Quad Mesh Quantization Without a T-Mesh

Yoann Coudert-Osmont, David Desobry, Martin Heistermann, David Bommes, Nicolas Ray, Dmitry Sokolov

Quad Mesh Quantization Without a T-Mesh

Yoann Coudert- -Osmont[1], David Desobry[1], Martin Heistermann[2], David Bommes[2], Nicolas Ray [1] and Dmitry Sokolov[1]

[1]Inria Nancy – Grand Est, LORIA, Villers-les-Nancy, France
{ yoann.coudert-osmont, david.desobry, nicolas.ray } @inria.fr, dmitry.sokolov@univ-lorraine.fr
[2]University of Bern, Bern, Switzerland
{ martin.heistermann, david.bommes } @inf.unibe.ch

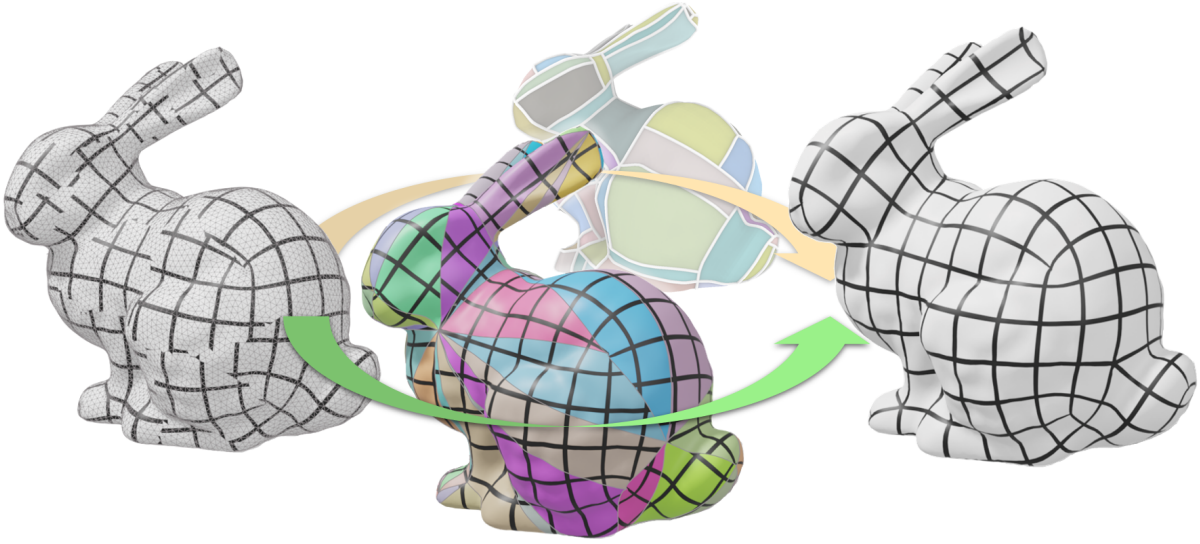


Figure 1: *Quantization algorithms take as input a seamless map (left) defined on a triangular mesh and compute a grid preserving map (right). Integer degrees of freedom are typically optimized on a T-mesh (middle—top) extracted from the surface and the corresponding seamless map. We show that replacing the T-mesh by a triangulation (middle—bottom) is simpler and more flexible.*

Abstract

Grid preserving maps of triangulated surfaces were introduced for quad meshing because the 2D unit grid in such maps corresponds to a subdivision of the surface into quad shaped charts. These maps can be obtained by solving a mixed integer optimization problem: real variables define the geometry of the charts and integer variables define the combinatorial structure of the decomposition. To make this optimization problem tractable, a common strategy is to ignore integer constraints at first, then to enforce them in a so-called quantization step. Actual quantization algorithms exploit the geometric interpretation of integer variables to solve an equivalent problem: they consider that the final quad mesh is a subdivision of a T-mesh embedded in the surface, and optimize the number of subdivisions for each edge of this T-mesh. We propose to operate on a decimated version of the original surface instead of the T-mesh. It is easier to implement and to adapt to constraints such as free boundaries, complex feature curves network, etc.

Keywords: *modelling, mesh generation, surface parameterization*

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

1. Introduction: remeshing via mapping

In computer graphics, quad meshes offer a versatile way to represent surfaces, they are used by many applications such as editing, texture mapping, subdivision, animation, etc. Naturally arises the question to convert other surface representations to quad meshes. For analytic surfaces, represented by spline functions $\mathbb{R}^2 \to \mathbb{R}^3$, a quad mesh is readily available as the image of the unit grid by the spline. Nevertheless, this method cannot be directly applied for producing

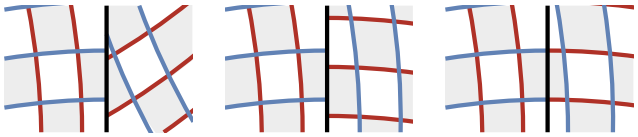


Figure 2: *Map classification according to its behavior across discontinuity (shown in black): arbitrary map (left), seamless map (middle), and integer-grid map (right).*

quad meshes from triangulated surfaces, because a parametric domain does not exist. Our goal is to generate maps to replace the parametric domain, and even introduce discontinuities in these maps to handle a much larger family of quad meshes that allows quad strip cycles and irregular vertices.

Remeshing with aid of map generation was introduced to the computer graphics community by the pioneering work [Alliez et al.(2002)]. The idea is, starting from a triangular mesh, to produce a better quality triangular mesh by generating a triangulation in a 2D map and projecting it onto the surface of interest. The method stems from texture mapping algorithms, and to represent the map, Alliez et al. used 2D coordinates attached to the triangle corners of the input triangulation. Then, by interpolating barycentric coordinates of three triangle corners, we can project any point of the map onto the surface.

Throughout this paper, we call a map of a triangulated surface $\mathcal{M}$ a function $U : \mathcal{M} \to \mathbb{C}$ that is linear per triangle, with positive Jacobian; where the Jacobian matrix of the map is defined by interpreting complex numbers $z \in \mathbb{C}$ as vector $\begin{pmatrix} \Re(z) \\ \Im(z) \end{pmatrix}$.

These maps are typically computed via numerical methods [Hormann et al.(2007)] minimizing a measure of the map distortion under some constraints (injectivity, feature alignment, etc.).

Few years later, this approach of remeshing via mapping was generalized to quad mesh generation [Ray et al.(2006), Kälberer et al.(2007), Bommes et al.(2009)]: the idea is to produce a map of an input triangulated surface such that a regular unit grid defined in the map forms a quad mesh on the surface. If the map is continuous everywhere (as in [Alliez et al.(2002)]), this approach is rather easy to implement. For most surfaces, however, maps without discontinuities do not exist. Fortunately, it is possible to form a valid quad mesh on the input surface despite the presence of discontinuities in the map, as long as discontinuities respect some constraints. Thus we are interested in a certain class of maps called *grid preserving maps*. We will define the notion shortly, but the main idea is very simple: the unit grid is invariant to integer translations and rotations by 90°. If an edge of the input triangulation has two distinct images in the map due to discontinuity, its images must be the same up to a composition of a rotation by a multiple of 90° and an integer translation, thus ensuring preservation of the grid across the cut (see Figure 2–right).

The challenge is to produce a grid-preserving map with as little distortion as possible, and typically it is formulated as an optimization problem, where we want to minimize an objective function $f(U)$ under grid preservation constraints. There are many ways to define the objective function, but most of them include an estimation of the map distortion. For example, it can be the distance between the columns of the Jacobian matrix of U and the branches of an input frame field [Kälberer et al.(2007), Bommes et al.(2009)]. For harder problems, more elaborate functions [Garanzha et al.(2021)] allow to generate valid maps (with positive Jacobian).

2. Integer-Grid Maps

In this section we define formally the constraints that must be satisfied by a map U to be grid-preserving. We use half-edges data structure [Muller and Preparata(1978)] (Figure 3—left) to access the mesh connectivity, and we represent the map $U = (U_1 \dots U_H)^\top$ by the coordinates U_h of the origin of half-edge h in the map, where H stands for the total number of half-edges in the input triangulation. In order to characterize the admissible discontinuities of U across edges, we need to add extra variables: given a half-edge h, the transition function between its facet and its opposite facet is defined by a Gaussian integer translation t_h and a rotation $r_h = \imath^k$, where $k \in \mathbb{Z}$ (Figure 3—right).

2.1. (Untractable) formalization

In this formulation, our optimization problem has the following set of variables: map coordinates $\{U_h\}_{h=1}^H$, rotations $\{r_h\}_{h=1}^H$ and translations $\{t_h\}_{h=1}^H$.

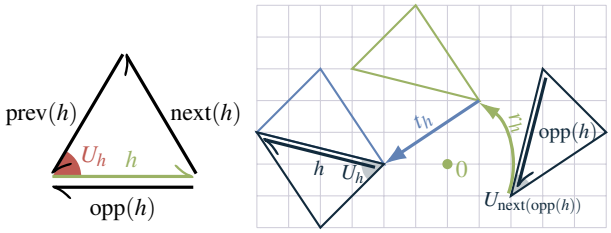


Figure 3: *Notations: half-edges data structure (left), and grid preserving transition function across an half-edge h given by a rotation r_h and a translation t_h (right).*

We are now ready to fully formulate the conditions that bind any half-edge $h \in [1 \dots H]$ with its opposite $\text{opp}(h)$.

