3D Bézier Guarding: Boundary-Conforming Curved Tetrahedral Meshing

PAYAM KHANTEIMOURI and MARCEL CAMPEN, Osnabrück University, Germany

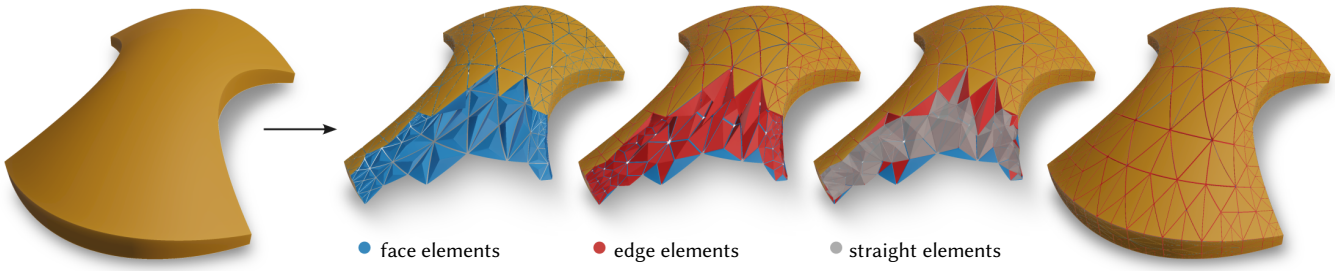

Fig. 1. Our method generates arbitrary-order tetrahedral meshes that conform to prescribed polynomial boundary surfaces, here formed by cubic B-spline surface patches (left). After conversion to a mesh of polynomial triangle patches and adaptive refinement, provably inversion-free curved tetrahedra (*face elements*) are constructed which conform to the surface triangles. Into the gaps between these a second type of (half curved, half planar) tetrahedra (*edge elements*) are placed. The remaining space can then be filled with non-curved tetrahedra (*straight elements*) using established meshing techniques, here shown semi-transparent. Together, this yields a conforming mesh of inversion-free higher-order tetrahedra that exactly conforms to the given boundary. Just for visual clarity, the tetrahedra are shown slightly shrunken.

We present a method for the generation of higher-order tetrahedral meshes. In contrast to previous methods, the curved tetrahedral elements are guaranteed to be free of degeneracies and inversions while conforming exactly to prescribed piecewise polynomial surfaces, such as domain boundaries or material interfaces. Arbitrary polynomial order is supported. Algorithmically, the polynomial input surfaces are first covered by a single layer of carefully constructed curved elements using a recursive refinement procedure that provably avoids degeneracies and inversions. These tetrahedral elements are designed such that the remaining space is bounded piecewise linearly. In this way, our method effectively reduces the curved meshing problem to the classical problem of linear mesh generation (for the remaining space).

CCS Concepts: • **Computing methodologies** → **Volumetric models**.

Additional Key Words and Phrases: tetrahedral mesh, high-order mesh, curved mesh, isogeometric analysis

ACM Reference Format:
Payam Khanteimouri and Marcel Campen. 2023. 3D Bézier Guarding: Boundary-Conforming Curved Tetrahedral Meshing. *ACM Trans. Graph.* 42, 6, Article 175 (December 2023), 19 pages. https://doi.org/10.1145/3618332

1 INTRODUCTION

Representing and discretizing geometric objects by means of meshes is a fundamental task in computational sciences, from computer graphics to computational engineering. We are concerned here specifically with the case of 3D volumetric meshes with simplex elements, i.e. tetrahedral meshes.

Authors' address: Payam Khanteimouri, payam.khanteimouri@uos.de; Marcel Campen, campen@uos.de, Osnabrück University, Germany.

While most often tetrahedral meshes with *linear* elements, i.e. straight-sided tetrahedra, are employed, for various applications advantages of using *higher-order* elements have been demonstrated or conjectured [Müller et al. 2015; Feng et al. 2018; Bargteil and Cohen 2014; Jiang et al. 2021; Suwelack et al. 2013; Ferguson et al. 2023; Mezger et al. 2008; Zlamal 1973; Babuška and Guo 1996; Wang et al. 2013]. In this case the geometry of each tetrahedral element, its edges and facets, is described by polynomials of order > 1. We are thus dealing with *curved* tetrahedra.

One important advantage is the ability to geometrically match curved domain boundaries, if represented polynomially, with the mesh, avoiding the geometric approximation error that is generally inevitable when working with linear elements [Luo et al. 2001; Ciarlet and Raviart 1972b; Bassi and Rebay 1997]. We say the mesh *conforms* to the boundary surface.

