152	1.10
Train Station (12 right)	2546	8	239	0.0048	0.0253	0.0233	1.14
Flat Roof (13 left)	40000	1	24	0.0056	0.0005	0.0005	8.13
Flat Roof (13 center)	40000	8	88	0.0102	0.0019	0.0389	13.0
Dome Roof (13 right)	10000	5	96	0.0086	0.0023	0.0098	1.35
Wing Profile (14)	14463	0	25	0.0021	0.0006	0.0005	1.76
Six (15)	3109	4	173	0.0038	0.0119	0.0135	0.62
Exclamation (16)	1976	0	52	0.0019	0.0033	0.0040	0.22

7 RESULTS

We present results of our algorithm for a variety of different use cases. Input parametrizations for our method were obtained by minimizing the (continuous) parametrization objective of [Bommes et al. 2009] with local injectivity constraints [Bommes et al. 2013a]. The objective favors directional alignment of the parametrization to a given cross field, which in turn is obtained as the smoothest cross field interpolating salient principal curvature direction as computed by [Campen et al. 2016]. Final optimization of the output, after locally-injective initialization as described in Section 6.2, was performed using a Newton solver with the objective of minimizing the difference between output integer grid map and input parametrization $\mathcal{F}$, while preserving local injectivity, seamlessness, and assigned integer values.

Runtimes. In Table 1 we give an overview of the times taken by the stages of our algorithm, measured on a desktop machine using a single-threaded implementation. Note that the overall runtime is dominated by the final off-the-shelf distortion minimization, while the generation of the initial quantized trimmed integer grid map by our three central stages – partitioning, quantization, and re-embedding – typically makes up less than 5% of the total runtime.

General Quad Meshing. Figures 1 and 11 show exemplary resulting quad meshes of our algorithm with and without boundary alignment on a variety of different input meshes. We also report histograms of the quad angle deviation from $\pi/2$ at inner vertices and of the quad edge lengths. Note how for meshes with free boundaries the angle stays closer to $\pi/2$ and the edge length distribution has lower variance. In some cases (e.g. the hand model) free boundaries do not give a significant benefit, but in general free boundaries should not lead to worse quality than aligned boundaries, as free boundaries are free to align if this is beneficial for lower distortion.

In Figure 12 we compare results obtained for varying user-specified target edge lengths for a model with a strongly curved boundary. In such cases the effect of free boundaries is particularly significant.

Architectural Design. In Figure 13 trimmed quad meshes computed with our method for two different structures, a flat and a

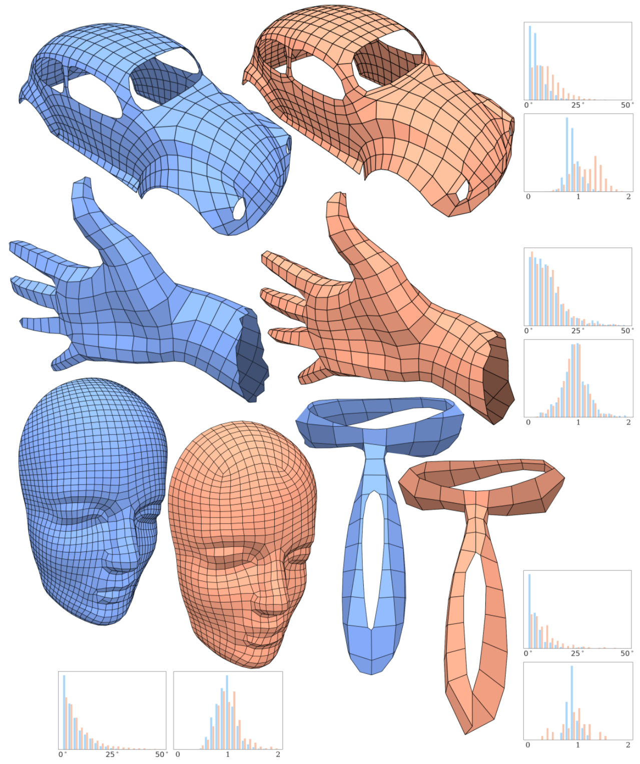


Fig. 11. Comparison of quad meshes extracted from integer grid maps with free (blue) and aligned (orange) boundaries. The upper histogram shows the absolute angle deviation from $\pi/2$, the lower one shows a histogram of the quad edge lengths normalized at 1 for the requested target edge length.

