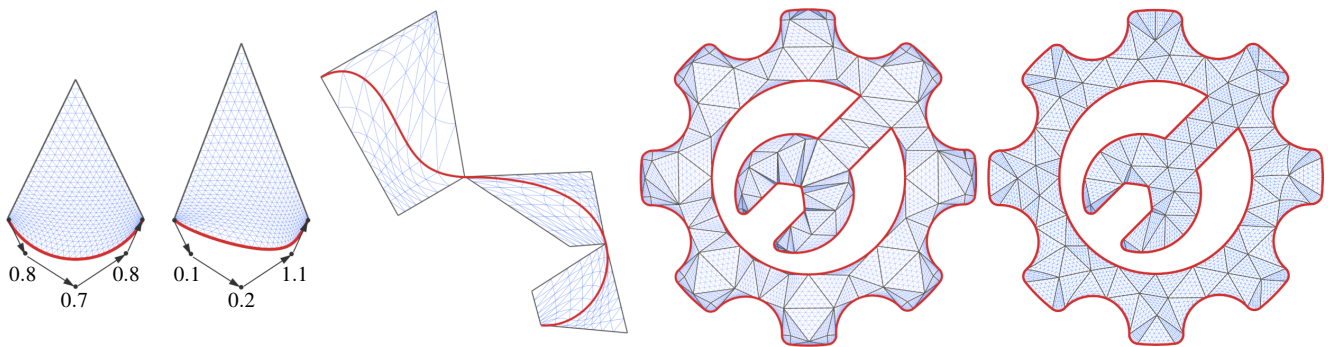# Rational Bézier Guarding

P. Khanteimouri · M. Mandad · M. Campen

Osnabrück University, Germany

Figure 1: *We enable the construction of nonlinear rational triangular elements that conform precisely to a given rational curve (left, red) of arbitrary degree (here quartic). Under a condition on the curve's complexity, we guarantee these elements' injectivity by construction (see the fold-free blue isolines). Arbitrarily complex rational curves can always be split into subcurves that all satisfy this condition (center left). By integrating this with the recent Bézier Guarding approach, we enable the generation of valid conforming planar triangle meshes with rational elements for curve-bounded domains (center right); these may serve as feasible starting point for further remeshing (right).*

Abstract

We present a reliable method to generate planar meshes of nonlinear rational triangular elements. The elements are guaranteed to be valid, i.e. defined by injective rational functions. The mesh is guaranteed to conform exactly, without geometric error, to arbitrary rational domain boundary and feature curves. The method generalizes the recent Bézier Guarding technique, which is applicable only to polynomial curves and elements. This generalization enables the accurate handling of practically important cases involving, for instance, circular or elliptic arcs and NURBS curves, which cannot be matched by polynomial elements. Furthermore, although many practical scenarios are concerned with rational functions of quadratic and cubic degree only, our method is fully general and supports arbitrary degree. We demonstrate the method on a variety of test cases.

CCS Concepts

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

1. Introduction

The generation of mesh representations for given domains, in particular for purposes of discretization, is a classical problem. We consider here the case of planar domains. One can distinguish between meshes that approximate, and meshes that exactly represent the domain, i.e. the mesh elements formally form a partition of it. In case of a planar domain this distinction comes down to the question whether the mesh precisely conforms to the domain boundary.

The generation of meshes with *linear* triangular elements is particularly well researched and many robust algorithms are available.

Unless the domain boundary is of piecewise linear nature, such meshes can only approximate the domain. Meshes with curved triangular elements defined by higher-order polynomials are more flexible in this regard. Recent results [MC20, MC21] enable the generation of such meshes with guarantees of element validity and exact conformance to curved domain boundaries – as long as they are polynomial. This still excludes domains with various important boundary shapes, including circular and elliptic arcs, and more generally the practically relevant class of NURBS curves (e.g. for *trimmed* surface patch domains), consisting of rational pieces.

We generalize the *Bézier Guarding* method [MC20] to support not only polynomial, but rational curves, which includes all of the above. We do this in such a way that the method's beneficial properties are preserved: the generated mesh with rational triangular elements conforms precisely to the rational domain boundary, as well as to potential feature or interface curves; each element is guaranteed to be valid (i.e. formed by an injective map) by construction; there are no smoothness assumptions on the input curves.

The central operation in this approach is the construction of a valid triangular element such that one of its edges takes the shape of a prescribed curve, while the other two edges are straight. While this is not generally feasible for arbitrary curves, the construction is such that any curve can be split into subcurves such that for each subcurve this construction is feasible. Our key contribution is a construction that enables this for rational curves and triangles, while provably possessing all the properties of the original polynomial construction that are required to replace it into the Bézier Guarding method, preserving its guarantees. This contribution includes:

- a *guardability* condition for rational curves that will eventually be satisfied under repeated bisection;
- a construction of rational triangular elements, suitable for any regular rational curve that satisfies this condition, such that:
  - the element conforms to the curve;
  - the element is valid, it possesses an injective geometric map;
  - under bisection, per subcurve the element's height over the curve converges to zero at a faster rate than its width, and, as a consequence, the inner angles at the curve converge to zero.

The latter convergence property can be ensured by careful variations of the approach used in the original method. Ensuring validity requires a major deviation due to the rational nature of the curve and the sought geometric map.

2. Related Work

The literature on mesh generation is vast. We focus our discussion on methods for 2D mesh generation with nonlinear elements, in particular those that support the prescription of curve constraints, such as domain boundaries and feature curves, that the mesh shall conform to. Key differences lie in whether a method guarantees (i) validity (non-degeneration and non-inversion) of elements, and (ii) precise conformance to constraints (rather than approximation). Furthermore, the type of employed elements and supported constraints (polynomial, rational) is a distinctive criterion.

Polynomial, without both Guarantees. There is quite a list of methods to generate a mesh of higher-order polynomial triangles that follow an indirect approach: first generate a mesh with linear elements, approximating the curve constraints, then perform an incremental deformation procedure with the goal of making the then curved elements conform the given curves [SP02, LSO*04, SFJ*05, Oli08, PP09, RGPS11, GB12, GPRPS13, XSHM13, TGRL13, XC14, RGSR16, MEK*16, PSG16, FP16, TPM18, Pau18, HSG*19]. While the deformation can be constrained to prevent the introduction of any degeneracies or inversions, guarantees of convergence to a conforming state cannot be given. The non-convexity of the involved deformation problems is a major obstacle in this regard.

Other methods favor conformance, for instance by forcing the element onto the curve [DOS99, DOS01, RL14, JQ14]. The downside is that this either invalidates any guarantee that the resulting deformation preserves validity, i.e. does not introduce inversions or degeneracies, or leads to deformation maps of more complex, non-polynomial kind – at least unless additional restrictions are imposed on the shape of the constraint curves.

Polynomial, with both Guarantees. Only recently, polynomial mesh generation methods offering validity *and* conformance guarantees surfaced. The Bézier Guarding method [MC20] constructs a polynomial triangular mesh precisely conforming to polynomial curves. It supports arbitrary polynomial order and the generated triangular elements are provably valid by construction. Another method [MC21] in addition guarantees lower bounds on various mesh quality measures like scaled Jacobian and MIPS distortion in the meshes it generates. Both methods, however, target polynomial constraint curves, and their constructions are tightly linked with specificities of polynomial curves.

Rational, without both Guarantees. The above mentioned indirect nonlinear mesh generation idea extends easily to rational curves and elements. First an approximating linear mesh is constructed, then adjusted to enforce conformance, essentially by replacing some coefficients, control points, or weights [Mäk05, LJ19, ADF14, JQ14, BECN*22]. Validity, i.e. injectivity of the elements, is not accounted for and may be lost in this adjustment.

Also direct approaches have been taken for rational curves [EE16], albeit without explicit regard for validity, and with a local view on a single curve, rather than a full mesh generation solution capable of handling complex curve arrangements with corners, tight passages, or similar features.

In some works, more complex elements (e.g. non-triangular or piecewise rational) are employed [SRH16, Miu93], or only part of the mesh generation problem is considered from a theoretical point of view [GH73, Zla73]. The verification, rather than construction, of rational elements in terms of injectivity has recently been addressed by providing conservative Jacobian determinant bounds [EE20], generalizing ideas from the polynomial case [DOS99].

Rational, with both Guarantees. In this category, to the best of our knowledge, no method exists. This is exactly the gap filled by our method presented in the following. Concurrent work [YLC*22] addresses a similar problem setting in a related manner.

3. Conforming Rational Bézier Elements

The central operation in the Bézier Guarding approach [MC20] is the construction of a polynomial triangular element that exactly conforms to a given polynomial curve and whose geometric map is, by construction, injective (Figure 1 left). The curve is assumed to meet a *guardability* condition that enables this construction. We recapitulate this condition and construction in Sec. 3.2.