A fundamental task is the automatic generation of such meshes for given domains. In this context, a key challenge lies in generating the mesh in such a way that it is boundary-conforming while at the same time guaranteeing that all elements are *regular*. Regularity refers to the polynomial function defining an element's shape being injective, thus free of parametric degeneracies or inversions, as is an important prerequisite for instance in the context of the finite element method [Mitchell et al. 1971; Barrett 1996].

Existing volumetric higher-order mesh generation methods follow the principle of *a posteriori* curving: First generate a linear tetrahedral mesh, then formally elevate the elements' order, and finally curve them, e.g., by optimizing a geometric deformation for minimal geometric approximation error towards the boundary. In this setting there are two options: Constraining the optimization to maintain regularity (but thereby potentially precluding it from reaching full conformance) or leaving it unconstrained to ensure reaching conformance (but thereby potentially losing regularity).

Our method presented in the following is an *a priori* approach instead. Tetrahedra are generated in a curved state right away, ensuring regularity and conformance *by construction*. The construction is

not limited to, e.g., quadratic or cubic elements, but arbitrary polynomial order is supported. On a high-level, it follows the paradigm of the recent Bézier Guarding approach [Mandad and Campen 2020a] for 2D mesh generation: A first layer of partially-curved elements covering the input boundary is carefully constructed, effectively *shielding off* its potentially complex curved nature, such that the remaining space can be meshed in a compatible way using established linear meshing techniques. We detail this starting from Section 3.

Let us point out that mesh *quality* (as opposed to mesh *validity*) is not the focus of this work. Due to the existence of validity preserving mesh optimization and remeshing techniques for quality improvement this can be considered a separate aspect.

We also remark that a relevant but different problem setting besides precise interpolation of a given domain boundary is that of (bounded) approximation of the domain [Jiang et al. 2021; Liu et al. 2021; Feng et al. 2018].

2 RELATED WORK

We review previous work on the problem of non-linear mesh generation, with a particular focus on simplicial meshes of triangles or tetrahedra, as well as on the properties of regularity and conformance.

2.1 2D Curved Meshing

An approach often taken for the generation of higher-order triangle meshes in the plane relies on *a posteriori* deformation: Initially, a linear mesh is generated, approximating the domain boundary in a piecewise linear manner. Afterwards, a deformation of the elements into a non-linear curved state is performed, driven by objectives promoting low approximation error and low distortion [Hu et al. 2019; Toulorge et al. 2013; Roca et al. 2011; Ruiz-Gironés et al. 2016; Abgrall et al. 2014; Xie et al. 2013; Persson and Peraire 2009; Paul 2018; Xu and Chernikov 2014; Oliver 2008; Moxey et al. 2016; Fortunato and Persson 2016; Turner et al. 2018; Poya et al. 2016; Shephard et al. 2005]. Extensions to non-planar triangles on curved parametric surfaces have likewise been considered [Gargallo-Peiró et al. 2013]. Besides such geometric deformation, structural mesh modifications (flips, splits, collapses) may be applied in the process, enabling improved results in some cases [Hu et al. 2019; Cardoze et al. 2004; Luo et al. 2004; Shephard et al. 2005; Dey et al. 1999]. Elements may even be snapped to the boundary [Dey et al. 1999; Rangarajan and Lew 2014; Engvall and Evans 2016; Jaxon and Qian 2014], so as to establish conformance. This, however, may destroy regularity.

Another approach is to construct elements that are curved *a priori*, so as to establish conformance from the start [Mansfield 1978; Gordon and Hall 1973; Zlamal 1973; Haber et al. 1981; Sevilla et al. 2016]. Then, however, yielding regularity turns out to be challenging, unless the input meets particular smoothness assumptions [Ciarlet and Raviart 1972a; Rangarajan and Lew 2014]. Untangling methods have been proposed to improve regularity in a post-process [Toulorge et al. 2013, 2016], though without guarantees of success.

A quite different approach is taken in recent work [Mandad and Campen 2020a]: Conformance *and* regularity are established in an *a priori* manner by means of a divide-and-conquer like strategy. This idea was subsequently extended to, in addition to validity guarantees, provide quality guarantees [Mandad and Campen 2021], and generalized from polynomial to rational elements [Khanteimouri et al. 2022; Yang et al. 2022]. These methods address the 2D triangle mesh setting and do not readily extend to 3D.

