DOI: 10.1111/cgf.14604
Eurographics Symposium on Geometry Processing 2022
M. Campen and M. Spagnuolo
(Guest Editors)

Volume 41 (2022), Number 5

Precise High-order Meshing of 2D Domains with Rational Bézier Curves

Jinlin Yang Shibo Liu Shuangming Chai[†] Ligang Liu Xiao-Ming Fu

University of Science and Technology of China

Abstract

We propose a novel method to generate a high-order triangular mesh for an input 2D domain with two key characteristics: (1) the mesh precisely conforms to a set of input piecewise rational domain curves, and (2) the geometric map on each curved triangle is injective. Central to the algorithm is a new sufficient condition for placing control points of a rational Bézier triangle to guarantee that the conformance and injectivity constraints are theoretically satisfied. Taking advantage of this condition, we provide an explicit construct that robustly creates higher-order 2D meshes satisfying the two characteristics. We demonstrate the robustness and effectiveness of our algorithm over a data set containing 2200 examples.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;***

1. Introduction

High-order meshes consist of curved elements rather than linear elements with straight edges, which are widely used in physics simulation, geometric modeling, animation, and nonphotorealistic rendering [HSG*19]. Many previous literatures have discussed and demonstrated that high-order meshes possess potential superiority over linear meshes [Zlá73, WFA*13]. Thus, high-order meshing is a fundamental task in physical simulation and geometric modeling.

Many methods have been proposed for high-order meshing. Most of them are able to either generate valid curved meshes or conform to given input domain curves. A high-order mesh is said to be valid if the geometric map of each curved element is injective, i.e., the Jacobian of the geometric map is strictly positive definite everywhere within the domain. Two recently proposed algorithms take both validity and conformance constraints into account [MC20, MC21]; however, only polynomial curves are considered in their work.

We study high-order meshing conforming to piecewise rational curves for 2D domains (Figure 1). Although geometric designs using higher-order rational curves are less common, it is common to elevate the order of the curve to achieve higher-order meshes for higher simulation accuracy of finite element analysis [SB21].

This is not a straightforward extension from polynomial curves to rational curves due to the following two reasons. It is difficult to find a sufficient condition to ensure the validity for rational inputs,

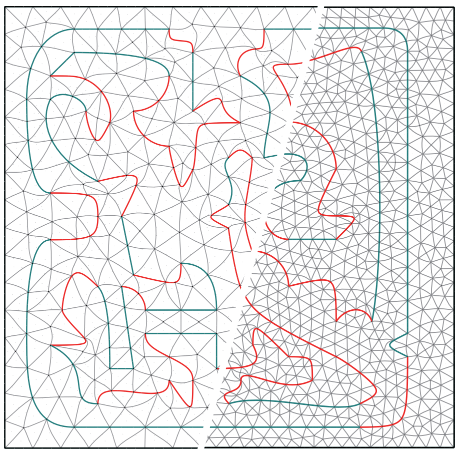
Figure 1: *Given input conics, cubic rational curves (red), and polynomial curves (green), our method generates valid high-order meshes conforming the input curves. We show the meshes in different resolutions. Rational curves have weights ranging from* 1 *to* 10.

and it is even more complicated to construct a valid curved mesh according to the sufficient condition. In [EE20], a sufficient condition is proposed to check the validity of rational Bernstein-Bézier elements, but it does not provide a geometrically intuitive and explicit way to construct control points while guaranteeing validity. To the best of our knowledge, no previous methods generate valid meshes that precisely conform to rational Bézier domain curves, especially

[†] The corresponding author

conics and cubic rational curves widely used in industry and design, with a theoretical guarantee.

In this paper, we propose a novel and efficient method to generate high-order triangular meshes that (1) have an injective geometric map on each curved triangle and (2) precisely conform to the input rational Bézier domain curves. By revisiting [JW94] that only calculates a sufficient condition to injectivity of rational quadratic Bézier triangles, we state and prove a new sufficient condition for rational Bézier triangles with arbitrary orders and positive weights. After embedding our condition into the Bézier guarding framework [MC20], we place the control points of quadratic and cubic curved triangles to meet the conformance and injectivity constraints, thereby constructing a valid high-order mesh. Then, the valid mesh is improved in reducing geometric map distortion by several developed practical strategies. We demonstrate the effectiveness and robustness of our method over a data set containing 2200 examples.

2. Related work

High-order curved mesh generation methods have gained increased popularity during the last decade owing to the potential of providing higher accuracy [DOS99]. We briefly review the most relevant work about high-order mesh generation considering injectivity and conformance constraints.

To theoretically guarantee injective geometric maps, the common high-order meshing methods first create a linear triangulation and then deform the linear mesh to be curved while aligning with the input curves and preserving injectivity [ADF14, DOS01, DOS99, HSG*19, MEK*16, GB12, CCM*04, LSO*04, SFJ*05, FP16, PP09, PSG16, RGPS11, RGSR16, SP02, TGRL13, TLR16, TPM18, XSHM14, XC14, LSL*21]. These methods mainly differ in the use of different methods and energies to drive the deformation. When preserving injectivity is a hard constraint of the deformation, there is no theoretical guarantee that the conformance constraint is always satisfied.

Using a curved mesh that strictly conforms to the characteristic curve and boundary has a lot of advantages as the geometry approximation errors are eliminated [SRH16, DOS99, EE16, JQ14, RL14, ADF14]. For instance, an accurate representation of the geometry enables to propagate vortices over long distances in fluid mechanics [SFMH08, SRH16]. In linear elasticity problems, when the order of the geometric approximation is lower than the order of the functional interpolation, the numerical solution has sizable errors [LSR01]. To theoretically guarantee that the conformance constraint is strictly satisfied, methods create elements with curved edges along the input curves (similar to linear advancing front methods) [DOS99] or directly fit (or snap) the nodes of a curved mesh to those of the input curves [GTNI16]. If the mesh is invalid, the untangling (or removing flipped mappings) techniques [TGRL13, TLR16] can be used to remove the invalid curved elements. However, these methods cannot theoretically guarantee that there are no invalid curved elements for any input curves [SRH16].

The recently proposed approaches [MC20, MC21] have a theoretical guarantee to generate valid 2D high-order meshes, which do not violate the injectivity or conformance constraints. However, they do not support rational curves, such as arcs and general conics, which

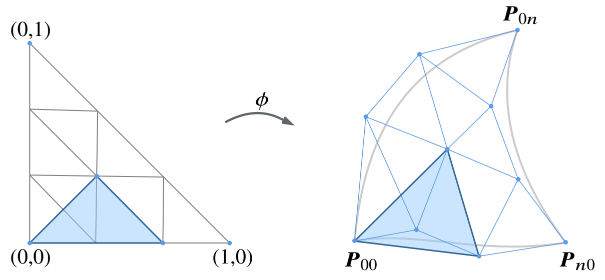

Figure 2: *A rational Bézier map ϕ (cubic in this example) from a reference unit triangle to the planar three-sided domain. A blue triangle in the parameter domain (left) corresponds to a straight triangle in the computational domain (right).*

are practically important. We generalize [MC20] to rational Bézier curves via a new explicit construction method based on a sufficient condition, while still ensuring injectivity and conformance. There is a concurrent work that solves the same problem [KMC22].

3. Problem statement

Input Our input is a set of curves C, called domain curves, containing line segments, polynomial curves of arbitrary orders, conic sections, and cubic rational curves. We assume that the parameter t of each domain curve $\boldsymbol{c} \in C$ is defined on $[0,1]$, and each control point has a positive weight. To ensure validity, we further assume each domain curve $\boldsymbol{c}$ meets the following conditions: (1) $\|\boldsymbol{c}'(t)\| \neq 0, \forall t \in [0,1]$, (2) there is no degeneracy, i.e., two curves do not form an angle-zero corner at coincident endpoints, and (3) the curve intersects with other curves only at the endpoints.

Goal Our goal is to construct a valid planar high-order mesh $M = (V,E,F)$ such that: (1) it conforms to a given domain formed by all the domain curves, i.e., the edges E are line segments or rational Bézier curves, and (2) each triangle element in F can be represented as a planar rational Bézier triangle with a valid geometric map, i.e., the Jacobian of the geometric map is strictly positive definite everywhere within the triangle.