Our goal in this section is to describe a construction of a *rational* triangular element conforming to a given *rational* curve. We also formulate a condition on the curve that guarantees the injectivity

of the constructed element, and show how any curve can be segmented into subcurves that meet this condition (Figure 1 center). Afterwards, in Sec. 4, we will make use of this construction for the purpose of mesh generation.

3.1. Background

Definition 1 (Bézier Curve) Given control points $p_0, \ldots, p_n \in \mathbb{R}^m$, a (polynomial) Bézier curve $\boldsymbol{c}(t) : [0, 1] \to \mathbb{R}^m$ is defined as:

$$\boldsymbol{c}(t) = \sum_{i=0}^{n} p_i B_i^n(t),$$

where $B_i^n(t)$ are the Bernstein polynomials of degree n.

Definition 2 (Rational Bézier Curve) Given control points $p_0, p_1, \ldots, p_n \in \mathbb{R}^m$ and associated non-negative weights $w_0, w_1, \cdots, w_n \in \mathbb{R}$, a rational Bézier curve $\boldsymbol{c}(t) : [0, 1] \to \mathbb{R}^m$ is defined as [Far02, Ch. 13]:

$$\boldsymbol{c}(t) = \frac{\sum_{i=0}^{n} w_i p_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)}.$$

Equivalently, we can use homogeneous coordinates, i.e. $p_i^* = (w_i x_i, w_i y_i, w_i)$ for $p_i = (x_i, y_i)$ in the here relevant two-dimensional case ($m = 2$), to define

$$\boldsymbol{c}^*(t) = \sum_{i=0}^{n} p_i^* B_i^n(t).$$

Then, for $(x, y, w) = \boldsymbol{c}^*(t)$, we have $\boldsymbol{c}(t) = (x/w, y/w)$, i.e. $\boldsymbol{c}$ is obtained by central projection of $\boldsymbol{c}^*$ onto the plane $w = 1$, as illustrated in Figure 2.

Due to the following lemma [Far02, Ch. 13], we may assume $w_0 = w_n = 1$, which will simplify some constructions in the following.

Lemma 1 (Weight Normalization) For any given positive values c_0, c_n, any rational Bézier curve can be reparameterized (without changing its image) such that $w_0 = c_0$ and $w_n = c_n$. In particular, it can be *normalized* to assume $w_0 = w_n = 1$.

The sequence $(p_i) = p_0, \ldots, p_n$ of control points we will also refer to as *control polygon*, and their differences $v_i = p_{i+1} - p_i$ ($0 \leq i \leq n-1$) as *control vectors*.

Definition 3 (Rational Bézier Triangle) A Bézier triangle $\boldsymbol{p}(u, v) : \triangle \to \mathbb{R}^m$ of degree n over $\triangle = \{(u, v) \mid u \geq 0, v \geq 0, u + v \leq 1\}$ is defined by its control points (*control net*) $p_{ij} \in \mathbb{R}^m$, ($i \geq 0, j \geq 0, i + j \leq n$) as [Far02, Ch. 17]:

$$\boldsymbol{p}(u, v) = \sum_{i+j \leq n} p_{ij} B_{ij}^n(u, v),$$

where $B_{ij}^n(u, v)$ are the bi-variate Bernstein polynomials. A *rational* Bézier triangle with positive weights w_{ij} is defined by:

$$\boldsymbol{p}(u, v) = \frac{\sum_{i+j \leq n} w_{ij} p_{ij} B_{ij}^n(u, v)}{\sum_{i+j \leq n} w_{ij} B_{ij}^n(u, v)}.$$

Since a (rational) Bézier triangle conforms to the (rational) Bézier curves defined by its *boundary control points*, i.e. those p_{ij} with $i = 0$, $j = 0$, or $n - i - j = 0$, constructing *some* triangle $\boldsymbol{p}(u, v)$ (by setting its control points and weights) such that it conforms to
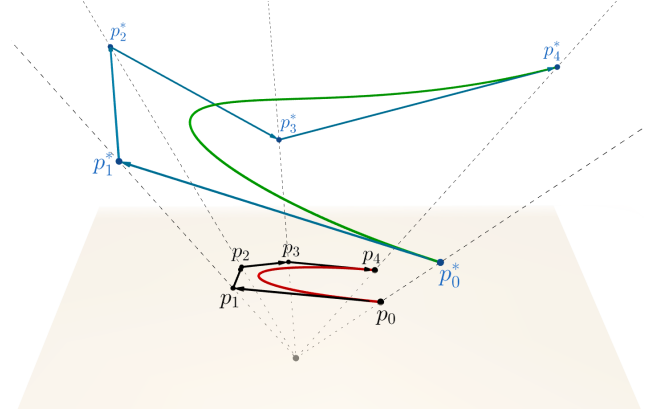
Figure 2: *A rational curve (red) in the plane can be interpreted as the central projection of a polynomial curve (green) in $\mathbb{R}^3$. The polynomial curve's control points p_i^* are given by the weight-scaled control points p_i of the rational curve, elevated to a height given by the corresponding weights w_i.*

a given curve is trivial; the challenge lies in achieving injectivity of the map $\boldsymbol{p}(u, v)$. This is addressed in Sec. 3.2.

Definition 4 (Bisection) A (rational) Bézier curve (p_i) can be bisected, i.e. subdivided into two subcurves at some parameter value $0 < t < 1$. In case of a polynomial Bézier curve, the subcurves' control points can be computed using de Casteljau's algorithm, starting from $p_i^0 = p_i$:

$$p_i^j = (1 - t) p_i^{j-1} + t p_{i+1}^{j-1}. \tag{1}$$

The sequences of intermediate points $(p_0^0, p_0^1, \cdots, p_0^n)$ and $(p_0^n, p_1^{n-1}, \cdots, p_n^0)$ are the control points of the two subcurves. In case of a rational Bézier curve, the subcurves' control points and weights can be obtained analogously using the rational version of de Casteljau's algorithm:

$$p_i^j = \frac{(1-t) w_i^{j-1} p_i^{j-1} + t w_{i+1}^{j-1} p_{i+1}^{j-1}}{w_i^j}, \tag{2}$$

$$w_i^j = (1-t) w_i^{j-1} + t w_{i+1}^{j-1}. \tag{3}$$

3.2. Injectivity

A (rational) Bézier triangle $\boldsymbol{p}(u, v)$ is *locally injective* if for all $(u, v) \in \triangle$:

$$\frac{\partial}{\partial u} \boldsymbol{p} \times \frac{\partial}{\partial v} \boldsymbol{p} \neq 0. \tag{4}$$

If the map furthermore acts injectively on the domain boundary $\partial \triangle$, we can conclude that $\boldsymbol{p}$ is (not only locally) injective.

The Bézier Guarding method constructs an injective *polynomial* Bézier triangle such that one of its edges takes the shape of a given curve based on the following principle: Assuming that all of the curve's control vectors are contained in a convex cone (this is the guardability condition referred to above in Sec. 3), the other control points of the triangle (away from that curved edge) are placed such
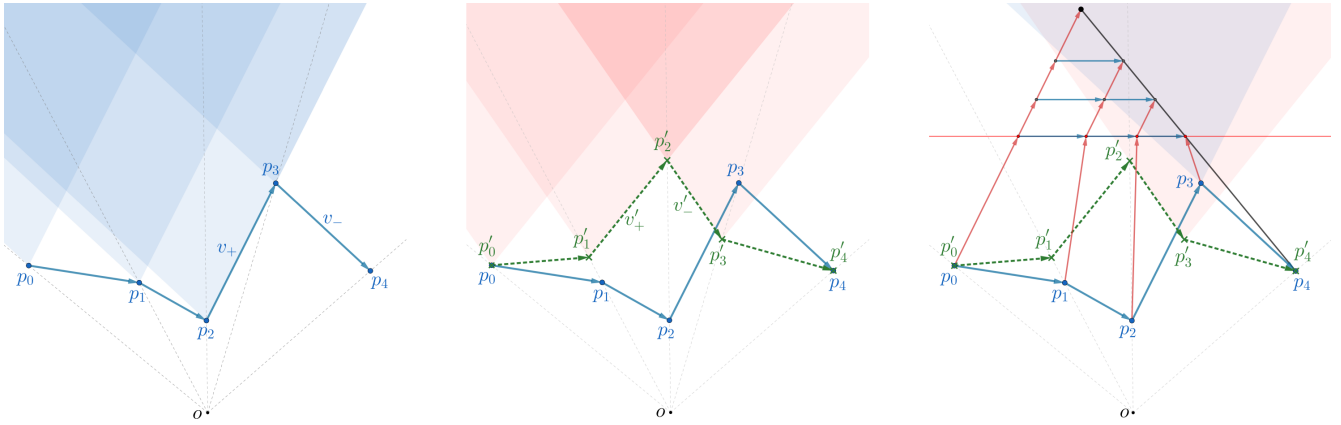
Figure 3: *Left: control polygon* (p_i) *with blue cones* $\vee_i^1$ *rooted at each control point (except the last, which is not needed). Center: auxiliary control polygon* (p_i'), *displaced according to the associated weights relative to a point o, with red cones* $\vee_i^2$ *rooted at these. Right: exemplary placement of control points* p_{ij} *for a rational Bézier triangle. These satisfy the conditions of Theorem 1, in particular: the control points on the dashed horizontal line lie in both, the respective blue and red cone (as illustrated for* $\vee_3^1 \cap \vee_3^2$*); all vectors* Δ_{ij}^0 *(blue) in the top part are parallel to d (horizontal); all control net vectors* Δ_{ij}^1 *(red) in the top part are parallel to the left edge. Therefore this triangle is injective.*

that certain vectors of the control net are contained in this cone, while others are contained in a complementary cone (a cone such that the union of both cones is a halfplane). The partial derivatives in (4) can be shown to be convex combinations of these vectors, i.e. they are contained in these two disjoint cones, respectively. This implies that their cross product cannot vanish, conservatively guaranteeing local injectivity by construction.