2.2 3D Curved Meshing

Similar to the 2D case, for the generation of 3D higher-order tetrahedral meshes there are mainly indirect methods, starting with a linear mesh and deforming it so as to get closer towards conformance [Abgrall et al. 2014; Persson and Peraire 2009; Moxey et al. 2016; Fortunato and Persson 2016; Poya et al. 2016; Xie et al. 2013]. This can be done in an interior-point manner, so as to maintain regularity [Persson and Peraire 2009; Ruiz-Gironés et al. 2017; Jiang et al. 2021]. Some deformation approaches with a sound solid mechanics foundation come with favorable theoretical properties with regard to reaching conformance [Turner et al. 2018]; in the discrete setting (fixed mesh, fixed order), however, the situation is less clear. Enforcing conformance by snapping or projecting to the boundary has been considered in 3D as well [Dey et al. 1999; Luo et al. 2004; Gargallo-Peiró et al. 2015b], at the cost of losing regularity in general.

Some direct methods, constructing curved elements right away, have been described as well, often restricted to a specific (low) order, such as for quadratic elements [George and Borouchaki 2012; Dey et al. 2001]. Advancing front type algorithms can be generalized to the curved case, though without regularity guarantees [Mohammadi and Shontz 2021]. Untangling methods can also be applied in the curved 3D setting [Toulorge et al. 2013, 2016; Escobar et al. 2003], possibly with additional mesh structure modification operators [Luo et al. 2004], though achieving regularity in such an *a posteriori* manner is even more challenging in 3D than in 2D.

The approach of Feng et al. [2018] optimizes geometry and topology of the curved mesh in tandem based on the optimal Delaunay triangulation principle. Regularity and conformance are both promoted (the latter through soft fitting term) but not guaranteed. Liu et al. [2021] and Jiang et al. [2021] likewise modify mesh geometry and topology in tandem, in an incremental remeshing manner, preserving regularity. While not yielding a conforming mesh, this incremental approach allows bounding the approximation error.

None of these previous works generally generate tetrahedral meshes with polynomial elements that are guaranteed to be regular *and* conforming to the given boundary surfaces—not all of them even aim to, due to different problem settings. This is the gap for which our method provides a first solution.

3 METHOD OVERVIEW

The input to our method is a description of the surfaces in $\mathbb{R}^3$ that the generated tetrahedral mesh shall conform to. This can be the boundary of the domain to be meshed, but may also include material interfaces or any other feature surfaces to be respected. For generality, we assume these surfaces are represented as arbitrarily structured meshes of curved polynomial triangles, Bézier triangles in $\mathbb{R}^3$. Note that various smooth surface representations, such as polynomial tensor product patches or (trimmed) B-spline surfaces,

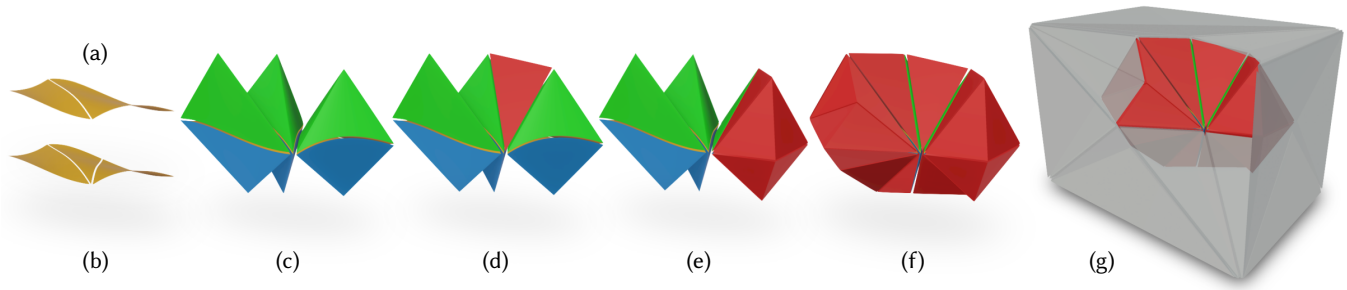

Fig. 2. Overview of our 3D Bézier Guarding approach, from input surface triangles (a) to output volume tetrahedral mesh (g). See Section 3 for an explanation. Note that here both sides of the input are treated (blue and green), whereas in Fig. 1 meshing was restricted to the interior of a closed object. For visual clarity, triangles and tetrahedra are shown slightly shrunken.

can be exactly partitioned into triangles of sufficient polynomial order. Details on input assumptions follow in Section 3.2.