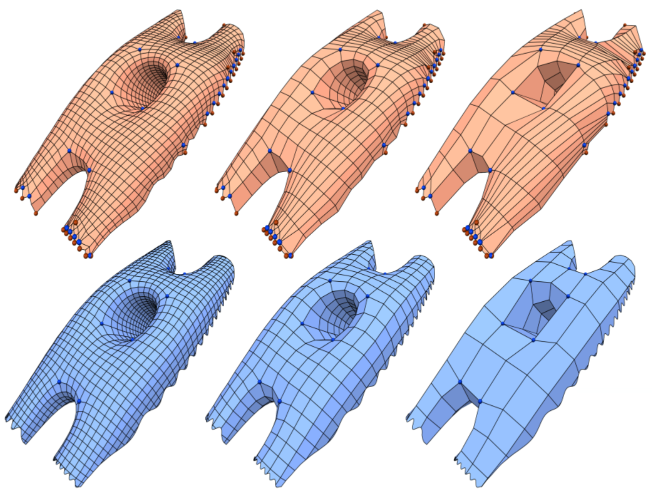


Fig. 12. Comparison of quad meshes with free (blue) and aligned boundaries, for a range of different user-prescribed target edge Notice how, while a principal curvature direction driven guiding f used, the forced boundary alignment in the top row leads to an ir number of singularities (red and blue spheres) which in turn prevent solution and cause badly oriented elements.

domed roof, are shown. The free-boundary setting enables accurate alignment of mesh edges to the principal directions of stress. The

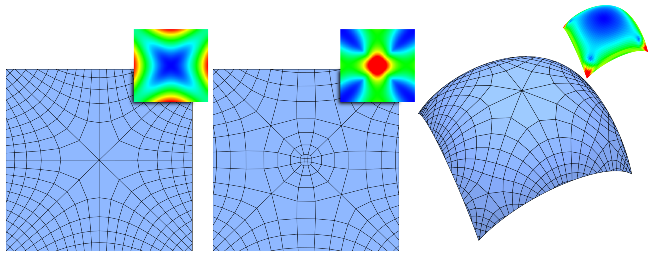


Fig. 13. Three examples of quad meshes aligned to principal stress directions on simple flat and domed roofs. These roofs are supported on their four corners and are subject to gravitational forces due to their own weight. In the central example an additional force is applied at the center. The insets show the magnitude of the maximal stress vector color coded from minimal (blue) to maximal (red) for illustration.

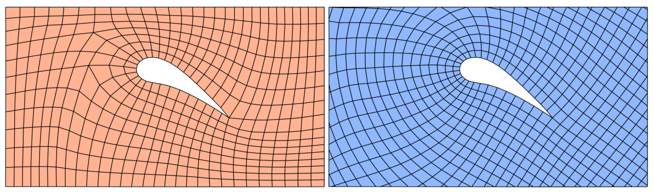


Fig. 14. Meshes for wing profiles. Left with alignment to all boundaries, right only aligned to the wing.

resulting quad meshes could serve as a basis for the construction of efficient structures (e.g. consisting of beams or folds) which benefit from such alignment.

Finite Element Simulation. Many numeric simulation methods benefit from well structured quad meshes in order to yield quick and accurate results. Typically singularities are not desired as they require special treatment or lead to less accuracy. Meshing the simulation domain around the wing profile in Figure 14 without boundary alignment allows for a resulting quad mesh with fewer singularities.