In the more general case of rational Bézier triangles, the partial derivatives are more complex, in particular not simply convex combinations of control vectors. Therefore the above argument does not apply, it does not generalize to the rational case.

3.2.1. Guardability

Instead we proceed as follows for the rational case. We first define the *auxiliary points* $p_i' = (1 - w_i)o + w_i p_i$, where o is an arbitrary fixed point – its choice in practice is discussed in Sec. 5. See Figure 3 center for an illustration. The point o can be imagined as an origin relative to which points p_i are scaled by w_i to yield p_i'. Moreover, let $v_i' = p_{i+1}' - p_i'$ for $0 \leq i \leq n - 1$ denote the *auxiliary control vectors*. We call a set of vectors $\{v_i\}$ *monotone* with respect to a direction d iff $d^\mathsf{T} v_i > 0$ for all i.

Definition 5 (Rational Guardability) We say a rational Bézier curve is *guardable* if its control vectors (v_i) as well as its auxiliary control vectors (v_i') are monotone with respect to a common direction d and a point o.

Notice that the first half of this definition (related to (v_i) only) is exactly the guardability condition used for the polynomial case in previous work.

Proposition 3.2 from [MC20] asserts that the control vectors v_i become monotone under repeated bisection, and therefore a polynomial curve becomes guardable. The following lemma asserts that this extends to the auxiliary control vectors v_i'. Hence, if a rational

curve is not guardable, it can be subdivided into a set of subcurves, such that each subcurve is guardable.

Lemma 2 (Auxiliary Point Convergence) Under repeated bisection, the subcurves' normalized weights w_i converge to 1, therefore the auxiliary control points p_i' converge to the control points p_i, and the auxiliary control vectors v_i' converge to the control vectors v_i.

Proof When subdividing a polynomial Bézier curve, computing control points using recursion (1), each subcurve's control polygon converges to a point (on the curve, thus inside the original control points' convex hull) under repeated subdivision [PK94]. When bisecting a rational curve, the two subcurves' weights are computed from the original curve's weights using (3). As this is the same recursion as (1), it follows that these, per subcurve, converge to a common positive value. With normalization, this value is 1. $\quad\square$

Furthermore, we can show that once a subcurve is guardable, this property is preserved under further subdivision. This is important to ensure convergence of the overall meshing algorithm (Sec. 4).

Lemma 3 (Guardability Preservation) Bisection of a guardable rational curve c yields two guardable rational subcurves.

Proof We need to show that monotonicity of $\{v_i\}$ and of $\{v_i'\}$ is preserved. The former is guaranteed due to the fact that, by (2), the control vectors of a subcurve are convex combinations of the original control vectors. For the latter, notice that substituting $p' - (1 - w)o = wp$, the auxiliary point definition, into (2) yields

$$p_i'^j = (1 - t)p_i'^{j-1} + t p_{i+1}'^{j-1}. \tag{5}$$

This is precisely the de Casteljau recursion (1) for polynomial curves, i.e., the auxiliary points p' behave under subdivision of the rational curve just like the control points of a polynomial curve. Therefore, by an argument completely analogous to Lemma 3.6 from [MC20], the preservation of monotonicity of auxiliary control vectors under subdivision is shown – essentially because they, as well, are obtained as convex combinations. $\quad\square$

3.2.2. Sufficient Injectivity Condition

We now describe, for an arbitrary guardable rational curve c, a condition on the placement of control points of a rational Bézier triangle p, such that it conforms to the curve and is injective.

Let d be a direction for which c is guardable, and v_+ (v_-) the control vector v_i minimizing $d^\mathsf{T} v_i/||v_i||$ that points counterclockwise (clockwise) relative to d. Analogously, we define v'_+, v'_- with respect to the auxiliary control vectors v'_i (and direction d). With these *extreme vectors* we define two sets of 2D (infinite, half) cones:

- *blue* cone $\vee^1_i$ with apex at point p_i and boundary directions v_+ and $-v_-$ (Figure 3 left),
- *red* cone $\vee^2_i$ with apex at point p'_i and boundary directions v'_+ and $-v'_-$ (Figure 3 center).

Note that for a guardable rational curve, with monotone (auxiliary) control vectors, these cones are non-empty, and the intersections $\vee^1_i \cap \vee^2_i$ are non-empty as well – as both cones contain the direction $d^\perp$, counterclockwise perpendicular to d. These cones and this fact play a central role in our injectivity condition stated in the following. For this, let $\Delta^0_{ij} = p_{i(j+1)} - p_{ij}$ and $\Delta^1_{ij} = p_{(i+1)j} - p_{ij}$ ($i,j \geq 0$ and $i + j \leq n - 1$) denote vectors of the control net of a Bézier triangle.

Theorem 1 (Rational Injectivity Condition) If, for a given guardable rational curve (p_i) with weights (w_i), we choose the control points of a rational Bézier triangle p such that

1. $p_{0i} = p_i$ and $w_{0i} = w_i$,
2. $w_{ij} = 1, i \geq 1$,
3. $p_{1i} \in \vee^1_i \cap \vee^2_i$,
4. $\Delta^0_{ij} \parallel d, i \geq 1$,
5. $\Delta^1_{ij} \parallel \Delta^1_{00}, i \geq 1$.

then p conforms to the curve and is injective. $a \parallel b$ here means there is an $s > 0$ such that $a = sb$.

Conformance is straightforward due to Condition 1 (making the first row of control points and weights coincide with the curve's), and injectivity is essentially ensured by carefully placing the second row of control points inside the cone intersections (Figure 3 right). The further rows can easily be placed such that the latter two conditions are satisfied (see the red and blue control net vectors on the rows $i \geq 1$ in Figure 3 right).

To prove this theorem, we interpret rational Bézier triangle p as a composition of two maps:

The first of these two maps is $p^*: \triangle \to \mathbb{R}^3$ with $p^*(u,v) = \sum_{i+j \leq n} p^*_{ij} B^n_{ij}(u,v)$, $p^*_{ij} \in \mathbb{R}^3$ (the homogeneous version of the control points p_{ij}). Technically, this is a *polynomial* triangle in $\mathbb{R}^3$.

The second map is $h: \mathbb{R}^3 \to \mathbb{R}^2$ with $h(x,y,w) = (x/w, y/w)$. This is the central projection onto the plane $w = 1$, followed by omitting the third coordinate.

Observe that $p = h \circ p^*$. Our approach is to, rather than considering the injectivity of the rational triangle map as a whole, show that p^* is injective (on the triangular domain $\triangle$) and h is injective on $p^*(\triangle)$, as long as the conditions from Theorem 1 are satisfied. Together, this implies the injectivity of the composed map.

Injectivity of First Map

We will assume $o = \mathbf{0}$. This is without loss of generality: due to the affine equivariance of Bézier constructions, translations do not affect differential properties like injectivity. In the following we denote by (v^*_i) the homogeneous control vectors corresponding to the homogeneous control points (p^*_i) of the curve.

Condition 1 of the theorem yields $p^*_{0i} = p^*_i$ ($0 \leq i \leq n$) for the first row of the control net.

Let $\overline{E}_i$ denote the (infinite) extrusion of the red cone $\vee^2_i$ along the third dimension. Note that p^*_{0i}, like p'_i, lies on the apex line of this extruded cone, because p^*_{0i} and p'_i agree in the first two coordinates. When we talk about containment of vectors (rather than points) in such a cone in the following, we can drop the index i, as these cones only differ by translation.

