Bézier Guarding: Precise Higher-Order Meshing of Curved 2D Domains

MANISH MANDAD and MARCEL CAMPEN, Osnabrück University, Germany

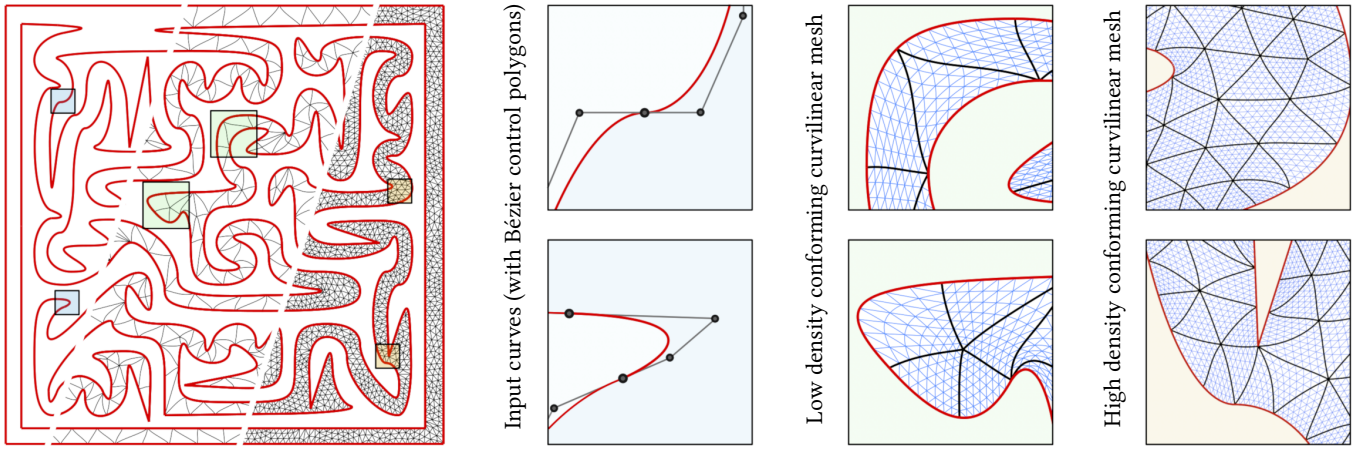

Fig. 1. Example planar domain with piecewise polynomial boundary curve (red). Our method constructs higher-order (here: cubic) curved triangle meshes (of controllable density) conforming to the prescribed curved boundary. The triangular elements (black edges) are provably regular, i.e., have injective geometric maps, and provably conform precisely to the domain's boundary. The blue blow-ups show the input curve's Bézier control polygons, while the green and orange blow-ups visualize the triangles' regular geometric maps using blue parametric iso-curves between their black and red edges.

We present a mesh generation algorithm for the curvilinear triangulation of planar domains with piecewise polynomial boundary. The resulting mesh consists of regular, injective higher-order triangular elements and precisely conforms with the domain's curved boundary. No smoothness requirements are imposed on the boundary. Prescribed piecewise polynomial curves in the interior, like material interfaces or feature curves, can be taken into account for precise interpolation by the resulting mesh's edges as well. In its core, the algorithm is based on a novel explicit construction of guaranteed injective Bézier triangles with certain edge curves and edge parametrizations prescribed. Due to the use of only rational arithmetic, the algorithm can optionally be performed using exact number types in practice, so as to provide robustness guarantees.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Mesh models**; **Mesh geometry models**; *Shape modeling*; • **Applied computing** → *Computer-aided design*; • **Mathematics of computing** → **Mesh generation**.

Additional Key Words and Phrases: Bézier triangle, Bézier simplex, curvilinear mesh, isogeometric analysis

Author's addresses: Manish Mandad, Marcel Campen, Institute for Computer Science, Osnabrück University, Germany.

ACM Reference Format:
Manish Mandad and Marcel Campen. 2020. Bézier Guarding: Precise Higher-Order Meshing of Curved 2D Domains. *ACM Trans. Graph.* 39, 4, Article 103 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392372

1 INTRODUCTION

The meshing of given domains using conforming triangular elements is a cornerstone in graphics, geometry processing, numerical simulation, and other fields. While often *linear* elements, triangles with straight edges, are used, the potentially significant benefits of higher-order polynomial elements with curved edges have been discussed and demonstrated [Babuška and Guo 1996; Hu et al. 2019; Oden 1994; Wang et al. 2013; Zlámal 1973]. The importance of accurate curved boundary conformance was studied as well, e.g. [Bassi and Rebay 1997; Ciarlet and Raviart 1972b; Luo et al. 2001].

We describe a method, based on a construction we call *Bézier guarding*, to generate higher-order polynomial triangle meshes with curved edges for arbitrary 2D domains with piecewise polynomial boundary curves, interface curves, and constraint curves—collectively referred to as *domain curves* in the following. We do not impose any smoothness requirements on these curves. The algorithm is general in that it supports arbitrary polynomial order.

To the best of our knowledge this method is the first to offer both of the following output properties in combination:

(1) the meshes precisely *conform* to (instead of approximate) the domain curves;
(2) the elements come with strictly *injective* and *polynomial* geometric maps.

The latter point guarantees that each curved element is the image of a straight-edge reference triangle under some injective polynomial map—and that this map is known explicitly. In other words,

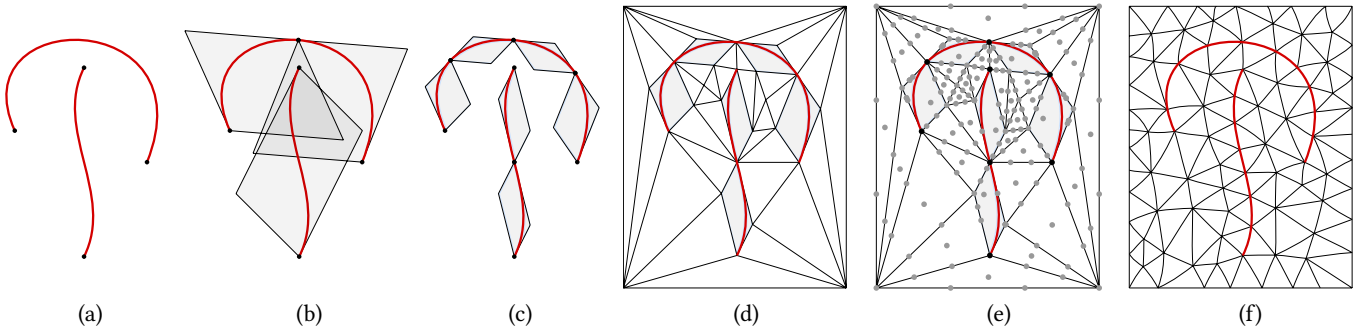

Fig. 2. Approach overview. Input curves (a) are covered from both sides by *guarding triangles* (b). Selective refinement yields disjoint guarding triangles (c). The rest of the domain is triangulated using straight-edge triangles (d). For each triangular element (straight or curved), Bézier control point positions defining regular geometric maps are constructed (e). Finally, the mesh can be optimized structurally and geometrically (f), preserving regularity and curve conformance.

each element can be represented as a planar Bézier triangle [Farin 1986] (or triangular Lagrange element) with strictly positive Jacobian determinant (referred to as *regular* herein). This often is an essential prerequisite for the use of these meshes in the context of isogeometric analysis and FEM [Barrett 1996; Mitchell et al. 1971]. This is a nontrivial request: even if all edges bounding a triangular region are formed by regular non-intersecting polynomial curves, such a map for the interior may not exist for a given order, cf. Fig. 3. If it does exist, there is no algorithm to provably find and construct it. Our method provides this map per element by construction.

In cases of non-polynomial domain curves, conformance obviously cannot be achieved using polynomial elements. Our method may still prove beneficial also in this scenario: while other methods approximate such generic domain curves as integral (not fully controllable) part of the meshing process, our method provides the opportunity of a priori approximation. One can prescribe precisely by which polynomial curves the curves shall be approximated in the mesh, which provides additional control and potential advantages depending on use case and specific domain knowledge.

1.1 Approach Overview

Given a set of domain curves (Fig. 2a), in a first stage our method covers these with a single layer of triangular elements per side (Fig. 2b). Each one of these elements has two straight edges and part of a domain curve as third edge. The number and distribution of these elements is adaptively chosen such that they do not intersect (Fig. 2c) and the geometric complexity of the curve part covered by an individual element is bounded.

In a second stage, the rest of the domain, which due to the straight edges of the covering elements is a straight-edge polygon with holes, is meshed with straight-edge elements via simple polygon triangulation (Fig. 2d).

Then, in a third stage, the triangular regions formed by the edges (some straight, some curved) are equipped with geometric maps, i.e., control points are constructed per element such that they define a regular Bézier triangle that precisely fits the edges (Fig. 2e). To this end we provide an explicit construction (Bézier guarding) that provably yields regular elements only. Continuity between adjacent elements is guaranteed by construction as well.

Finally, the generated valid curved mesh can be optimized geometrically and combinatorially through incremental remeshing in a manner that preserves its regularity, continuity, and conformance to the domain curves (Fig. 2f).

2 RELATED WORK

Methods tackling the problem of higher-order 2D mesh generation can be classified as indirect or direct [Dey et al. 1999]; we review the most relevant aspects and relations to our method in the following, with a focus on questions of injectivity and conformance.

2.1 Indirect Curved Meshing

Indirect approaches start by generating a mesh with linear straight-edge elements. Subsequently, some or all of these are incrementally curved with the goal of conforming to the domain curves. A variety of deformation models have been considered for this task [Abgrall et al. 2014; Fortunato and Persson 2016; Gargallo-Peiró et al. 2013; George and Borouchaki 2012; Hu et al. 2019; Luo et al. 2004; Moxey et al. 2016; Oliver 2008; Paul 2018; Persson and Peraire 2009; Poya et al. 2016; Roca et al. 2011; Ruiz-Gironés et al. 2016; Shephard et al. 2005; Sherwin and Peiró 2002; Toulorge et al. 2013; Turner et al. 2018; Xie et al. 2013; Xu and Chernikov 2014], some additionally interleaving mesh modification operators to increase robustness [Cardoze et al. 2004; Dey et al. 1999, 2001; Hu et al. 2019; Luo et al.

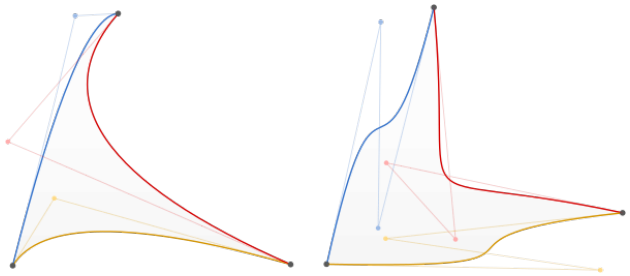

Fig. 3. Examples of planar triangles with curved polynomial edges (left: order 2; right: order 3; with Bézier control polygons) for which no injective geometric map of the same order exists. In other words: no regular Bézier triangle (of order 2 or 3, respectively) has these curves as its edges.

2004; Shephard et al. 2005]. In part due to the non-convexity of these deformation formulations, guarantees in the sense that eventually conformance will be achieved are unavailable.

Other methods employ some form of projection or replacement of initially straight edges by curves [Dey et al. 1999, 2001; Engvall and Evans 2016; Jaxon and Qian 2014; Rangarajan and Lew 2014]. While this, for some approaches, can yield elements formed by intersection-free curved edges, it does not generally guarantee the existence (or knowledge) of injective polynomial geometric maps of the desired order, cf. Fig. 3.

2.2 Direct Curved Meshing

Direct methods create elements with curved edges right away. A common approach is to proceed in an advancing front manner, creating elements along domain curves first. Those variants aiming to ensure element regularity commonly employ non-polynomial geometric maps, e.g., based on transfinite interpolation [Gordon and Hall 1973; Haber et al. 1981; Mansfield 1978; Zlámal 1973], or impose smoothness assumptions on the curves [Ciarlet and Raviart 1972a; Rangarajan and Lew 2014]. Often the construction of single elements is considered; a complete mesh generation strategy in its entirety is discussed in a few works only. An example is [Sevilla et al. 2016]; again non-polynomial geometric maps are obtained, targeting a non-standard FEM approach. The question of how structural and geometric optimization of the initially generated meshes could be performed in a regularity and conformance preserving manner in these particular settings is typically not addressed.