Challenges The main challenges are twofold. First, there is no existing suitable condition to guarantee injectivity for rational Bézier triangles due to the complexity of the Jacobian. Second, it is nontrivial to use this condition to construct a valid geometric map on each triangular element. To overcome these challenges, we introduce an explicit construction method for the control point placement (Section 5.2) based on the sufficient condition (Section 4).

4. A sufficient condition for validity

Preliminaries Let T be a unit isosceles right triangle in the parameter domain $\mathbb{R}^2$ with barycentric coordinates $\xi = (\xi_1, \xi_2, \xi_3)$, $\xi_1 + \xi_2 + \xi_3 = 1$. Denote $\phi : T \rightarrow \mathbb{R}^2$ as a rational Bézier map [Far86] of degree n from the parameter domain to the computational domain:

$$\phi(\xi) = \frac{\sum_{|\lambda|=n} w_\lambda \boldsymbol{P}_\lambda B_\lambda^n(\xi)}{\sum_{|\lambda|=n} w_\lambda B_\lambda^n(\xi)}, \tag{1}$$

where $\lambda = (i,j,k), i,j,k \in \{0,1\ldots,n\}, |\lambda| := i+j+k, \boldsymbol{P}_\lambda = (x_\lambda, y_\lambda)$ are control points, with associated weights w_λ, and the basis functions $B^n_\lambda(\xi) = \frac{n!}{\lambda!}\xi_1^i\xi_2^j\xi_3^k$, where $\lambda! := i!j!k!$. Sometimes, we also use the subscript (i,j) instead of (i,j,k).

The map (1) can be written in a homogeneous manner:

$$\tilde{\phi}(\xi) = (X(\xi), Y(\xi), W(\xi)) = \sum_{|\lambda|=n} \tilde{\boldsymbol{P}}_\lambda B^n_\lambda(\xi), \tag{2}$$

where $\tilde{\boldsymbol{P}}_\lambda = (w_\lambda x_\lambda, w_\lambda y_\lambda, w_\lambda)$. The "Dir" function is defined as [SWS95]:

$$\mathrm{Dir}(\tilde{\boldsymbol{P}}_{\lambda_1}, \tilde{\boldsymbol{P}}_{\lambda_2}) := w_{\lambda_1} w_{\lambda_2}(\boldsymbol{P}_{\lambda_2} - \boldsymbol{P}_{\lambda_1}).$$

This function can be applied to any two homogeneous coordinates. Then, we use the "Dir" function to calculate the partial derivatives:

$$\phi_{\xi_1}(\xi) = \frac{\partial}{\partial \xi_1}\left(\frac{X(\xi)}{W(\xi)}, \frac{Y(\xi)}{W(\xi)}\right) = \frac{\mathrm{Dir}(\tilde{\phi}(\xi), \tilde{\phi}_{\xi_1}(\xi))}{(W(\xi))^2}. \tag{3}$$

Analogously, we have:

$$\phi_{\xi_2}(\xi) = \frac{\mathrm{Dir}(\tilde{\phi}(\xi), \tilde{\phi}_{\xi_2}(\xi))}{(W(\xi))^2}. \tag{4}$$

In addition, we use $\det(\lambda_1, \lambda_2, \lambda_3)$ to denote the mixed product,

$$\det(\lambda_1,\lambda_2,\lambda_3) = \begin{vmatrix} i_1 & i_2 & i_3 \\ j_1 & j_2 & j_3 \\ k_1 & k_2 & k_3 \end{vmatrix}.$$

In [JW94], a sufficient condition for injectivity of quadratic rational Bézier triangles is calculated with the help of the Maple algebraic manipulation language. We generalize the condition to any order n.

Theorem 1 Suppose a rational Bézier triangle map ϕ of order n satisfies the following conditions:

1. The weights $\{w_\lambda\}$ are positive;
2. No degeneracy at corners, i.e. $\det(\tilde{\boldsymbol{P}}_{00}, \tilde{\boldsymbol{P}}_{10}, \tilde{\boldsymbol{P}}_{01}) > 0$, $\det(\tilde{\boldsymbol{P}}_{0n}, \tilde{\boldsymbol{P}}_{0(n-1)}, \tilde{\boldsymbol{P}}_{1(n-1)}) > 0$, $\det(\tilde{\boldsymbol{P}}_{n0}, \tilde{\boldsymbol{P}}_{(n-1)1}, \tilde{\boldsymbol{P}}_{(n-1)0}) > 0$;
3. $\det(\lambda_1,\lambda_2,\lambda_3)\det(\tilde{\boldsymbol{P}}_{\lambda_1}, \tilde{\boldsymbol{P}}_{\lambda_2}, \tilde{\boldsymbol{P}}_{\lambda_3}) \geq 0$ for arbitrary three control points $\boldsymbol{P}_{\lambda_1}, \boldsymbol{P}_{\lambda_2}, \boldsymbol{P}_{\lambda_3}$.

Then ϕ is valid, i.e. $\det J_\phi > 0$ for any ξ in T.

Proof The homogeneous Bézier map and its partial derivatives are written as

$$\tilde{\phi}(\xi) = \xi_1 \tilde{F}_1(\xi) + \xi_2 \tilde{F}_2(\xi) + \xi_3 \tilde{F}_3(\xi),$$

$$\tilde{\phi}_{\xi_1}(\xi) = n(\tilde{F}_1(\xi) - \tilde{F}_3(\xi)),$$

$$\tilde{\phi}_{\xi_2}(\xi) = n(\tilde{F}_2(\xi) - \tilde{F}_3(\xi)),$$

with

$$\tilde{F}_1(\xi) = \sum_{|\mu|=n-1} \tilde{\boldsymbol{P}}_{\mu+e_1} B^{n-1}_\mu(\xi) \triangleq (\tilde{X}_1, \tilde{Y}_1, \tilde{W}_1),$$

$$\tilde{F}_2(\xi) = \sum_{|\mu|=n-1} \tilde{\boldsymbol{P}}_{\mu+e_2} B^{n-1}_\mu(\xi) \triangleq (\tilde{X}_2, \tilde{Y}_2, \tilde{W}_2),$$

$$\tilde{F}_3(\xi) = \sum_{|\mu|=n-1} \tilde{\boldsymbol{P}}_{\mu+e_3} B^{n-1}_\mu(\xi) \triangleq (\tilde{X}_3, \tilde{Y}_3, \tilde{W}_3),$$

where $\mu = (i,j,k)$, $e_1 = (1,0,0)$, $e_2 = (0,1,0)$, and $e_3 = (0,0,1)$. With the properties of the "Dir" function, we have:

$$\begin{aligned}
\det J_\phi &= \begin{vmatrix} \frac{\partial \phi(\xi)}{\partial \xi_1} & \frac{\partial \phi(\xi)}{\partial \xi_2} \end{vmatrix} \\
&= \frac{1}{(W(\xi))^4} \begin{vmatrix} \mathrm{Dir}(\tilde{\phi}(\xi), \tilde{\phi}_{\xi_1}(\xi)) & \mathrm{Dir}(\tilde{\phi}(\xi), \tilde{\phi}_{\xi_2}(\xi)) \end{vmatrix} \\
&= \frac{n^2}{(W(\xi))^3}\left(\tilde{W}_3 \begin{vmatrix} \tilde{X}_1 & \tilde{X}_2 \\ \tilde{Y}_1 & \tilde{Y}_2 \end{vmatrix} + \tilde{W}_2 \begin{vmatrix} \tilde{X}_3 & \tilde{X}_1 \\ \tilde{Y}_3 & \tilde{Y}_1 \end{vmatrix} + \tilde{W}_1 \begin{vmatrix} \tilde{X}_2 & \tilde{X}_3 \\ \tilde{Y}_2 & \tilde{Y}_3 \end{vmatrix}\right) \\
&= \frac{n^2}{(W(\xi))^3} \sum_{|\mu_1|=n-1} \sum_{|\mu_2|=n-1} \sum_{|\mu_3|=n-1} \Big[c_{\mu_1\mu_2\mu_3} \\
&\qquad B^{n-1}_{\mu_1}(\xi) B^{n-1}_{\mu_2}(\xi) B^{n-1}_{\mu_3}(\xi)\Big],
\end{aligned} \tag{5}$$

where the coefficient

$$\begin{aligned}
c_{\mu_1\mu_2\mu_3} &= w_{\mu_1+e_1} w_{\mu_2+e_2} w_{\mu_3+e_3} \Big(\begin{vmatrix} \boldsymbol{P}_{\mu_1+e_1} & \boldsymbol{P}_{\mu_2+e_2} \end{vmatrix} \\
&\quad + \begin{vmatrix} \boldsymbol{P}_{\mu_3+e_3} & \boldsymbol{P}_{\mu_1+e_1} \end{vmatrix} + \begin{vmatrix} \boldsymbol{P}_{\mu_2+e_2} & \boldsymbol{P}_{\mu_3+e_3} \end{vmatrix} \Big) \\
&= \det(\tilde{\boldsymbol{P}}_{\mu_1+e_1}, \tilde{\boldsymbol{P}}_{\mu_2+e_2}, \tilde{\boldsymbol{P}}_{\mu_3+e_3}).
\end{aligned}$$