Our goal is to algorithmically generate a higher-order tetrahedral mesh that perfectly conforms to these input surfaces, i.e. sets of curved triangles. The main conceptual idea is as follows:

(1) On each input triangle, create a tetrahedron such that, by construction, it is regular and conforms to the triangle.
(2) On each input edge, between two input triangles, create a fan of tetrahedra such that, by construction, they are regular, conform to the curved edge, and together fill the "gap" between the adjacent tetrahedra constructed in step 1.

Importantly, we create all these tetrahedra such that they are not only conforming and regular, but such that all their exposed facets (not conforming to an input triangle or a neighboring tetrahedron) are planar, and all their exposed edges are straight. The remaining space therefore is polyhedral, bounded in a piecewise linear manner.

(3) Apply a standard linear tetrahedral conforming meshing algorithm to mesh the remaining space.
(4) Combine all three types of tetrahedra, from steps 1–3, to yield the output mesh.

The key challenge lies in finding constructions (for steps 1 and 2) that ensure conformance and regularity, as well as pairwise disjointness of all these tetrahedra. Unfortunately, this is not even feasible in general. However, as we show, there is a subdivision of the input surface meshes for which this problem is feasible and for which we are able to devise a suitable construction. This subdivision is incrementally discovered in our method. We start with the original input mesh, and wherever the construction is not applicable or yields intersecting elements, local subdivision is performed on the mesh. Termination is guaranteed by carefully designing the constructions to exhibit some specific convergence properties under subdivision.

Relation to 2D Bézier Guarding. Conceptually, our method can be seen as a 3D version of the 2D method of Mandad and Campen [2020a] that generates higher-order triangle meshes conforming to given curves in the plane. The idea of that method is to cover the curves by conforming triangle elements, subdividing where necessary, and finally filling the remaining space with a linear triangle mesh. A straightforward generalization of this idea to 3D, however, is hindered by multiple challenges. Localized subdivision of a triangle mesh is more complex than bisecting curves; the condition (*guardability*) that drives subdivision does not naturally extend to 3D; the triangle construction that ensures regularity does not apply to tetrahedra; in 2D one type of elements is sufficient to cover the curves, in 3D we need multiple types, to cover the curved input faces as well as to cover the curved input edges before the reduction to a linear meshing problem is complete.

Illustration. Fig. 2 illustrates the the key steps of our 3D Bézier Guarding approach. The input (a) is a curved triangulation of a piecewise polynomial surface, i.e. a set of non-intersecting Bézier triangles (called patches, yellow), for visual clarity here just two. First, the guardability of the patches is checked; here one of them is unguardable. This triggers refinement (b) by means of a split. Then, *face elements* (c) are constructed on the guardable patches from both sides (blue and green). These are Bézier tetrahedra that conform exactly to the input by construction. Their edges are straight, but their side facets are still curved since they need to conform to the curved edges of their yellow base triangle. Therefore, in the next step, *edge elements* (red) are constructed, for the inner curved edges (d) as well as for the boundary curved edges (e). These conform to the edge as well as to the adjacent face elements by construction. Together, the face elements and the edge elements form a layer around the input patches that entirely conceals its curved nature; the outer facets of the edge elements are planar triangles by construction (f). Whenever some of these elements are not constructible or intersect mutually, this is resolved by further subdividing the responsible patches and reconstructing face and edge elements for the newly created sub-triangles. Afterwards, all compartments of the remaining space are bounded in a piecewise planar manner, such that they can be meshed with straight-sided tetrahedra (gray) using standard meshing methods (g). All of the constructed elements including face, edge, and straight elements are provably regular by construction, and they join conformingly in a C^0 manner.

In the following we clarify the technical background (Section 3.1), the input specification (Section 3.2), and present the main algorithm (Section 3.3). Details on the subroutines employed by the algorithm follow (Section 4).

3.1 Bézier Elements

Without loss of generality, we express all polynomial elements (input triangles and output tetrahedra) in the simplicial Bernstein-Bézier basis of the desired polynomial order n. To this end, let

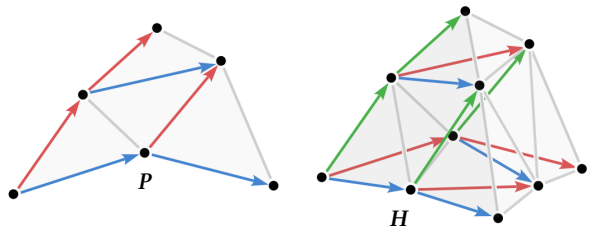

Fig. 3. Bézier triangle $\boldsymbol{P}$ (left) and Bézier tetrahedron $\boldsymbol{H}$ (right), here of degree 2, with blue, red, and green control vectors Γ^0, Γ^1, Γ^2.