By contrast, our method yields standard polynomial elements (which can be preferable [Solin et al. 2003, §3.3]), it supports arbitrary order, we describe the generation process for an entire mesh, and we show that mesh optimization, preserving regularity and conformance, can practically be performed in this setting.

An alternative strategy for triangles with no regular polynomial map available is to initialize with irregular polynomial maps, and subsequently try to *untangle* the mesh [Toulorge et al. 2013, 2016], i.e., optimize the maps and the mesh's geometry (or even structure) with the goal of increasing the Jacobian determinant where it is negative. Existing methods for this purpose, however, do not provide guarantees of achieving both regularity and conformance.

2.3 Non-Conforming Meshing

Depending on the use case, it may be possible to work with meshes that do not conform to the domain boundary or interface curves, and therefore may use simpler elements—even Cartesian grids. Examples are cut cell methods [Burman and Hansbo 2010] and X-FEM [Belytschko and Black 1999; Fries and Belytschko 2010]. It has been noted that working with higher-order sub-elements can be of benefit in this context as well [Cheng and Fries 2010].

2.4 Injectivity

A variety of algorithms have been described to *test* whether a given higher-order triangular element is regular, i.e. whether its geometric map is injective [Dey et al. 1999; George and Borouchaki 2012; Gravesen et al. 2014; Hernandez-Mederos et al. 2006; Johnen et al. 2012; Luo et al. 2002]. We, by contrast, aim to *construct* elements for which this is guaranteed by construction.

A conservative injectivity test of the above kind is employed in our subsequent mesh optimization stage in order to guarantee regularity preservation.

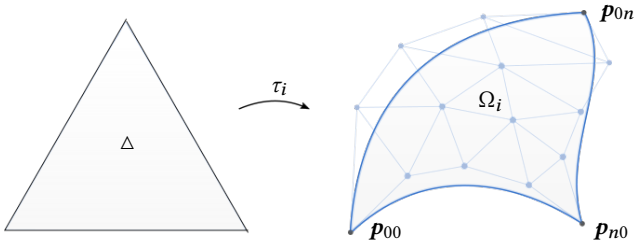

Fig. 4. Map τ_i from reference triangle $\triangle$ to planar three-sided domain Ω_i with curved boundary edges, represented in the Bernstein basis, i.e., as a (here quartic) 2D Bézier triangle.

3 CONFORMING CURVED TRIANGULATION

The high-level goal of the method described in this section is to construct a planar curvilinear mesh of triangular elements t_i of arbitrary polynomial order n that conformingly partitions a given curvilinear domain formed by domain curves of order $\leq n$. This entails partitioning the domain into curved three-sided regions Ω_i such that for each region a regular *geometric map* $\tau_i : \triangle \to \Omega_i$ can be defined. Here $\triangle$ is an abstract reference triangle, also referred to as parameter space, while Ω_i is also referred to as physical space. The map is called *regular* iff $\det J_{\tau_i} \neq 0$ everywhere, where J_{τ_i} is the map's Jacobian. Regularity implies that the map is (locally) injective—an essential prerequisite, e.g., for finite element techniques [Mitchell et al. 1971; Solin et al. 2003].

We represent and define the map τ_i (and thus $\Omega_i = \tau_i(\triangle)$) in the Bernstein basis, i.e. as a Bézier triangle [Farin 1986]. A 2D Bézier triangle of order n is specified by control points $\boldsymbol{p}_{ijk} \in \mathbb{R}^2$ with $i, j, k \geq 0$ and $i + j + k = n$. For brevity we omit the third (implied) index $k = n - i - j$ in the following. Fig. 4 shows an example.

From an operational point of view, our algorithm does not require heavy machinery. Besides primitive geometric operations (like line intersections and vector algebra) the only non-trivial ingredients are well-known standard algorithms, namely:

- triangle-triangle intersection test
- polygon triangulation (with holes)
- Bézier curve bisection (de Casteljau)

3.1 Input

The input to our method is a set C of prescribed polynomial curves $\boldsymbol{c}_i : [t_{i,0}, t_{i,1}] \to \mathbb{R}^2$ of arbitrary order, called domain curves. Without loss of generality, we may assume $[t_{i,0}, t_{i,1}] = [0, 1]$ in the following to keep the exposition simple. The (directed) unit tangent vector of a curve $\boldsymbol{c}_i$ is denoted $\boldsymbol{c}_i^{\mathrm{t}} : [0, 1] \to S^1$, with $\boldsymbol{c}_i^{\mathrm{t}} = \boldsymbol{c}_i' / \|\boldsymbol{c}_i'\|$.

The following is assumed about the set C of domain curves (Fig. 5):

(a) **No Irregularity:** curves are regular, i.e., $\|\boldsymbol{c}_i'(t)\| \neq 0, \forall t \in [0, 1]$.

(b) **No Degeneracy:** at coincident end points, two curves do not form an angle-zero corner, i.e., $\boldsymbol{c}_i(s) = \boldsymbol{c}_j(t) \Rightarrow \boldsymbol{c}_i^{\mathrm{t}}(s) \neq \pm \boldsymbol{c}_j^{\mathrm{t}}(t)$ ('+' if $s = t$, '−' otherwise), for $s, t \in \{0, 1\}$.

(c) **No Intersection:** curves only (self-)intersect at their end points, i.e., $\boldsymbol{c}_i(s) \neq \boldsymbol{c}_j(t)\ \forall s \in [0,1], t \in (0,1), (i \neq j \vee s \neq t)$.

Assumption (a) is a fundamental requirement (not specific to our method): if a curve is irregular it cannot, in general, be the boundary curve of a regular Bézier triangle. In certain cases a regular reparametrization, i.e., a regular curve of the same order with the same image, may exist, but this is not generally the case.

Assumption (b) likewise is a fundamental requirement: the corner formed by two such curves (or one closed curve) cannot be the corner of a regular Bézier triangle; the geometric map's Jacobian determinant would vanish at the corner point.

Assumption (c) is to keep the exposition focused. In case it is not satisfied by a given set of domain curves, a satisfying set can be derived by splitting violating curves at their intersection points [Zukerman et al. 2019]. As intersection points may not be numerically representable without error, approximation techniques for consistent rounding to limited precision numbers have been proposed [Eigenwillig et al. 2007].

3.2 Bézier Guarding

The goal of the algorithm described in the following is to generate a set of curvilinear triangular elements covering both sides of an individual domain curve in a conforming way. As a first step, curves are (virtually) split into *guardable* sub-curves (cf. Fig. 6) via (repeated) bisection, i.e., subdivision at $t = 1/2$ [Farin 2002, §5.4].

Definition 3.1 (Guardable Curve). Let $\boldsymbol{p}_i$, $i \in \{0, \dots, n\}$ be the Bézier control points of an order n curve $\boldsymbol{c}$, forming the control polygon $P_{\boldsymbol{c}}$. The *control vectors* of $P_{\boldsymbol{c}}$ are $\boldsymbol{s}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$. We call the curve $\boldsymbol{c}$ *guardable* iff there exists a direction $\boldsymbol{d}$ such that $\boldsymbol{d}^{\mathsf{T}}\boldsymbol{s}_i > 0$ for all i.

Note that this implies that for each side of the Bézier control polygon $P_{\boldsymbol{c}}$ there is a direction (a point at infinity) from which the entire side is visible. As a side note, by implication of the hodograph property of Bézier curves [Sederberg and Meyers 1988] this furthermore implies (but is not equivalent to) the entire curve being visible from these two directions (one per side).

Proposition 3.2. *Repeated bisection of a non-guardable curve eventually yields sub-curves that all are guardable.*

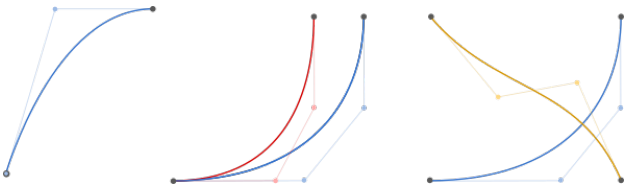

Fig. 5. Curve configurations excluded by the input assumptions (Sec. 3.1). Left: irregular cubic curve with vanishing derivative (first and second Bézier control points are coincident). Center: two curves forming an angle-zero corner (due to coinciding tangents). Right: two curves intersecting.

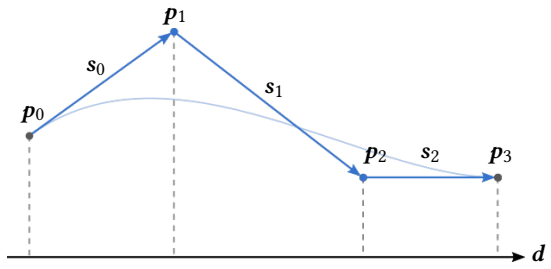

Fig. 6. A *guardable* cubic curve (Def. 3.1) with its three Bézier control vectors $\boldsymbol{s}_i$ and their projection onto an *axis* $\boldsymbol{d}$ of the curve (Def. 3.4).

Proof. This follows from the convergence of control polygons to a flat state, i.e., parallel control vectors $\boldsymbol{s}_i$, under curve subdivision [Li et al. 2012; Morin and Goldman 2001]. □

For each side of a guardable curve $\boldsymbol{c}$, with endpoints $\boldsymbol{p}_0$ and $\boldsymbol{p}_n$, we now construct a guard point $\boldsymbol{o}$. This guard is chosen such that together with curve $\boldsymbol{c}$, the straight line segments $\overline{\boldsymbol{p}_0\boldsymbol{o}}$ and $\overline{\boldsymbol{p}_n\boldsymbol{o}}$ form a triangular region (with one curved edge and two straight edges), and that this region admits a regular geometric map of the same order as $\boldsymbol{c}$. The explicit construction of this map is detailed in Sec. 3.3.

Definition 3.3 (Guarding Triangle). For a side of a curve $\boldsymbol{c}$ with guard $\boldsymbol{o}$, the triangular region formed by $\boldsymbol{c}$, $\overline{\boldsymbol{p}_0\boldsymbol{o}}$, and $\overline{\boldsymbol{p}_n\boldsymbol{o}}$ is called this side's *guarding triangle* (see Fig. 7).

Definition 3.4 (Curve Axis). A direction $\boldsymbol{d}$ for which $\boldsymbol{d}^{\mathsf{T}}\boldsymbol{s}_i > 0$ for all Bézier control vectors $\boldsymbol{s}_i$ of a guardable curve $\boldsymbol{c}$, we call an *axis* of the curve.

For some curve axis $\boldsymbol{d}$, let $\boldsymbol{s}^+$ be the control vector minimizing $\boldsymbol{d}^{\mathsf{T}}\boldsymbol{s}_i$ that points counterclockwise relative to $\boldsymbol{d}$, and $\boldsymbol{s}^-$ the minimizer that points clockwise. Consider a line L^+ through curve endpoint $\boldsymbol{p}_0$ in direction $\boldsymbol{s}^+$, and a line L^- through curve endpoint $\boldsymbol{p}_n$ in direction $\boldsymbol{s}^-$. Let $\boldsymbol{x}$ be the intersection point of L^+ and L^-. If both lines are parallel, this implies the curve is straight and $L^+ = L^-$; in this case the intersection point is not unique and we let $\boldsymbol{x}$ be the mid point of $\boldsymbol{p}_0$ and $\boldsymbol{p}_n$. We define $w = \|\boldsymbol{p}_0 - \boldsymbol{p}_n\|$ as the *width* of the curve. The guard $\boldsymbol{o}_l$ of the left curve side is defined as $\boldsymbol{o}_l = \boldsymbol{x} + \mu\,(w^2/\hat{w})\boldsymbol{n}$, where $\boldsymbol{n}$ is a unit vector perpendicular to and counterclockwise of $\boldsymbol{d}$, and $\mu > 0$ is a parameter. The normalization factor $\hat{w}$ denotes the curve's *initial* width—it remains constant when the curve is subdivided (in step (2) of the mesh generation algorithm of Sec. 3.4).[1]

The parameter μ allows for a trade-off: smaller values lead to triangles that hug the domain curves more tightly (leading to quicker termination of the meshing algorithm as a whole, cf. Sec. 3.4) but that at the same time commonly have higher initial map distortion. The construction is correct for any positive choice $\mu > 0$; our default choice of $\mu = 10^{-2}$ is justified in Sec. 5.6.

Analogously, a guard $\boldsymbol{o}_r$ for the curve's right side is obtained by applying the construction to the reverse curve. In this way a guarding triangle for each curve side is defined.