Hence, by introducing an auxiliary subscript $\mu = \mu_1 + \mu_2 + \mu_3$, the summation in (5) is reorganized as

$$\det J_\phi = \frac{n^2}{(W(\xi))^3} \sum_{|\mu|=3n-3} C_\mu B^{3n-3}_\mu(\xi), \tag{6}$$

where

$$C_\mu = \frac{\mu!(n-1)!^3}{(3n-3)!} \sum_{\substack{\mu_1+\mu_2+\mu_3=\mu \\ |\mu_1|=|\mu_2|=|\mu_3|=n-1}} \frac{\det(\tilde{\boldsymbol{P}}_{\mu_1+e_1}, \tilde{\boldsymbol{P}}_{\mu_2+e_2}, \tilde{\boldsymbol{P}}_{\mu_3+e_3})}{\mu_1!\mu_2!\mu_3!}. \tag{7}$$

We merge (7) with different triangles. Consider a subset of subscripts $\{(\mu_1,\mu_2,\mu_3), (\mu_1, \mu_3+e_3-e_2, \mu_2+e_2-e_3), (\mu_2+e_2-e_1, \mu_1+e_1-e_2, \mu_3), (\mu_2+e_2-e_1, \mu_3+e_3-e_2, \mu_1+e_1-e_3), (\mu_3+e_3-e_1, \mu_1+e_1-e_2, \mu_2+e_2-e_3), (\mu_3+e_3-e_1, \mu_2, \mu_1+e_1-e_3)\}$, denoted as Ω. By summing them up, we have:

$$\begin{aligned}
&\sum_{(\bar{\mu}_1,\bar{\mu}_2,\bar{\mu}_3)\in\Omega} \frac{\det(\tilde{\boldsymbol{P}}_{\bar{\mu}_1+e_1}, \tilde{\boldsymbol{P}}_{\bar{\mu}_2+e_2}, \tilde{\boldsymbol{P}}_{\bar{\mu}_3+e_3})}{\bar{\mu}_1!\bar{\mu}_2!\bar{\mu}_3!} \\
&\quad = \frac{\det(\mu_1+e_1, \mu_2+e_2, \mu_3+e_3)\det(\boldsymbol{P}_{\mu_1+e_1}, \boldsymbol{P}_{\mu_2+e_2}, \boldsymbol{P}_{\mu_3+e_3})}{(\mu_1+e_1)!(\mu_2+e_2)!(\mu_3+e_3)!}.
\end{aligned}$$

Then, (7) has the form:

$$C_\mu = \frac{\mu!(n-1)!^3}{(3n-3)!} \sum_{\substack{\lambda_1+\lambda_2+\lambda_3=\mu+\mathbf{1} \\ |\lambda_1|=|\lambda_2|=|\lambda_3|=n}} \frac{\det(\lambda_1,\lambda_2,\lambda_3)\det(\tilde{\boldsymbol{P}}_{\lambda_1}, \tilde{\boldsymbol{P}}_{\lambda_2}, \tilde{\boldsymbol{P}}_{\lambda_3})}{\lambda_1!\lambda_2!\lambda_3!}. \tag{8}$$

where $\mathbf{1} = (1,1,1)$ and $\lambda_1 < \lambda_2 < \lambda_3$ in the lexicographical order. The positive weights w_{ij} in the condition (1) imply that $W(\xi)$ is positive for all ξ in T. The condition (3) implies that all $C_\lambda \geq 0$, while condition (2) further ensures that $C_{00(3n-3)}$, $C_{0(3n-3)0}$, $C_{(3n-3)00}$ are nonzero, and thus $\det J_\phi > 0$. □

Remark Note that this theorem is the general form of the second-order case in [JW94]. If the subscript $\lambda = (i,j,k)$ is considered as a

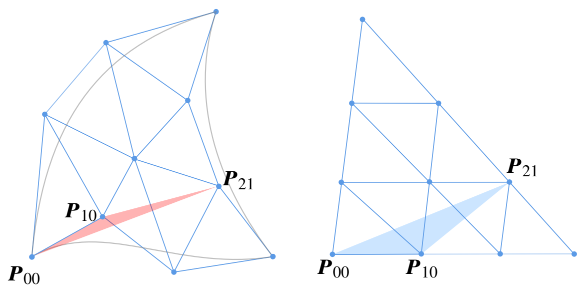

Figure 3: *Two cubic rational Bézier triangles' control nets. The control points have arbitrary positive weights. The area of $\triangle P_{00}P_{10}P_{21}$ is negative (left), so Theorem 1 cannot be used to show that the map is valid. The control net on the right with straight boundary edges is uniform, so the map is valid.*

planar point $(i/n, j/n)$ in the parameter domain T, then all the subscripts are uniform nodes in the parameter domain T, as illustrated in Figure 2 - Left. The triangle formed by arbitrary three nodes in T corresponds to a straight-edge triangle in the computational domain, as illustrated in Figure 2 - Right. From a geometric point of view, $\det(\lambda_1, \lambda_2, \lambda_3)$ equals to the signed area of $\triangle\lambda_1\lambda_2\lambda_3$ multiplied by $2n$ in the parameter domain, and $\det(\tilde{P}_{\lambda_1}, \tilde{P}_{\lambda_2}, \tilde{P}_{\lambda_3})$ equals to the signed area of $\triangle P_{\lambda_1}P_{\lambda_2}P_{\lambda_3}$ multiplied by $2w_{\lambda_1}w_{\lambda_2}w_{\lambda_3}$ in the computational domain. Thus, condition (3) in Theorem 1 is equivalent to that the sign of the corresponding triangle areas in the parameter and computational domains must be the same, or either of the area is zero. In other words, any pair of corresponding triangles have the same orientation. Since the orientations of triangles in the parameter domain are all fixed, we only need to check the orientations of triangles in the computational domain. We list in Appendix B the triangles that need to be checked for the quadratic case. This is a rather strict condition, but we develop an explicit construction method for conic and cubic rational curves, and it provides a potential construction in the case of a higher order. We show an example of a violation of the theorem in Figure 3.

5. High-order meshing

If the input Bézier curves only contain polynomial curves, the Bézier guarding algorithm [MC20] can generate satisfactory results. We first briefly review its key concepts and then introduce our algorithm.

5.1. Bézier guarding

Guardable curve Let p_i, $i \in \{0, \ldots, n\}$, denote the control points of an order-n Bézier curve c, forming the control polygon P_c. The weight corresponding to each point is w_i. For a polynomial curve, the weight of each point is 1. The control vectors of c are $s_i = p_{i+1} - p_i$. If there is a vector d so that $d^T s_i > 0$ for all i, the curve c is called *guardable*, and the vector d is called an *axis* of the curve.

Guarding triangle and envelope For a guardable curve c with axis d, let s^+ be the control vector minimizing $d^T s_i$ that points counterclockwise relative to d, and s^- the minimizer that points clockwise. Let x be the intersection point of L^+ and L^-, where L^+ is a line through p_0 in direction s^+, and L^- is a line through p_n in direction s^-. If $L^+ = L^-$, we let x be the midpoint of p_0 and p_n. For each side of the curve c, we construct a guard point o by adding some amount of translation on x in normal direction. We define o as $x + 0.01(w^2/\hat{w})n$, where $w = ||p_0 - p_n||$ is the width of the curve, $\hat{w}$ denotes the curve's initial width and n is a unit vector perpendicular to and counterclockwise of d. The triangular region formed by c, $\overline{p_0 o}$, and $\overline{p_n o}$ admits a regular geometric map, which is called this side's *guarding triangle*. The curve c is contained in the quadrilateral formed by the two guarding triangles, and we call the quadrilateral the *envelope* of c, denoted as $E(c)$ (Figure 4).