Condition 3 implies $p_{1i} \in \vee^2_i$, which implies (due to $w_{1i} = 1$ by Condition 2) $p^*_{1i} \in \overline{E}_i$. Note that this implies that the vector $\Delta^{1*}_{i0} = p^*_{1i} - p^*_{0i}$ is contained in $\overline{E}$. In Figure 4 this is illustrated for the case $i = 1$. Note that we use the shorthands $q_i = p_{1i}$ and $q^*_i = p^*_{1i}$ to refer to the second row of control points.
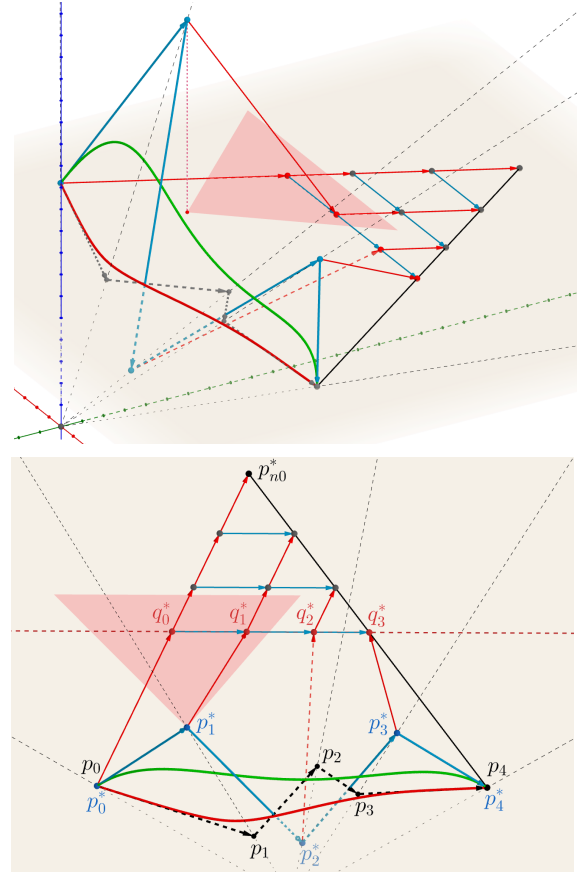
Figure 4: *Perspective side view and orthographic top view of the (polynomial) 3D Bézier triangle p^*. The red cone $\vee^2_1$ (a cross section of extruded cone $\overline{E}_1$) is shown. Notice that the vector $\Delta^{1*}_{10} = q^*_1 - p^*_1$ is contained in the extruded cone $\overline{E}_1$, therefore its image in the orthographic view is contained in the cone $\vee^2_1$.*

Condition 5 together with the constancy of weights due to Condition 2 then implies that not only Δ_{i0}^{1*} but actually all vectors Δ_{ij}^{1*} are contained in $\overline{E}$.

Condition 4 together with the constancy of weights due to Condition 2 asserts that all vectors Δ_{ij}^{0*} for $i \geq 1$ are contained in E, an extruded cone such that the disjoint union $E \cup \overline{E}$ forms a halfspace. The monotonicity of the auxiliary control polygon (p'_i) together with the definition of red cones $\vee^2$ implies that this is also the case for Δ_{ij}^{0*} with $i = 0$ (i.e. the curve's homogeneous control vectors).

Based on these observations, the following lemma can be shown:

Lemma 4 $\boldsymbol{p}^*(u,v)$ is injective on the domain $\triangle$.

Proof The partial derivatives of the map $\boldsymbol{p}^*(u,v)$ are $\partial/\partial_u \boldsymbol{p}^* = n \sum \Delta_{ij}^{0*} B_{ij}^{n-1}$ and $\partial/\partial_v \boldsymbol{p}^* = n \sum \Delta_{ij}^{1*} B_{ij}^{n-1}$ [Far86]. From the above we know that the vectors Δ_{ij}^0 are contained in E, while the vectors Δ_{ij}^1 are contained in $\overline{E}$. Therefore the two partial derivatives, being convex combinations thereof, are contained in E and $\overline{E}$, respectively, as well. As E and $\overline{E}$ are convex (apex angle $< \pi$), these convex combinations do not vanish. Due to the disjointness of E and $\overline{E}$, and their containment in a common halfspace, the partial derivatives are not parallel. Thus, based on (4), $\boldsymbol{p}^*$ is injective. $\square$

Injectivity of Second Map

In order to prove the injectivity of $\boldsymbol{h}$ on $\boldsymbol{p}^*(\triangle)$, we employ the following lemma.

Lemma 5 For a Bézier triangle $\mathcal{T} \subset \mathbb{R}^3$ with control net $\mathcal{N}$, if $\boldsymbol{h}(\mathcal{N})$ forms a simple triangulation (without degenerate or overlapping triangles) then $\boldsymbol{h}(\mathcal{T})$ is injective.

Proof If $\boldsymbol{h}(\mathcal{N})$ is a simple triangulation this implies that any ray emitted from the origin intersects $\mathcal{N}$ in at most one point. Due to the variation diminishing property of Bézier triangles [Far86], any such ray does therefore not intersect $\mathcal{T}$ in more than one point. As $\boldsymbol{h}$ is a projection along these rays, this concludes the proof. $\square$

The fact that the conditions of Theorem 1 imply a simple triangulation under $\boldsymbol{h}$ can then be used to finally show the following:

Lemma 6 If the map $\boldsymbol{p}^*$ is defined by a triangle $\boldsymbol{p}$ that satisfies the five conditions of Theorem 1, then $\boldsymbol{h}$ is injective on $\boldsymbol{p}^*(\triangle)$.

Proof The conditions of Theorem 1 imply that the control points $p_{ij} = \boldsymbol{h}(p_{ij}^*)$ form a simple triangulation: the first row of triangles is non-degenerate and non-inverted due to Condition 3 ($p_{1i} \in \vee_i^1$) (see Figure 5). Due to Conditions 4 and 5, the remaining rows are non-degenerate and non-inverted. Therefore, the entire triangulation $\boldsymbol{h}(\mathcal{N})$ is simple, such that Lemma 5 applies. $\square$

3.3. Construction

We now describe one concrete way to place, in accordance with Theorem 1, the control points and weights defining a rational Bézier triangle conforming to a given guardable rational curve.

The essential degrees of freedom are the placement of points q_i in $\vee_i^1 \cap \vee_i^2$, as these then imply the remaining rows via Conditions 4 and 5. To yield a particularly simple construction, we here consider the placement of these points in the common intersection region of all cones, $\bigcap_i (\vee_i^1 \cap \vee_i^2)$.

1. **Intersection of cones** Let $\vee^1 = \bigcap_i \vee_i^1$ denote the cone that is the intersection of all blue cones, and $\vee^2 = \bigcap_i \vee_i^2$ the intersection of all red cones, as illustrated in Figure 6.

2. **Placing the tip control point.** Now, to define the tip p_{n0} of the conforming element, we first compute point p as the lowest point of intersection region $\vee^1 \cap \vee^2$ with respect to direction $d^\perp$, as shown in Figure 6. Then, we set $p_{n0} = p + (w^2/\hat{w})\mu d^\perp$, where w is the *width* of the curve (the length of the segment $\overline{p_0 p_n}$), $\hat{w}$ is the initial width before any bisections were applied to the curve, and $\mu > 0$ is a parameter. This specific choice in particular of the factor w^2, adopted from [MC20], is important for the termination argument in Sec. 4.1. From the fact that the cones $\vee^1$ and $\vee^2$ contain direction $d^\perp$ by construction, we conclude that p_{n0} is contained in $\vee^1 \cap \vee^2$. The placement of p_{n0} immediately defines the two straight sides of the triangular element.

3. **Placing the remaining control points.** In order to place the remaining points of the control net, we first locate a line parallel to d which intersects both straight sides of the triangle inside the region $\vee^1 \cap \vee^2$ (see Figure 6). For this purpose, we intersect the boundaries of $\vee^1 \cap \vee^2$ with the straight sides and let the line run through the highest intersection point (with respect to $d^\perp$). The line's intersections with the straight sides define q_0 and q_{n-1}. The remaining points can be placed by arbitrary (but ordered) placement of the other q_i in between, arbitrary (but ordered) placement of the other p_{i0} along the left straight edge, and then intersecting two families of parallel lines through these control points to yield the rest.

In Appendix A we describe two more elaborate alternatives (that can be used as drop-in replacements in the main mesh generation algorithm of Sec. 4). One employs a placement in the *individual* cone intersections, the other furthermore employs individual *visibility* cones generalizing $\vee_i^1$, so as to obtain even more flexibility. This allows constructing elements that are less tall (and less distorted) and therefore enables simpler output meshes on average.