- The first condition is that the transition function is a combination of a rotation r_h and a translation t_h:

$$\begin{cases} U_h & = & r_h U_{\text{next}(\text{opp}(h))} & + & t_h \\ U_{\text{next}(h)} & = & r_h U_{\text{opp}(h)} & + & t_h \\ r_h & \in & \{1, \imath, \imath^2, \imath^3\} \end{cases} \tag{1}$$

- The second condition is that some variables must be integer. Obviously, translations t_h must be Gaussian integer, but in addition to that, singular vertices also need to be Gaussian integer. Indeed, when angles of triangle corners (in the map) sharing a vertex v do not sum to 360°, the vertex v is called singular, and needs to be Gaussian integer to produce an irregular vertex in the final quad mesh. These constraints can be expressed as follows:

$$\begin{cases} t_h \in \mathbb{Z}[\imath] & \forall h \in \{1 \dots H\} \\ U_h \in \mathbb{Z}[\imath] \text{ if the origin of } h \text{ is singular} \end{cases} \tag{2}$$

where $\mathbb{Z}[\imath] = \{a + \imath b \mid a, b \in \mathbb{Z}\}$ are the Gaussian integers.

Finally, grid-preserving map generation can be stated as the optimization problem:

$$\underset{\{U_h,r_h,t_h\}_{h=1}^{H}}{\arg\min} \quad f(U) \quad \text{under constraints (1) and (2).} \tag{3}$$

2.2. Make it tractable, step №1: frame field

While being elegant, in this general form the Prob. (3) is intractable. The mixture of integer and real variables, and more especially, high number of integer variables makes the problem extremely difficult to solve. In practice, the vast majority of attempts to solve the problem reduces the number of degrees of freedom[†] by decomposing the problem into two parts. The first part defines the desired orientation for the quads (the frame field), and the second one defines two scalar fields (texture coordinates) such that their iso-values are aligned with the frame field. As observed in [Bommes et al.(2009)] both problems involve integer values and can be solved by the same type of solver.

So, the idea is to drastically reduce the number of integer variables in the Prob. (3) by computing a frame field $\mathcal{F}$ [Vaxman et al.(2016), Bommes et al.(2009), Desobry et al.(2021)] over the input triangulation. This allows us to fix all rotations $\{r_h\}$, so they are no longer variables of our problem. We can go even further, and eliminate a large part of the translations $\{t_h\}$. Having fixed the frame field, we have fixed singular vertices, so we can cut the surface into a topological disc along a set of half-edges called *cut graph* $\mathcal{C} \subset \{1 \dots H\}$. The idea is to obtain a topological disc with all the singularities of $\mathcal{F}$ located on the boundary [Kälberer et al.(2007)]. It allows us to set all translations t_h and rotations r_h to respectively zero and one through every edge that doesn't belong to $\mathcal{C}$ (refer to Figure 4). In that way, the conditions $t_h \in \mathbb{Z}[i]$ are satisfied for concerned half-edges. Let us express the constraints formally:

$$\begin{cases} t_h = 0 & \forall h \notin \mathcal{C} \\ r_h = 1 & \forall h \notin \mathcal{C} \\ r_h \text{ constrained by } \mathcal{F} & \forall h \in \mathcal{C} \end{cases} \tag{4}$$

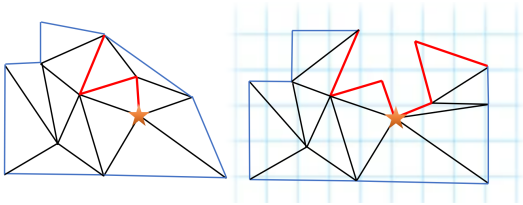


Figure 4: *The mesh (left) admits a seamless map (right) that is discontinuous across the cut graph $\mathcal{C}$ (red edges). Here, discontinuities are characterized by a rotation of $\pi/2$ around the singularity (star).*

To sum up, in the rest of this paper we are interested by the following problem:

$$\underset{U}{\arg\min}\, f(U) \quad \text{under constraints (1), (2) and (4).} \tag{5}$$

[†] Note that reducing the search space leads to a smaller set of attainable solutions.

Note that the translations $t_h \in \mathcal{C}$ in this formulation do not represent degrees of freedom as they can be computed from $\{U_h\}_{h=1}^{H}$ and $\{r_h\}_{h=1}^{H}$.

2.3. Make it tractable, step №2: quantization

The optimization problem (5) is a minimization of a function subject to linear equality constraints (feature alignments, discontinuities in the map), integer values (for features and singularities) and to inequality constraints (positive Jacobian). The difficulty with such a mixed integer problem is that it requires both combinatorial optimization for the integer variables and numerical optimization for the other variables. For grid preserving maps, quantization-based algorithms separate combinatorial and numerical optimizations in different steps, refer to Algorithm 1 for a typical outline. A seamless map (Figure 2) is obtained by numerical optimization without considering integer constraints, then the quantization step finds a set of linear constraints that fixes integer degrees of freedom, and finally these constraints replace integer constraints in a second numerical optimization step.

In this work, we address the quantization step: given a seamless map, we search a set of constraints that would enforces the variable integrity while minimizing distortion.

Algorithm 1: Generation of grid preserving maps

Input: A mesh $\mathcal{M}$, represented by half-edges $\{h\}_{h=1}^{H}$, frame field $\mathcal{F}$, cut graph $\mathcal{C}$
Output: Grid preserving map U^{grid}

```
/* Compute a seamless map: ignore integer
   constraints                                  */
```
$U^{\text{seamless}} \leftarrow \underset{U}{\arg\min}\, f(U)$ under constraints (1) and (4);
```
/* Find linear constraints to enforce integer
   values                                       */
```
$[A, \omega] \leftarrow \text{quantize}(\mathcal{M}, \mathcal{F}, U^{\text{seamless}})$;
```
/* Compute a grid preserving map: numerical
   optimization of a real-valued problem under
   affine constraints                           */
```
$U^{\text{grid}} \leftarrow \underset{U}{\arg\min}\, f(U)$ s.t. $AU = \omega$ under constraints (1), (4);

3. State of the art for quantization

Rounding The pioneer quantization algorithms [Kälberer et al.(2007), Bommes et al.(2009)] simply round integer variables to their nearest integers. However, the existence and quality of a map subject to these new constraints are not guaranteed. Alternating between map optimization and rounding [Bommes et al.(2012)] improves the situation, and can even solve the problem [Wang et al.(2022)] when all singularities are located on the boundary of the domain. However, such iterative optimizations do not provide robustness in the general case.

Branch and bound Robustness can be achieved by a branch and bound algorithm that minimizes a function of integer valued variables. However, it requires to run a parametrization algorithm for

Algorithm 2: Quantize — *T-mesh version*

Input: Triangular mesh $\mathcal{M}$, seamless map U^{seamless}
Output: Linear system A, ω such that $AU = \omega$ implies (2).

```
// Compute a simpler proxy problem
```
1 $\mathcal{M}^{\text{proxy}} \leftarrow$ extract T-mesh $(\mathcal{M}, U^{\text{seamless}})$;
2 $U^{\text{proxy}} \leftarrow$ propagate map $(\mathcal{M}, \mathcal{M}^{\text{proxy}}, U^{\text{seamless}})$;
```
// Optimize integer degrees of freedom
```
3 $\{l_h\}_{h=1}^{H^{\text{proxy}}} \leftarrow$ find integer edge lengths(U^{proxy});
```
/* Move the solution back to the original
   problem                                   */
```
4 $[A, \omega] \leftarrow$ get constraints $(\mathcal{M}, \{l_h\}_{h=1}^{H^{\text{proxy}}})$;

Algorithm 3: Quantize — *Decimated mesh version*

Input: Triangular mesh $\mathcal{M}$, seamless map U^{seamless}
Output: Linear system A, ω such that $AU = \omega$ implies (2).

```
// Compute a simpler proxy problem (§4.1)
```
1 $\mathcal{M}^{\text{proxy}}, D \leftarrow$ decimate $(\mathcal{M})$;
2 $U^{\text{proxy}} \leftarrow D \cdot U^{\text{seamless}}$;
```
// Optimize integer degrees of freedom (§4.2)
```
3 $\{\omega_h\}_{h=1}^{H^{\text{proxy}}} \leftarrow$ solve edge vector(U^{proxy});
```
/* Move the solution back to the original
   problem (§4.3)                            */
```
4 $A \leftarrow D$;

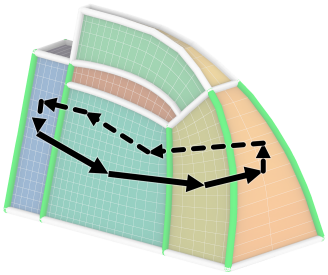


Figure 5: *QGP atomic operation: given a loop (black) the adjacency graph of the T-mesh partition, it increases the length of each T-mesh arc (green) that is dual to an edge of the loop.*

each evaluation of the objective function i.e. each set of integer variables to be evaluated. To make it tractable, [Bommes et al.(2013)] apply it to a decimated mesh and propagate the result to the original mesh. This approach is pretty slow, but allows to produce coarse meshes. It can be used for interactive re-meshing [Ebke et al.(2016)] by local updates.