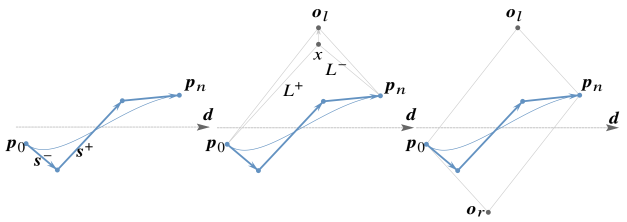

Figure 4: *Construction of a curve envelope (right). A guarding triangle (middle) above a guardable curve (left) with an axis d. Point o_l and o_r are guard points on each side of the curve.*

Workflow The Bézier guarding algorithm works as follows to build an initial mesh:

1. Bisect non-guardable curves c into guardable curves, then reparameterize each sub-curve to $t \in [0, 1]$.
2. When there is a pair of guardable curves whose envelopes intersect except at the curves' endpoints, bisect the one with a larger envelope.
3. Triangulate a bounding polygon encompassing all envelopes that has all envelopes as holes.

In our algorithm, we use a similar workflow to generate guarding triangles and envelopes for inputs of rational Bézier curves, and show that these steps terminate after a finite number of iterations.

Proposition 1 Steps (1) and (2) terminate.

Proof The termination of steps (1) and (2) for polynomial curves has been stated in [MC20]. A rational curve converges to piecewise polynomial curves after repeated bisection(Appendix C), and thus steps (1) and (2) terminate. □

The difference with respect to [MC20] is that the input curves contain rational curves. Therefore, for some control points of mesh elements, we need to specify weights to achieve conforming, and modify the structure of geometric mapping. We describe the modifications in the following section.

5.2. Meshing triangles

We now explain how to construct control points on each Bézier triangle to serve as a valid geometric map for a guarding triangle above each domain curve. To conform to the domain curves, the control points and weights on the side corresponding to the domain curve in the guarding triangle are the same as the curve. We need to specify additional weights to the other control points for the guarding triangle on the rational curve. We adopt different construction methods for guarding triangles above different kinds of curves.

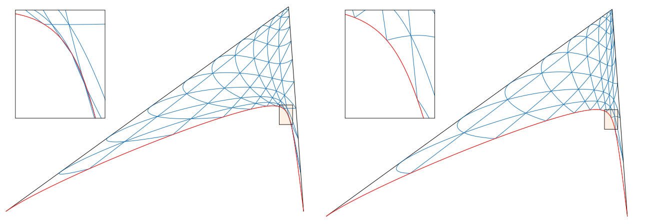
Figure 5: *A guarding triangle above a rational cubic curve (red). One interior point has a weight of* 8*, and the others are all* 1*. We use several barycentric iso-curves (blue) to represent the geometric map. Left: map constructed by Bézier guarding, which is not valid. Right: map constructed by our method.*

The same construction strategy [MC20] is adopted for polynomial curves. For rational curves, the guarding triangles above the curves can be represented as rational Bézier triangles. The polynomial version of the construction method cannot guarantee validity of the map (Figure 5). We solve this problem based on Theorem 1.

Reparameterization In order to ensure that each point on the mesh has a unique weight, some preprocessing needs to be performed first on the rational curves. When two rational curves share the same endpoint, they may have different weights. Therefore, we adopt the following parameter transformation for each rational Bézier curve so that the weights at the endpoint of the curve are 1 [Far02]:

$$\hat{t} = \frac{at}{at+b(1-t)}. \tag{9}$$

If we choose a as $w_0^{-1/n}$ and b as $w_n^{-1/n}$, then $\hat{w_0} = \hat{w_n} = 1$, and $\hat{w_i} = w_i b^i a^{n-i}$. This substitution is also beneficial for constructing maps on straight-edge triangles.

To construct a rational Bézier triangle map satisfying our sufficient condition, we need to check the orientations of straight-edge triangles on the control net, as stated in the remark in Section 4. The triangles for the quadratic cases are listed in Appendix B.

5.2.1. Conic sections

For guarding triangles above conic sections, we only need to select the control points on the two straight edges. On the straight edge $\overline{\boldsymbol{p}_0\boldsymbol{o}}$, we select a point such that it is located on the left side of the control vectors $\boldsymbol{s}_0$ and $\boldsymbol{s}_1$, denoted as $\boldsymbol{q}_1$. Then we select a point on the straight edge $\overline{\boldsymbol{p}_2\boldsymbol{o}}$ analogously, denoted as $\boldsymbol{q}_2$. In practice, we define $\boldsymbol{q}_1 = 0.8\boldsymbol{i}_1 + 0.2\boldsymbol{o}$ and $\boldsymbol{q}_2 = 0.8\boldsymbol{i}_2 + 0.2\boldsymbol{o}$, where $\boldsymbol{i}_2$ is the intersection of $\overline{\boldsymbol{p}_0\boldsymbol{p}_1}$ and $\overline{\boldsymbol{p}_2\boldsymbol{o}}$, $\boldsymbol{i}_1$ is the intersection of $\overline{\boldsymbol{p}_2\boldsymbol{p}_1}$ and $\overline{\boldsymbol{p}_0\boldsymbol{o}}$. We set all the weights of $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, $\boldsymbol{o}$ to 1, which is beneficial for subsequent constructions. Figure 6 (middle) illustrates the construction.

Proposition 2 If the conic control net is constructed by the above procedure, then the geometric map is valid.

Proof According to the construction of our guard point $\boldsymbol{o}$, the 3 triangles $\triangle \boldsymbol{o}\boldsymbol{p}_0\boldsymbol{p}_1$, $\triangle \boldsymbol{o}\boldsymbol{p}_0\boldsymbol{p}_2$, $\triangle \boldsymbol{o}\boldsymbol{p}_1\boldsymbol{p}_2$ has the correct orientations. The construction of the control point $\boldsymbol{q}_1$ ensures that the 5 triangles $\triangle \boldsymbol{q}_1\boldsymbol{p}_0\boldsymbol{p}_1$, $\triangle \boldsymbol{q}_1\boldsymbol{p}_0\boldsymbol{p}_2$, $\triangle \boldsymbol{q}_1\boldsymbol{p}_1\boldsymbol{p}_2$, $\triangle \boldsymbol{q}_1\boldsymbol{p}_1\boldsymbol{o}$, $\triangle \boldsymbol{q}_1\boldsymbol{p}_2\boldsymbol{o}$ has the correct orientations. The construction of the control point $\boldsymbol{q}_2$ ensures that the 9 triangles $\triangle \boldsymbol{q}_2\boldsymbol{p}_0\boldsymbol{p}_1$, $\triangle \boldsymbol{q}_2\boldsymbol{p}_0\boldsymbol{p}_2$, $\triangle \boldsymbol{q}_2\boldsymbol{p}_1\boldsymbol{p}_2$, $\triangle \boldsymbol{q}_2\boldsymbol{q}_1\boldsymbol{p}_0$, $\triangle \boldsymbol{q}_2\boldsymbol{q}_1\boldsymbol{p}_1$, $\triangle \boldsymbol{q}_2\boldsymbol{q}_1\boldsymbol{p}_2$, $\triangle \boldsymbol{q}_2\boldsymbol{o}\boldsymbol{p}_0$, $\triangle \boldsymbol{q}_2\boldsymbol{o}\boldsymbol{p}_1$, $\triangle \boldsymbol{q}_2\boldsymbol{o}\boldsymbol{q}_1$ has the correct orientations. Thus all 17 triangles have the correct orientations, which satisfies our sufficient condition and is valid. □

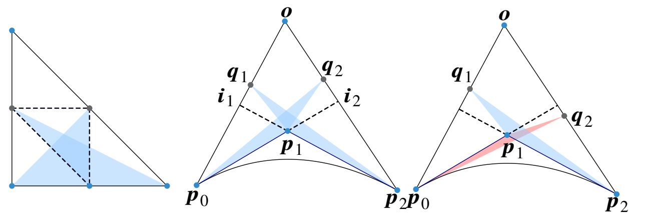

Figure 6: *Construction of a geometric map for a guarding triangle above a conic (middle). Each triangle has the same orientation as the corresponding triangle in the parametric domain, such as the blue triangles. The right figure shows an invalid construction.*