$\triangle = \{(u,v) \mid u,v \geq 0, u+v \leq 1\}$ denote a unit triangular domain, $\blacktriangle = \{(u,v,w) \mid u,v,w \geq 0, u+v+w \leq 1\}$ a unit tetrahedral domain.

Definition 1 (Bézier Triangle). *A 3D Bézier triangle* $\mathbf{P}(u,v) : \triangle \to \mathbb{R}^3$ *of degree* n *is defined by its control points (forming its control net)* $\boldsymbol{p}_{ij} \in \mathbb{R}^3$, $(i,j \geq 0,\ i+j \leq n)$ *as:*

$$\mathbf{P}(u,v) = \sum_{i+j \leq n} \boldsymbol{p}_{ij} B^n_{ij}(u,v),$$

where $B^n_{ij}(u,v)$ *are the triangular Bernstein polynomials. Moreover,* $\Gamma^0 = \{\Gamma^0_{ij}\} = \{\boldsymbol{p}_{(i+1)j} - \boldsymbol{p}_{ij}\}$ *and* $\Gamma^1 = \{\Gamma^1_{ij}\} = \{\boldsymbol{p}_{i(j+1)} - \boldsymbol{p}_{ij}\}$ *$(i,j \geq 0,\ i+j \leq n-1)$ we call blue and red vectors of the control net, respectively.*

This is illustrated in Fig. 3, and naturally extends to tetrahedral elements as follows:

Definition 2 (Bézier Tetrahedron). *A Bézier tetrahedron* $\mathbf{H}(u,v,w) : \blacktriangle \to \mathbb{R}^3$ *of degree* n *is defined by its control net* $\boldsymbol{p}_{ijk} \in \mathbb{R}^3$, $(i,j,k \geq 0,\ i+j+k \leq n)$ *as:*

$$\mathbf{H}(u,v,w) = \sum_{i+j+k \leq n} \boldsymbol{p}_{ijk} B^n_{ijk}(u,v,w)$$

where $B^n_{ijk}(u,v,w)$ *are the tetrahedral Bernstein polynomials.* $\Delta^0 = \{\Delta^0_{ijk}\} = \{\boldsymbol{p}_{(i+1)jk} - \boldsymbol{p}_{ijk}\}$, $\Delta^1 = \{\Delta^1_{ijk}\} = \{\boldsymbol{p}_{i(j+1)k} - \boldsymbol{p}_{ijk}\}$ *and* $\Delta^2 = \{\Delta^2_{ijk}\} = \{\boldsymbol{p}_{ij(k+1)} - \boldsymbol{p}_{ijk}\}$ *$(i,j,k \geq 0,\ i+j+k \leq n-1)$ we call blue, red, and green vectors of the control net, respectively. The control points* $\{\boldsymbol{p}_{..0}\}$ *(with* $k=0$*) form a Bézier triangle, one of the four facets of tetrahedron* $\mathbf{H}$*, that we call the* base *of* $\mathbf{H}$*.*

3.1.1 Regularity. A curved tetrahedral element defined by a Bézier tetrahedron $\mathbf{H}(u,v,w)$ is *regular* if $\mathbf{H}$ is *locally injective*. This is the case iff $\det(J_\mathbf{H}) > 0$ for all $(u,v,w) \in \blacktriangle$, where $J_\mathbf{H}$ is the Jacobian of $\mathbf{H}$. Concretely, $J_\mathbf{H}$ is defined as:

$$J_\mathbf{H}(u,v,w) = \begin{pmatrix} \frac{\partial x}{\partial u}(u,v,w) & \frac{\partial x}{\partial v}(u,v,w) & \frac{\partial x}{\partial w}(u,v,w) \\ \frac{\partial y}{\partial u}(u,v,w) & \frac{\partial y}{\partial v}(u,v,w) & \frac{\partial y}{\partial w}(u,v,w) \\ \frac{\partial z}{\partial u}(u,v,w) & \frac{\partial z}{\partial v}(u,v,w) & \frac{\partial z}{\partial w}(u,v,w) \end{pmatrix},$$

where:

$$\begin{aligned} \frac{\partial x}{\partial u}(u,v,w) &= n \sum_{i+j+k \leq n-1} \left(x_{(i+1)jk} - x_{ijk}\right) B^{n-1}_{ijk}(u,v,w), \\ \frac{\partial x}{\partial v}(u,v,w) &= n \sum_{i+j+k \leq n-1} \left(x_{i(j+1)k} - x_{ijk}\right) B^{n-1}_{ijk}(u,v,w), \\ \frac{\partial x}{\partial w}(u,v,w) &= n \sum_{i+j+k \leq n-1} \left(x_{ij(k+1)} - x_{ijk}\right) B^{n-1}_{ijk}(u,v,w), \end{aligned} \tag{1}$$

and analogously for y and z, where $\boldsymbol{p}_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk})$ are the control points of $\mathbf{H}$.

The Jacobian determinant can equivalently be written as

$$\det(J_\mathbf{H}) = \left(\frac{\partial}{\partial u}\mathbf{H} \times \frac{\partial}{\partial v}\mathbf{H}\right) \cdot \frac{\partial}{\partial w}\mathbf{H} \tag{2}$$

$$= n^3 \left(\left(\sum \Delta^0_{ijk} B^{n-1}_{ijk}\right) \times \left(\sum \Delta^1_{ijk} B^{n-1}_{ijk}\right)\right) \cdot \left(\sum \Delta^2_{ijk} B^{n-1}_{ijk}\right). \tag{3}$$

This determinant is positive iff the three columns of $J_\mathbf{H}$ are in *right-hand formation*, i.e. the third column has a positive dot product with the cross product of the first and second. Note that the columns are, for every $(u,v,w) \in \blacktriangle$, *convex* combinations of the blue, red, and green vectors of $\mathbf{H}$, respectively.

3.2 Input

The input to our method is a set of polynomial 3D Bézier triangles $\{\mathbf{P}_i(u,v) : \triangle \to \mathbb{R}^3\}$, called *patches* in the following, of arbitrary degree n. These triangles may coincide along their edges, forming (possibly non-manifold) meshes. They are assumed to meet the following conditions, so as to define a feasible regular conforming meshing problem:

- **No Irregularity:** Each patch $\mathbf{P}_i$ is regular, i.e. $\frac{\partial \mathbf{P}_i}{\partial u} \times \frac{\partial \mathbf{P}_i}{\partial v}(u,v) \neq 0$ for $(u,v) \in \triangle$.
- **No Intersection:** Patches may only intersect at their boundaries, i.e. $\mathbf{P}_i(u,v) = \mathbf{P}_j(w,r)$, $i \neq j \Rightarrow (u,v),(w,r) \in \partial\triangle$. If two patches intersect, then in an entire edge or a corner.
- **No Degeneracy:** At coincident edge or corner points, no two patches form a zero angle, i.e. $\mathbf{P}_i(u,v) = \mathbf{P}_j(w,r) \Rightarrow \frac{\partial \mathbf{P}_i}{\partial u} \times \frac{\partial \mathbf{P}_i}{\partial v}(u,v) \neq -\frac{\partial \mathbf{P}_j}{\partial u} \times \frac{\partial \mathbf{P}_j}{\partial v}(w,r)$.

Note that *rectangular* Bézier patches of tensor-product type, which are a popular model for polynomial surface representation, can be exactly converted into pairs of Bézier triangles of sufficient order. In this sense they are supported as input as well. A 3D Bézier rectangle $\mathbf{T}(u,v) : [0,1]^2 \to \mathbb{R}^3$ of degree (m,n) is defined by its control points $\boldsymbol{p}_{ij} \in \mathbb{R}^3$ as:

$$\mathbf{T}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \boldsymbol{p}_{ij} B^m_i(u) B^n_j(v),$$

where $B^m_i(u)$ and $B^n_j(v)$ are the Bernstein basis functions of degree m and n, respectively. Then, the Bézier triangle $\mathbf{P}_1(u,v)$ of degree $m+n$ with the following control points $\{\boldsymbol{p}'_{ab}\}$ represents the first part of the same surface [Goldman and Filip 1987]:

$$\boldsymbol{p}'_{ab} = \frac{1}{\binom{m+n}{n}} \sum_{j=0}^{a} \sum_{k=\max\{0,b-m+j\}}^{\min\{b,n-a+j\}} \boldsymbol{p}_{jk} \binom{a}{j}\binom{b}{k}\binom{m+n-a-b}{m+k-j-b}.$$

The second Bézier triangle $\mathbf{P}_2(u,v)$ can be obtained from the reverse array of the rectangular patch's control points, i.e. $\{\boldsymbol{p}_{m-i,n-j}\}$.