[1] The amount of translation in normal direction being relative to $\|\boldsymbol{p}_0 - \boldsymbol{p}_n\|$ (or another curve size dependent measure), and depending superlinearly (w^2) on this measure, is required to ensure termination of the meshing algorithm as a whole, cf. Lemma 3.7.

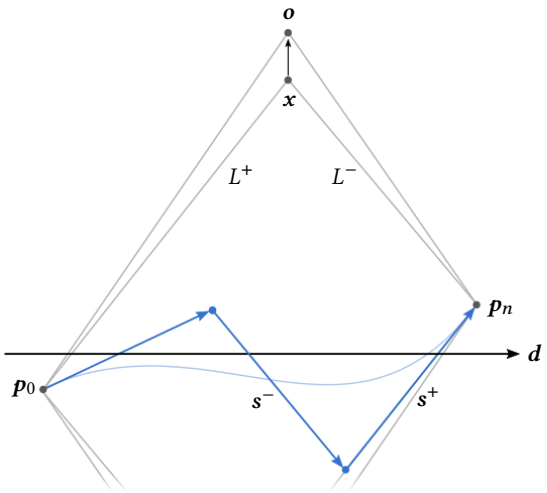

Fig. 7. Construction of a *guarding triangle* (Def. 3.3) using guard $\boldsymbol{o}$. Point $\boldsymbol{x}$ is the intersection of lines L^+, L^- parallel to the control vectors $\boldsymbol{s}^+$, $\boldsymbol{s}^-$ that are the steepest (relative to the curve's axis $\boldsymbol{d}$).

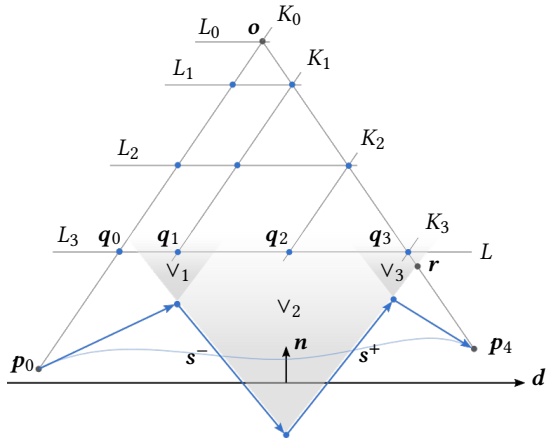

Fig. 8. Construction of a geometric map for a guarding triangle (Sec. 3.3.1), exemplarily for a quartic curve. The cones $\vee_i$ are shown in light grey. In this example the points $\boldsymbol{q}_i$, $0 < i < n-1$, were chosen as the center of the respective cone segment. The blue points, together with the three black corner points, form control points of a regular Bézier triangle.

Definition 3.5 (Curve Envelope). The quadrilateral formed by the four straight edges of the two guarding triangles of a curve $\boldsymbol{c}$, i.e., spanned by endpoints $\boldsymbol{p}_0$, $\boldsymbol{p}_n$ and guards $\boldsymbol{o}_l$, $\boldsymbol{o}_r$, is called the *envelope* $E(\boldsymbol{c})$.

By construction, $\boldsymbol{c}$ is contained in $E(\boldsymbol{c})$, and only the curve's endpoints lie on the boundary $\partial E(\boldsymbol{c})$.

Any axis $\boldsymbol{d}$ according to Def. 3.4 can be chosen for the above guard and envelope construction. In order to yield small envelopes on average it is advisable to symmetrically choose the bisector direction of $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$, i.e., $\boldsymbol{d} = \frac{\boldsymbol{s}^+}{\|\boldsymbol{s}^+\|} + \frac{\boldsymbol{s}^-}{\|\boldsymbol{s}^-\|}$.

3.3 Geometric Maps

3.3.1 Curved Elements. We now show how control points for a regular Bézier triangle of order n to serve as geometric map for a guarding triangle above a curve $\boldsymbol{c}$ of order n can be constructed. As before, $\boldsymbol{d}$ denotes the curve axis direction used in the guard points' construction, and $\boldsymbol{n}$ the corresponding normal. Fig. 8 illustrates the construction.

Let $\vee_i$ be a cone with apex $\boldsymbol{p}_i$, bounded by the minimizing directions $-\boldsymbol{s}^-$ and $\boldsymbol{s}^+$ (cf. Sec. 3.2). Note that $\vee_i$ contains $\boldsymbol{n}$. Let $\boldsymbol{r}$ be the intersection point of the cone's boundary $\partial\vee_{n-1}$ and $\overline{\boldsymbol{p}_n\boldsymbol{o}}$.

Consider a line L in axis-direction $\boldsymbol{d}$, with normal $\boldsymbol{n}$, such that it separates the guard $\boldsymbol{o}$ from control polygon $P_{\boldsymbol{c}}$ and $\boldsymbol{r}$. Then L has one intersection point with each of the guarding triangle's straight edges; denote these $\boldsymbol{q}_0$ and $\boldsymbol{q}_{n-1}$. On this line we choose $n-2$ further distinct points $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{n-2}$, ordered from $\boldsymbol{q}_0$ to $\boldsymbol{q}_{n-1}$.

These points are chosen such that $\boldsymbol{q}_i$, $1 \le i \le n-2$ lies in the interior of the cone $\vee_i$. One easily verifies that, by construction, such a choice is always possible: the in-cone positions for $\boldsymbol{q}_i$ on line L are in the intersection segment $S_i = L \cap \vee_i$. Let $\ell(t)$ be a linear parametrization of L, with $\ell(0) = \boldsymbol{q}_0$ and $\ell(1) = \boldsymbol{q}_{n-1}$; then segments S_i correspond to parameter intervals $[l_i, r_i]$ on ℓ. The segments are ordered along L in the following sense: $l_i \le l_{i+1}$ and $r_i \le r_{i+1}$, $0 < r_1$ and $l_{n-2} < 1$. One possible valid construction therefore is to define $\boldsymbol{q}_i = \ell(t_i)$ with $t_0 = 0$, $t_i = \frac{1}{2}(\max(t_{i-1}, l_i), \min(r_i, 1))$, incrementally from $i = 1$ to $n-2$. Intuitively, point $\boldsymbol{q}_i$ is put in the center of its valid segment S_i clamped from below by the previous point $\boldsymbol{q}_{i-1}$ and clamped from above by the last point $\boldsymbol{q}_{n-1}$.

To conclude the construction, we define the control points as intersection points (cf. Fig. 8) of the following lines for $0 \le i < n$:

- line K_i parallel to $\overline{\boldsymbol{p}_0\boldsymbol{o}}$, through point $\boldsymbol{q}_i$.
- line L_i parallel to L, through point $\boldsymbol{h}_i = \text{intersect}(K_i, \overline{\boldsymbol{p}_n\boldsymbol{o}})$.

Let $\boldsymbol{x}_{i,j} = \text{intersect}(K_i, L_j)$. Note that $\boldsymbol{x}_{i,n-1} = \boldsymbol{q}_i$ and $\boldsymbol{x}_{0,0} = \boldsymbol{o}$.

The Bézier triangle control points are denoted $\boldsymbol{p}_{ij}$, $i, j \ge 0$, $i+j \le n$. We define them as

$$\boldsymbol{p}_{ij} = \begin{cases} \boldsymbol{p}_i & j = 0 \\ \boldsymbol{x}_{i,n-j} & j > 0 \end{cases}$$

In Sec. 3.6 we prove that these control points define a regular Bézier triangle in any case.

3.3.2 Straight Elements. For step (5) of the algorithm in Sec. 3.4 we furthermore need to be able to construct regular Bézier elements for straight-edge triangles—with control points along edges prescribed in a certain manner to ensure C^0-continuity across edges. On edges adjacent to a guarding triangle, we adopt the edge control points constructed for this guarding triangle; on edges adjacent to another straight-edge triangle, we prescribe a linear map (i.e., uniformly distributed edge control points). Due to the manner of triangulation in step (3) (with splitting of certain triangles) each straight-edge triangle is adjacent to at most one guarding triangle.

Given a straight-edge triangle with corner points $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, and at most edge $(\boldsymbol{a}, \boldsymbol{b})$ with non-uniform prescribed control points $(\boldsymbol{p}_0, \ldots, \boldsymbol{p}_n$ in strictly monotone order along the edge, with $\boldsymbol{p}_0 = \boldsymbol{a}$ and $\boldsymbol{p}_n = \boldsymbol{b})$, we define Bézier triangle control points, with $k = n - i - j$, as

$$\boldsymbol{p}_{ij} = \begin{cases} \boldsymbol{p}_i & j = 0 \\ \frac{k}{n}\boldsymbol{a} + \frac{i}{n}\boldsymbol{b} + \frac{j}{n}\boldsymbol{c} & j > 0 \end{cases}$$

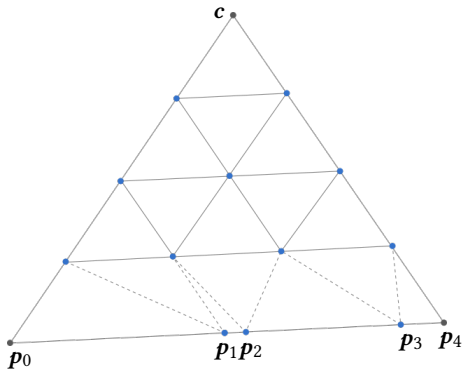

Fig. 9. Regular Bézier control points for a straight-edge triangle with non-uniform (but monotone) control point distribution along one edge (Sec. 3.3.2).

These control points are illustrated in Fig. 9. Regularity of this construction is shown in Sec. 3.6.

3.4 Complete Meshing Algorithm

Given the input set of domain curves, the following steps are executed. The output is a curve conforming mesh of regular higher-order triangles.

(1) While there is a non-guardable curve $\boldsymbol{c}$, bisect it, i.e., replace it by two sub-curves, each reparametrized to $t \in [0, 1]$.
(2) While there is a pair $\boldsymbol{c}_i, \boldsymbol{c}_j$ of (guardable) curves whose envelopes $E(\boldsymbol{c}_i), E(\boldsymbol{c}_j)$ intersect except at the curves' endpoints, bisect the one with larger envelope (Fig. 10).
(3) Triangulate a bounding polygon (e.g., an axis-aligned rectangle) encompassing all envelopes that has all envelopes as holes. Split triangles that are edge-adjacent to two guarding triangles.
(4) Construct Bézier control points defining a regular geometric map per guarding triangle (Sec. 3.3.1).
(5) Construct Bézier control points defining a regular geometric map per non-guarding (straight-edge) triangle (Sec. 3.3.2).

Remark (envelope size): In step (2), any notion of envelope size can be used, as long as it is positive and converges to 0 under curve bisection, cf. Lemma 3.7. Choosing the envelope's *height*, defined as $|\boldsymbol{n}^\mathsf{T}(\boldsymbol{o}_l - \boldsymbol{o}_r)|$, proved to be a good heuristic to minimize the total number of bisections.

Remark (closed domain): In the case that the input boundary curves form a closed domain Ω, one can easily restrict the algorithm to this domain (instead of a bounding box). To this end, one simply excludes the outer sides of boundary curves from the algorithm, i.e., boundary curves have one-sided envelopes in step (2), and triangulation is applied to the polygon $\Omega \setminus \cup_i E(\boldsymbol{c}_i)$ in step (3).

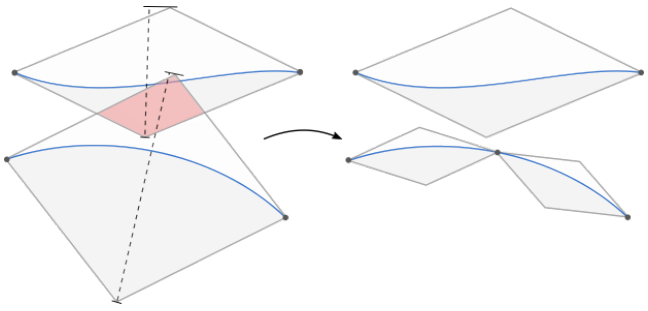

Fig. 10. Illustration of algorithm step (2): two curves' envelopes are intersecting (red). Here bisecting one curve (the one with larger envelope height, dashed) resolves the conflict; in general repeated bisection may be required.

3.5 Proof of Termination

Termination of (1) follows directly from Prop. 3.2. Termination of (2) is shown using the following proposition.