4. Meshing Algorithm

Equipped with the construction of conforming injective rational elements, we can now formulate the overall meshing algorithm. As we have ensured that the constructed elements have a number of specific properties, we can closely follow the overall Bézier Guarding algorithm, substituting our rational guardability condition and element construction.
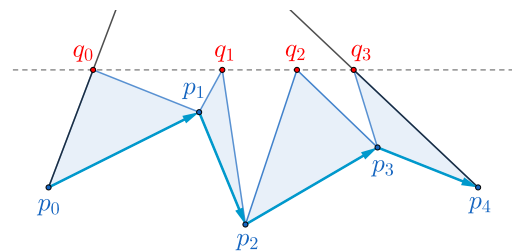
Figure 5: *The first row of control net triangles is non-degenerate and non-inverted due to containment of the second row of control points in cones $\vee_i^1$.*
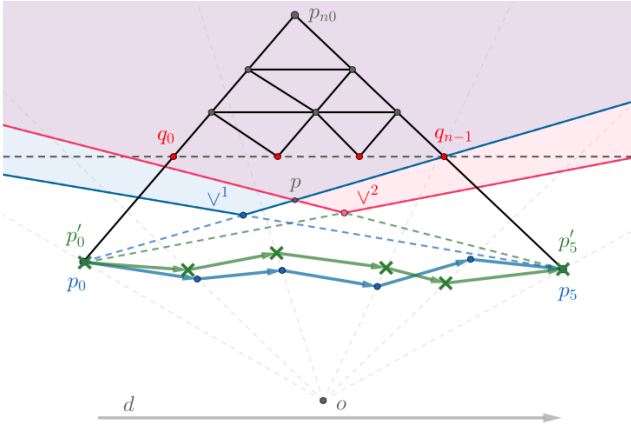
Figure 6: *Placement of the control net for a conforming rational Bézier triangle, using the* simple *placement strategy (Sec. 3.3).*

Input Assumptions The input is a set of rational Bézier curves in 2D, $\boldsymbol{c}_i : [0,1] \to \mathbb{R}^2$, which satisfies the following conditions: (1) they are *regular*, i.e. their derivatives are non-vanishing, $\|\boldsymbol{c}_i'(t)\| \neq 0$ for any $t \in [0,1]$, (2) they do not intersect except perhaps at their endpoints, and (3) they do not form infinitely sharp corners with zero-angles. Reasons for these requirements are analogous to those discussed in [MC20].

Algorithm For such a set of input curves, the algorithm proceeds as follows to generate a valid mesh that conforms to the input.

1. As long as there is a non-guardable rational curve, bisect it (at $t = \frac{1}{2}$). Once guardable, construct a conforming element using the presented construction.

2. As long as any two triangular elements overlap, bisect the curve with the tallest element and reconstruct the conforming elements for the two subcurves.

3. Triangulate the remaining space (or a bounding box), considering the conforming elements as holes.

4. Equip the generated straight-edge elements with a suitable geometric map conforming to neighboring elements (i.e. with shared control points and weights).

For the last step, we employ the same placement rules for control points like the original Bézier Guarding method, with the difference that we additionally set all weights to 1. Note that the curved elements constructed following Sec. 3.3 have unit weights along their edges as well, so this choice is compatible.

4.1. Termination of the Algorithm

The following lemma supports the termination of the algorithm.

Lemma 7 Under repeated bisection, inner angles $\angle p_{n0}p_0p_n$ and $\angle p_{n0}p_np_0$ of elements constructed after Sec. 3.3 converge to zero.

Proof Under subdivision, the control polygon of a regular rational curve converges to the curve in a pointwise manner and the control vectors of a subcurve converge to all be parallel [PK94, LPR12, MG01]. Hence, the blue and red cones converge towards an apex angle of π. As additionally apices of red cones converge to

the control points, thus to the curve, by Lemma 2, the point p constructed in Sec. 3.3 (always well-defined because subcurves remain guardable by Lemma 3) converges to the curve. At the same time, the width w of a curve converges to 0. Hence the tip control point $p_{n0} = p + (w^2/\hat{w})\mu d^\perp$ converges to p, thus to the curve as well. As the convergence of the tip control point to p is asymptotically faster (quadratic in w) than the convergence of the curve width to 0 (linear in w), the inner angles converge to 0. $\qquad\square$

This shows that Lemma 3.7 from [MC20] applies to our construction of rational triangular elements as well. As a consequence, our construction satisfies the requirements for their proof of termination, in particular Proposition 3.8.

5. Results

In the following we demonstrate the construction of single elements as well as the generation of entire meshes on various examples.

Figure 7 illustrates the effect of the choice of point o. Note that the question of guardability as well as the shape of the cones $\vee_i^2$
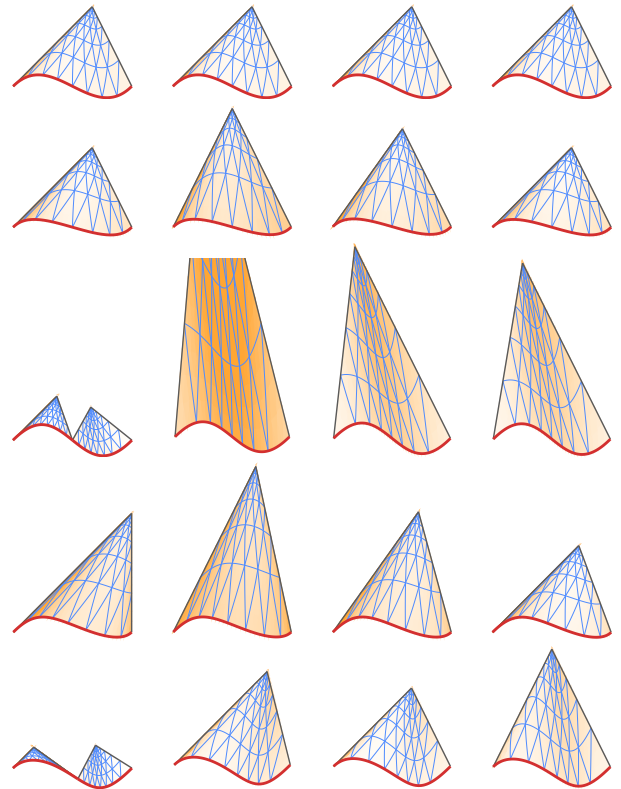
Figure 7: *Effect of weights and choice of point o on the element construction. Shown is a curve with control points $(0,0)$, $(1,1)$, $(2,-1)$, $(3,0)$. From left to right: point o is chosen at p_0, at distance w (curve width) below the midpoint, at distance $\frac{1}{2}w$ below the midpoint, at the midpoint $\frac{1}{2}(p_0 + p_n)$. From top to bottom: weights are $(1,1,1,1)$, $(1,0.5,0.5,1)$, $(1,2,2,1)$, $(1,1,0.5,1)$, $(1,1,2,1)$. Color indicates local MIPS distortion, with white indicating isometry. For the two instances where two triangular elements are shown on the curve, it had to be subdivided for guardability.*

is relative to this choice. This means, different choices will lead to differently sized triangle elements and different numbers of bisections. We find that the simple choice of $o = \frac{1}{2}(p_0 + p_n)$ leads to small elements and a low number of bisections on average, and therefore generally use this fixed choice in all other experiments. Other simple options like $o = p_0$ or o below the midpoint $\frac{1}{2}(p_0 + p_n)$ at various distances (as used in Figures 3 and 6 for less cluttered illustration) proved to behave worse.

Figure 8 illustrates the proper handling of curves forming (non-polynomial) circular and elliptical arcs by our algorithm. Also note the effect of the parameter μ: a larger value results in taller elements which, while having less parametric distortion, may require more curve bisections to generate disjoint elements. We use $\mu = 0.01$ in all other experiments. Figure 9 shows our output on a Fibonacci spiral as well as on a complex mechanical part.

Figure 10 illustrates an extreme scenario. It consists of a sequence of seven rational curves of degree 3 with the weights of inner control points ranging from 0.001 (left curve) to 1000 (right curve). Smaller weights pull auxiliary points p' close to the curve, implying large cone intersections, enabling small elements. By contrast, larger weights push them far away, implying steeper extreme slopes and taller elements, requiring more curve bisections.