Quantized Global Parametrization (QGP) The reference method for quantization [Campen et al.(2015)] takes another point of view. Starting from a seamless map, it proceeds in three steps:

1. *Compute a simple proxy problem (Algorithm 2, lines 1 and 2).* A motorcycle graph [Eppstein et al.(2008)] is constructed by simultaneously tracing iso-curves starting from all singular points of the parametrization. These traces form a T-mesh, cutting the surface map into rectangles aligned with the coordinate axes of the parametric domain, and partitioning the input surface into curved rectangular-shaped regions.
2. *Optimize for integer degrees of freedom (Algorithm 2, line 3).* The quantization step assigns integer lengths to all motorcycle graph arcs or, equivalently, finds the size for all rectangles in the map. The atomic operation for this computation is an increase/decrease of the length of each edge crossed by a loop defined on the adjacency graph of the partition, refer to Figure 5 for an illustration. Each such loop is computed by Dijkstra's algorithm to be as short as possible, with weights crafted to favor edge lengths close to their length in the seamless map.
   Initially, all edge lengths are set to zero, and then iteratively increased until there are no more zero length edge path joining two singularities. Once this validity condition is achieved, QGP continues to apply atomic operations as long as they do not violate the validity criterion and decrease the difference between the edge length and corresponding length in the seamless map.
3. *Deduce the constraints for the original problem (Algorithm 2, line 4).* A grid preserving map is then generated by solving the seamless map problem, with the additional constraint that the difference between pairs of singular points are constrained to match integer values founded by the quantization.

Built on top of QGP, more recent works can bound angular deviation from the frame field [Lyon et al.(2021a)], support free boundaries [Lyon et al.(2019)] or even merge singular vertices [Lyon et al.(2021b)].

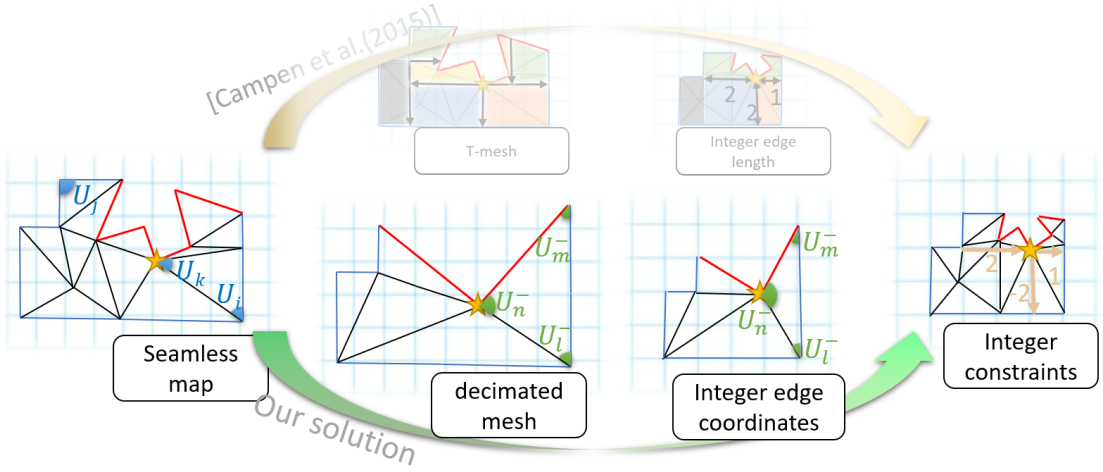
Our contributions The main contribution of our paper is the adaptation of QGP's framework to a simpler proxy: instead of relying on T-meshes, we propose to perform a standard decimation of the input object, directly working on triangulated meshes. Thus, the goal is to have a provably robust quantization method that produces results similar to QGP both in terms of speed and quality.

Our intuition behind this is that a triangular mesh is a more versatile proxy than a T-mesh. While being out of scope for the present paper, we discuss in Section 6 several possible extensions that should directly benefit from our adaptation.

4. Our quantization

This is the main technical section of the paper, here we propose a new quantization method that is an adaptation of QGP where the T-mesh is replaced by a decimated mesh (compare Algorithms 2 and 3). Our pipeline follows the same three steps as QGP, and the rest of this section focuses on each of the steps. Figure 6 provides an illustration and gives the outline of the section:

1. The first step (§4.1) decimates the mesh and assembles a matrix D that produces map coordinates U_h^- in the coarse mesh when multiplied by original map coordinates U.
2. The second step (§4.2) computes the (Gaussian integer valued) geometry $\omega_h = U^-_{next(h)} - U^-_h$ of each coarse edge h in the map.
3. The last step (§4.3) basically chains both relations to obtain the constraints that will force the map to be grid-preserving by matching the geometry of the coarse edges.



STEP	§4.1 Decimation	§4.2 Optimization of integer edge geometry	§4.3 Extraction of integer constraints
Equation	$U_h^- = D_h U$	$U^-_{next(h)} - U^-_h = \omega_h$	$(D_{next(h)} - D_h)U = \omega_h$
Example	$\begin{cases} \dots \\ U_l^- = U_i \\ U_m^- = U_k - \imath(U_j - U_k) \\ U_n^- = U_k \\ \dots \end{cases}$	$\begin{cases} \dots \\ U_m^- - U_l^- = (0,4) \\ U_n^- - U_m^- = (-1,-2) \\ U_l^- - U_n^- = (1,-2) \\ \dots \end{cases}$	$\begin{cases} \dots \\ U_k - \imath(U_j - U_k) - U_i = (0,4) \\ U_k - (U_k - \imath(U_j - U_k)) = (-1,-2) \\ U_i - U_k = (1,-2) \\ \dots \end{cases}$

Figure 6: *Our pipeline illustrated on the mesh of Figure 4. The reference algorithm [Campen et al.(2015)] (upper path) generates a T-mesh, optimizes its edge lengths and converts them to integer constraints. We instead (lower path) decimate the mesh, optimize its edge geometry and convert it to integer constraints. The table shows how equations that link these steps are instantiated on a triangle of this example.*

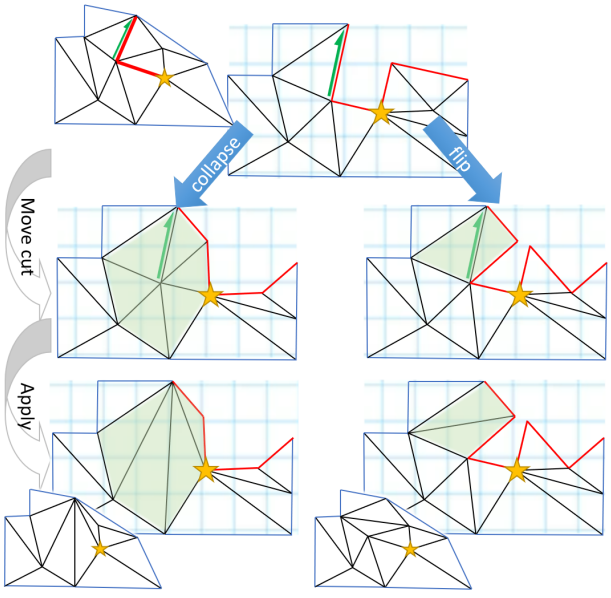


Figure 7: *Decimation operations (edge collapse and flip) need to update the seamless map. Whenever an affected edge is situated on a cut, we move the cut before applying the operation.*

Algorithm 4: *Optimize for integer degrees of freedom*

Input: Mesh $\mathcal{M}^{\text{proxy}}$, map U^{proxy}
Output: ω that corresponds to a grid preserving map U^-

/* Initialize ω (see §4.2.3) */
$n \leftarrow 0$
repeat
  for $h \in \{1 \dots H\}$ **do**
    $\omega_h \leftarrow U^{\text{proxy}}_{next(h)} - U^{\text{proxy}}_h$
    $\omega_h \leftarrow a \in \mathbb{Z}[\imath] \setminus 0$ that minimizes $\|a - 2^n \omega_h\|$
  end
  improve_closeness(ω)
until ω *is closed and valid*;
/* Modify ω to optimize $E(U^-)$ */
repeat
  for $(h_{seed}, a_{seed}) \in \{1 \dots H\} \times \{1, \imath, \imath^2, \imath^3\}$ **do**
    $\alpha \leftarrow$ find_atomic_operation $(\omega, h_{seed}, a_{seed})$
    if $E(\omega + \alpha) < E(\omega)$ *and* $\omega + \alpha$ *is valid* **then**
      $\omega \leftarrow \omega + \alpha$
    end
  end
until ω *remains the same*;

4.1. Decimated mesh as a proxy problem

We decimate the original mesh with the algorithm proposed by [Bommes et al.(2013)]: it collapses as many edges as possible, then flips all edges that do not respect the Delaunay criteria in the map space, and repeats until convergence. Conditions for an edge to be collapsed are that: it does not remove a singular vertex, the resulting mesh is manifold, and new triangles are oriented counter-clockwise in the map. The algorithm tries to collapse edges in decreasing length order.

After each editing operation, new triangle map coordinates are copied from the map coordinates of triangles prior the operation. Note that map discontinuities require special care: before applying an operation, some triangles are moved in the map to push the discontinuity outside of the region affected by the operation, as illustrated in Figure 7.

An important thing to note here is that both moving the discontinuity and applying a collapse/flip operation produce new map coordinates that can be expressed as a linear combination of the old map coordinates. We chain these linear expressions during the decimation process, registering it in a matrix D that produces the coarse map coordinates when multiplied by the original map coordinates, and, in particular, we have

$$U^{\text{proxy}} = D \cdot U^{\text{seamless}}.$$

4.2. Optimize integer degrees of freedom

Having decimated the input mesh and computed the corresponding valid seamless map U^{proxy}, we need to find a valid initial grid preserving map U^- that we will further optimize. Let us put aside for a brief moment the initialization of U^- (we return to the question in §4.2.3) and focus on the optimization.