Tiled Texturing. Covering surfaces with repetitive structures or tiled textures [Akleman et al. 2005] essentially requires a parametrization that induces a grid structure, i.e. an integer grid map. Depending on the concrete scenario, singular points may or may not be acceptable or required, and boundary-alignment may be beneficial or uncalled for. Figure 15 shows an example of a symmetric structural pattern imposed on a surface by means of a free-boundary integer grid map.

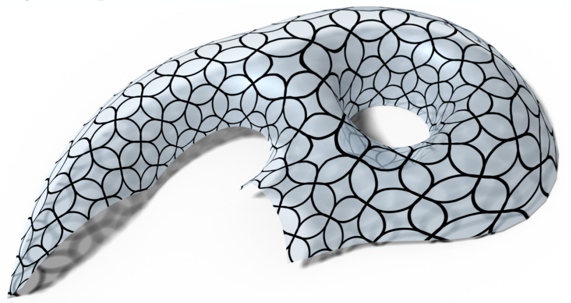


Fig. 15. Example of a semi-regular structure created on a surface by means of a free-boundary integer grid map.

Trimmed Parametric Surfaces. When parametrically representing surfaces (e.g. using NURBS), gridded surface parametrizations are naturally involved. For trimmed variants thereof, these correspond to free-boundary parametrizations. Figure 16 shows a very simple example obtained using our method for illustration; note that in the depicted free-boundary case there are no singularities (i.e. an integer grid map is unnecessarily general for this case), but for more complex surfaces and more general representations (e.g. NURCCS or NURSSes [Sederberg et al. 2003]) singular points can as well be of interest.

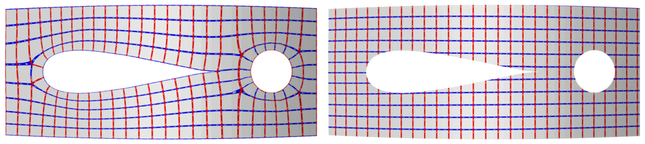
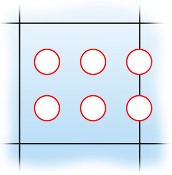


Fig. 16. Left: boundary-aligned integer grid map. Right: free-boundary integer grid map. Such parametrizations are relevant for constructing representations using untrimmed or trimmed parametric surfaces, respectively.

8 LIMITATIONS & FUTURE WORK

As an extension of the quantized global parametrization algorithm [Campen et al. 2015] our algorithm shares many of its properties, including some of its limitations. Most importantly, our algorithm, like QGP, employs a geometrically guided but greedy quantization approach for efficiency. Thus, the results are not guaranteed to be globally optimal.

We opted to tailor our method to the setting of topologically non-complex (simply-connected) trimmed quads such that our T-mesh construction ensures all patches to be sub-rectangular and only of the types illustrated in Figure 5. We believe this limited structural element variety is reasonable and even of value in many use cases. If however, for some application one would like to support non-simply-connected trimmed quads in the output mesh (i.e. quad elements with (multiple) internal holes, as shown in the inset), this would require some non-trivial modifications since our approach, as it is, always refines the motorcycle graph such that no boundary loops remain fully contained within one patch. In situations with holes that are smaller than the target quad size, one could fill such small holes in the input mesh, compute the integer grid map, and finally remove the hole fillings, thereby re-introducing the holes – even inside single quads (cf. Section 3.2.1).



Like previous work, e.g. [Bommes et al. 2013a, 2009; Campen et al. 2015; Kälberer et al. 2007], our method makes use of a continuous (i.e. non-quantized) seamless parametrization as input; the quantization is determined based on it. Computing such a parametrization is a hard problem, in particular when additional constraints (concerning singularity positions and indices, iso-line connectivity, etc.) are to be respected. In particular, when formulated as an optimization problem, non-convexity poses robustness challenges.