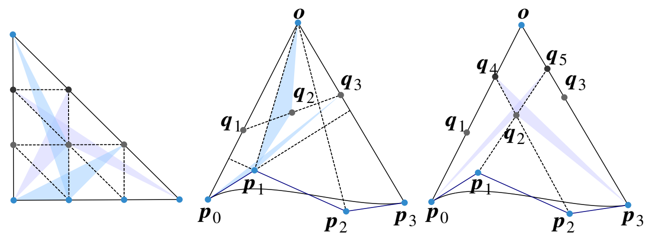

Figure 7: *Construction of a geometric map for a guarding triangle above a cubic curve (right). We construct control points layer by layer and always keep sufficient conditions satisfied. For instance, we ensure the orientations of blue triangles are correct when constructing the points in the second layer, and so are the purple triangles when constructing the points in the third layer.*

5.2.2. Cubic rational curves

We need to determine the positions of five additional control points on the guarding triangles above cubic rational curves. We select a point on the straight edge $\overline{\boldsymbol{p}_0\boldsymbol{o}}$, such that it is located on the left side of the vectors $\boldsymbol{s}_0$, $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, denoted as $\boldsymbol{q}_1$. Then we select a point on the straight edge $\overline{\boldsymbol{p}_3\boldsymbol{o}}$ analogously, denoted as $\boldsymbol{q}_3$. The selection method is similar to the conic sections. The segment $\overline{\boldsymbol{p}_1\boldsymbol{o}}$ has one intersection point with $\overline{\boldsymbol{q}_1\boldsymbol{q}_3}$, and the segment $\overline{\boldsymbol{p}_2\boldsymbol{o}}$ has one intersection point with $\overline{\boldsymbol{q}_1\boldsymbol{q}_3}$. We select a point on the line segment formed by these two intersection points, denoted as $\boldsymbol{q}_2$. Let $\boldsymbol{q}_4$ be the intersection point of $\overline{\boldsymbol{p}_0\boldsymbol{o}}$ and the line through $\boldsymbol{p}_2$ and $\boldsymbol{q}_2$, let $\boldsymbol{q}_5$ be the intersection point of $\overline{\boldsymbol{p}_3\boldsymbol{o}}$ and the line through $\boldsymbol{p}_1$ and $\boldsymbol{q}_2$. Figure 7 illustrates the construction. We set the weight of $\boldsymbol{q}_1$, $\boldsymbol{q}_3$, $\boldsymbol{q}_4$, $\boldsymbol{q}_5$, $\boldsymbol{o}$ to 1. The weight of $\boldsymbol{q}_2$ can be specified arbitrarily. In our algorithm, we set this weight as the average of the other weights, making the distortion lower, especially when the difference in weights is large.

To evaluate our algorithm, we construct geometric maps on guarding triangles of 10000 random rational cubic guardable curves and show the scaled-Jacobian values in Figure 8.

Proposition 3 If the cubic control net is constructed by the above procedure, then the geometric map is valid.

Proof A straightforward way is that we can check the orientations of all triangles. Due to the construction of the guarding triangle, many triangles already satisfy the condition by default. We adopt a

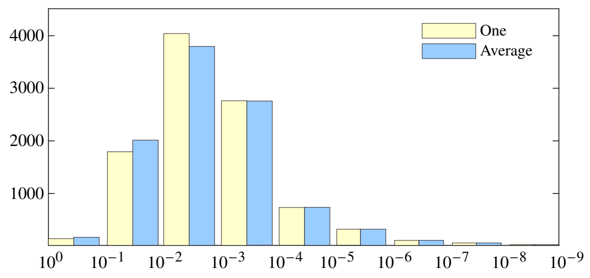

Figure 8: *The weight of each control point ranges from* 1 *to* 10. *The scaled-Jacobian values (defined as* $\frac{\min|\det J_\phi|}{\max|\det J_\phi|}$*, where* J_ϕ *is the Jacobian matrix of* ϕ*) are all positive. The horizontal axis is the scaled-Jacobian values, and the vertical axis is the number of maps in the corresponding range. The weight of the interior point is one or the average of the other weights.*

layer-by-layer construction and guarantee that our sufficient theorem holds when constructing each control point. □

Degree elevation Suppose the order of the rational Bézier triangle map is less than the specified order n. We perform degree elevation until the order becomes n after constructing the control points and weights on the guarding triangle above a rational curve. In that case, the new control points and weights are what we finally constructed.

Due to the construction and reparameterization (9), the weights of the control points on the straight edges of the triangle are all 1. Therefore, for straight-edge triangles, we can take advantage of the sufficient conditions in polynomial situation [MC20] to construct a valid map.

5.2.3. Straight elements

We furthermore need to construct a valid geometric map for straight-edge triangles. To ensure C^0-continuity across edges, if an edge is adjacent to a guarding triangle, we adopt the edge control points constructed for this guarding triangle. As mentioned above, these weights are all 1. Due to the manner in step (2), each straight-edge triangle is adjacent to at most two guarding triangles. Given a straight-edge triangle with corner point $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$. We uniformly distributed control points if the straight triangle is not adjacent to any guarding triangle. If there is only one edge $(\boldsymbol{a}, \boldsymbol{b})$ adjacent to other guarding triangle with non-uniform prescribed control points $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_n$ ($\boldsymbol{p}_0 = \boldsymbol{a}$ and $\boldsymbol{p}_n = \boldsymbol{b}$), we define Bézier triangle control points as

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

If edge $(\boldsymbol{a}, \boldsymbol{c})$ is adjacent to another guarding triangle at the same time, with prescribed control points $\boldsymbol{q}_0, \ldots, \boldsymbol{q}_n$ ($\boldsymbol{q}_0 = \boldsymbol{a} = \boldsymbol{p}_0$ and $\boldsymbol{q}_n = \boldsymbol{c}$), let K_i be the line parallel to $\overline{\boldsymbol{p}_0\boldsymbol{p}_n}$, passing through point $\boldsymbol{q}_i, 1 \le i < n$. Let $\boldsymbol{r}_i$ be the intersection of K_i and $\overline{\boldsymbol{p}_n\boldsymbol{q}_n}$, and we uniformly distributed points on the line segment $\overline{\boldsymbol{q}_i\boldsymbol{r}_i}$ as inner control points. To conclude the construction, we define Bézier triangle control points as

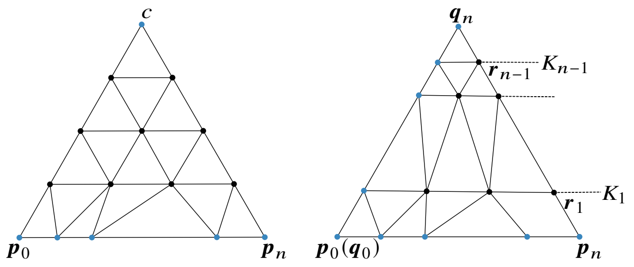

Figure 9: *Bézier control points for a straight-edge triangle. Left: triangle adjacent to one guarding triangle. Right: triangle adjacent to two triangles. All 1-vectors* $\boldsymbol{p}_{i(j+1)} - \boldsymbol{p}_{ij}$ *points counterclockwise relative to 0-vectors* $\boldsymbol{p}_{(i+1)j} - \boldsymbol{p}_{ij}$.

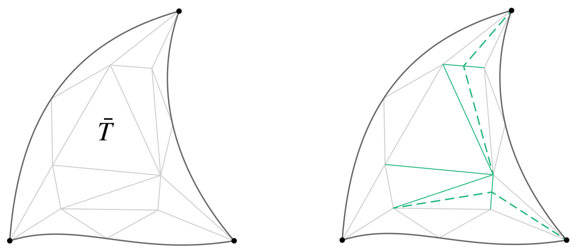

Figure 10: *Non-uniform control points on gray edges and curved edges (black). There exists a straight-edge triangle (for example* $\bar{T}$*) that is not adjacent to any guarding triangle, with non-uniform control points on three edges (left). We split some special triangles via green dotted lines (right), then these green edges have uniform control points.*

$$\boldsymbol{p}_{ijk} = \begin{cases} \boldsymbol{p}_i, & j = 0, \\ \boldsymbol{q}_j, & i = 0, \\ \boldsymbol{r}_j, & k = 0, i \neq 0, j \neq 0, \\ \frac{i}{n-j}\boldsymbol{q}_j + \frac{k}{n-j}\boldsymbol{r}_j, & i, j, k > 0. \end{cases}$$

These control points are illustrated in Figure 9, and all weights are set to 1.