We are looking for an integer valued map coordinates U^- with edges as close as possible to the corresponding edges in the seamless map U^{proxy} i.e. U^- must minimize:

$$E(U^-) = \sum_h \frac{\|\omega_h^{\text{proxy}} - \omega_h\|^2}{\|\omega_h^{\text{proxy}}\|^2},$$

where $\omega_h = U^-_{\text{next}(h)} - U^-_h$ and $\omega_h^{\text{proxy}} = U^{\text{proxy}}_{\text{next}(h)} - U^{\text{proxy}}_h$ are respectively the geometry of half-edge h in the grid preserving map U^- and in the proxy map U^{proxy}.

It is more convenient to manipulate ω_h rather than U^-_h. Both representations are equivalent up to a translation per triangle if and only if opposite half-edges have opposite geometry up to the map rotation $\omega_h = -r_h \omega_{\text{opp}(h)}$, and ω is closed i.e. the boundary of each triangle T is a closed loop in the map $\sum_{h \in T}$. Moreover, $\omega_h \in \mathbb{Z}[\imath], \forall h$ ensures that we can find $U^-_h \in \mathbb{Z}[\imath], \forall h$ e.g. by setting $U^-_h = 0$ on a corner h of each triangle. The last condition for ω to match a grid preserving map U^- is the map validity i.e. $\det(J_T) > 0$. In the rest of this section, the energy and validity of the map U^- are abusively attributed to ω to simplify the notations.

As done in QGP, our optimization (Algorithm 4) relies on applying atomic operations that preserve the map validity. An initial valid ω is given (lines 1–8) by scaling-and-rounding the input seamless map U^{proxy} (again, we dive into details in §4.2.3). Then, the algorithm applies energy decreasing atomic operations (lines 9-16) until no more can be found. As in QGP, this strategy reaches a local minimum of the energy but not necessary the global minimum.

We first present the core of the optimization method: the definition §4.2.1 and computation §4.2.2 of our atomic operation. Then, we explain how to optimize the initialization of ω_h.

4.2.1. Our atomic operation

Our atomic operation edits the quad mesh combinatorial structure via our variables ω_h. To better illustrate it, let us instantiate[‡] a quad loop insertion (suppression) operation (Figure 8–top row).

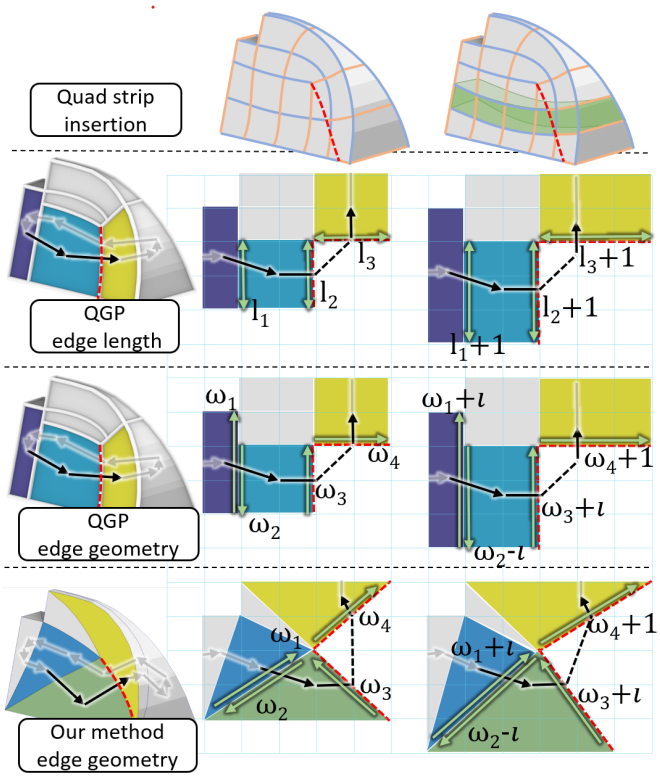


Figure 8: *Quad loop (green) insertion (top row) using QGP atomic operation (second row), our formalism applied to a T-mesh (third row) and the decimated mesh (bottom row).*

QGP is manipulating the combinatorial structure of the quad mesh by adjusting edge lengths in the T-mesh l_i (Figure 8–second row). Having found a cycle of T-mesh facets (shown in black), QGP adjusts by 1 the length of crossed edges.

While QGP does not alter the direction of the crossed edges in the map, it simply elongates (shortens) them, it is possible to perform exactly the same operation by using another set of variables for our optimization: we can use the geometry of the edges instead of the

[‡] Here we show a quad loop insertion, but atomic operations both in QGP and in our method are more rich, refer to Appendix A for an illustration.

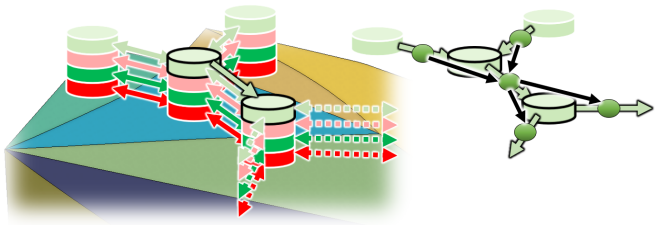


Figure 9: *Dual edge graph $\mathcal{G}$ of the quad covering (left). Nodes (cylinders) are defined by a triangle t and a value $a \in \{1, \imath, \imath^2, \imath^3\}$ (cylinders color). Nodes (t_i, a_i) and (t_j, a_j) are connected by the arc (h, a_i) if t_i is adjacent to t_j by crossing half-edge h and $a_i = r_h a_j$. To remove the path-dependent weights, shortest paths are computed in another graph where each oriented edge (left–black arrow) is transformed into a node (right–green sphere) and four oriented edges (right–black arrows).*

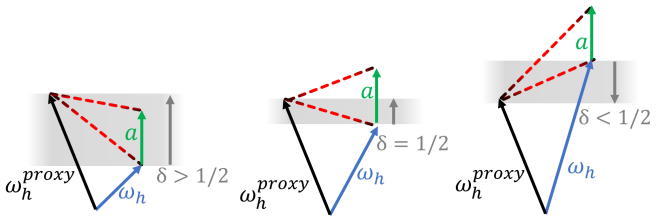


Figure 10: *δ measures the difference $\omega_h^{proxy} - \omega_h$ along the axis of a. The contribution of h to the energy is proportional to the squared length of the dotted red segments: bottom segment before adding a and upper segment after adding a. Comparing δ to $1/2$ allows to detect if the operation increases or decreases the energy.*

lengths (Figure 8–third row). In this case, each T-mesh arc in the map ω_h is updated by $\omega_h \leftarrow \omega_h + \alpha_h$ where $\alpha_h \in \{1, \imath, \imath^2, \imath^3\}$ is selected to increase (or decrease) the arc length by one. Note that this coherent choice of α_h preserves the equivalence of U^- and the updated value of ω.

In our case we do exactly this, but the edge orientation is not necessarily aligned with a possible value of α_h (Figure 8–last row). Therefore, the operation is defined by an oriented cycle (shown in black) but also the value α_h associated to each half-edge crossed by this loop. Here, values of α_h are constrained to preserve the equivalence of U^- and the updated value of ω (i.e. the closeness of the one form ω).

4.2.2. Computing atomic operations

We represent the atomic operation as a cycle in the dual edge graph $\mathcal{G}$ of the quad covering [Kälberer et al.(2007)] of the decimated mesh (see Figure 9). It encodes both a loop in the dual edge graph and the values $\alpha_h \in \{1, \imath, \imath^2, \imath^3\}$ for each half-edge h of the loop. This atomic operation updates ω by $\omega_h \leftarrow \omega_h + \alpha_h$ for each edge of the cycle.

To select an atomic operation to apply, Algorithm 4–line 11 needs a cycle that passes through an arc (h_{seed}, a_{seed}). We compute it by chaining the arc (h_{seed}, a_{seed}) with the minimal cost path that joins the extremities of (h_{seed}, a_{seed}), and where the path cost is the sum of weights of its edges.

Ideally, the path cost should be the energy gain, and paths breaking the map validity should not be considered. Unfortunately, this problem has negative and path dependent weights that makes the problem hard to solve. On the other hand, setting all weights to 1 is likely to find atomic operations that will be rejected in Algorithm 4–line 12.

As a consequence, we need to carefully choose weights $w_{h,a}$ that improve the probability for a cycle to yield a valid ω that decreases $E(\omega)$. We estimate the impact of adding a to ω_h by $\delta = \Re(\bar{a} \cdot (\omega_h^{\text{proxy}} - \omega_h))$ as illustrated in Figure 10:

- If $\delta > 1/2$, adding a to ω_h decreases the energy. To favor these edges, we set $w_{h,a} = 1/(1/2 + \delta)$; it is small ($< 1$), positive and decreasing with respect to δ.
- If $\delta \leq 1/2$, adding a to ω_h increases the energy. To avoid such edges, we set $w_{h,a} = (1 - \delta)/\varepsilon$; it is finite, positive, increasing with respect to δ, and larger than weight of edges that decrease the energy when $\varepsilon \to 0$.
- **Pay attention to the validity of the map:** If adding a to edge h breaks the map validity on the triangle of h, the weight $w_{h,a}$ is replaced by $1/\varepsilon^2$ that is larger than all other weights. As a consequence, this edge will be considered only if there exists no other path without such edges.