Lemma 3.6. *Bisection of a guardable curve yields guardable curves.*

Proof. The control vectors of the sub-curves are convex combinations of the control vectors of $\boldsymbol{c}$ [Farin 2002, §5.4]. Therefore, if each side of $P_{\boldsymbol{c}}$ is visible from some directions, so are the sub-curves' control polygons' sides. □

Lemma 3.7. *For a guardable curve $\boldsymbol{c}$, let $H(\boldsymbol{c}) = \partial E(\boldsymbol{c}) \setminus \{\boldsymbol{p}_0, \boldsymbol{p}_n\}$. $H(\boldsymbol{c})$ converges to the interior of $\boldsymbol{c}$ under bisection, and the two inner angles of $E(\boldsymbol{c})$ at $\boldsymbol{p}_0$ and $\boldsymbol{p}_n$ converge to 0.*

Proof. Under subdivision of $\boldsymbol{c}$, $P_{\boldsymbol{c}}$ converges to that curve in a pointwise manner (with respect to a uniform parametrization) [Prautzsch and Kobbelt 1994]. Furthermore, the control polygon of an individual sub-curve resulting from subdivision converges to a straight line segment, the control vectors $\boldsymbol{s}_i$ converge to all be parallel [Li et al. 2012; Morin and Goldman 2001]. Hence, the intersection points $\boldsymbol{x}$ (cf. Sec. 3.2) converge to the curve, in particular to the center of the sub-curve. Points $\boldsymbol{x}$ are well-defined because the sub-curves remain guardable (Lemma. 3.6).

At the same time, the width w of a curve converges to 0 under subdivision. Hence, the guards $\boldsymbol{o}$ (defined as $\boldsymbol{x}+\mu\,(w^2/\hat{w})\boldsymbol{n}$) converge to $\boldsymbol{x}$, thus to the center of the sub-curve as well. Therefore the union of $\overline{\boldsymbol{p}_0\boldsymbol{o}}$ and $\overline{\boldsymbol{o}\boldsymbol{p}_n}$ converges to the curve.

It follows that the envelope boundary without the curve's endpoints, $H(\boldsymbol{c})$, converges to the interior of the curve.

As the convergence of $\boldsymbol{o}$ to $\boldsymbol{x}$ is asymptotically faster (quadratic in w) than the convergence of the curve width to 0 (linear in w), the inner angles of $E(\boldsymbol{c})$ at $\boldsymbol{p}_0$ and $\boldsymbol{p}_n$ converge to 0. □

Fig. 11 illustrates this behavior.

Proposition 3.8. *Step (2) of the algorithm from Sec. 3.4 terminates.*

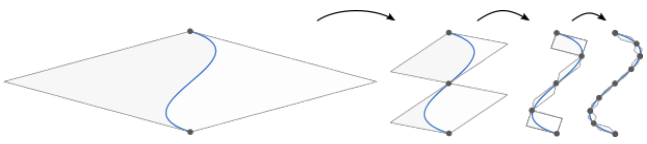

Fig. 11. Envelopes converging to a curve (blue) under repeated bisection.

Proof. Input curves do not intersect except at their endpoints by assumption (c), and bisection in steps (1) and (2) does not change that. Due to Lemma 3.7, under sufficient (finite) subdivision the envelopes of two *non-intersecting* curves do not intersect. For two general curves *intersecting* at their endpoints, this could be the case in the limit only: if they form a zero angle, no finite number of subdivisions will suffice. Due to assumption (b), however, intersecting curve pairs form non-zero angles only, such that due to Lemma 3.7 under sufficient (finite) subdivision non-intersecting envelopes result.

Assume for a moment that in step (2) both curves, $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, were bisected if their envelopes intersect. For this variant it is obvious that eventually sufficient subdivision will be achieved. In case, however, one chooses to only bisect one of the two curves in conflict, care must be taken. If the same of two curves in conflict, say $\boldsymbol{c}_i$, was constantly chosen, this could lead to an infinite recursion (if $E(\boldsymbol{c}_j)$ intersects the curve $\boldsymbol{c}_i$ itself, not just $E(\boldsymbol{c}_i)$). By assigning to curves some positive measure that converges to 0 under bisection (like the envelope height, cf. Sec. 3.4; alternatives are the curve width or the envelope's area or circumference) and always choosing the one with larger measure value, one guarantees that both conflicting curves will eventually be bisected until the conflict is resolved. □

3.6 Proof of Regularity

Let $\boldsymbol{p}_{ij} \in \mathbb{R}^2$, $i, j \geq 0$, $i + j \leq n$, be the control points of a Bézier triangle τ. The vectors $\Delta^0_{ij} = \boldsymbol{p}_{(i+1)j} - \boldsymbol{p}_{ij}$, $i, j \geq 0$, $i + j \leq n - 1$, we call the 0-vectors and $\Delta^1_{ij} = \boldsymbol{p}_{i(j+1)} - \boldsymbol{p}_{ij}$ the 1-vectors of τ. Fig. 12 illustrates these vectors.

Proposition 3.9. *If all 0-vectors of a 2D Bézier triangle τ are contained in a sector $r_0 \subset S^1$, all 1-vectors in a sector $r_1 \subset S^1$, $r_0 \cap r_1 = \emptyset$, and both sectors are contained in the interior of a common sector r of angle π, then τ is regular.*

Proof. The Jacobian of map τ is $J_\tau = n[\sum \Delta^0_{ij} B^{n-1}_{ij}, \sum \Delta^1_{ij} B^{n-1}_{ij}]$, where the B^{n-1}_{ij} are triangular Bernstein polynomials [Farin 1986]. Due to their non-negativity, $\sum \Delta^{0/1}_{ij} B^{n-1}_{ij}$ is a convex combination of the 0/1-vectors, and therefore contained in the sector r_0 or r_1, respectively. The counterclockwise angle θ from the Jacobian's first to its second column vector is thus $0 < \theta < \pi$, thus $\det J_\tau > 0$. □

A variant of this, with less direct proof, was noted in [Vavasis 2003].

The curved Bézier triangle from Sec. 3.3.1 satisfies the premise of the proposition: Each 0-vector is either one of the curve's control vectors $\boldsymbol{s}_i$ or parallel to the chosen curve axis $\boldsymbol{d}$. They are all contained in a sector spanned by $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$ (with a angle $\alpha < \pi$ by the premise that the curve is guardable (i.e., $\boldsymbol{d}^\mathsf{T} \boldsymbol{s}^i > 0$). Each 1-vector is one of the vectors $\boldsymbol{q}_i - \boldsymbol{p}_i$ with $0 \leq i \leq n - 1$, or parallel to $\boldsymbol{q}_0 - \boldsymbol{p}_0$. By construction (strict containment in cones $\vee_i$, cf. Sec. 3.3.1) these vectors are contained in the *interior* of the sector spanned by $\boldsymbol{s}^+$ and $-\boldsymbol{s}^-$ (with angle $\beta = \pi - \alpha$).

For the straight-edge triangle from Sec. 3.3.2 the case is particularly simple: all 0-vectors are parallel to $\boldsymbol{b} - \boldsymbol{a}$ while all 1-vectors are not parallel to $\boldsymbol{b} - \boldsymbol{a}$ and pointing to the left of it by construction.

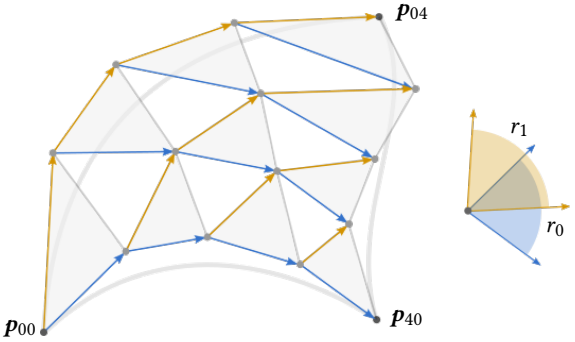

Fig. 12. 0-vectors (blue) and 1-vectors (yellow) of a quartic 2D Bézier triangle's control net. On the right the minimal sectors r_0 and r_1 they are contained in are shown; in this example the sectors are not disjoint, so Prop. 3.9 cannot be used to show regularity in this case.

3.7 Numerical Considerations

When, instead of the usual 2-norm, employing the 1-norm (for curve width and axis computation, normal vector normalization) all arithmetic operations in our algorithm are rational. As a consequence, the algorithm can be carried out using exact rational number types, e.g. [Granlund et al. 2019], to guarantee success and valid output not just in theory but for an actual implementation.

Of course in practice, using standard (double precision) floating point numbers can be desirable, for speed and for interoperability (when it comes to representing the method's output for further use). One option is to use exact predicates only (for envelope intersection tests, polygon triangulation) but inexact constructions, i.e., standard floating point numbers represent control and guard points. Possible numerical issues in extreme cases are: curve bisection may introduce numerical error (in the computed sub-curve control points) that may lead to sub-curves that violate one of the three assumptions stated in Sec. 3.1, and control points computed for the Bézier triangles may be numerically coincident. However, even such challenging configurations as the very sharp corner in Fig. 19 are still handled correctly by an implementation following this principle.

Another option is executing the entire algorithm with exact constructions and only in the end converting (e.g., rounding) the output to standard floating point numbers. Performing the conversion in such a way that regularity is provably preserved in any case is a challenge: already in the linear case the analogous structure preserving geometric rounding problem is an (NP-complete) issue [Milenkovic and Nackman 1990]. Note that these are fundamental and general issues not particular to our method.

4 MESH OPTIMIZATION

The mesh generated by our method described in the previous section conforms with the prescribed domain curves, and its triangular elements are regular by construction. The quality of this mesh, in terms of element shape, geometric map distortion, and grading, is uncontrolled, cf. Fig. 2e. We can now apply geometrical as well as combinatorial mesh optimization operators to improve these gradual quality aspects, while strictly preserving the established hard validity properties of regularity and conformance.

For the purpose of mesh optimization we make use of previously proposed techniques, following a mesh improvement strategy like the one recently applied in [Hu et al. 2019, 2018]. It is based on two principles, for combinatorial and geometrical mesh improvement.

- On the combinatorial side, local mesh modification operators (edge flips, edge splits, edge collapses) are applied, which have already proven their effectiveness in linear remeshing scenarios [Botsch and Kobbelt 2004; Freitag and Ollivier-Gooch 1997; Hoppe et al. 1993; Kobbelt et al. 2000].
- On the geometrical side, a continuous optimization of the vertex and control point positions is performed, directly driven by a distortion objective evaluated on the triangles' geometric maps [Fu et al. 2015; Hormann and Greiner 2000]. Preservation of regularity is ensured by explicitly testing for violations [Hernandez-Mederos et al. 2006] in the optimization's line search.

We found a few changes to this strategy to be beneficial for our scenario. The particularity of this scenario is the fact that the initial mesh is generated with a strict focus on regularity, not on element quality (in terms of aspect ratios, sizing, etc.). The optimization thus may have to start from meshes with highly distorted elements of strongly varying sizes. Our changes allow this to be handled more efficiently. Further details are modified to support all edges being curved, instead of only the curve-conforming edges.

Because our contribution is not in the overall mesh optimization strategy, we restrict ourselves to briefly describing and discussing these changes and clarifying potential ambiguities in the following. For the sake of self-containedness and clarity we describe the strategy in its entirety in Appendix C.

4.1 Geometric Optimization

To optimize control point positions, we employ second-order optimization. Newton steps are applied globally to a conformal distortion objective $E_{\text{conf}} = (\Sigma^2 + \sigma^2)/\Sigma\sigma$ (with singular values Σ, σ of the Jacobian) defined in terms of all free control points' positions. To obtain a positive semi-definite Hessian we make use of composite majorization [Shtengel et al. 2017] based on a *convex-concave* decomposition, detailed in Appendix A. Due to the higher-order setting, integration is performed using quadrature, cf. Appendix B.

In this setting, all edges are free to curve if this serves distortion reduction. If, however, having straight edges away from the domain curves (as in [Hu et al. 2019]) is of benefit for an application, this is easily achieved: as our initial mesh has straight edges away from the curves, we simply need to preserve this. To this end, the respective edge and facet control points are expressed as fixed linear combinations of the variable vertex control points, such that only initially curved edges have the ability to deform freely during optimization.

4.2 Combinatorial Optimization

We briefly clarify the details of the local mesh modification operators that are relevant for regularity preservation.

Edge Split. An edge is split by bisecting the (one or two) adjacent triangles using Bézier triangle bisection [Farin 1986], which yields control points for the new triangles. These again define regular maps by construction: per triangle, the two new geometric maps differ from the former one by an affine domain transformation only.