Note that if the element is a guarding triangle or a straight-edge triangle adjacent to two guarding triangles, the control points on the straight edges are usually non-uniform. In order to prevent the appearance of a straight-edge triangle with three non-uniform control points on the three sides, we take the following measures. Let $\mathcal{S}$ denote the set of all straight-edge triangles adjacent to two triangles. For any triangle t in $\mathcal{S}$, denote the edge adjacent to a straight triangle as e, the adjacent triangle as t_1. If t_1 adjacent to another triangle belonging to A, we split the triangle t at e (Figure 10). We perform it before constructing geometric maps.

Another strategy is to split triangles that are edge-adjacent to two guarding triangles so that each straight-edge triangle is adjacent to at most one guarding triangle, but this may result in more triangles, increasing the complexity of subsequent mesh optimizations.

5.2.4. Curves of arbitrary order

For rational curves of arbitrary order n, one possible approach is to use the strategy in [MC21] to construct triangles and use gen-

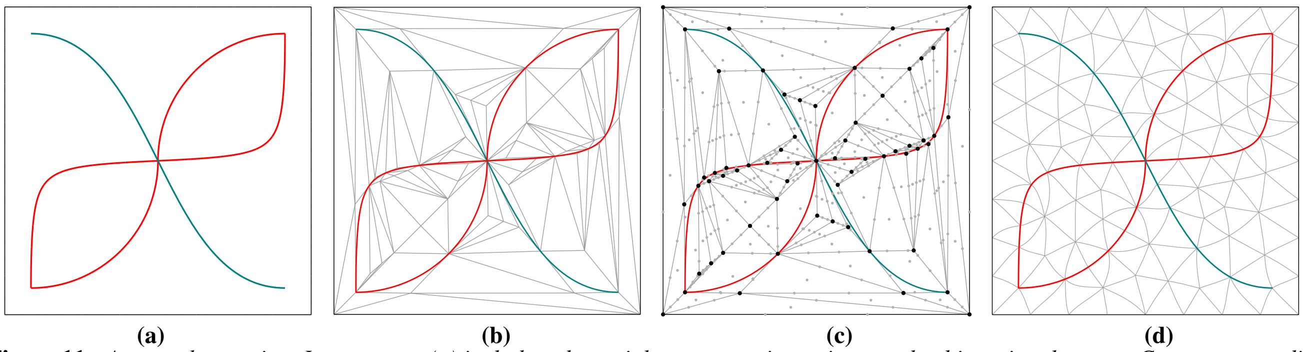

Figure 11: *Approach overview. Input curves (a) include polynomial curves, conic sections, and cubic rational curves. Construct guarding triangles on each sub-curve and perform triangulation (b). Construct control points on each element to serve as a valid geometric map (c). Finally, some optimization strategies are used to improve the mesh quality (d).*

eralized barycentric coordinates for placing control points. Then, we check whether the triangle is valid and bisect the curve if not. This strategy has proven effective for polynomial curves in [MC21]. In Proposition 1, we introduced that rational curves will converge to piecewise polynomial curves after repeated bisection. So the strategy is also feasible for rational curves. On the other hand, the triangle converges to a linear reference configuration under repeated bisection, i.e., converges to a state of uniformly distributed control points [MC21], and our sufficient condition is satisfied after a sufficient number of subdivisions, just like the example in Figure 3 - Right. It is a posterior-based strategy, i.e., construct first and then check. A construction method based on sufficient conditions can avoid additional validity checks.

5.3. Our meshing algorithm

The above construction method strictly guarantees that the mesh is valid and conforms to the input domain curves, but the quality is uncontrolled. Many previous methods have been proposed for mesh optimization [HSG*19, Pan20, BK04, MC20]. We take use of previously proposed techniques based on two ways, combinatorial and geometric mesh improvement [MC20, LSL*21]. Since our contribution is not in the mesh optimization, we restrict ourselves to briefly describing and discussing some modifications.

5.3.1. Combinatorial Optimization

We use an edge-length driven strategy [BK04], and assign a target edge length l to each edge. The length of an edge between vertices p and q is denoted as $l(p,q)$, and it is approximated by $l(p,q) = \|p-q\|$. Then we perform the following steps.

Edge split If $l(p,q) > \frac{4}{3}l$, we split the edge by bisecting the adjacent triangles using Bézier triangle bisection [FH99], which yields control points and corresponding weights for the new triangles. When we split a domain curve, the weight of the new point is usually not one. In this case, we perform a reparameterization on each new triangle such that each triangle's endpoint has a weight of one.

Edge collapse In order to maintain conformance, an edge is considered "collapsible" if this collapse operation does not cause movement of the domain curves. For example, vertices representing the endpoints of domain curves are never collapsed, and those in the interior of domain curves are only collapsed along edges on domain curves. We collapse the edge if it is "collapsible" and $l(p,q) < \frac{4}{5}l$.

Edge flip Analogously, edges representing domain curves are never flipped. We flip those edges if that reduces the deviation from valence 6 (or 4 on boundaries).

After each operation, we perform a quasi-uniform distribution for each changed triangle except for the edges representing domain curves. To ensure that the mesh is valid, we perform an injectivity test [HMESM06] on each locally modified area. If there exists an invalid triangle, we reject this operation.

5.3.2. Geometric Optimization

In order to promote equilateral elements, we apply a second-order solver [SPSH*17] on a scale invariant conformal distortion measure [HG00], computed from the map's Jacobian J_ϕ (with singular values Σ, σ):

$$E_{\text{conf}} = (\Sigma^2 + \sigma^2)/\Sigma\sigma.$$

It is numerically integrated over the triangles using a quadrature scheme [WV15, MC20]. Preservation of validity is ensured by the injectivity test in the line search in Newton method during optimization. All edges are free to curve, which contributes to distortion reduction in the optimization.

5.3.3. Complete meshing algorithm

Then, we have the following steps, as shown in Figure 11. The target edge length is l, and the order of the mesh is n. The output is a mesh conforming input curves with a valid geometric map.

1. Building the initial mesh as described in Section 5.1.
2. Reparameterize each curve so that each endpoint has a weight of 1.
3. Construct Bézier control points per guarding triangle. For the guarding triangles above rational curves, we elevate the order to n.
4. Construct Bézier control points for straight-edge triangles.
5. Perform geometric and combinatorial optimization.

Dataset:		(A)	(B)	(C)	(D)
FLOAT	FAILURE	0	0	0	4
	SUCCESS	500	400	500	796
EXACT	FAILURE	0	0	0	0
	SUCCESS	500	400	500	800

Table 1: *In the* EXACT *version, we generate a valid mesh for each example in the dataset. For the* FLOAT *version, a few examples have invalid triangular elements in the mesh due to numerical errors.*

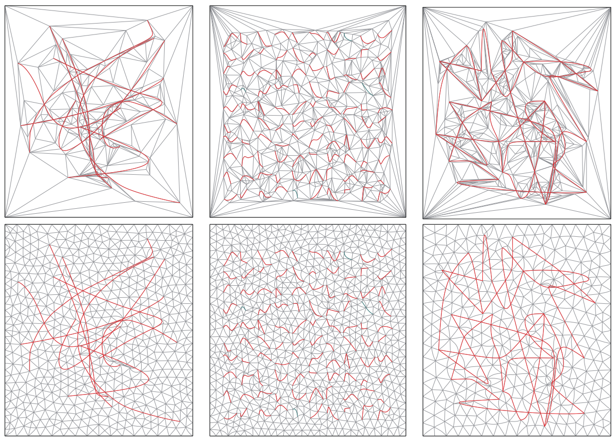
Figure 12: *Examples generated by our algorithm. The upper row shows the initial valid meshes, and the bottom row shows a remeshed version.*

Remark Except the reparameterization step in our algorithm, other steps can be performed using exact rational operations. Specifically, the reparameterization step, which ensures that each vertex has the same weight and that all straight-edged triangles are non-rational, contains a mathematical power expression with a fractional power, thereby bringing a truncation error. The error is about the floating point precision. When we mention EXACT below, it means that we perform a floating-point operation in the reparameterization step and exact rational operations in other steps.

6. Experiments and evaluations

Our method is implemented in C++, and all the experiments are performed on a desktop PC with a 4.00 GHz Intel Core i7-4790K and 16 GB of RAM.