To sum up, the cycle will contain only energy decreasing edges if it is possible. Otherwise, it may accept some energy increasing edges. If there is no better solution, the cycle may invalid the map by crossing a triangle (by two consecutive edges), as it is possible that the cycle makes it valid again by crossing the same triangle on different edges and value of a.

Our **shortest path algorithm** is Dijkstra's algorithm, but we need to run it on the dual graph [Añez et al.(1996)] of $\mathcal{G}$ (see Figure 9–right) to account for the path-dependent weights. Indeed, the validity of the map on the triangle depends of both the current and the previous edge of the cycle, see Figure 8.

4.2.3. Initialization

Starting with $\omega \leftarrow 0$ is almost a good solution, but it fails to respect the validity of the map. In QGP, validity only requires each edge to have a positive length, and can always be obtained from $\omega = 0$ by a series of atomic operations. This nice property does not hold in our case, so we opted for an alternative approach.

If we take into account the fact that real numbers are represented by floating points, the input seamless map scaled by 2^n is actually grid preserving when n is large enough. It provides a certified way to find a valid initial ω.

In practice, Algorithm 4 minimizes the scaling in lines 1-8. The first step rounds ω_h to the nearest Gaussian integer of $2^n \omega_h^{\text{proxy}}$ that is not null. The resulting ω may not be closed, so Algorithm 4–line 7 tries to fix it by pairing triangles with opposite sum of ω. In fact, we compute a path in $\mathcal{G}$ that links both triangles using the algorithm that finds paths for the atomic operations. The existence of such a path is guaranteed by Theorem 3.1 of [Noma et al.(2022)], but the final map may still not be valid, leading to increasing n and retrying.

4.3. Move the solution back to the original problem

By definition $\omega_h = U^-_{\text{next}(h)} - U^-_h$, and U^- is related to the final map by the linear system $U^- = DU^{\text{grid}}$ computed during the decimation. It gives the equation $\omega_h = D_{\text{next}(h)}U^{\text{grid}} - D_h U^{\text{grid}}$ that constrains the path between singularities (in the map) to match the integer degrees of freedom computed in ω. A last constraint is needed to fix the translation of the map: placing a singularity at position 0 is sufficient to ensure that all singularities are on a Gaussian integer position. Then the final map U^{grid} is computed in Algorithm 1–line 3.

4.4. Implementation details

In this section we give few implementation ingredients we found important to reproduce our results.

Numerical optimization As in most previous works, we generate maps aligned with a frame field. We start by integrating the frame field in the least squares sense, then untangle the result by rescaling the frame field branches and using [Garanzha et al.(2021)] if needed. This optimization is performed to produce the input seamless map, and to generate the integer-grid map subject to the quantization constraints. These two optimizations are performed with a variable reduction matrix M such that $U = MX$ satisfies all linear constraints for all vector X, as suggested in [Bommes et al.(2012)]. It allows to minimize $f(MX)$ without constraints, and get the result by $U = MX$.

Feature curves and boundaries Feature or boundary curves of the surface have to be represented by a subset of quad edges in the final mesh. To do so, each half-edge vector h must be real or purely imaginary in the map U. This constraint is enforced during the seamless map computation, and propagated to the quantization step, where it removes some edges (h,a) of the graph. For meshes with boundaries, the atomic operation can have cycles that are partially outside the mesh i.e. paths joining two boundary edges.

Non-integer DoF Note that if we compute U^{grid} as in Algorithm 1–line 3, it places each vertex that has its counterpart in the decimated mesh to a Gaussian integer position. This constraint is too strong for vertices that are not singular but remain in the decimated mesh e.g. to preserve feature curves. To get back these degrees of freedom, we define a sparse vector Y such that Y is zero on integer variables, U respects (1), and $Y+U$ respects (1) and (2). The map U^{grid} is the map U that minimizes $f(U)$ under these constraints, where f is the map distortion used to formulate our problem in the introduction section.

Graph weights for cycle search The weights cannot be represented by floating point numbers because ε must be infinitely smaller than all other numbers. Our implementation simply stores three floating point numbers x, y, and z to represent $w_{h,a} = x + y/\varepsilon + z/\varepsilon^2$ and compare weights with $\varepsilon \to 0$.

5. Results

Performances of our algorithm have been evaluated on the same CAD database as in [Reberol et al.(2021)]. It contains 1000 models of the ABC dataset [Koch et al.(2019)] and the 114 models of the MAMBO dataset [Ledoux(2019)]. These CAD models have many feature curves that makes it challenging for producing grid preserving maps. We were able to generate a valid input for our algorithm (seamless maps) with boundary/feature alignment for 1111 models and without for the last 3 models. On each model, we performed two quantization tests: one to produce a coarse result (Figure 11), and the other to generate a quad mesh at the resolution of the input (Figure 12). Coarse results are obtained by scaling down U^{proxy} before applying the optimization of integer degrees of freedom (§ 4.2).

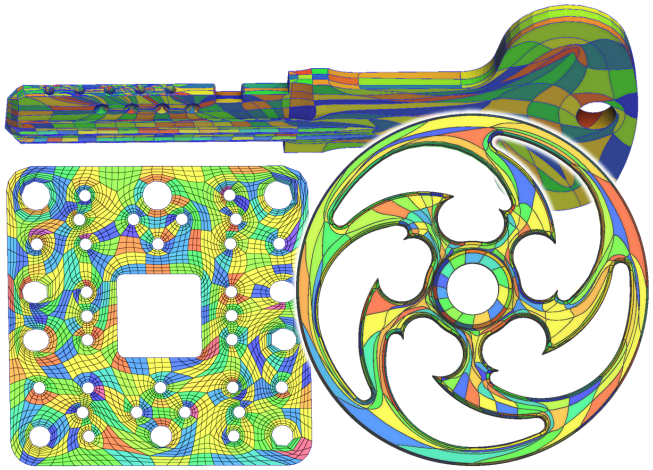


Figure 11: *Coarse quantization segments the model into quad shaped charts. It is often necessary to subdivide charts (bottom–left) to get a quad mesh with fair geometry.*

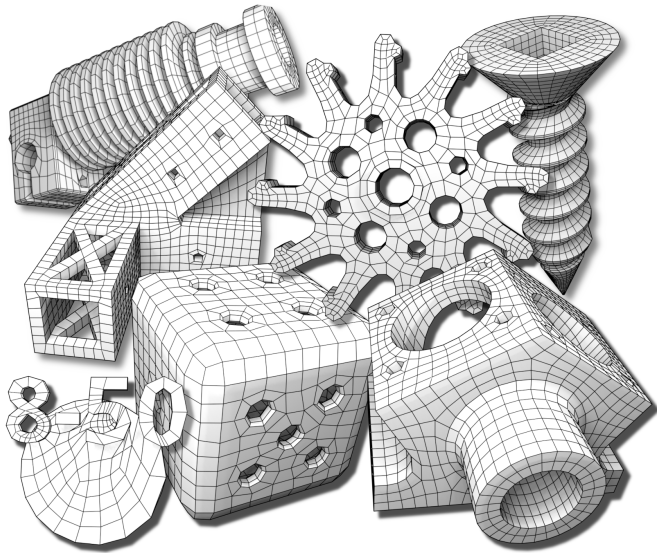


Figure 12: *Examples of quad meshes generated by our algorithm.*

5.1. Running times

Our running times have the same order of magnitude to those of QGP, which is almost negligible when compared to the full quad mesh generation pipeline. To be more specific, we conducted a compari-