The work of [Myles et al. 2014], based on cross field tracing, provides a particularly robust method to construct such a continuous seamless parametrization, with only mild limitations concerning the exact preservation of prescribed singularities. It would thus be one natural choice for the generation of input parametrizations for our method in a reliable quad mesh generation pipeline – if it supported non-aligned boundaries.

Assuming that method could be generalized to the free-boundary setting, its combination with our method would be a pretty redundant approach, though: both methods make use of a number of similar tools and data structures, e.g. (different variants of) the motorcycle graph, that would be recomputed and reapplied. Investigation of how the two methods could best be married to obtain an efficient integrated approach (which furthermore supports the free-boundary scenario) is thus an interesting avenue for future work. Essentially, one can imagine our method being adapted to operate on a (much easier to compute) input cross field rather than a seamless parametrization. Our free-boundary extensions of the motorcycle graph construction and the T-mesh re-embedding would have to be adjusted to this setting. To which extent the observations and results of [Myles et al. 2014] can be adopted in this free-boundary context will have to be evaluated.

ACKNOWLEDGMENTS

We would like to thank Jan Möbius for creating and maintaining the geometry processing framework OpenFlipper [Möbius and Kobbelt 2012], Juan Musto for providing the simulation results for Figure 13, Amir Vaxman, the Stanford shape repository and AIM@Shape for providing models, as well as the reviewers for their helpful feedback. This work was supported by the European Research Council under the European Union's Seventh Framework Programme, ERC grant agreement 340884 (ACROSS) as well as the Gottfried-Wilhelm-Leibniz Programme of the Deutsche Forschungsgemeinschaft DFG.

REFERENCES

Ergun Akleman, Avneet Kaur, and Lori Green. 2005. Tiled Textures. *ISAMA'2008* (2005).
Cecil G. Armstrong, Harold J. Fogg, Christopher M. Tierney, and Trevor T. Robinson. 2015. Common themes in multi-block structured quad/hex mesh generation. *Procedia Engineering* 124 (2015), 70–82.
David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-Grid Maps for Reliable Quad Meshing. *ACM Transactions on Graphics* 32, 4 (2013), 98:1–98:12.
David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Cláudio Silva, Marco Tarini, and Denis Zorin. 2013b. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum* 32, 6 (2013), 51–76.
David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-Integer Quadrangulation. *ACM Transactions on Graphics* 28, 3 (2009), 77:1–77:10.
Marcel Campen. 2017. Partitioning Surfaces Into Quadrilateral Patches: A Survey. In *Computer Graphics Forum*, Vol. 36. 567–588.
Marcel Campen, David Bommes, and Leif Kobbelt. 2015. Quantized global parametrization. *ACM Transactions on Graphics* 34, 6 (2015).
Marcel Campen, Moritz Ibing, Hans-Christian Ebke, Denis Zorin, and Leif Kobbelt. 2016. Scale-Invariant Directional Alignment of Surface Parametrizations. *Computer Graphics Forum* 35, 5 (2016).
Marcel Campen and Leif Kobbelt. 2014. Quad Layout Embedding via Aligned Parameterization. *Computer Graphics Forum* 33, 8 (2014), 69–81.
Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum* 29, 5 (2010), 1525–1533.
Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. 2006. Spectral surface quadrangulation. *ACM Transactions on Graphics* 25, 3 (2006).
Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. 2013. QEx: Robust Quad Mesh Extraction. 32, 6 (2013), 168:1–168:10.
Hans-Christian Ebke, Marcel Campen, David Bommes, and Leif Kobbelt. 2014. Level-of-Detail Quad Meshing. *ACM Transactions on Graphics* 33, 6 (2014), 184:1–184:11.
Hans-Christian Ebke, Patrick Schmidt, Marcel Campen, and Leif Kobbelt. 2016. Interactively Controlled Quad Remeshing of High Resolution 3D Models. *ACM Trans. Graph.* 35, 6 (2016), 218:1–218:13.
David Eppstein and Jeff Erickson. 1999. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete &*