We furthermore applied our algorithm to examples from the ABCD dataset of [MC20], adapted to our setting by randomly assigning weights in range $(0.1, 10)$ to the control points, so as to turn the originally polynomial curves into rational curves, of degree 3. Figure 11 shows the output (with up to 5K rational triangles) on one such example from each of the four dataset categories. Our implementation (on a commodity laptop) has the following average run time, start to end of the algorithm, for the input instances from the four dataset categories:

	A	B	C	D
Steps 1+2 (rational elements)	3.2s	3.8s	1.3s	0.35s
Steps 3+4 (linear elements)	0.3s	0.4s	0.1s	0.03s

Output Quality Note that our method focuses on the binary criteria conformance and validity: It guarantees that there is no approximation error and that every element in the resulting mesh is valid. It does not guarantee any controllable lower bounds on continuous
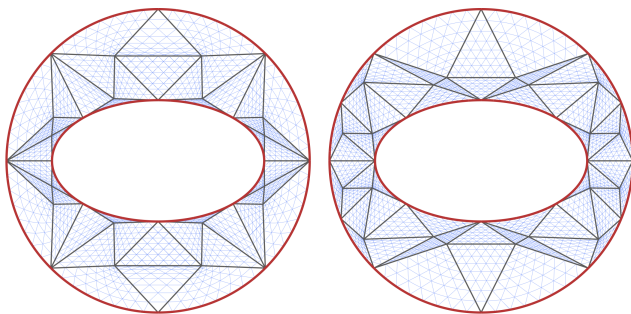

Figure 8: *Meshing of a region enclosed by eight rational curves (degree 2) forming a circle and an ellipse for $\mu = 0.01$ (left) and $\mu = 0.1$ (right); generally, a larger μ generates taller elements which require more subdivision to ensure that the elements are disjoint.*
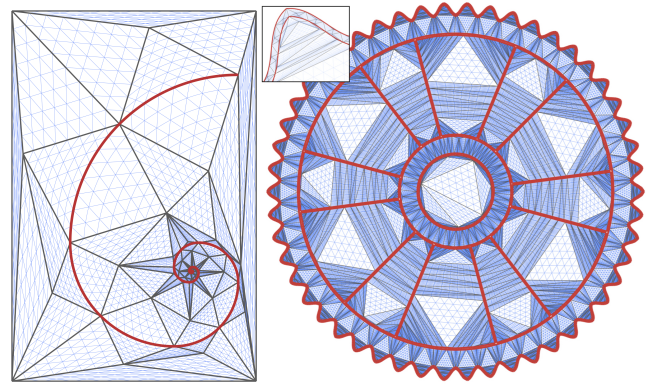

Figure 9: *Results on a Fibonacci spiral and a complex practical configuration. The Fibonacci spiral on the left is made of 13 rational curves of degree 2. The mechanical gear consists of more than 500 rational curves, running very close to each other. This is not an issue for the algorithm, due to its solid foundation.*

mesh *quality* measures beyond that, e.g. regarding angles or geometric map distortion [EE20]; for instance in the complex example of Figure 9 right, the popular *scaled Jacobian* measure goes down to around 10^{-5}. One may adapt quality-improving remeshing techniques to the rational case, for which our output can serve as valid starting point, just as demonstrated for the original Bézier Guarding method; conservative validity checks [EE20] can be used to ensure the preservation of validity throughout. Figure 1 right shows a preliminary demonstration; in-depth exploration of this route is a topic for future work.

6. Conclusion and Future Work

We have introduced the first method to generate rational triangle meshes for planar domains, bounded and constrained by rational curves such as general NURBS (see also Figure 12), such that these meshes are valid and conform to the curves *by construction*. The key contribution is a construction of injective conforming individual elements, with all the specific properties required to incorporate it into the Bézier Guarding approach, that previously only supported conformance to polynomial curves.

For future work we imagine the exploration of a targeted exploitation of the degrees of freedom that the construction offers. For instance, for simplicity we have used a fixed choice of the point o,
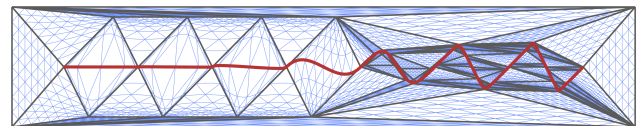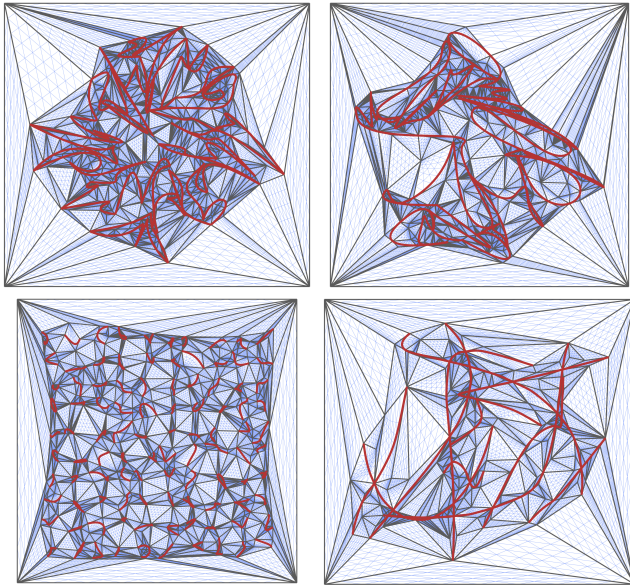

Figure 10: *Extreme cases. The elements resulting from our method are conforming and valid by construction, regardless of weights. Shown in red is a sequence of seven consecutive curves with identical control polygon each (zig-zag shape) but varying weights assigned to their inner control points, from 10^{-3} (left) to 10^3 (right).*

Figure 11: *Sample results for instances from the ABCD stress test dataset [MC20], made rational by assigning random weights.*

while its variation could lead to smaller elements, to faster convergence, to more parsimonious results, and/or to geometric maps of lower distortion. Similarly, the apex point could be flexibly chosen within an entire region rather than at a fixed point.

While the polynomial Bézier Guarding approach was shown to be implementable exclusively using rational arithmetic (enabling exact computation, free of rounding error), weight normalization in the rational case involves roots. While there are exact computation techniques supporting roots, these likely come with a significant performance overhead.

An extension of the overall approach to the three-dimensional setting (e.g. using tetrahedral meshes) is another worthwhile research direction, which comes with a wealth of additional challenges not present in the two-dimensional scenario.

Acknowledgements
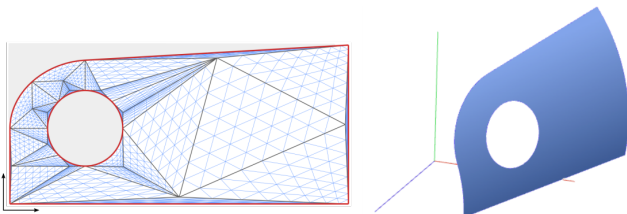
Figure 12: *Rational mesh generated for the parameter domain (left), bounded by NURBS trimming curves (linear and rational quadratic pieces), of a 3D trimmed NURBS surface patch (right).*

References

[ADF14] ABGRALL R., DOBRZYNSKI C., FROEHLY A.: 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. 2

[BECN*22] BARROSO E. S., EVANS J. A., CAVALCANTE-NETO J. B., VIDAL C. A., PARENTE E.: An efficient automatic mesh generation algorithm for planar isogeometric analysis using high-order rational Bézier triangles. *Engineering with Computers* (2022), 1–22. 2

[DOS99] DEY S., O'BARA R. M., SHEPHARD M. S.: Curvilinear mesh generation in 3D. In *Proc. International Meshing Roundtable* (1999), pp. 407–417. 2

[DOS01] DEY S., O'BARA R. M., SHEPHARD M. S.: Towards curvilinear meshing in 3D: the case of quadratic simplices. *Computer-Aided Design 33*, 3 (2001), 199–209. 2

[EE16] ENGVALL L., EVANS J. A.: Isogeometric triangular Bernstein–Bézier discretizations: Automatic mesh generation and geometrically exact finite element analysis. *Comput. Methods Appl. Mech. Eng. 304*, C (2016). 2

[EE20] ENGVALL L., EVANS J. A.: Mesh quality metrics for isogeometric Bernstein–Bézier discretizations. *Computer Methods in Applied Mechanics and Engineering 371* (2020). 2, 8

[Far86] FARIN G.: Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design 3*, 2 (1986), 83–127. 6

[Far02] FARIN G.: *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. Morgan Kaufmann, San Francisco, CA, USA, 2002. 3

[FP16] FORTUNATO M., PERSSON P.-O.: High-order unstructured curved mesh generation using the winslow equations. *Journal of Computational Physics 307* (2016), 1 – 14. 2

[GB12] GEORGE P., BOROUCHAKI H.: Construction of tetrahedral meshes of degree two. *Int. J. Numer. Methods Eng. 90*, 9 (2012). 2

[GH73] GORDON W. J., HALL C. A.: Transfinite element methods: Blending-function interpolation over arbitrary curved element domains. *Numerische Mathematik 21*, 2 (1973), 109–129. 2

[GPRPS13] GARGALLO-PEIRÓ A., ROCA X., PERAIRE J., SARRATE J.: High-order mesh generation on cad geometries. *International Conference on Adaptive Modeling and Simulation VI* (2013). 2