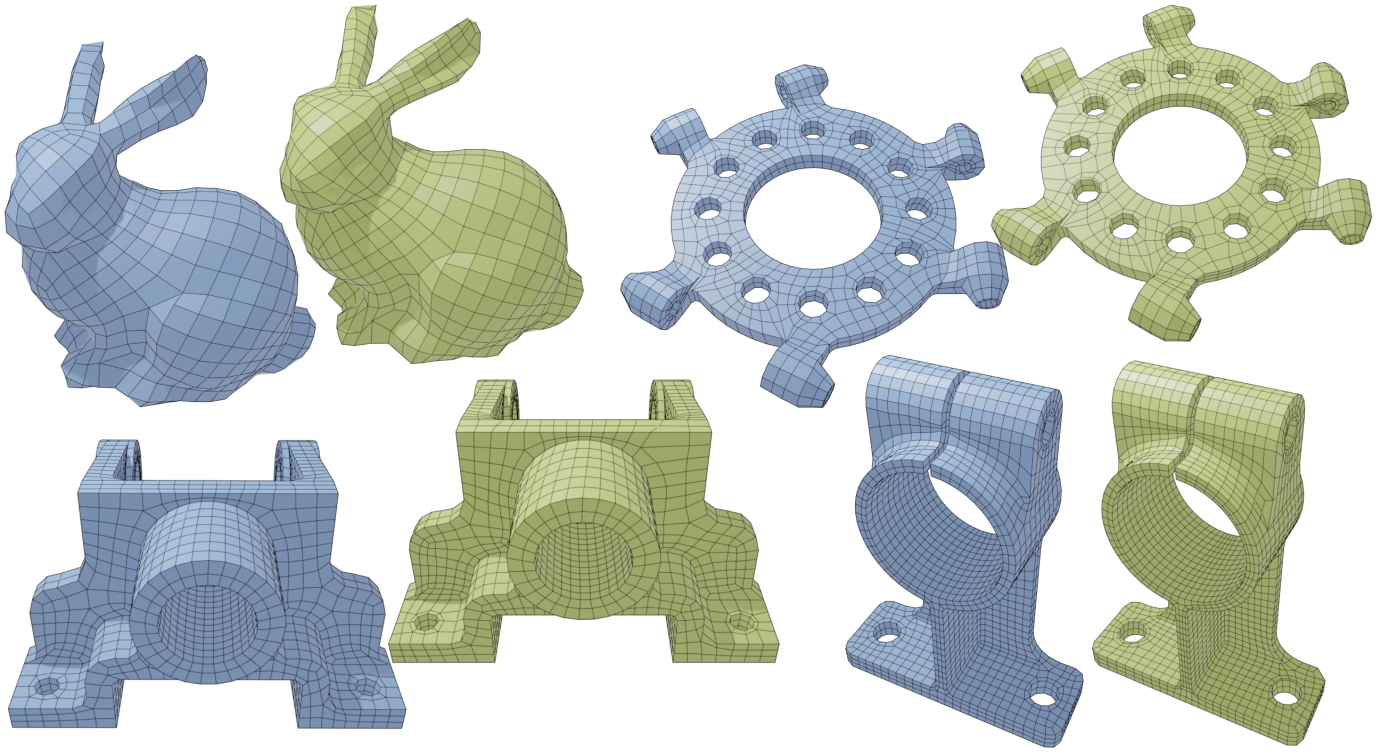


Figure 13: *Quadrilateral meshes produced by our method (blue) and [Campen et al.(2015)] (green) exhibit similar quality.*

son of the running times of both methods on our database. Within each method (Algorithms 2 and 3), we separate the proxy generation step (lines 1–2) from the quantization step (line 3). We executed our pipeline on a PC equipped with an 11th generation Intel Core i5-11500Hx12 processor, while the timing measurements for QGP were obtained using a PC featuring two AMD 7742 CPUs.

Upon analysis, we observed that our decimation is directly proportional to the input mesh size and proves to be faster than the process of generating a T-mesh, as depicted in Figure 14–top. Conversely, QGP's quantization scales better than ours with respect to the proxy size, as illustrated in Figure 14–bottom.

In the most challenging scenario (Figure 16), our rounding process causes the red triangle to degenerate, snapping the singularity (star) onto the feature edge (white) for scalings below 64. In this scenario, our algorithm was 50 times slower than QGP, resulting in the entire quad mesh generation pipeline being 3 times slower. Fortunately, it's worth noting that this particular case is an outlier in our dataset (refer to Figure 14).

5.2. Quality

There exists many criteria to evaluate the quality of a quad mesh, but the impact of the quantization algorithm is rather difficult to estimate independently of the other steps of the pipeline (frame field, parametrization, quad mesh extraction, smoothing and so on).

From visual inspection on few models (see Figure 13 and supplemental materials), quad mesh quality obtained from QGP and our method are very similar. When compared with [Pietroni et al.(2021)] (see Figure 17), the overall quality is also very similar, except that our method produces simpler configurations because it does not produce singularities that are not present in the input frame field. Our algorithm works well on models with thin features Figure 18 that are known to be difficult to quantize.

6. Possible extensions

Recall that we have modified QGP framework to manipulate a decimated mesh and edit edge geometry in the map instead of working with T-meshes and adjusting only edge lengths. Being built upon QGP's framework, our method is likely to support similar extensions such as bounded angle deviation [Lyon et al.(2021a)] or introducing singularities in facets of the decimated mesh when no valid quantization can be found. Moreover, our representation offers new advantages that can be directly exploited or opens new research opportunities. While being out of scope of the present paper, we list few of them here.

6.1. Free boundary

As opposed to T-meshes, edges of the decimated mesh are not necessarily axis aligned in the map. As a consequence, if we do not explicitly force this alignment on boundary edges, the grid preserving map will be free of boundary conditions, see Figure 19. With

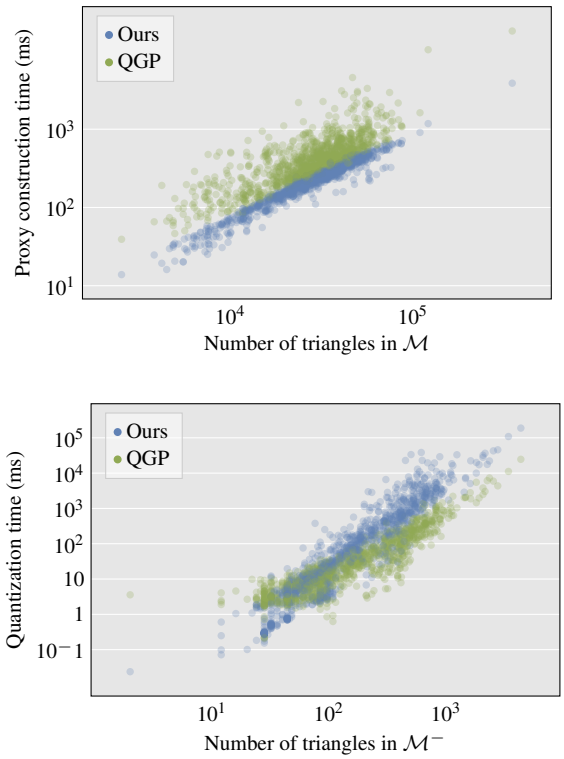


Figure 14: *Timing for proxy generation (top) and quantization (bottom): Each model in the database generates an blue dot for our decimation method and a green one for the T-mesh generation using QGP.*

T-meshes [Lyon et al.(2019)] a substantial effort is required to obtain this behavior: the T-mesh has to be refined and virtual arcs must be introduced.

6.2. Enforcing feature constraints at the quantization step

Another benefit of having edges not aligned with map axes is that we can support input seamless maps misaligned with feature curves and boundaries. This feature is especially important for CAD objects with so many feature curves constraints that seamless maps may not exist, or be too distorted and very difficult to compute.

To respect as many feature curve constraints as possible, we try to enforce them during the quantization step. We do so by introducing the objective to set to 0 either the real of imaginary part of feature edges in both the energy function and the shortest path edge weights.

A direct benefit of managing feature curves alignment at quantization step is to ease finding a seamless map with acceptable distortion (see Figure 20). Moreover, when the feature curves network is impossible to satisfy, we can produce a grid preserving map that satisfies as many constraints as possible. In this case, the alignment to the last feature curves may be achieved by post-processing of the mesh (see Figure 21).

6.3. Degenerated facets

One more advantage of edges not being aligned with the map axes, is that the decimated mesh can represent a large range of quad mesh combinatorial structures without degenerated ($\det(J) = 0$) facets.

QGP must degenerate some T-mesh facets (zero edge length) to align singularities (see Figure 22), a feature that is highly expected for quad remeshing. Having degenerated facets is however counter-intuitive if we consider that each T-mesh facet will be replaced by a regular grid of quads: it would produce empty spaces for degenerated facets. In QGP, zero length edges can be accepted because the embedding of T-junctions is not constrained for the generation of the grid preserving map: a weaker condition acting on paths of T-mesh arcs is sufficient. As opposed to this, our method does not need to degenerate triangles to represent aligned singularities. Importance of this feature is illustrated in two use cases: the coarse to fine map transfer and the direct quad mesh extraction.

6.3.1. Coarse-to-fine map transfer

Generation of a grid preserving map with quantization constraints (Algorithm 1–line 3) is very hard due to the $\det J > 0$ constraints. Transferring the map U^- to the original mesh is a much simpler alternative solution.

By considering that the coarse mesh is embedded in the original mesh (via the seamless map), it is not necessary to solve a complex optimization problem to obtain the final grid preserving map: it can be directly transferred from the coarse map. This solution requires to imprint edges of the decimated mesh into the original mesh, but is nevertheless more robust. A similar strategy was used in previous work [Myles et al.(2014), Lyon et al.(2019)] despite the necessity to manage zero length T-mesh arcs: in this case, the T-mesh undergoes the tedious process of being edited and re-embedded prior to produce the grid preserving map.

Note that our method can represent a large class of quad meshes without degenerating triangles, including aligned singularities (see Figure 22). As a consequence, we can transfer U^- to U^{grid} without editing and re-embedding the coarse mesh.

6.3.2. Quad mesh extraction on a decimated mesh

Maps U^{seamless} and U^{proxy} provide a cross parametrization between the original mesh and the decimated mesh. Therefore, a quad mesh can be extracted directly on the coarse mesh and then moved to the original mesh. This strategy has some advantages: the quad extraction is easier (integer coordinates have less numerical imprecision issues) and it opens the possibility to fill coarse facets with meshes that are more complex than just a regular grid, as done in other quad meshing methods [Myles et al.(2014), Pietroni et al.(2021)] .

For this approach, degenerated facets will once again be difficult to manage, making it easier to go with a decimated mesh than a T-mesh.