Computational Geometry 22, 4 (1999), 569–592.
David Eppstein, Michael T. Goodrich, Ethan Kim, and Rasmus Tamstorf. 2008. Motorcycle Graphs: Canonical Quad Mesh Partitioning. *Computer Graphics Forum* 27, 5 (2008), 1477–1486.
Xianzhong Fang, Hujun Bao, Yiying Tong, Mathieu Desbrun, and Jin Huang. 2018. Quadrangulation Through Morse-parameterization Hybridization. *ACM Trans. Graph.* 37, 4 (2018), 92:1–92:15.
Gerald E. Farin and Dianne Hansford. 2000. *The Essentials of CAGD.* A. K. Peters, Ltd., Natick, MA, USA.
Thomas-Peter Fries and Ted Belytschko. 2010. The extended/generalized finite element method: an overview of the method and its applications. *Internat. J. Numer. Methods Engrg.* 84, 3 (2010), 253–304.
Huanhuan Gu, Jean Gotman, and Jon P. Webb. 2011. Computed Basis Functions for Finite Element Analysis Based on Tomographic Data. *IEEE Transactions on Biomedical Engineering* 58 (2011), 2498–2505.
Erkan Gunpinar, Masaki Moriguchi, Hiromasa Suzuki, and Yutaka Ohtake. 2014. Feature-aware partitions from the motorcycle graph. *Computer-Aided Design* 47 (2014).
Jin Huang, Muyang Zhang, Jin Ma, Xinguo Liu, Leif Kobbelt, and Hujun Bao. 2008. Spectral quadrangulation with orientation and alignment control. In *ACM Transactions on Graphics (TOG)*, Vol. 27. 147.
Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J Guibas. 2018. QuadriFlow: A Scalable and Robust Method for Quadrangulation. In *Computer Graphics Forum*, Vol. 37. 147–160.
Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant field-aligned meshes. *ACM Transactions on Graphics* 34, 6 (2015), 189.
Felix Kälberer, Matthias Nieser, and Konrad Polthier. 2007. QuadCover – Surface Parameterization using Branched Coverings. *Computer Graphics Forum* 26, 3 (2007).
Zhang Kedi, Najafi Ahmad Raeisi, Jin Jian-Ming, and Geubelle Philippe H. 2015. An interface-enriched generalized finite element analysis for electromagnetic problems with non-conformal discretizations. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* 29, 2 (2015), 265–279.
Andrei Khodakovsky, Nathan Litke, and Peter Schröder. 2003. Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics* 22, 3 (2003).
Denis Kovacs, Ashish Myles, and Denis Zorin. 2011. Anisotropic quadrangulation. *Computer Aided Geometric Design* 28, 8 (2011), 449–462.
Ruotian Ling, Jin Huang, Bert Jüttler, Feng Sun, Hujun Bao, and Wenping Wang. 2014. Spectral quadrangulation with feature curve alignment and element size control. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 11.
Celong Liu, Wuyi Yu, Zhonggui Chen, and Xin Li. 2017. Distributed poly-square mapping for large-scale semi-structured quad mesh generation. *Computer-Aided Design* 90 (2017), 5–17.
Manish Mandad and Marcel Campen. 2019. Exact Constraint Satisfaction for Truly Seamless Parametrization. *Computer Graphics Forum* 38, 2 (2019).
Giorgio Marcias, Nico Pietroni, Daniele Panozzo, Enrico Puppo, and Olga Sorkine-Hornung. 2013. Animation-aware quadrangulation. In *Proc. Symposium on Geometry Processing*. 167–175.
Jan Möbius and Leif Kobbelt. 2012. OpenFlipper: An Open Source Geometry Processing and Rendering Framework. In *Curves and Surfaces*. Lecture Notes in Computer Science, Vol. 6920.
Ashish Myles, Nico Pietroni, Denis Kovacs, and Denis Zorin. 2010. Feature-aligned T-meshes. *ACM Transactions on Graphics* 29, 4 (2010), 117:1–117:11.
Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-aligned Global Parametrization. *ACM Transactions on Graphics* 33, 4 (2014).
Ashish Myles and Denis Zorin. 2012. Global parametrization by incremental flattening. *ACM Transactions on Graphics* 31, 4 (2012).
Moës Nicolas, Dolbow John, and Belytschko Ted. 1999. A finite element method for crack growth without remeshing. *Internat. J. Numer. Methods Engrg.* 46, 1 (1999).
Matthias Nieser, Ulrich Reitebuch, and Konrad Polthier. 2011. CubeCover – Parameterization of 3D Volumes. *Computer Graphics Forum* 30, 5 (2011), 1397–1406.
Steven J Owen. 1998. A survey of unstructured mesh generation technology.. In *IMR*.
Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. 2014. Frame fields: Anisotropic and non-orthogonal cross fields. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 134.
Nico Pietroni, Marco Tarini, Olga Sorkine, and Denis Zorin. 2011. Global parametrization of range image sets. In *ACM Transactions on Graphics (TOG)*, Vol. 30. 149.
Helmut Pottmann, Alexander Schiftner, Pengbo Bo, Heinz Schmiedhofer, Wenping Wang, Niccolo Baldassini, and Johannes Wallner. 2008. Freeform Surfaces from Single Curved Panels. *ACM Trans. Graph.* 27, 3 (2008), 76:1–76:10.
Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 4, Article 37a (2017).
Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic global parameterization. *ACM Transactions on Graphics* 25, 4 (2006), 1460–1485.
Nicolas Ray, Vincent Nivoliers, Sylvain Lefebvre, and Bruno Lévy. 2010. Invisible seams. In *Computer Graphics Forum*, Vol. 29. 1489–1496.
Faniry H. Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. 2015. Perfect Matching Quad Layouts for Manifold Meshes. *Computer Graphics Forum* 34, 5 (2015), 219–228.
Nico Schertler, Daniele Panozzo, Stefan Gumhold, and Marco Tarini. 2018. Generalized motorcycle graphs for imperfect quad-dominant meshes. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 155.
Nico Schertler, Marco Tarini, Wenzel Jakob, Misha Kazhdan, Stefan Gumhold, and Daniele Panozzo. 2017. Field-aligned online surface reconstruction. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 77.
Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad Nasri. 2003. T-splines and T-NURCCs. *ACM Transactions on Graphics* 22, 3 (2003), 477–484.
Kenji Shimada. 2006. Current trends and issues in automatic mesh generation. *Computer-Aided Design and Applications* 3, 6 (2006), 741–750.
Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Trans. Graph.* 36, 4, Article 38 (2017), 11 pages.
Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. 2006. Designing quadrangulations with discrete harmonic forms. In *Proc. SGP '06*. 201–210.
William T. Tutte. 1963. How to Draw a Graph. *Proceedings of the London Mathematical Society* s3-13, 1 (1963), 743–767.
Mirko Zadravec, Alexander Schiftner, and Johannes Wallner. 2010. Designing quad-dominant meshes with planar faces. In *Computer Graphics Forum*, Vol. 29. 1671–1679.
Muyang Zhang, Jin Huang, Xinguo Liu, and Hujun Bao. 2010. A wave-based anisotropic quadrangulation method. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 118.
Jiaran Zhou, Marcel Campen, Denis Zorin, Changhe Tu, and Claudio T. Silva. 2018. Quadrangulation of non-rigid objects using deformation metrics. *Computer Aided Geometric Design* 62 (2018), 3–15.