More generally, *trimmed* rectangular patches can conceptually be partitioned exactly into a set of Bézier triangles as well, though with challenges due to high resulting degrees, cf. Section 7.1.

3.3 Algorithm

We now present our overall algorithm to generate a higher-order tetrahedral mesh that conforms to the given set $\{\mathbf{P}_i\}$ of triangular

Algorithm 1: Higher-Order Mesh Generation Algorithm

```
Data: Triangular 3D Bézier patches {P_i} of order n
Result: Regular conforming tetrahedral mesh of order n
queue q ← all edges of input mesh {P_i}
while q is not empty do
    e ← q.pop()
    {T_i} ← incident input triangle patches of e
    if any T_i is not strongly guardable then                ▷ Sec. 4.1
        subdivide T_i                                        ▷ Sec. 4.5
        update(q)
        continue
    end
    for each T_i do
        for both sides construct a face element              ▷ Sec. 4.2
    end
    if e is not coverable then
        subdivide the T_i with tallest face element          ▷ Sec. 4.5
        update(q)
        continue
    end
    construct edge elements for e                            ▷ Sec. 4.3
    if q is empty then
        for each pair of non-adjacent edges e, e' do
            if R_e ∩ R_e' ≠ ∅ then                           ▷ Sec. 4.6
                subdiv. patch with tallest face elem.        ▷ Sec. 4.5
                update(q)
                continue the while loop
            end
        end
    end
end
tetrahedralize the remaining space
compute control points for these tetrahedra                  ▷ Sec. 4.4
```

patches. Algorithm 1 list the algorithm's high-level structure. The employed subroutines are detailed in Section 4.

The algorithm considers all edges from an (arbitrarily ordered) queue q that initially contains all edges of the input triangular mesh; shared edges of two (or more) adjacent triangular patches are considered one edge, and are included in the queue only once. For each edge e in q, if any of its incident Bézier triangles $\mathbf{T}_i$ (two or more for inner edges, one for boundary edges) is unguardable we subdivide it. Generally, whenever a triangle is subdivided, we remove its edges (if any) from the queue q and add all of the new triangles' edges to q; this is denoted as update(q) in the algorithm. Otherwise, for each $\mathbf{T}_i$, we construct two corresponding face elements (if not already constructed), one for each side. Then, we check whether edge e is *coverable*. We say edge e is coverable if our edge element construction is able to create edge elements covering it. If e is not coverable, we subdivide an incident patch of e; the choice of the patch with the tallest face element for this purpose is justified in Section 5.3. Otherwise, the edge elements for e are constructed. The above process continues until the queue q becomes empty which indicates that all of the face and edge elements have been successfully constructed.

In the next part of the algorithm, potential intersections of the elements are resolved. Each edge e is surrounded by a cyclic fan of incident curved tetrahedra (the constructed face and edge elements). These tetrahedra of one edge are pairwise disjoint by construction. But the fans of two (non-adjacent) edges may overlap. Let $\mathbf{R}_e$ be the set of these tetrahedra. For any non-adjacent pair of edges $e \neq e'$ we check whether these tetrahedra of $\mathbf{R}_e$ are disjoint from the ones in $\mathbf{R}_{e'}$. If they overlap we subdivide the patch (incident to e or e') whose face element has the maximum height among all face elements involved in $\mathbf{R}_e$ and $\mathbf{R}_{e'}$.

Whenever an input patch is subdivided in the algorithm, its incident face and edge elements (if already constructed) are deleted.

In the end, a linear tetrahedral mesh generation method is applied to mesh the remaining space, e.g. by applying it to a bounding box, specifying the boundary of the set of generated face and edge elements as holes. Our constructions ensure that these holes are piecewise planar.

4 SUBROUTINES

To clarify the main steps of Algorithm 1, we first describe the construction method of the main elements, namely face, edge, and straight elements in detail. We show that they are regular by construction and join with C^0 continuity. Afterwards, we turn to the subdivision and disjointness test methods, which form further important parts of the algorithm.

4.1 Guardability

We first define *guardability* of 3D Bézier triangles (in analogy to the notion used for 2D Bézier curves by [Mandad and Campen 2020a]) and give an algorithm to test whether an input patch is guardable. In Section 4.2 we then present an algorithm to construct a Bézier tetrahedron that conforms to a Bézier triangle—and is applicable exactly if the triangle is guardable.