We test our algorithm on the dataset of random domain curves provided by [MC20], which are divided into four groups: (A) curves forming a closed C^0 domain boundary loop with corners; (B) curves forming a closed C^1 domain boundary loop without corners; (C) isolated random curves, spread uniformly over a rectangular domain; (D) curves forming a curve network, generated by intersecting random curves. We randomly assign weights to the control points of each curve, ranging from 1 to 10. Since adding weights may generate some bad data like degenerate curves or overlapping curves, we generate 2200 data after removing those.

We tested our algorithm in both exact rational numbers and standard double precision numbers, and presented the statistics of the results in Table 1. In Figure 12, we show three resulting meshes generated from the dataset.

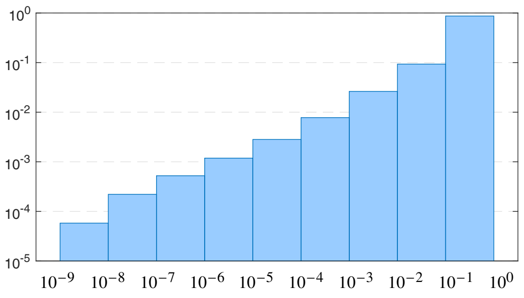

Figure 13: *Logarithmic histogram of scaled-Jacobian values (horizontal axis) of the initial meshes.*

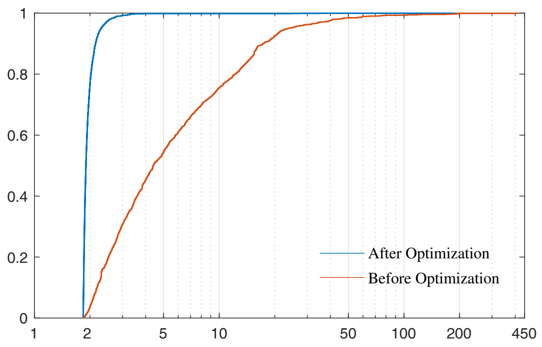

Figure 14: *Cumulative distribution functions of E_{conf} before and after optimization. The minimally possible value of E_{conf} is 2 (for an ideal equilateral element). Most of the values are less than 5 after optimization.*

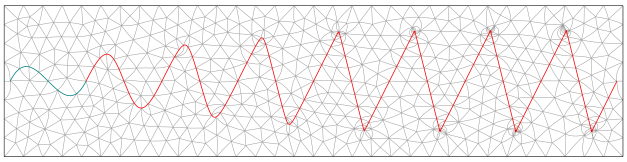
Figure 15: *Curves with different weights. The weights of the inner points increase from left to right. The maximum weight of control points is* 100000.

The distribution of the scaled-Jacobian values, which aggregate over all elements of the initial meshes of the datasets, is shown in Figure 13. As guaranteed by our algorithm, these values are strictly positive (here $> 10^{-9}$), such that the meshes are valid starting points for validity preserving optimization. In Figure 14, we show the cumulative distribution functions of E_{conf}. There may exist highly distorted elements of strongly varying sizes in initial meshes, which will be greatly reduced after optimization.

In Figure 15, we demonstrate that the proposed method is able to properly handle curves with extremely large weights. Though large weights rarely appear in real-world data, this experiment illustrates the reliability of the method.

In Table 2, we record the timing of our implementation of the

	FLOAT			EXACT		
#Curves	~50	~100	~200	~50	~100	~200
STEP (1)	91.8	143.6	249.1	116.8	217.7	284.9
STEP (3)+(4)	3.4	5.5	8.8	80.0	179.6	184.0
TOTAL	95.2	149.1	257.9	196.8	397.3	468.9
STEP (5)	44000.0	72000.0	108000.0	-	-	-

Table 2: *Timings of our algorithm (Section 5.3) in milliseconds. Obtained by averaging over runs on the input domains of the datasets containing ~50, ~100, and ~200 curves (±%10), respectively.*

algorithm. The time difference from the polynomial case mainly comes from the clipping of the rational curve. The optimization running time is greatly affected by the quality of input curves and the target length. We perform optimization under floating-point arithmetic and detect validity using exact rational computation.

7. Conclusion and Discussion

We propose a sufficient condition for the injectivity of rational Bézier triangles of arbitrary order. Our method uses this sufficient condition in combination with the algorithm stated in [MC20] to generate a valid mesh that conforms to domain curves, including rational and non-rational curves. It is effective for curves with arbitrary weights.

Tightness of our sufficient condition Our sufficient condition is sometimes too restrictive to be used for extremely higher-order cases. A general construction of valid geometric maps for guarding triangles above rational curves of arbitrary order is an obvious avenue for future work.

Guaranteed quality Furthermore, generating quality-guaranteed higher-order mesh by extending the method stated in [MC21] is another interesting research direction. It is also beneficial to consider the optimization of weights. In [EE20], a new set of quality metrics for curvilinear finite elements and a new mesh optimization are proposed, which can be taken into account as well.

Intersection of rational curves In our work, the input curves are assumed to only intersect at endpoints. If the input curves have general intersections, the algorithm cannot generate a mesh that precisely conforms to the input since the intersection point is computed by solving nonlinear equations, which is not a rational operation. Thus, when applying our algorithm to intersected curves, we propose to split curves at the intersection points and approximate them by rounding to limited precision numbers.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive suggestions and comments. The work is supported by Anhui Center for Applied Mathematics, the Major Project of Science and Technology of Anhui Province (No. 202203a05020050), and the National Natural Science Foundation of China (62025207).

References

[ADF14] ABGRALL R., DOBRZYNSKI C., FROEHLY A.: A method for computing curved meshes via the linear elasticity analogy. *Int. J. Numer. Methods Fluids 76*, 4 (2014), 246–266. 2

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (New York, NY, USA, 2004), Association for Computing Machinery, pp. 185–192. 7

[CCM*04] CARDOZE D., CUNHA A., MILLER G. L., PHILLIPS T., WALKINGTON N.: A Bézier-based approach to unstructured moving meshes. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry* (New York, NY, USA, 2004), Association for Computing Machinery, pp. 310–319. 2

[DOS99] DEY S., O'BARA R. M., SHEPHARD M. S.: Curvilinear mesh generation in 3D. In *Proceedings of the 8th 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. *Comput. Aided Des. 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. Engrg. 304* (2016), 378–407. 2

[EE20] ENGVALL L., EVANS J. A.: Mesh quality metrics for isogeometric bernstein bézier discretizations. *Comput. Methods Appl. Mech. Engrg. 371* (2020), 113305. 1, 9

[Far86] FARIN G.: Triangular Bernstein-Bézier patches. *Comput. Aided Geom. Des. 3*, 2 (1986), 83–127. 2, 10

[Far02] FARIN G.: *Curves and surfaces for CAGD: A practical guide*, 5th ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. 5

[FH99] FARIN G., HANSFORD D.: Discrete coons patches. *Computer Aided Geometric Design 16*, 7 (1999), 691 – 700. 7

[FP16] FORTUNATO M., PERSSON P.-O.: High-order unstructured curved mesh generation using the Winslow equations. *J. Comput. Phys. 307* (2016), 1–14. 2

[GB12] GEORGE P., BOROUCHAKI H.: Construction of tetrahedral meshes of degree two. *Int. J. Numer. Methods Eng. 90* (06 2012), 1156–1182. 2

[GTNI16] GHASEMI A., TAYLOR L. K., NEWMAN III J. C.: Massively parallel curved spectral/finite element mesh generation of industrial CAD geometries in two and three dimensions. In *Proceedings of the ASME 2016 Fluids Engineering Division Summer Meeting* (07 2016), p. 50299. 2

[HG00] HORMANN K., GREINER G.: MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*. Vanderbilt University Press, 2000, pp. 153–162. 7

[HMESM06] HERNANDEZ-MEDEROS V., ESTRADA-SARLABOUS J., MADRIGAL D. L.: On local injectivity of 2D triangular cubic Bézier functions. *Investigación Operacional 27*, 3 (2006), 261–276. 7

[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), 52:1–52:15. 1, 2, 7

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

[JW94] JOE B., WANG W.: Reparameterization of rational triangular Bézier surfaces. *Comput. Aided Geom. Des. 11*, 4 (1994), 345–361. 2, 3

[KMC22] KHANTEIMOURI P., MANDAD M., CAMPEN M.: Rational Bézier Guarding. *Computer Graphics Forum 41*, 5 (2022). 2