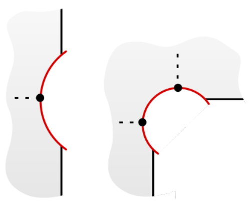
A APPENDIX

A.1 Sub-Rectangularity

On a patch p, let $d\theta/dx$ be the angular change of the parametric boundary tangent direction when traveling along the patch boundary ∂p. Our sign convention is such that $d\theta/dx > 0$ where p is locally convex (i.e. the boundary bends inwards with respect to the patch's interior), and $d\theta/dx < 0$ where p is locally concave. Note that $d\theta/dx$ is not defined at corners, where ∂p is non-smooth, but we will only argue in terms of (still meaningful) integrals $\int d\theta$ across corners.

The patch is bounded by arcs and parts of $\partial\mathcal{M}$. We define as *black* segment (cf. Figure 5) a maximal connected sub-curve of ∂p that runs along one or more arcs of the same parametric direction; the case of more than one such arc may occur due to T-joints. A *red* segment (cf. Figure 5) is defined as a maximal connected sub-curve of ∂p coinciding with a part of $\partial\mathcal{M}$. Note that $d\theta/dx = 0$ everywhere along each black segment, as they are parametrically straight.

Proof of Proposition 4.1: Along the boundary ∂p of each patch p, a black segment can directly follow a black segment, or exactly one red segment can be in between. A red segment cannot directly follow a red segment by definition as maximal connected component. Traveling from a point a on a black segment to a point b on a *directly* following black segment, we have $\int_a^b d\theta = \pi/2$ (by the way arcs intersect). Traveling from a point a on an black segment to a point b on an *indirectly* following black segment, we have $\int_a^b d\theta \geq \pi/2$; this is because this value necessarily is an integer multiple of $\pi/2$ (as arcs are aligned with iso-lines) and $\int_a^b d\theta = -k\pi/2$ for an integer $k \geq 0$ would imply that there are at least $k+1 > 0$ peak points along the intermediate red segment, as depicted in the inset for $k = 0$ and $k = 1$ – which would have spawned motorcycles, creating arcs, thus black segments, contradicting the premise that these points are in the interior of a red segment. Thus in any case, the



signed total rotation traveling from one black segment to the (not necessarily distinct) next is at least $\pi/2$, thus *positive*.

Let $b_1, \ldots, b_n$ be the $n \geq 1$ connected components (boundary loops) of ∂p. For each loop b_i we distinguish two cases:

- b_i contains a black segment: It is not possible that b_i consists solely of a single black segment; the corresponding arc(s) must stem from a seed point, which would have spawned orthogonal traces as well. Thus there is at least one more black or red segment. In both cases it follows that the total rotation $\Theta_i := \int_{b_i} d\theta$ is *positive*.
- b_i contains no black segment: This means b_i is a single red segment, so we have $\Theta_i > -\pi/2$, as otherwise a peak would necessarily be contained. As $d\theta/dx$ is defined relative to a seamless parametrization (with transitions involving rotations about multiples of $\pi/2$ only), it generally holds $\Theta_i = k\pi/2$ for some integer k. It follows that the total rotation Θ_i is *non-negative*.

Lemma A.1 (stated in Appendix A.2) asserts that $\sum_i \Theta_i \leq 0$ if there is more than one boundary loop, and $\sum_i \Theta_i < 0$ if there are more than two.

This leaves only three options:

(1) there is one purely red boundary loop
(2) there are two purely red boundary loops
(3) there is one boundary loop involving black segments

Cases 1 and 2 imply that the patch covers the entire surface (a topological disk or annulus, respectively, free of singular points) and the motorcycle graph is empty. As, however, at least one node is added as seed in any case (cf. Section 4), these cases cannot occur.

It follows that the only possible case is that of a single boundary loop b_1 containing black segments and, according to Lemma A.1, $\Theta_1 = 2\pi$. This implies that there are at most four black segments, one of each parametric direction (cf. Figure 5). Thus there exist axis-aligned rectangles (a unique one if there are four black segments) that match the black segments in the parametric image Ω of the patch; in the terms of Definition 3.2: $\partial_a\Omega \subseteq \partial R$. Furthermore, if there are red segments involved, these cannot fall back behind their adjacent black segments (they would contain peak points), thus the rectangle R is (or can be chosen) such that all red segments are contained in its interior, thus $\Omega \subseteq R$. Hence patch p is sub-rectangular by Definition 3.2. □