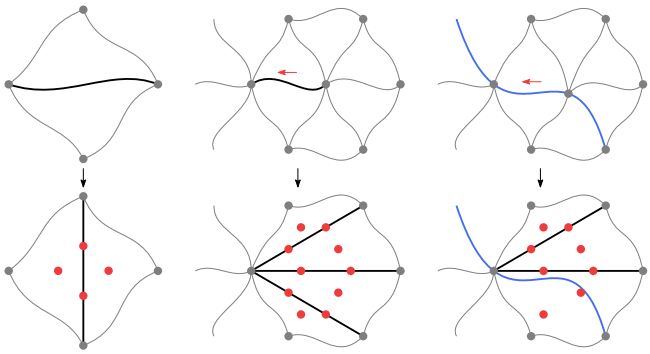

Fig. 13. Flip and collapse operators for local mesh modification. The red dots indicate control points (here for the cubic case) whose coordinates need to be set. On the right the special case of collapsing along a domain curve (blue) is illustrated.

Edge Collapse & Flip. In order to maintain conformance, edges representing domain curves are never flipped, vertices representing domain curve endpoints are never collapsed, and vertices in the interior of domain curves are only collapsed along edges on domain curves. Furthermore, in contrast to the linear case, where these operations come without any geometric degrees of freedom, positions for the control points associated with the triangles affected by the operation need to be chosen. We start with a quasi-uniform distribution, i.e., except for the control points shared with adjacent unaffected triangles (whose positions are preserved in order to preserve continuity) for each triangle with corner points $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ we set $\boldsymbol{p}_{ij} = \frac{k}{n}\boldsymbol{a} + \frac{i}{n}\boldsymbol{b} + \frac{j}{n}\boldsymbol{c}$ for all free control points (red in Fig. 13). A special case occurs when collapsing an edge on a domain curve; in this case two adjacent edges on the same domain curve, $\boldsymbol{c}([t_0, t_1])$ and $\boldsymbol{c}([t_1, t_2])$, are replaced by one, $\boldsymbol{c}([t_0, t_2])$, as illustrated in Fig. 13 right. Accordingly, the control points associated with this edge are set to the control points of the Bézier curve $\boldsymbol{c}([t_0, t_2])$.

If the map defined by these control points cannot be certified regular [Hernandez-Mederos et al. 2006], the collapse or flip operation is considered illegal, and is not performed. More involved strategies to determine control points (such as attempted untangling, cf. Appendix 4.2) proved to not provide benefits in the overall process. The next iteration of geometric mesh improvement will take care of optimizing the initial control point positions for low distortion.

4.3 Mesh Gradation

The mesh improvement strategy is driven by a target edge length. Uniformly sized meshes are obtained by prescribing a globally constant target edge length l. Alternatively, one can specify a predetermined, spatially varying sizing field, or adapt it, following [Hu et al. 2018], automatically in a dynamical manner depending on element quality, so as to obtain graded meshes, cf. Fig. 23.

When the mesh is initially of very low quality, verbatim application of the latter strategy can lead to the mesh getting highly

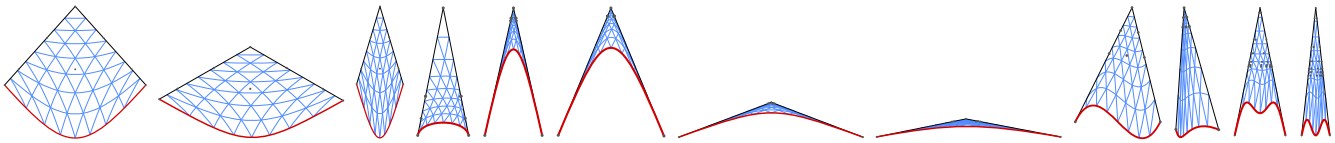

Fig. 14. Bézier triangles with one prescribed guardable curved edge. The underlying map, visualized here using several barycentric iso-curves (blue), is regular by construction. Most curves and their Bézier triangles shown here are cubic (the practically particularly relevant case); the two rightmost cases are of order 7.

overrefined in the early iterations, before ultimately being coarsened again—strongly slowing down the optimization due to the high intermediate mesh complexity. We therefore deviate in the following details:

- Instead of using a constant distortion threshold to decide whether the local target edge length is increased or decreased, we use a threshold that tightens in the course of optimization. We start with the maximum distortion of the initial mesh as threshold, which is then halved after each iteration of optimization.
- The local target edge lengths are updated after instead of already during one iteration.
- Instead of requiring all collapse and flip operations to, in addition to serving sizing improvement, monotonically reduce distortion, we allow them to be performed as long as the (intermediately potentially higher) distortion stays below a bound. We use the above dynamic threshold as this bound.

5 RESULTS

In this section we illustrate the proposed algorithm's behaviour and demonstrate its characteristics on a variety of example inputs.

5.1 General Overview

In Fig. 14 we demonstrate, on a variety of individual guardable curves, what the regular guarding Bézier triangles commonly look like. The geometric maps are visualized using barycentric iso-curves. As guaranteed by the proven regularity by construction, two iso-curves of the same parametric direction never intersect. Notice that, in contrast to some previous approaches, we do not need to compute and split at curve inflection points, or to limit the total curvature per curve piece to a small amount.

In Fig. 15, by virtue of the method's support of arbitrary order, we demonstrate the handling of a curve of order 10 by our algorithm. In stage (1) (as defined in Sec. 3.4), the curve was split into 9 guardable curves, in stage (2) they were split into a total of 17 curves because their envelopes were intersecting, in stage (3) the rest of the domain was triangulated. The result is shown on the right in the figure.

In Fig. 16 the handling of a complex spiral-like arrangement of cubic curves, exhibiting a wide range of shapes and curvature profiles, is shown. The proposed algorithm constructs a regular initial curved mesh, which can be used as valid starting point for mesh improvement and optimization methods.

In Fig. 17 we demonstrate that besides domain boundary curves also interface curves, feature curves, branch points, and holes (i.e., non-simply-connected domains) are naturally handled by our method.

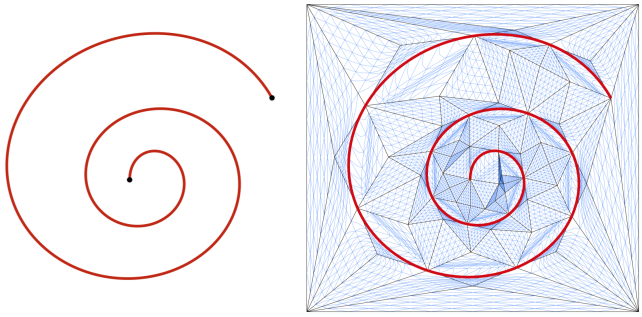

Fig. 15. Our construction supports arbitrary order. Here an example of a single decic (degree 10) curve is shown, meshed in a conforming way using triangles of order 10. Shown is the initial mesh without optimization. The blue curves are barycentric iso-curves of the regular geometric maps, shown here for visualization.

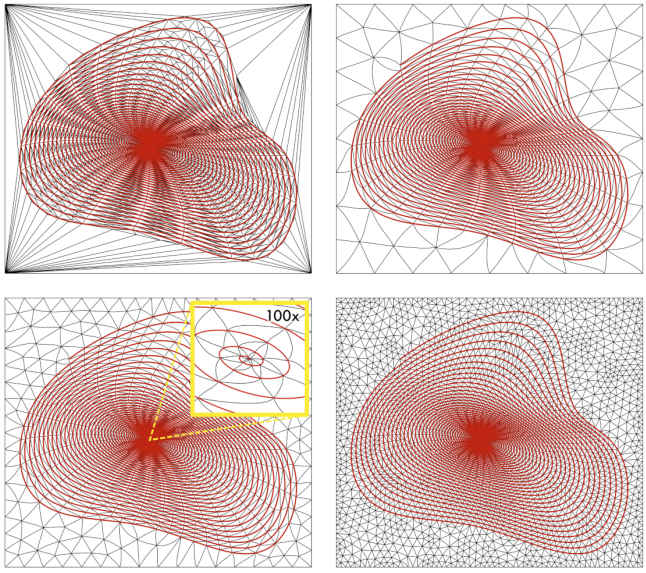

Fig. 16. Meshing of a complex spiral-like arrangement of 411 cubic curves. Top left: initial triangulation with regular triangle elements; the curves were split into 1,043 pieces to reach a state of non-intersecting envelopes. Starting from this initial state, the mesh was improved through incremental regularity preserving remeshing operations towards three different user-specified target edge lengths. On the top right, notice that formerly split curve segments may be recombined (via edge collapses, cf. Sec. C.2) in this process. The inset blow-up (100×) shows how the narrow configuration in the center is handled properly.

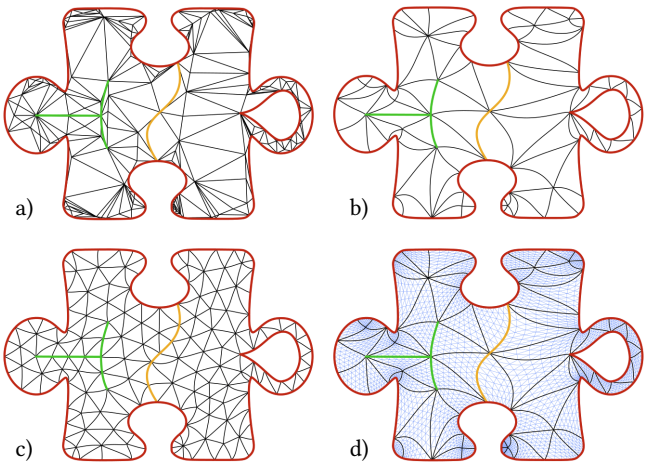

Fig. 17. Domain boundary curves (red), interface curves (yellow), and feature curves (green), including branch points and holes, are naturally handled by our method. a) initial regular mesh. b) improved mesh, large target edge length. c) improved mesh, shorter target edge length. d) visualization of geometric maps using iso-curves.

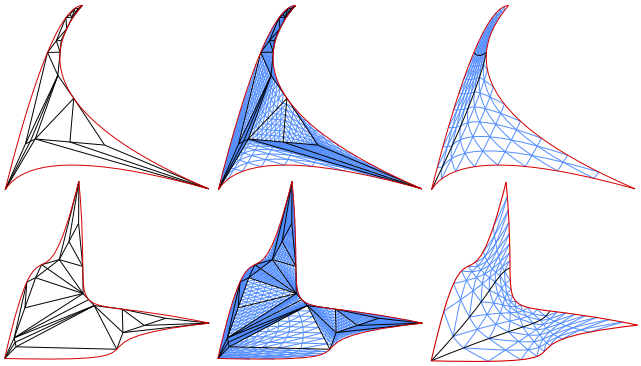

Fig. 18. Meshing of the triangular regions from Fig. 3 that, as a whole, do not admit a regular geometric map. Left: initial mesh. Center: initial mesh with parametric iso-curves (blue). Right: simplified through mesh optimization with large target edge length.

In Fig. 18 the meshing of the regions from Fig. 3 is shown. In the course of our algorithm the domain is suitably subdivided, achieving a regular conforming result with multiple triangular elements.

In Fig. 19 we exemplarily demonstrate that the proposed method is able to properly handle corners with extremely small angles, in this case 10^{-30} degrees. This is likely of low practical relevance, but illustrates the method's solid foundation.

In all cases the regularity of the produced meshes' elements was verified via a positive lower bound on the Jacobian determinant [Hernandez-Mederos et al. 2006], computed using exact arithmetic.

5.2 Random Domain Curves

For further empirical verification of the proposed method, we applied it to four datasets (A, B, C, D) with different characteristics,

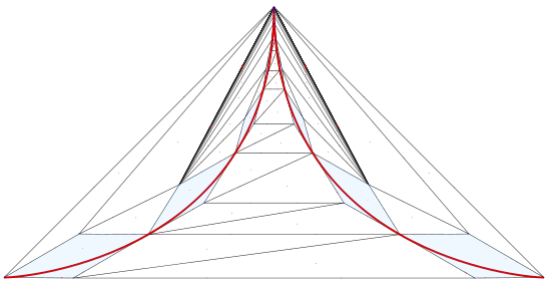

Fig. 19. Two curves forming an angle of just 10^{-30} degrees. The proposed method reliably meshes such very sharp corners. Here the two curves were guarded by 204 increasingly small triangles each. The initial mesh (of the curves' convex hull) is shown, with guarding triangles highlighted in blue.

each with 1000 randomly generated domain curve arrangements. Each exemplar in these datasets has the following characteristics:

(A) 50 curves forming a closed C^0 domain boundary loop with corners.
(B) 50 curves forming a closed C^1 domain boundary loop without corners.
(C) 100 isolated random curves, spread uniformly over a rectangular domain.
(D) around 100 curves forming a curve network, generated by intersecting random curves.