6.4. Flexible data structure implications

A decimated mesh is simply a set of triangles whereas a T-mesh is a polygonal surface where the boundary of each facet is decomposed

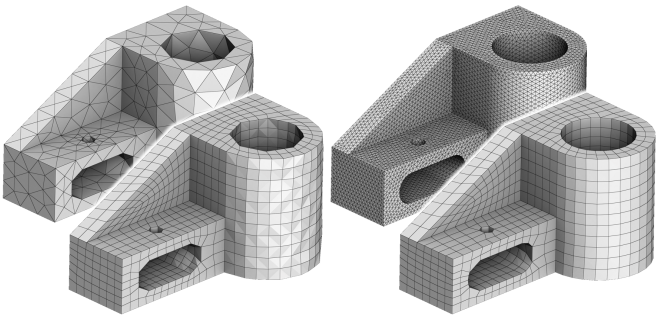


Figure 15: *Solving integer variables is independent of mesh resolution (49ms in this example). Only the timing of our decimation step is impacted: 6ms for the coarse model (left, 802 triangles) and 232ms for the fine mesh (right, 20402 triangles).*

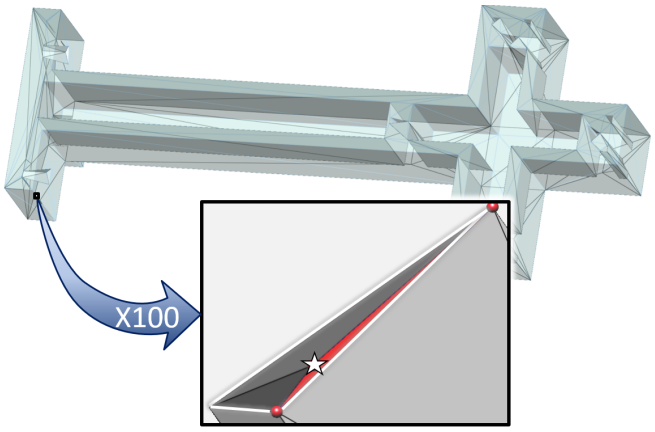


Figure 16: *Quantization initialization with high scale. The red triangle of this decimated mesh (visible in the second close-up) requires scaling the seamless map by a factor of 64 to avoid degenerated map due to the star vertex being snapped to its white edge.*

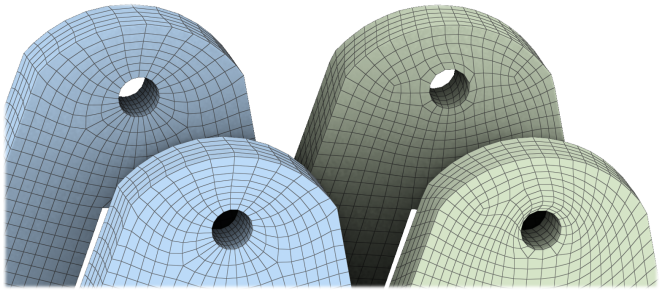


Figure 17: *A quad mesh (blue) extracted from a grid preserving map has the singularities of the input frame field. Introducing extra singularities [Pietroni et al.(2021)] (green) increases the robustness, at the expense of producing more complex structures.*

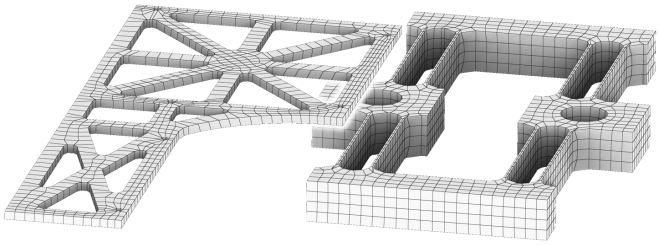


Figure 18: *Models with thin features.*

into four set of edges aligned in the map. A triangular mesh (as opposed to T-meshes) is a standard versatile representation with many off-the-shelf tools readily available.

To reembed or not? Moreover, keeping track of the matrix D is sufficient to represent the embedding in the original mesh in our case, whereas T-mesh embedding requires tracing paths (crossing triangles or following edges) on the surface.

Expressivity increase №1. Interestingly, with our decimated mesh, finding the integer degrees of freedom §4.2 is exactly the problem of the generation of a grid preserving map Prob. (5), where all variables would have to be Gaussian integer. It opens the possibility to keep and manage real valued variables in the decimated mesh for more expressivity.

Expressivity increase №2. Another way to broaden the class of quad meshes that can be represented by the decimated mesh is to edit it during the optimization of integer degrees of freedom. For example, if a triangle is to be degenerated by applying an atomic operation, it is often possible to make it valid by an edge flip (and updating D and ω accordingly); the coarse mesh is so versatile that is can be optimized at the same time as ω. Refer to Appendix B for an illustration.

Conclusion

This work demonstrates that quantization algorithms can work with a decimated mesh instead of a T-mesh. A first benefit is that the decimated mesh structure is more common, simpler to generate and to manipulate. The other advantage is that its edges are not necessary axis aligned in the map, making the structure more flexible. While we do not improve the state-of-the art robustness nor quality, we believe that a major contribution of this work is its aesthetic aspect, just like a more concise proof of a theorem in mathematics. Moreover, a simpler formulation opens the opportunity for future works to increase the complexity to reach new objectives.

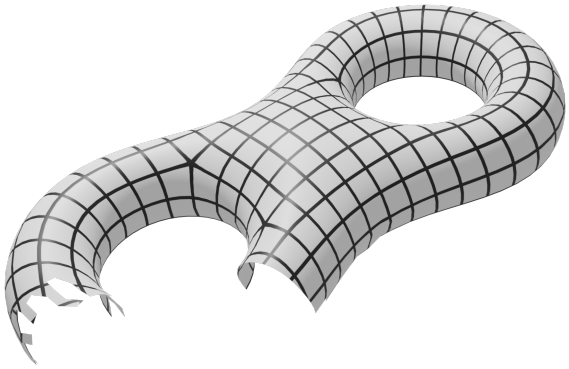


Figure 19: *Free boundary integer grid map obtained with our quantization algorithm.*

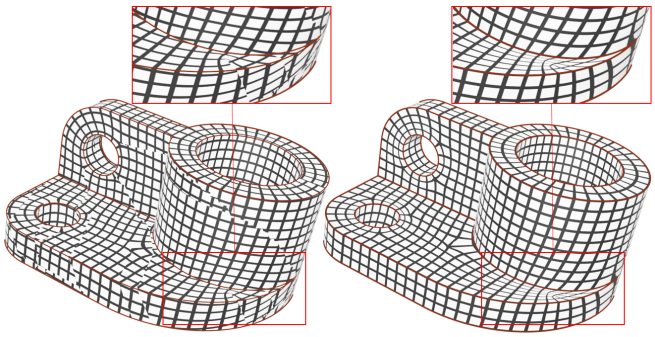


Figure 20: *Our method can take as input a seamless map that is not perfectly aligned with feature curves (left). These constraints are easier to enforce during the quantization step (right).*

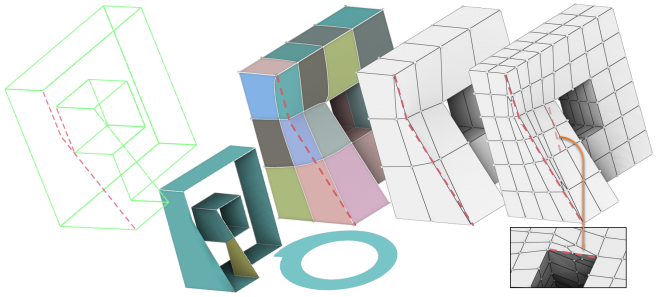


Figure 21: *This model have incompatible feature curves: two subsets of blue facets (bottom left) are topologically equivalent to the famous annulus failure case (bottom, Figure 5 of [Myles et al.(2014)]). We can discover conflicting feature curves at the quantization step (red dot lines), and remove them (middle left). A feature aligned mesh is then given by imprinting feature curves (middle right) and applying mid point subdivision (right).*

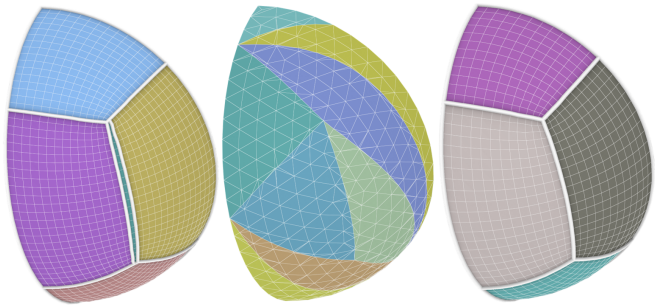


Figure 22: *Without degenerated charts, T-mesh quantization (left) cannot remove the thin chart to align singularities (right). Our decimated mesh (middle) does not need to degenerate facets to do it.*

Appendix A: Combinatorial optimization versus quantization

The intuition behind QGP and our quantization algorithm is to manipulate the final quad mesh combinatorial structure by applying series of atomic operations. Figure 8 illustrates the quad strip insertion, but atomic operations correspond to more general editing operations.

The T-mesh atomic operation is a simple chord insertion/deletion (Figure 23–blue) when applied to a T-mesh without T-junctions. When T-junctions are present, local drift of both sides of the cycle is possible (Figure 23–orange). In our case (on a decimated mesh), the atomic operation also allows to insert and delete quads in the same cycle (Figure 23–green). It exactly corresponds to the operation introduced in [Bommes et al.(2011)] to align singularities by removing q-helixes.