[HSG*19] HU Y., SCHNEIDER T., GAO X., ZHOU Q., JACOBSON A., ZORIN D., PANOZZO D.: Triwild: Robust triangulation with curve constraints. *ACM Trans. Graph. 38*, 4 (2019). 2

[JQ14] JAXON N., QIAN X.: Isogeometric analysis on triangulations. *Computer-Aided Design 46* (2014), 45–57. 2

[LJ19] LIU N., JEFFERS A. E.: Feature-preserving rational Bézier triangles for isogeometric analysis of higher-order gradient damage models. *Comput. Meth. in Applied Mechanics and Engineering 357* (2019). 2

[LPR12] LI J., PETERS T. J., ROULIER J. A.: Angular convergence during Bézier curve approximation. *arXiv:1210.2686* (2012). 7

[LSO*04] LUO X.-J., SHEPHARD M. S., O'BARA R. M., NASTASIA R., BEALL M. W.: Automatic p-version mesh generation for curved domains. *Eng. with Comput. 20*, 3 (2004), 273–285. 2

[Mäk05] MÄKIPELTO J.: Exact geometry description with unstructured triangular meshes for shape optimization. In *Proc. 6th World Congress of Structural and Multidisciplinary Optimization, Brazil* (2005). 2

[MC20] MANDAD M., CAMPEN M.: Bézier Guarding: Precise higher-order meshing of curved 2D domains. *ACM Trans. Graph. 39*, 4 (2020). 1, 2, 4, 6, 7, 8, 9

[MC21] MANDAD M., CAMPEN M.: Guaranteed-quality higher-order triangular meshing of 2D domains. *ACM Trans. Graph. 40*, 4 (2021), 1–14. 1, 2

[MEK*16] MOXEY D., EKELSCHOT D., KESKIN U., SHERWIN S., PEIRÓ J.: High-order curvilinear meshing using a thermo-elastic analogy. *Comput. Aided Des. 72*, C (2016), 130–139. 2

[MG01] MORIN G., GOLDMAN R.: On the smooth convergence of sub-division and degree elevation for Bézier curves. *Comput. Aided Geom. Des. 18*, 7 (2001), 657–666. 7

[Miu93] MIURA K. T.: G^2 continuous interpolation over rational curve meshes. In *Communicating with Virtual Worlds*. Springer, 1993. 2

[Oli08] OLIVER T. A.: *A High-Order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-Averaged Navier-Stokes Equations*. PhD thesis, MIT, 2008. 2

[Pau18] PAUL J.: Orientation preserving mesh optimisation and preconditioning. *Int. J. Numer. Methods Eng. 114*, 7 (2018), 749–776. 2

[PK94] PRAUTZSCH H., KOBBELT L.: Convergence of subdivision and degree elevation. *Advances in Computational Mathematics 2*, 1 (1994), 143–154. 4, 7

[PP09] PERSSON P.-O., PERAIRE J.: Curved mesh generation and mesh refinement using lagrangian solid mechanics. *47th AIAA Aerospace Sciences Meeting* (2009). 2

[PSG16] POYA R., SEVILLA R., GIL A. J.: A unified approach for a posteriori high-order curved mesh generation using solid mechanics. *Computational Mechanics 58*, 3 (2016), 457–490. 2

[RGPS11] ROCA X., GARGALLO-PEIRÓ A., SARRATE J.: Defining quality measures for high-order planar triangles and curved mesh generation. In *Proc. Int. Meshing Roundtable*. 2011, pp. 365–383. 2

[RGSR16] RUIZ-GIRONÉS E., SARRATE J., ROCA X.: Generation of curved high-order meshes with optimal quality and geometric accuracy. *Procedia Engineering 163* (2016), 315 – 327. 2

[RL14] RANGARAJAN R., LEW A. J.: Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes. *Int. J. Numer. Methods Eng. 98*, 4 (2014), 236–264. 2

[SFJ*05] SHEPHARD M. S., FLAHERTY J. E., JANSEN K. E., LI X., LUO X., CHEVAUGEON N., REMACLE J.-F., BEALL M. W., O'BARA R. M.: Adaptive mesh generation for curved domains. *Appl. Numer. Math. 52*, 2 (2005), 251–271. 2

[SP02] SHERWIN S. J., PEIRÓ J.: Mesh generation in curvilinear domains using high-order elements. *Int. J. Numer. Methods Eng. 53*, 1 (2002), 207–223. 2

[SRH16] SEVILLA R., REES L., HASSAN O.: The generation of triangular meshes for NURBS-enhanced FEM. *Int. J. Num. Methods Engrg. 108*, 8 (2016), 941–968. 2

[TGRL13] TOULORGE T., GEUZAINE C., REMACLE J.-F., LAMBRECHTS J.: Robust untangling of curvilinear meshes. *Journal of Computational Physics 254* (2013), 8 – 26. 2

[TPM18] TURNER M., PEIRÓ J., MOXEY D.: Curvilinear mesh generation using a variational framework. *Computer-Aided Design 103* (2018), 73–91. 2

[XC14] XU J., CHERNIKOV A. N.: Automatic curvilinear quality mesh generation driven by smooth boundary and guaranteed fidelity. *Procedia Engineering 82* (2014), 200 – 212. 2

[XSHM13] XIE Z. Q., SEVILLA R., HASSAN O., MORGAN K.: The generation of arbitrary order curved meshes for 3D finite element analysis. *Computational Mechanics 51*, 3 (2013), 361–374. 2

[YLC*22] YANG J., LIU S., CHAI S., LIU L., FU X.-M.: Precise high-order meshing of 2D domains with rational Bézier curves. *Computer Graphics Forum 41*, 5 (2022). 2

[Zla73] ZLAMAL M.: The finite element method in domains with curved boundaries. *Int. J. Numer. Methods Eng. 5*, 3 (1973), 367–373. 2

Appendix A: Alternative Element Constructions

The control point placement strategy from Sec. 3.3 is a particularly simple one. In the following we describe a more flexible construction that exploits the individual cone intersections $\vee_i^1 \cap \vee_i^2$, as well as a further generalization, which uses larger *visibility cones* $\vee_i^3$ in
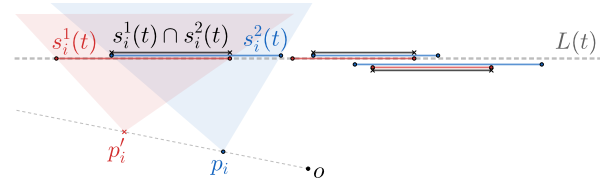
Figure 13: *The red and blue cones v_i induce segments $s_i^1(t)$ and $s_i^2(t)$ on the line $L(t)$. Their intersections, kernel segments, are indicated in black. All segments are well-ordered within their class.*
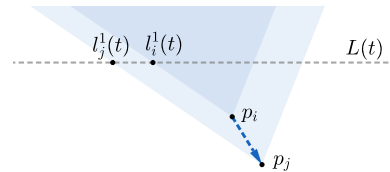
Figure 14: *The chain of control vectors connecting p_i to p_j must have a vector with slope greater than the boundaries'.*

place of guarding cones $\vee_i^1$. Both enable the construction of less tall triangle elements, with efficiency benefits for the overall algorithm.

We assume that a (sub)curve $\{p_i, w_i\}$ is given that is rational-guardable with respect to a direction d, and define:

- Let $L(t)$ a line parallel to d, with signed distance t to p_0 (increasing in direction d^T). On this line (for a suitable t) the second row of the control net, the points $\{q_i\}$, will be placed.
- Let $s_i^1(t) = [l_i^1(t), r_i^1(t)] = \vee_i^1 \cap L(t)$ denote the segment of $L(t)$ inside the cone $\vee_i^1$. Analogously, the segments s_i^2 are defined based on cones $\vee_i^2$. Their intersections $k_i(t) = [l_i'(t), r_i'(t)] = s_i^1(t) \cap s_i^2(t)$ we call *kernel segments*. See Figure 13.
- Let $t_{\min}$ denote the smallest value of t such that $k_i(t) \neq \varnothing$ for all $0 \le i \le n-1$. As the direction d^T is contained in both types of cones, the segments $k_i(t)$ grow strictly monotonically with t.

Proposition 1 For any fixed value $t > t_{\min}$, the segments $s_i^1(t)$ are *well-ordered* along L, i.e. $l_i^1(t) \le l_{i+1}^1(t)$ and $r_i^1(t) \le r_{i+1}^1(t)$. The same holds for segments $s_i^2(t)$ and kernel segments $k_i(t)$.