In Fig. 20 we show the resulting mesh (initial and optimized) for one case from each dataset for illustration of these characteristics. In Table 1 statistics for test runs of two implementations of our algorithm on these datasets are reported. These implementations differ in terms of numerics (cf. Sec. 3.7): the Exact version makes use of exact rational numbers [Granlund et al. 2019] throughout, while the Float version relies on standard double precision numbers for coordinate representation (of control points, guard points, mesh points) and exact predicates for intersection tests. Code containing both implementations (Exact and Float) and the four datasets are available on the authors' websites.

Table 1. The Exact version (top rows) successfully processes all test cases from the random datasets. Success was verified by computing a lower bound on the Jacobian determinant; it was strictly positive for every element in every case. The distribution of the Jacobian determinant values is shown in Fig. 21. For the Float version (lower rows), in a few cases some subdivided envelopes or final triangles became numerically degenerate such that no control points forming regular elements could be obtained in the final steps.

	Dataset:	(A)	(B)	(C)	(D)
Exact	Failure	0	0	0	0
	Success	1000	1000	1000	1000
Float	Failure	0	2	0	13
	Success	1000	998	1000	987
	Total	1000	1000	1000	1000

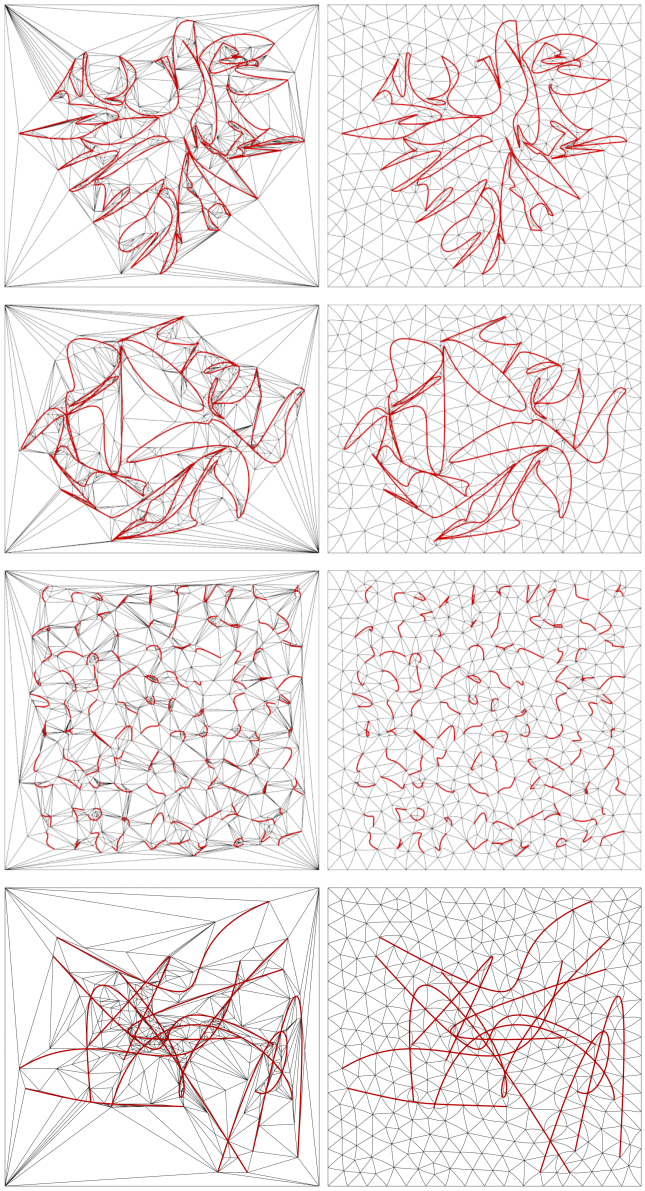

Fig. 20. Examples from the randomly generated test datasets. The left column shows the initial meshes (Sec. 3), the right column a uniformly remeshed version (Sec. C). From top to bottom: dataset A (C^0 domain with sharp corners), dataset B (C^1 domain), dataset C (isolated curves) and, dataset D (curve networks).

5.3 Clipart Dataset

The dataset used for testing by [Hu et al. 2019] contains around 20K cubic curve configurations from clipart files. This data is quite messy: there are degenerate curves, intersections, overlaps, duplicates, zero-angles. We applied the clean-up routine proposed in the above work and furthermore approximated occasional rational curves by polynomial curves. Irregular curves (whose first or last two control points coincide) are not explicitly treated by this clean-up procedure and remain in almost half the cases[2]. We applied our algorithm to the subset of regular exemplars. Statistics are reported in Table 2.

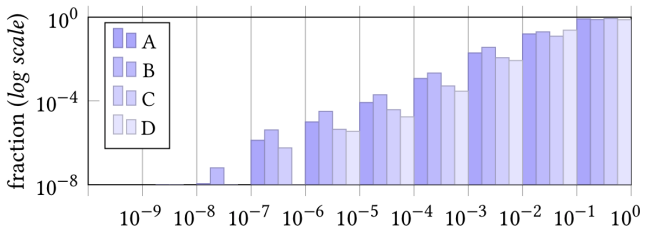

Fig. 21. Logarithmic histogram of scaled-Jacobian values (horizontal axis), aggregated over all elements of the initial result meshes for the datasets of Sec. 5.2. As guaranteed by our algorithm these values, while initially occasionally small, are strictly positive (here $> 10^{-9}$ in any case), such that the meshes are valid starting points for regularity preserving optimization.

Table 2. As expected, the Exact version processes the regular inputs from the clipart dataset without issues. With the Float version issues due to rounding errors can occur in the algorithm's steps: in step (1) curve bisection can lead to irregular or intersecting sub-curves; in step (2) this can be the case as well; in step (3) degenerate envelopes can cause ill-defined polygons that are to be triangulated; in step (4) the regular control point computation according to Sec. 3.3.1 can be hampered by numerical degeneracies.

Clipart Dataset	Float	Exact
Step (1) Failure	0	0
Step (2) Failure	38	0
Step (3) Failure	32	0
Step (4) Failure	23	0
Success	10,613	10,706
Total	10,706	10,706

5.4 Curve Conformance

To the best of our knowledge, among general methods for the generation of higher-order meshes with polynomial elements, the proposed method is the first to offer guarantees concerning curve conformance and regularity in combination. To demonstrate the benefit, in Fig. 22 we exemplarily show several cases where a recent method [Hu et al. 2019] for higher-order meshing of curved domains shows limitations: in these cases it can output regular but non-conforming, or conforming but irregular elements. By construction, our method yields regular conforming elements also in these cases.

5.5 Timing

In Table 3 we give some insight into the runtime of our (single-threaded) implementation of the algorithm from Sec. 3.4.

The expectation of a time complexity roughly linear in the number of curves for steps (1), (4), and (5) is confirmed.

[2]The fact that the meshing method from [Hu et al. 2019] can yield regular meshes for at least some of these turns out to be merely due to numerical inaccuracies introduced by the Bézier-to-Lagrange conversion that this method employs, which quite often turns truly irregular curves into slightly different nearly irregular curves.

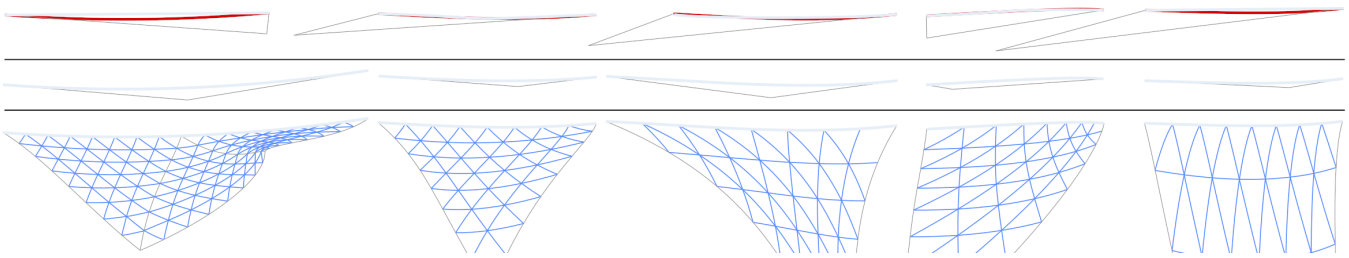

Fig. 22. Top row: examples of curve conformance errors of a previous curved meshing method [Hu et al. 2019] (using code published by the authors, on files 219104, 222226, 245616, 183643, and 169618 from their clipart dataset). Shown are single triangle excerpts from larger result meshes. Prescribed domain curves are drawn in red, while actual triangle element edges corresponding to these curves are drawn on top in very light gray-blue; hence, any visibility of red color indicates non-conformance. In these cases, a conforming regular map for the triangular region created by the mesher either did not exist or could not be found. Second row: regular results created by our proposed algorithm at the exact same curve regions; due to strict conformance by construction, no red is visible. Bottom row: the same region after mesh optimization; conformance is preserved.

In step (2), which is dominated by intersection tests between envelopes, we employ an interval-tree for the envelopes' bounding boxes as search structure for potential intersections.

In our current implementation step (3) (linear triangulation of the remaining domain) is performed using the Triangle library (in our Float version) or using CGAL (in our Exact version).

Table 3. Timings of our algorithm (Sec. 3.4) in milliseconds. Obtained by averaging over runs on hundreds of input domains (taken from the clipart dataset) containing ~10, ~100, and ~1000 curves (±10%), respectively.

	Float			Exact		
#Curves=	~10	~100	~1000	~10	~100	~1000
Step (1)	0.4	7.4	82.6	2.3	26.0	228.8
Step (2)	21.1	419.2	5903.7	20.8	411.3	6010.6
Step (3)	1.1	10.7	89.5	11.3	166.0	1246.7
Step (4)+(5)	0.4	3.4	30.8	19.6	285.5	2418.4
Total	23.0	440.7	6106.6	54.0	888.8	9904.5

5.6 Choice of μ

Choosing a large value for the parameter μ leads to overly large envelopes, requiring more bisection, leading to unnecessarily dense initial meshes and longer run times. Very small μ can imply nearly irregular elements (very small $\det J$), which—in a non-exact arithmetic implementation—can cause numerical degeneracies. This is reflected in the statistics shown in the table below, computed over 1000 input configurations from the dataset from Sec. 5.3: shown is the total number of triangle elements created (summed over all inputs) and the percentage out of these elements that are degenerate (or inverted) when represented using standard double precision floating point numbers. It can be observed that the choice is rather uncritical in the range between 10^{-1} and 10^{-3}. We use 10^{-2} by default and in all other experiments.

$\mu =$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
Total elements	$1897K$	$1416K$	$1355K$	$1353K$	$1352K$	$1352K$
Degen. elements	0	0	0	0.04%	0.14%	0.17%

6 LIMITATONS & FUTURE WORK

Various generalizations and extensions of the proposed approach will be interesting objects for future investigation.

For instance, adding support for rational curves in addition to polynomial curves, i.e., Bézier curves with an additional weight per control point, is an obvious avenue for future work. In this way practically important non-polynomial curves like arcs and general conics could be supported as input domain curves.

Likewise, generalization to the three-dimensional setting, i.e., higher-order tetrahedral meshes conforming to curved piecewise polynomial or rational domain surfaces, is of interest. A major challenge will be the construction of regular geometric maps for this case.

Depending on the use case, smoothness of the geometric maps across edges (beyond the C^0-continuity ensured by our construction) can be of interest, e.g., in the context of isogeometric analysis based on spline spaces. There may be ways to achieve this in our framework using macro-element schemes on top [Jaxon and Qian 2014]. The key issue will be ensuring regularity at the same time.

Performance improvements can potentially be achieved in a variety of ways. One tuning option is to perform bisections not at $\frac{1}{2}$ but at points carefully selected to reduce the required number of bisections. Another option is to investigate different, perhaps tighter envelope definitions. Prop. 3.9, the proof of regularity based on control vectors, is quite generic and could be applied to modified control point constructions that possibly lead to less refinement, better elements, and faster convergence.

The focus herein is on guaranteeing regularity and conformance. In the field of subsequent mesh optimization interesting challenges remain, in particular towards quality guarantees, angle bounds, etc.

On a minor note, a general construction of regular geometric maps for a straight-edge triangle with non-uniform control point distribution along more than one of its edges would avoid the need to split certain edges in step (3) of the algorithm in Sec. 3.4.