[LSL*21] LIU Z.-Y., SU J.-P., LIU H., YE C., LIU L., FU X.-M.: Error-bounded edge-based remeshing of high-order tetrahedral meshes. *Computer-Aided Design 139* (2021), 103080. 2, 7

[LSO*04] LUO X.-J., SHEPHARD M., O'BARA R., NASTASIA R., BEALL M.: Automatic p-version mesh generation for curved domains. *Engineering with Computers 20* (09 2004), 273–285. 2

[LSR01] LUO X., SHEPHARD M., REMACLE J.-F.: *The influence of geometric approximation on the accuracy of higher order methods*. Tech. rep., Scientific Computation Research Center, Rensselaer Polytechnic Institute, 2001. 2

[MC20] MANDAD M., CAMPEN M.: Bézier guarding: precise higher-order meshing of curved 2D domains. *ACM Trans. Graph. 39*, 4 (2020), 103:1–103:15. 1, 2, 4, 5, 6, 7, 8, 9

[MC21] MANDAD M., CAMPEN M.: Guaranteed-quality higher-order triangular meshing of 2D domains. *ACM Trans. Graph. 40*, 4 (2021), 154:1–154:14. 1, 2, 6, 7, 9

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

[Pan20] PANETTA J.: Analytic eigensystems for isotropic membrane energies, 2020. arXiv:2008.10698. 7

[PP09] PERSSON P.-O., PERAIRE J.: Curved mesh generation and mesh refinement using Lagrangian solid mechanics. In *Proceeding of the 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition* (2009). 2

[PSG16] POYA R., SEVILLA R., GIL A. J.: A unified approach for a posteriori high-order curved mesh generation using solid mechanics. *Comput. Mech. 58* (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 *Proceedings of the 20th International 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

[SB21] SZABÓ B., BABUŠKA I.: *Finite Element Analysis: Method, Verification and Validation, 2nd Edition*. John Wiley & Sons Inc., 2021. 1

[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

[SFMH08] SEVILLA R., FERNÁNDEZ-MÉNDEZ S., HUERTA A.: NURBS-enhanced finite element method (NEFEM). *Int. J. Numer. Methods Eng. 76*, 1 (2008), 56–83. 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

[SPSH*17] SHTENGEL A., PORANNE R., SORKINE-HORNUNG O., KOVALSKY S. Z., LIPMAN Y.: Geometric optimization via composite majorization. *ACM Transactions on Graphics 36*, 4 (2017), 1–11. 7

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

[SWS95] SAITO T., WANG G.-J., SEDERBERG T. W.: Hodographs and normals of rational curves and surfaces. *Comput. Aided Geom. Des. 12*, 4 (1995), 417–430. 3

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

[TLR16] TOULORGE T., LAMBRECHTS J., REMACLE J.-F.: Optimizing the geometrical accuracy of curvilinear meshes. *Journal of Computational Physics 310* (2016), 361–380. 2

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

[WFA*13] WANG Z., FIDKOWSKI K., ABGRALL R., BASSI F., CARAENI D., CARY A., DECONINCK H., HARTMANN R., HILLEWAERT K., HUYNH H., KROLL N., MAY G., PERSSON P.-O., VAN LEER B., VISBAL M.: High-order CFD methods: current status and perspective. *Int. J. Numer. Methods Fluids 72*, 8 (2013), 811–845. 1

[WV15] WITHERDEN F., VINCENT P.: On the identification of symmetric quadrature rules for finite element methods. *Computers & Mathematics with Applications 69*, 10 (2015), 1232–1241. 7

[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

[XSHM14] XIE Z. Q., SEVILLA R., HASSAN O., MORGAN K.: The generation of arbitrary order curved meshes for 3D finite element analysis. *Comput. Mech. 51* (2014), 361–374. 2

[Zlá73] ZLÁMAL M.: The finite element method in domains with curved boundaries. *Int. J. Numer. Methods Eng. 5*, 3 (1973), 367–373. 1

Appendix A: Details of theorem 1

Bernstein polynomials satisfy the following recursion [Far86]:

$$B^n_\lambda(\xi) = \xi_1 B^{n-1}_{\lambda-e_1}(\xi) + \xi_2 B^{n-1}_{\lambda-e_2}(\xi) + \xi_3 B^{n-1}_{\lambda-e_3}(\xi).$$
$$\frac{\partial}{\partial \xi_1} B^n_\lambda(\xi) = n(B^{n-1}_{\lambda-e_1}(\xi) - B^{n-1}_{\lambda-e_3}(\xi)).$$
$$\frac{\partial}{\partial \xi_2} B^n_\lambda(\xi) = n(B^{n-1}_{\lambda-e_2}(\xi) - B^{n-1}_{\lambda-e_3}(\xi)).$$

Substitute them into (2), we get the recursion of $\tilde{\phi}(\xi)$, $\tilde{\phi}_{\xi_1}(\xi)$ and $\tilde{\phi}_{\xi_2}(\xi)$.

The determinant is related to the area as follows:

$$\begin{aligned}\det(\tilde{\boldsymbol{P}}_{\lambda_1}, \tilde{\boldsymbol{P}}_{\lambda_2}, \tilde{\boldsymbol{P}}_{\lambda_3}) &= w_{\lambda_1} w_{\lambda_2} w_{\lambda_3} \begin{vmatrix} x_{\lambda_1} & x_{\lambda_2} & x_{\lambda_3} \\ y_{\lambda_1} & y_{\lambda_2} & x_{\lambda_3} \\ 1 & 1 & 1 \end{vmatrix} \\ &= 2 w_{\lambda_1} w_{\lambda_2} w_{\lambda_3} \operatorname{area}(\triangle \boldsymbol{P}_{\lambda_1} \boldsymbol{P}_{\lambda_2} \boldsymbol{P}_{\lambda_3}).\end{aligned}$$

This area is positive if three vertices are in counterclockwise order; negative if they are in clockwise order; and zero if they are collinear.

Appendix B: Triangles that need to be verified

For the quadratic case, we need to ensure that the areas of the following 17 triangles are non-negative, that is, the vertices of the triangles are in counterclockwise order or they are collinear.

$$\triangle \boldsymbol{P}_{00}\boldsymbol{P}_{10}\boldsymbol{P}_{01}, \triangle \boldsymbol{P}_{00}\boldsymbol{P}_{10}\boldsymbol{P}_{11}, \triangle \boldsymbol{P}_{00}\boldsymbol{P}_{10}\boldsymbol{P}_{02}, \triangle \boldsymbol{P}_{00}\boldsymbol{P}_{20}\boldsymbol{P}_{01},$$
$$\triangle \boldsymbol{P}_{00}\boldsymbol{P}_{20}\boldsymbol{P}_{11}, \triangle \boldsymbol{P}_{00}\boldsymbol{P}_{20}\boldsymbol{P}_{02}, \triangle \boldsymbol{P}_{10}\boldsymbol{P}_{20}\boldsymbol{P}_{01}, \triangle \boldsymbol{P}_{10}\boldsymbol{P}_{20}\boldsymbol{P}_{11},$$
$$\triangle \boldsymbol{P}_{10}\boldsymbol{P}_{20}\boldsymbol{P}_{02}, \triangle \boldsymbol{P}_{00}\boldsymbol{P}_{11}\boldsymbol{P}_{01}, \triangle \boldsymbol{P}_{00}\boldsymbol{P}_{11}\boldsymbol{P}_{02}, \triangle \boldsymbol{P}_{10}\boldsymbol{P}_{11}\boldsymbol{P}_{01},$$
$$\triangle \boldsymbol{P}_{10}\boldsymbol{P}_{11}\boldsymbol{P}_{02}, \triangle \boldsymbol{P}_{20}\boldsymbol{P}_{11}\boldsymbol{P}_{01}, \triangle \boldsymbol{P}_{10}\boldsymbol{P}_{02}\boldsymbol{P}_{01}, \triangle \boldsymbol{P}_{20}\boldsymbol{P}_{02}\boldsymbol{P}_{01},$$
$$\triangle \boldsymbol{P}_{01}\boldsymbol{P}_{11}\boldsymbol{P}_{02}.$$

Appendix C: Bisection of Rational Curves

W.l.o.g consider the first sub-curve of a curve after ℓ bisections. Its k-th weight is $w'_k = \sum_{i=0}^{k} w_i B^k_i(0.5^\ell)$. As $\ell \to \infty$, $w'_k \to w'_0$, which shows convergence to a polynomial curves.