A.2 Total Rotation along Patch Boundary

Lemma A.1. *Let $b_1, \ldots, b_n$ be the $n \geq 1$ connected components (loops) of the boundary ∂p of a patch p, and $\Theta_i = \int_{b_i} d\theta$ the signed total rotation along boundary loop b_i (sign convention as in Section 4.1) in input parametrization $\mathcal{F}$. Then it holds $\sum_i \Theta_i = 2\pi(2 - n)$.*

Proof. Consider $n - 1$ mutually non-intersecting simple curves γ_i embedded in the patch such that γ_i connects boundary loops b_i and b_{i+1}. Let p' be the disk-topology patch obtained by cutting p along these curves. Remember that the input parametrization $\mathcal{F}$ is regular everywhere inside p' – because singular points are nodes, thus not inside patches. Like for any disk-topology region with regular parametrization, for p' it thus holds $\int_{\partial p'} d\theta = 2\pi$. By the structure of the boundary $\partial p'$, on the other hand, we also have $\int_{\partial p'} d\theta = \sum_i \Theta_i + 2\pi(n - 1)$. Note that terms involving integrals along γ_i cancel, as each such curve appears twice, in forward and backward direction, along $\partial p'$. The term $2\pi(n - 1)$ is due to the corners formed where γ curves and b loops meet. Assuming, w.l.o.g., the curves were chosen such that they intersect the boundary loops at parametrically smooth points, at each such intersection point they form a pair of parametric angles summing to π – and there are $2(n - 1)$ such points.

Combining these two equations for $\int_{\partial p'} d\theta$ yields the result. □

A.3 Re-Embedding of Zero-Arcs

Proof of Proposition 6.1: The only restriction for the applicability of the zero-arc collapse is that the arc needs to be collapsible. For an arbitrary quantization, it could thus be the case that no operator can be applied anymore while zero-arcs are still present. We show that this does not occur with the quantizations obtained using the method from Section 5, satisfying the free-boundary separation test (Section 5.2.1).

There are three types of non-collapsible arcs, depending on the types of incident nodes:

(1) critical—critical
(2) critical—border (if the arc is non-border)
(3) border—border (if the arc is non-border)

Case 1. Two critical nodes are separated already by the original separation test recapped in Section 5.2, due to (virtual) rectangularity of all patches also in our free-boundary motorcycle graph. Hence, even after collapsing other zero-arcs and zero-patches, no such zero-arc between the two critical points can occur.

Case 2. As we treat boundary arcs and virtual arcs as critical in our free-boundary separation test, critical nodes are separated from these. A border node (whether initially on the boundary, or in the course of the re-embedding process collapsed onto it), by contrast, obviously is non-separated from these boundary/virtual arcs (otherwise it would not be on, or would not have been collapsed onto the boundary). Hence no such zero-arc between a critical point and a border node can occur.

Case 3. As boundary nodes are used as separation test source in our free-boundary separation test, the non-separation of two boundary points is recognized – but ignored if these are connected by a zero-path along the surface boundary (cf. Section 5.2). A non-border zero-arc a between two non-critical border nodes may thus indeed occur (in contrast to cases 1 and 2); however, only if these two border nodes are additionally connected by a path of border zero-arcs. As these border zero-arcs are collapsible, the two incident nodes of a will eventually be collapsed onto a common point on the surface boundary via these. The arc a thereby turns into a loop (same start and end point), which can be contracted to a point.

Finally, it is easy to see that the number of zero-arcs to be collapsed is finite (i.e. the re-embedding process terminates): While operator 2 creates (a finite number of) additional zero-arcs within a (non-simple) zero-patch, the subsequent application of operators 1 and 3 is able to collapse these while at the same time reducing the total number of yet-to-be-collapsed zero-patches relative to the state before operator 2 was applied. This strictly monotonic decrease asserts termination after a finite number of operations. □