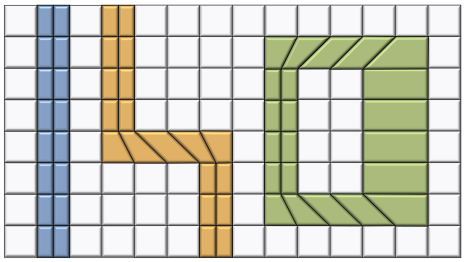


Figure 23: *Quad mesh editing applied on a grid: Chord insertion (blue), basic operation applied by T-mesh quantization can drift the chord insertion (orange), and our basic operation can also switch between chord insertion and removal. (green)*

Appendix B: Expressivity of a triangular mesh versus T-mesh

As illustrated on the toy example Figure 24, our decimated mesh can represent some maps where the T-mesh needs zero length arcs (third column) and even maps that T-mesh cannot represent without folds (last column). Conversely, T-meshes can represent maps that would be invalid in our (fixed connectivity) decimated mesh (see Figure 25). Note, however, that an edge flip during the U^- optimization phase would solve the problem, as proposed in § 6.4.

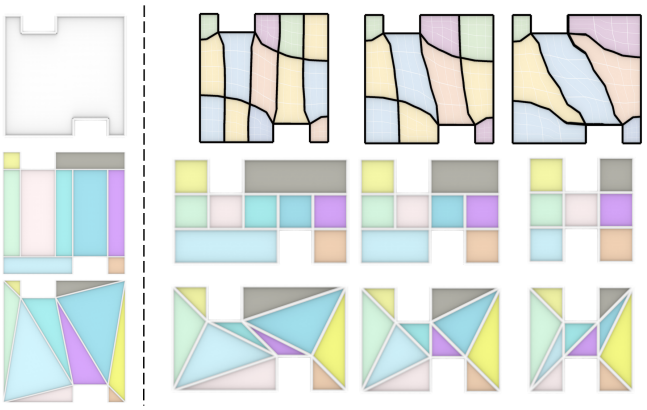


Figure 24: *Left side: a model (top) with a T-mesh proxy (middle) and a decimated mesh proxy (bottom). Right side: 3 possible block decompositions (top), the corresponding map on the T-mesh (middle) and on the decimated mesh (bottom). A quad of the T-mesh needs to be degenerate in the middle column, and inverted quads are required in the right column, whereas maps are always valid with our decimated mesh.*

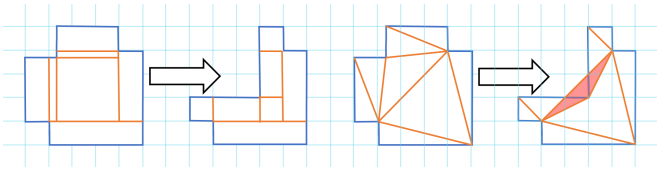


Figure 25: *A seamless map deformed into a grid preserving map: the map is correctly represented by the T-mesh decomposition (left), but the red triangle of the the decimated mesh (right) is invalid (negative Jacobian).*

References

[Alliez et al.(2002)] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. 2002. Interactive Geometry Remeshing. *ACM Trans. Graph.* 21, 3 (jul 2002), 347–354. https://doi.org/10.1145/566654.566588 2

[Añez et al.(1996)] J. Añez, T. De La Barra, and B. Pérez. 1996. Dual graph representation of transport networks. *Transportation Research Part B: Methodological* 30, 3 (1996), 209–216. https://doi.org/10.1016/0191-2615(95)00024-0 7

[Bommes et al.(2013)] David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.* 32, 4 (2013), 98:1–98:12. https://doi.org/10.1145/2461912.2462014 4, 6

[Bommes et al.(2011)] David Bommes, Timm Lempfer, and Leif Kobbelt. 2011. Global structure optimization of quadrilateral meshes. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 375–384. 12

[Bommes et al.(2009)] David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph.* 28, 3 (2009), 77. https://doi.org/10.1145/1531326.1531383 2, 3

[Bommes et al.(2012)] David Bommes, Henrik Zimmer, and Leif Kobbelt. 2012. Practical mixed-integer optimization for geometry processing. In *Proceedings of the 7th international conference on Curves and Surfaces* (Avignon, France). Springer-Verlag, Berlin, Heidelberg, 193–206. 3, 8

[Campen et al.(2015)] Marcel Campen, David Bommes, and Leif Kobbelt. 2015. Quantized global parametrization. *ACM Trans. Graph.* 34, 6 (2015), 192:1–192:12. https://doi.org/10.1145/2816795.2818140 4, 5, 9

[Desobry et al.(2021)] David Desobry, François Protais, Nicolas Ray, Etienne Corman, and Dmitry Sokolov. 2021. Frame Fields for CAD Models. In *Advances in Visual Computing - 16th International Symposium, ISVC 2021, Virtual Event, October 4-6, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 13018)*, George Bebis, Vassilis Athitsos, Tong Yan, Manfred Lau, Frederick Li, Conglei Shi, Xiaoru Yuan, Christos Mousas, and Gerd Bruder (Eds.). Springer, 421–434. https://doi.org/10.1007/978-3-030-90436-4_34 3

[Ebke et al.(2016)] 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, Article 218 (nov 2016), 13 pages. https://doi.org/10.1145/2980179.2982413 4

[Eppstein et al.(2008)] David Eppstein, Michael T. Goodrich, Ethan Kim, and Rasmus Tamstorf. 2008. Motorcycle Graphs: Canonical Quad Mesh Partitioning. *Computer Graphics Forum* (2008). https://doi.org/10.1111/j.1467-8659.2008.01288.x 4

[Garanzha et al.(2021)] Vladimir A. Garanzha, Igor E. Kaporin, Liudmila N. Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. 2021. Foldover-free maps in 50 lines of code. *CoRR* abs/2102.03069 (2021). arXiv:2102.03069 https://arxiv.org/abs/2102.03069 2, 8

[Hormann et al.(2007)] Kai Hormann, Bruno Lévy, and Alla Sheffer. 2007. Mesh Parameterization: Theory and Practice Video Files Associated with This Course Are Available from the Citation Page. In *ACM SIGGRAPH 2007 Courses* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 1–es. https://doi.org/10.1145/1281500.1281510 2

[Kälberer et al.(2007)] Felix Kälberer, Matthias Nieser, and Konrad Polthier. 2007. QuadCover - Surface Parameterization using Branched Coverings. *Comput. Graph. Forum* 26, 3 (2007), 375–384. https://doi.org/10.1111/j.1467-8659.2007.01060.x 2, 3, 7

[Koch et al.(2019)] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 8

[Ledoux(2019)] Franck Ledoux. 2019. Mambo dataset. https://gitlab.com/franck.ledoux/mambo 8

[Lyon et al.(2019)] 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 (jul 2019), 14 pages. https://doi.org/10.1145/3306346.3323019 4, 10

[Lyon et al.(2021a)] Max Lyon, Marcel Campen, and Leif Kobbelt. 2021a. Quad Layouts via Constrained T-Mesh Quantization. *Comput. Graph. Forum* 40, 2 (2021), 305–314. https://doi.org/10.1111/cgf.142634 4, 9

[Lyon et al.(2021b)] Max Lyon, Marcel Campen, and Leif Kobbelt. 2021b. Simpler Quad Layouts using Relaxed Singularities. *Computer Graphics Forum* 40, 5 (2021). 4

[Muller and Preparata(1978)] D.E. Muller and F.P. Preparata. 1978. Finding the intersection of two convex polyhedra. *Theoretical Computer Science* 7, 2 (1978), 217–236. https://doi.org/10.1016/0304-3975(78)90051-8 2

[Myles et al.(2014)] Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-Aligned Global Parametrization. *ACM Trans. Graph.* 33, 4, Article 135 (jul 2014), 14 pages. https://doi.org/10.1145/2601097.2601154 10, 12

[Noma et al.(2022)] Yuta Noma, Nobuyuki Umetani, and Yoshihiro Kawahara. 2022. Fast Editing of Singularities in Field-Aligned Stripe Patterns.

In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 37, 8 pages. https://doi.org/10.1145/3550469.3555387 7

[Pietroni et al.(2021)] Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. 2021. Reliable Feature-Line Driven Quad-Remeshing. *ACM Trans. Graph.* 40, 4, Article 155 (jul 2021), 17 pages. https://doi.org/10.1145/3450626.3459941 9, 10, 11

[Ray et al.(2006)] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic Global Parameterization. *ACM Trans. Graph.* 25, 4 (oct 2006), 1460–1485. https://doi.org/10.1145/1183287.1183297 2

[Reberol et al.(2021)] Maxence Reberol, Christos Georgiadis, and Jean-François Remacle. 2021. Quasi-structured quadrilateral meshing in Gmsh - a robust pipeline for complex CAD models. *CoRR* abs/2103.04652 (2021). arXiv:2103.04652 https://arxiv.org/abs/2103.04652 8

[Vaxman et al.(2016)] Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional Field Synthesis, Design, and Processing. *Comput. Graph. Forum* 35, 2 (2016), 545–572. https://doi.org/10.1111/cgf.12864 3

[Wang et al.(2022)] Shiyi Wang, Jingwen Ren, Xianzhong Fang, Hongwei Lin, Gang Xu, Hujun Bao, and Jin Huang. 2022. IGA-suitable planar parameterization with patch structure simplification of closed-form polysquare. *Computer Methods in Applied Mechanics and Engineering* 392 (2022), 114678. https://doi.org/10.1016/j.cma.2022.114678 3