ACKNOWLEDGMENTS

The authors thank Hendrik Brückler for his support in the development of demonstrational implementations.

REFERENCES

Remi Abgrall, Cécile Dobrzynski, and Algiane Froehly. 2014. A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems. *International Journal for Numerical Methods in Fluids* 76, 4 (2014), 246–266.

Ivo Babuška and B.Q. Guo. 1996. Approximation properties of the hp version of the finite element method. *Comput. Methods Appl. Mech. Eng.* 133, 3-4 (1996), 319–346.

K. E. Barrett. 1996. Jacobians for isoparametric finite elements. *Communications in Numerical Methods in Engineering* 12 (1996), 755–766.

Francesco Bassi and Stefano Rebay. 1997. High-order accurate discontinuous finite element solution of the 2D Euler equations. *J. Comp. Physics* 138, 2 (1997), 251–285.

T. Belytschko and T. Black. 1999. Elastic crack growth in finite elements with minimal remeshing. *Int. J. Numer. Methods Eng.* 45, 5 (1999), 601–620.

Mario Botsch and Leif Kobbelt. 2004. A Remeshing Approach to Multiresolution Modeling. In *Proc. Symposium on Geometry Processing*. ACM, 185–192.

Erik Burman and Peter Hansbo. 2010. Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method. *Comput. Methods Appl. Mech. Eng.* 199, 41 (2010), 2680 – 2686.

David Cardoze, Alexandre Cunha, Gary L. Miller, Todd Phillips, and Noel Walkington. 2004. A Bézier-based Approach to Unstructured Moving Meshes. In *Proc. Symposium on Computational Geometry*. ACM, 310–319.

Kwok Wah Cheng and Thomas-Peter Fries. 2010. Higher-order XFEM for curved strong and weak discontinuities. *Int. J. Numer. Methods Eng.* 82, 5 (2010), 564–590.

P.G. Ciarlet and P.-A. Raviart. 1972a. The Combined Effect of Curved Boundaries and Numerical Integration in Isoparametric Finite Element Methods. In *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, A.K. Aziz (Ed.). Academic Press, 409 – 474.

P.G. Ciarlet and P.-A. Raviart. 1972b. Interpolation theory over curved elements, with applications to finite element methods. *Comput. Methods Appl. Mech. Eng.* 1, 2 (1972), 217 – 249.

Saikat Dey, Robert M. O'Bara, and Mark S. Shephard. 1999. Curvilinear Mesh Generation In 3D. In *Proc. International Meshing Roundtable*. John Wiley & Sons, 407–417.

Saikat Dey, Robert M O'Bara, and Mark S Shephard. 2001. Towards curvilinear meshing in 3D: the case of quadratic simplices. *Computer-Aided Design* 33, 3 (2001), 199–209.

Arno Eigenwillig, Lutz Kettner, and Nicola Wolpert. 2007. Snap rounding of Bézier curves. In *Proc. Symposium on Computational Geometry*. ACM, 158–167.

Luke Engvall and John A. Evans. 2016. Isogeometric triangular Bernstein–Bézier discretizations: Automatic mesh generation and geometrically exact finite element analysis. *Comput. Methods Appl. Mech. Eng.* 304, C (2016).

Gerald Farin. 1986. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design* 3, 2 (1986), 83–127.

Gerald Farin. 2002. *Curves and Surfaces for CAGD: A Practical Guide* (5th ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Meire Fortunato and Per-Olof Persson. 2016. High-order unstructured curved mesh generation using the Winslow equations. *J. Comput. Phys.* 307 (2016), 1 – 14.

Lori A. Freitag and Carl Ollivier-Gooch. 1997. Tetrahedral mesh improvement using swapping and smoothing. *Int. J. Num. Methods in Engrg.* 40, 21 (1997), 3979–4002.

Thomas-Peter Fries and Ted Belytschko. 2010. The extended/generalized finite element method: An overview of the method and its applications. *Int. J. Numer. Methods Eng.* 84, 3 (2010), 253–304.

Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015. Computing Locally Injective Mappings by Advanced MIPS. *ACM Trans. Graph.* 34, 4 (2015), 71:1–71:12.

Abel Gargallo-Peiró, Xevi Roca, Jaime Peraire, and Josep Sarrate. 2013. High-order mesh generation on CAD geometries. *International Conference on Adaptive Modeling and Simulation* VI (2013).

P.L. George and H. Borouchaki. 2012. Construction of tetrahedral meshes of degree two. *Int. J. Numer. Methods Eng.* 90, 9 (2012), 1156–1182.

William J. Gordon and Charles A. Hall. 1973. Transfinite element methods: Blending-function interpolation over arbitrary curved element domains. *Numer. Math.* 21, 2 (1973), 109–129.

Torbjörn Granlund, , and the GMP development team. 2019. *GNU MP: The GNU Multiple Precision Arithmetic Library.* http://gmplib.org/.

Jens Gravesen, Anton Evgrafov, Dang-Manh Nguyen, and Peter Nørtoft. 2014. Planar Parametrization in Isogeometric Analysis. In *Mathematical Methods for Curves and Surfaces*. Springer Berlin Heidelberg, 189–212.

Robert Haber, Mark S. Shephard, John F. Abel, Richard H. Gallagher, and Donald P. Greenberg. 1981. A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mappings. *Int. J. Numer. Methods Eng.* 17, 7 (1981), 1015–1044.

Victoria Hernandez-Mederos, Jorge Estrada-Sarlabous, and Dionne León Madrigal. 2006. On local injectivity of 2D triangular cubic Bezier functions. *Investigación Operacional* 27, 3 (2006), 261–275.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1993. Mesh optimization. In *Proc. SIGGRAPH*. 19–26.

Kai Hormann and Günther Greiner. 2000. MIPS: An efficient global parametrization method. *Curve and Surface Design '99* (2000), 153–162.

Yixin Hu, Teseo Schneider, Xifeng Gao, Qingnan Zhou, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2019. TriWild: Robust Triangulation with Curve Constraints. *ACM Trans. Graph.* 38, 4 (2019).

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4 (2018).

Noah Jaxon and Xiaoping Qian. 2014. Isogeometric analysis on triangulations. *Computer-Aided Design* 46 (2014), 45–57.

Amaury Johnen, Jean-François Remacle, and Christophe Geuzaine. 2012. Geometrical Validity of Curvilinear Finite Elements. *J. Comput. Phys.* 233 (01 2012).

Leif P Kobbelt, Thilo Bareuther, and Hans-Peter Seidel. 2000. Multiresolution shape deformations for meshes with dynamic vertex connectivity. In *Computer Graphics Forum*, Vol. 19. 249–260.

J. Li, T. J. Peters, and J. A. Roulier. 2012. Angular Convergence during Bézier Curve Approximation. arXiv:math.GT/1210.2686

Xiaojuan Luo, Mark S Shephard, and Jean-Francois Remacle. 2001. The influence of geometric approximation on the accuracy of high order methods. *Rensselaer SCOREC report* 1 (2001).

Xiaojuan Luo, Mark S. Shephard, Jean-François Remacle, Robert M. O'Bara, Mark W. Beall, Barna A. Szabó, and Ricardo Actis. 2002. p-Version Mesh Generation Issues. In *IMR 2002*.

Xiao-Juan Luo, Mark S. Shephard, Robert M. O'Bara, Rocco Nastasia, and Mark W. Beall. 2004. Automatic p-Version Mesh Generation for Curved Domains. *Eng. with Comput.* 20, 3 (2004), 273–285.

Manish Mandad and Marcel Campen. 2020. Efficient Piecewise Higher-Order Parametrization of Discrete Surfaces with Local and Global Injectivity. *Computer-Aided Design* (to appear) (2020).

Lois Mansfield. 1978. Approximation of the Boundary in the Finite Element Solution of Fourth Order Problems. *SIAM J. Numer. Anal.* 15, 3 (1978), 568–579.

Victor Milenkovic and Lee R. Nackman. 1990. Finding Compact Coordinate Representations for Polygons and Polyhedra. In *Proc. Symp. Comp. Geom.* 244–252.

A. R. Mitchell, G. Phillips, and E. Wachapress. 1971. Forbidden Shapes in the Finite Element Method. *IMA Journal of Applied Mathematics* 8, 2 (1971), 260–269.

Géraldine Morin and Ron Goldman. 2001. On the smooth convergence of subdivision and degree elevation for Bézier curves. *Comput. Aided Geom. Des.* 18, 7 (2001), 657–666.

D. Moxey, D. Ekelschot, Ü. Keskin, S.J. Sherwin, and J. Peiró. 2016. High-order Curvilinear Meshing Using a Thermo-elastic Analogy. *Comput. Aided Des.* 72, C (2016), 130–139.

J. Tinsley Oden. 1994. Optimal hp finite element methods. *Comput. Methods Appl. Mech. Eng.* 112, 1-4 (1994), 309–331.

Todd A. Oliver. 2008. *A High-Order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-Averaged Navier-Stokes Equations.* Ph.D. Dissertation. MIT.

Jordi Paul. 2018. Orientation preserving mesh optimisation and preconditioning. *Int. J. Numer. Methods Eng.* 114, 7 (2018), 749–776.

Per-Olof Persson and Jaime Peraire. 2009. Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. *47th AIAA Aerospace Sci Meet* (2009).

Roman Poya, Ruben Sevilla, and Antonio J. Gil. 2016. A unified approach for a posteriori high-order curved mesh generation using solid mechanics. *Computational Mechanics* 58, 3 (2016), 457–490.

Hartmut Prautzsch and Leif Kobbelt. 1994. Convergence of subdivision and degree elevation. *Advances in Computational Mathematics* 2, 1 (1994), 143–154.

Ramsharan Rangarajan and Adrián J. Lew. 2014. Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes. *Int. J. Numer. Methods Eng.* 98, 4 (2014), 236–264.

Xevi Roca, Abel Gargallo-Peiró, and Josep Sarrate. 2011. Defining quality measures for high-order planar triangles and curved mesh generation. In *Proc. International Meshing Roundtable*. Springer, 365–383.

Eloi Ruiz-Gironés, Josep Sarrate, and Xevi Roca. 2016. Generation of Curved High-order Meshes with Optimal Quality and Geometric Accuracy. *Procedia Engineering* 163 (2016), 315 – 327.

Thomas W Sederberg and Ray J Meyers. 1988. Loop detection in surface patch intersections. *Computer Aided Geometric Design* 5, 2 (1988), 161–171.

Ruben Sevilla, Luke Rees, and Oubay Hassan. 2016. The generation of triangular meshes for NURBS-enhanced FEM. *Int. J. Num. Methods Engrg.* 108, 8 (2016), 941–968.

Mark S. Shephard, Joseph E. Flaherty, Kenneth E. Jansen, Xiangrong Li, Xiaojuan Luo, Nicolas Chevaugeon, Jean-François Remacle, Mark W. Beall, and Robert M. O'Bara. 2005. Adaptive mesh generation for curved domains. *Appl. Numer. Math.* 52, 2 (2005), 251–271.

S. J. Sherwin and J. Peiró. 2002. Mesh generation in curvilinear domains using high-order elements. *Int. J. Numer. Methods Eng.* 53, 1 (2002), 207–223.

Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z Kovalsky, and Yaron Lipman. 2017. Geometric optimization via composite majorization. *ACM Trans. Graph.* 36, 4 (2017), 38–1.

Pavel Solin, Karel Segeth, and Ivo Dolezel. 2003. *Higher-Order Finite Element Methods.* CRC Press.

Hiromasa Suzuki, Yusuke Sakurai, Takashi Kanai, and Fumihiko Kimura. 1998. Interactive mesh dragging with adaptive remeshing technique. In *Proc. Pacific Graphics*. IEEE, IEEE Computer Society, 188–197.

Thomas Toulorge, Christophe Geuzaine, Jean-François Remacle, and Jonathan Lambrechts. 2013. Robust untangling of curvilinear meshes. *J. Comput. Phys.* 254 (2013), 8 – 26.

Thomas Toulorge, Jonathan Lambrechts, and Jean-François Remacle. 2016. Optimizing the geometrical accuracy of curvilinear meshes. *J. Comput. Phys.* 310 (2016), 361–380.

Michael Turner, Joaquim Peiró, and David Moxey. 2018. Curvilinear mesh generation using a variational framework. *Computer-Aided Design* 103 (2018), 73 – 91.

Stephen Vavasis. 2003. A Bernstein-Bézier Sufficient Condition for Invertibility of Polynomial Mapping Functions. arXiv:cs.NA/cs/0308021

Z.J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H.T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. 2013. High-order CFD methods: current status and perspective. *Int. J. Numerical Methods in Fluids* 72, 8 (2013), 811–845.