Proof Suppose $i < j$ but $l_i^1(t) > l_j^1(t)$ (see Figure 14). Then, the chain of control vectors from p_i to p_j must contain a vector v for which the value $d^\mathsf{T} v / ||v||$ is smaller than the value for v_-, which contradicts the definition of v_-. The argument for the right endpoints $r_i^1(t)$ is similar. Analogously, based on the monotonicity of the auxiliary control polygon $\{p_i'\}$ with respect to direction d, the well-orderedness of the segments $s_i^2(t)$ for $t > t_{\min}$ is shown. The kernel segments $k_i(t)$ are well-ordered because they are pairwise intersections of these well-ordered segments. $\square$

This well-orderedness implies that if one chooses a placement of a *subset* of the points $\{q_i\}$ on L such that $q_i \in k_i$, this partial placement can always be completed, i.e. there always is room to place each remaining point q_i such that it lies in k_i *and* between q_{i-1} and q_{i+1} along L. We exploit this property in the following: We first fix the outermost points q_0 and q_{n-1} (which together with points p_0 and p_n define the straight edges of the element), and know that the remaining control points of the second row can always be placed in between in accordance with the requirements of Theorem 1.
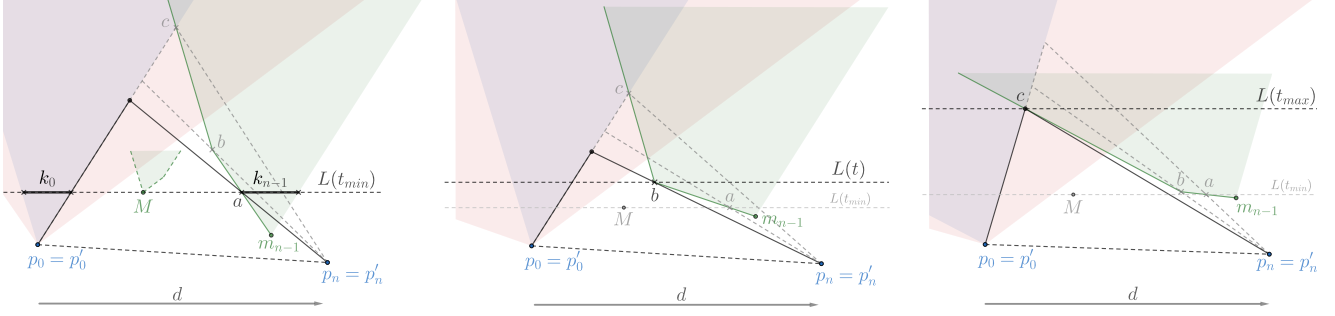
Figure 15: *Constructing the minimum height triangle based on individual cone pair intersections. Shown are the three possible height-minimal choices of t: near $t_{\min}$ (left), at the kink b of the region $\vee_{n-1}^1 \cap \vee_{n-1}^2$ (green, center), and near $t_{\max}$ (right).*

Using Individual Cone Pair Intersections

This element construction strategy considers the placement of points q_i inside the individual pairs of red and blue cones, i.e. $q_i \in \vee_i^1 \cap \vee_i^2$. The conceptual idea is to, in order to yield an element of low height, place q_0 rightmost in its feasible space, i.e. at the end point $r_0'(t)$ of kernel segment $k_0(t)$, and q_{n-1} leftmost, i.e. at the start point $l_{n-1}'(t)$ of kernel segment $k_{n-1}(t)$. These two points define the two straight sides of the triangle element, therefore they determine the position of the tip vertex p_{n0} and therefore the overall element height. See Figure 15 left for an illustration.

Our degree of freedom is the choice of $t > t_{\min}$. We first compute $t_{\min}$ by computing the lowest (relative to direction d^T) point m_i of each region $\vee_i^1 \cap \vee_i^2$. The boundary of such a region is an (unbounded) polygon, with at most three vertices; m_i is one of these three points. Let M denote the highest of these points m_i; this determines $t_{\min}$ such that line $L(t_{\min})$ runs through M (Figure 15 left).

Now let q_0 and q_{n-1} be the innermost points of the segments $k_0(t_{\min})$ and $k_{n-1}(t_{\min})$, respectively, shifted by $\mu(w^2/\hat{w})d^\mathsf{T}$ (similar to the shift performed in Sec. 3.3) to ensure they lie *inside* the cones, not on their boundary. Notice that they then lie on $L(t)$ with $t = t_{\min} + \mu(w^2/\hat{w}) > t_{\min}$ as required.

After defining q_0 and q_{n-1} in this way, we position the remaining $\{q_i\}$ by the following assignment (starting from $i = 1$), which due to well-orderedness of the kernel segments leads to properly ordered points: $q_i \leftarrow \frac{1}{2}\left(\max\{q_{i-1}, l_i'\} + \min\{q_{n-1}, r_i'\}\right)$. Finally, the rest of the control net is completed just as in Sec. 3.3 step 3.

Remark: Interestingly, in some cases performing the above construction not based on $t_{\min}$ but starting from a larger t can lead to an even less tall element. This is due to the fact that, while $\vee_0^1 \cap \vee_0^2$ (which induces $k_0(t)$) is simply a cone rooted at $p_0 = p_0'$, the region $\vee_{n-1}^1 \cap \vee_{n-1}^2$ has a more complex shape, bounded by up to four linear pieces. Depending on their slope relative to the element, increasing t can lead to the kernel segment $k_{n-1}(t)$ growing inwards at a rate that implies a smaller element. Due to the piecewise linearity of the cone intersection region, however, the optimal t can only lie at a kink of the region boundary (as illustrated in Figure 15 center), and we may not choose a $t \geq t_{\max}$, which is where $k_0(t)$ and $k_{n-1}(t)$ start overlapping (Figure 15 right), invalidating the construction. This small number of options can easily be checked to determine the one inducing the smallest element.

Using Visibility Cones

For even larger placement flexibility, we can actually replace the cones $\vee_i^1$ by larger *visibility* cones $\check{\vee}_i^1 \supseteq \vee_i^1$. Visibility cone $\check{\vee}_i^1$ contains all points r in the plane that can see the (oriented) control segment $\overline{p_i p_{i+1}}$, i.e. such that (p_i, p_{i+1}, r) is a positively oriented triangle whose interior does not intersect the control polygon. Let $\check{s}_i^1(t)$ denote the *visibility segment* $L(t) \cap \check{\vee}_i^1$, i.e. any point $p \in \check{s}_i^1(t)$ forms, with p_i and p_{i+1}, such a proper triangle.

The important observation is that placement of the second-row control points q_i in the larger segment $\check{s}_i^1(t)$ (if ordered properly) still ensures a simple triangulation $\mathbf{h}(\mathcal{N})$ (as required by Lemma 5); containment in $s_i^1(t)$, like we ensured before, is not a necessary requirement for this.

Therefore, we can use the larger intersection segments $\check{s}_i^1(t) \cap s_i^2(t) = [\bar{l}_i(t) \ \bar{r}_i(t)]$ rather than the above kernel segments. Note, however, that the well-ordering property does not generally hold for these segments. We therefore define *trimmed segments* $\check{k}_i(t)$, by replacing bounds (starting from $i = 0$) as follows:

$$\bar{l}_i(t) \leftarrow \max_{j \leq i}\{\bar{l}_j(t)\}, \quad \bar{r}_i(t) \leftarrow \min_{i \leq j}\{\bar{r}_j(t)\}.$$

Note that this is equivalent to:

$$\bar{l}_i(t) = \max_{j \leq i}\{\check{l}_j^1(t), l_j^2(t)\}, \quad \bar{r}_i(t) = \min_{i \leq j}\{\check{r}_j^1(t), r_j^2(t)\}.$$

These trimmed segments, which still are supersets of the kernel segments $k_i(t)$, are well-ordered by construction, and $\bar{l}_i(t)$ monotonically decreases with t, $\bar{r}_i(t)$ monotonically increases with t. We can therefore apply the same construction as before, just exchanging $k_i(t)$ for $\check{k}_i(t) = [\bar{l}_i(t) \ \bar{r}_i(t)]$. Some added complexity is due to the fact that $\bar{l}_i(t)$ and $\bar{r}_i(t)$ are piecewise linear functions with up to n linear pieces (because $\check{l}_j^1, \check{r}_j^1, l_j^2, r_j^2$ are linear), whereas the bounds of $k_i(t)$ have only one or two linear pieces (like the bold green left bound of k_{n-1} in Figure 15). This is relevant because

- for the computation of the points m_i (to determine M and $t_{\min}$), these two piecewise linear functions need to be intersected;
- for the determination of the height-optimal $t > t_{\min}$, the kinks of $\bar{r}_0(t)$ and $\bar{l}_{n-1}(t)$ need to be examined (which now can be more than just one).

It can be shown that, at least for a choice of d parallel to $\overline{p_0 p_n}$ and sufficiently small μ, the minimum element height is assumed when $L(t)$ runs through one of these kinks.