Freddie D Witherden and Peter E Vincent. 2015. On the identification of symmetric quadrature rules for finite element methods. *Computers & Mathematics with Applications* 69, 10 (2015), 1232–1241.

Zhong Q. Xie, Ruben Sevilla, Oubay Hassan, and Kenneth Morgan. 2013. The generation of arbitrary order curved meshes for 3D finite element analysis. *Computational Mechanics* 51, 3 (2013), 361–374.

Jing Xu and Andrey N. Chernikov. 2014. Automatic Curvilinear Quality Mesh Generation Driven by Smooth Boundary and Guaranteed Fidelity. *Procedia Engineering* 82 (2014), 200 – 212.

Miloš Zlámal. 1973. The finite element method in domains with curved boundaries. *Int. J. Numer. Methods Eng.* 5, 3 (1973), 367–373.

Baruch Zukerman, Ron Wein, and Efi Fogel. 2019. 2D Intersection of Curves. In *CGAL User and Reference Manual* (4.14.1 ed.). CGAL Editorial Board. https://doc.cgal.org/4.14.1/Manual/packages.html#PkgSurfaceSweep2

A MAJORIZATION

Using the following definitions of h and g, the objective E_{conf} can be written as $E_{\text{conf}} = h \circ g$:

$$h(u,v) = \frac{u^2}{v} \quad \text{and} \quad g = (\|J_\tau\|, \det(J_\tau)).$$

Let J_{ij} denote the entries of the Jacobian J_τ, then the function g can be further decomposed as $g = g^+ + g^-$ with

$$g^+ = \left(\|J_\tau\|, \frac{1}{4}\left[(J_{00}+J_{11})^2 + (J_{01}-J_{10})^2\right] \right),$$

$$g^- = \left(0\ , -\frac{1}{4}\left[(J_{00}-J_{11})^2 + (J_{01}+J_{10})^2\right] \right).$$

Note that the functions h and g^+ are convex, while g^- is concave. Using this convex-concave decomposition, a convex majorizer of E_{conf} with a positive semi-definite Hessian can be defined as described in [Shtengel et al. 2017, Eq. (9)], making it suitable for efficient second order Newton-type optimization.

A.1 Jacobian

To obtain the Jacobian J_τ of a Bézier triangle's geometric map τ, it is convenient to consider τ as a composition of two maps, $\tau = \phi \circ \psi$, via a right triangle R with unit length legs (coincident with the coordinate axes u, v). The map $\psi : \triangle \to R$ is a simple affine map, while $\phi : R \to \mathbb{R}^2$ is a Bézier map of degree n.

The Jacobian of the Bézier map ϕ is easily built from the partial derivatives by u and v:

$$J_\phi(u,v) = \begin{bmatrix} \frac{\partial x}{\partial u}(u,v) & \frac{\partial x}{\partial v}(u,v) \\ \frac{\partial y}{\partial u}(u,v) & \frac{\partial y}{\partial v}(u,v) \end{bmatrix},$$

where the partial derivatives [Farin 1986] are given by

$$\frac{\partial x}{\partial u}(u,v) = n \sum_{i+j+k=n-1} (x_{(i+1)jk} - x_{ij(k+1)}) B^{n-1}_{ijk}(u,v),$$

$$\frac{\partial x}{\partial v}(u,v) = n \sum_{i+j+k=n-1} (x_{i(j+1)k} - x_{ij(k+1)}) B^{n-1}_{ijk}(u,v),$$

and analogously for y, where $(x_{ijk}, y_{ijk}) = \boldsymbol{p}_{ijk}$ are the triangle's control points. The constant Jacobian of affine map $\psi : \triangle \to R$ is

$$J_\psi = \begin{bmatrix} 1 & -1/\sqrt{3} \\ 0 & 2/\sqrt{3} \end{bmatrix}.$$

The Jacobian of τ then is $J_\tau(u,v) = J_\phi(u,v) J_\psi$.

B QUADRATURE

The objective E_{conf} (more precisely: its majorizer) is numerically integrated over the higher-order triangles using a quadrature scheme [Witherden and Vincent 2015]. Let ξ_i, $i \in Q = \{1, \ldots, n\}$, denote the barycentric coordinates of the scheme's n quadrature points, and w_i the associated relative weights. Quadrature then yields

$$E^t_{\text{conf}} \approx A_t \sum_{i \in Q} w_i E_{\text{conf}}(t, \xi_i). \tag{1}$$

for a triangle t with area A_t.

C MESH OPTIMIZATION DETAILS

We spell out the details of a mesh optimization strategy that can be applied to improve the regular meshes generated by our algorithm from Sec. 3.4 in the following.

C.1 Geometric Optimization

In order to promote equilateral elements, we perform a variational optimization of control point positions with the objective of explicitly minimizing distortion of the geometric maps τ_i (defined on equilateral reference triangle $\triangle$). To avoid bias towards a particular element size, we employ a scale invariant conformal distortion measure [Hormann and Greiner 2000], computed from the map's Jacobian J_τ (or its singular values Σ, σ):

$$E_{\text{conf}} = (\Sigma^2 + \sigma^2)/\Sigma\sigma = \text{tr}(J_\tau^\intercal J_\tau)/\det(J_\tau). \tag{2}$$

Variants of this have been used for mesh optimization before, e.g. in [Hu et al. 2019]. For efficiency, as in [Mandad and Campen 2020], we employ second-order optimization, using global Newton steps with composite majorization [Shtengel et al. 2017] based on a *convex-concave* decomposition of E_{conf}, cf. Appendix A. Integration of E_{conf} over the mesh is performed using quadrature, cf. Appendix B.

During optimization, preservation of regularity is ensured by performing a (conservative) injectivity test [Hernandez-Mederos et al. 2006] on the Bézier triangles in the Newton method's line search, and curve conformance is preserved by constraining the corresponding control points: vertices on end points of the input domain curves remain fixed; vertices in the interior of such a curve are allowed to slide along the curve only (i.e., derivatives for such a vertex control point on curve point $\boldsymbol{c}(t)$, as well as for the curved edge control points, are taken w.r.t. parameter t instead of x, y).

C.2 Combinatorial Optimization

Like in a variety of previous works (mostly in the linear setting), e.g, [Botsch and Kobbelt 2004; Cardoze et al. 2004; Freitag and Ollivier-Gooch 1997; Hoppe et al. 1993; Hu et al. 2019; Kobbelt et al. 2000], we additionally perform combinatorial changes to the mesh connectivity. In particular, splits, collapses, and flips of edges are performed following a certain schedule and under certain conditions. Geometric and combinatorial optimization are performed alternatingly, until a termination criterion (sufficient mesh quality or maximum number of iterations) is met.

We make use of an edge-length driven strategy, which can be traced back at least to [Suzuki et al. 1998]. Like recent work on mesh optimization [Hu et al. 2019] we adopt the schedule proposed by [Botsch and Kobbelt 2004; Kobbelt et al. 2000], leading to the following algorithm. In this, the length of an edge between vertices v and w is denoted $\ell(v, w)$; we can approximate it using a linear mesh proxy (i.e. $\ell(v, w) = \|\boldsymbol{p}_v - \boldsymbol{p}_w\|$) as in previous work [Cardoze et al. 2004; Hu et al. 2019].

Initialization: Assign a target edge length l to each vertex. For an edge between vertices v and w let $l(v, w) = \frac{1}{2}(l(v) + l(w))$.

(1) *Split long edges:* split each edge (v, w) with $\ell(v, w) > \frac{4}{3}l(v, w)$, inserting new vertex z at its midpoint. Set $l(z) = l(v, w)$.
(2) *Collapse short edges:* collapse each collapsible edge (v, w) with $\ell(v, w) < \frac{4}{5}l(v, w)$.
(3) *Flip edges:* flip each flippable edge if that reduces the deviation of vertex valences from their optimum in a least squares sense.

The optimal vertex valence is 6 for interior vertices, 4 for general boundary vertices, and between 2 and 6 for corner boundary vertices (depending on the corner's angle).

An edge is considered "collapsible" or "flippable" if the implied geometric maps, defined as in Sec. 4.2, are regular. We note that instead of using the quasi-uniform control point distribution defined in Sec. 4.2, one could employ more sophisticated techniques that attempt to more flexibly find regular maps. In particular, one could instead try to *untangle* initially irregular configuration, for instance using a $\log(-\det J_\tau)$ objective [Toulorge et al. 2013]. We explored this option, but found it to mostly slow down the optimization process while ultimately not having a noticeable positive effect on the final result. A likely reason is that operations that lead to elements that are irregular despite part of their control points being in a uniform configuration are not very likely to be beneficial for mesh quality anyway.

Before proceeding with a global geometric optimization based on E_{conf} (Sec. C.1), it proved beneficial (speeding up the overall process) to perform a local optimization of E_{conf} per region affected by a local mesh modification operator.

Monotonicity. While such a purely edge length driven strategy works very well in the linear case [Botsch and Kobbelt 2004], in the higher-order case it can lead to nearly degenerate intermediate states, having adverse numerical effects on subsequent optimization steps. One can prevent this by discarding flips and collapses that do not strictly improve the mesh quality (as measured by E_{conf}) [Hu et al. 2019]. We, however, found that in the higher-order setting, with curved elements, restricting to strictly monotonic progress leads to unfavorable behavior. In particular if the initial mesh is of very low quality, the early removal of small or tiny elements through edge collapses often requires going through intermediate states with lower quality elements. Therefore, we allow non-monotonic but *bounded* behavior: an operation can be performed even if it increases E_{conf}, as long as $E_{\text{conf}} < E_{\text{limit}}$ for some limit value. This limit is incrementally tightened during optimization. We initialize E_{limit} as $\max E_{\text{conf}}$, i.e., the maximum over the entire mesh, and halve it after each iteration of the combinatorial optimization.

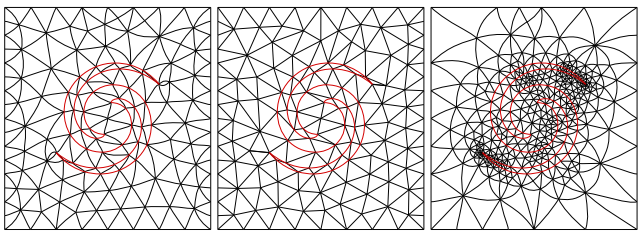

Fig. 23. Left: remeshing with uniform target edge length. Center: remeshing with uniform target edge length, preserving straightness of black edges. Right: graded remeshing with adaptive edge length, with $E_{\text{conf}} < 3$ everywhere.

C.3 Mesh Gradation

The above algorithm, with globally constant target edge length l, leads to uniformly sized meshes. Alternatively, one can specify a predetermined, spatially varying sizing field, or adapt it, following [Hu et al. 2019, 2018], automatically in a dynamical manner depending on element quality, cf. Fig. 23. This is achieved by specifying a target edge length range $[l_{\min}, l_{\max}]$, initializing $l = l_{\max}$ for each vertex, and updating the per vertex values after each iteration of combinatorial mesh optimization as follows:

- For each *bad* vertex v: $l(v) \leftarrow \frac{1}{2}l(v)$ clamped to $[l_{\min}, l_{\max}]$.
- For each *good* vertex v: $l(v) \leftarrow \frac{3}{2}l(v)$ clamped to $[l_{\min}, l_{\max}]$.

A vertex is *bad* if it has an adjacent triangle t with $\max_t E_{\text{conf}} > E_{\text{thres}}$; otherwise it is *good*. Using a constant E_{thres} (as in previous work [Hu et al. 2018]) proved to be suboptimal in our scenario: when the mesh is initially of very low quality, it will be strongly refined in the first iterations, before ultimately being coarsened again—unnecessarily slowing down the optimization due to the high intermediate mesh complexity. Initializing E_{thres} as $\max E_{\text{conf}}$, i.e., the maximum over the mesh, and halving it after each iteration of the combinatorial optimization proved to provide a major improvement in this regard. Note that the minimally possible value of E_{conf} is 2 (for an ideal equilateral element), therefore E_{thres} must not be reduced below a somewhat larger value. Clamping to a value of 4 offers a good balance; clamping to 3 commonly more than doubles the mesh complexity while reducing hardly reducing E_{conf} on average.

We point out that we update the l-field after all combinatorial operations are performed instead of immediately after step (1) (as described in [Hu et al. 2018]). When starting from a very low-quality state the latter can lead to significant overrefinement.