We say an oriented plane (through the origin) with normal vector $\boldsymbol{n}$ *supports* a vector $\boldsymbol{v}$ iff $\boldsymbol{n}^\top \boldsymbol{v} > 0$. It *weakly-supports* a vector $\boldsymbol{v}$ iff $\boldsymbol{n}^\top \boldsymbol{v} \geq 0$. We define the following:

Definition 3 (Guardability). *A 3D Bézier triangle* $\mathbf{P}$ *is guardable if the following conditions are satisfied:*

(1) ***(Supporting plane existence)*** *There is plane* P_1 *that supports all blue vectors and all red vectors* $\Gamma^0 \cup \Gamma^1$ *of* $\mathbf{P}$.
(2) ***(Separating plane existence)*** *There is plane* P_2 *that supports all blue vectors and all* negated *red vectors (it* separates *blue and red vectors).*

Let the *base plane* of Bézier triangle $\mathbf{P}$ be the plane that contains the three corner control points of $\mathbf{P}$. Its normal vector will be denoted $\boldsymbol{n}$ in the following.

Definition 4 (Orthogonal Guardability). *A 3D Bézier triangle* $\mathbf{P}$ *is orthogonally guardable if there exist supporting and separating planes orthogonal to the base plane.*

Fig. 4 illustrates a cross section view, along $\boldsymbol{n}$, of the vectors Γ^0 and Γ^1, an orthogonal supporting plane, and an orthogonal separating plane in a case where the above conditions are satisfied.

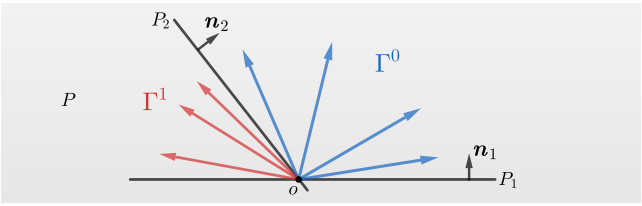

Fig. 4. Cross section view, along normal $\boldsymbol{n}$ of base plane P (gray), showing the vectors Γ^0 and vectors Γ^1, as well as a supporting plane P_1 and a separating plane P_2 with normal vectors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$.

4.1.1 Orthogonal Guardability Test. The existence of a supporting and a separating plane (and thus guardability) can be tested using a simple linear program requiring the proper dot product signs of the plane's unknown normal vector with the red and blue vectors. The existence of an *orthogonal* supporting or separating plane (and thus orthogonal guardability) can be tested by expressing this in a 2D projection on the base plane.

Besides testing for the existence of planes, we furthermore compute four particular planes explicitly, to be used in the construction of regular tetrahedra in the following subsections. We make use of the following definitions:

- Let *cones* C_0 and C_1 denote the conical hull of the vectors Γ^0 and Γ^1, respectively, i.e. $C_k = \left\{\sum \alpha_i \boldsymbol{b}_i \mid \alpha_i \in \mathbb{R}_{\geq 0}, \boldsymbol{b}_i \in \Gamma^k\right\}$. C_0 is called the blue cone, C_1 the red cone. Note that for a guardable Bézier triangle these cones are disjoint, due to the existence of a separating plane, and contained in a common halfspace, due to the existence of a supporting plane.
- Let *tangent planes* $T_{++}, T_{--}, T_{-+}, T_{+-}$ be oriented planes, with normals $\boldsymbol{n}_{++}$, etc., that are tangent to both cones, C_0 and C_1. Their orientation is chosen such that $(T \cap C_0, T \cap C_1, \boldsymbol{n})$ is right-handed for each of these four planes. They differ in whether C_0 and C_1 lie on their positive or negative side; the first subscript sign indicates the side that contains C_0, the second the side that contains C_1. Fig. 5 illustrates these four planes.

Note that T_{++} as well as $-T_{--}$ are weakly-supporting planes in the sense of Definition 3. T_{+-} as well as $-T_{-+}$ are weakly-separating planes. These four planes support the triangle's base normal iff the triangle is not only guardable but orthogonally guardable.

Regarding the computation of the cones C_0, C_1, note that this can be performed by means of a convex hull algorithm [Barber et al. 1996]. This yields the sets of spanning vectors $\bar{\Gamma}^k \subseteq \Gamma^k$ that form the edges of the cones. Regarding the computation of the tangent planes, consider the two convex polygons formed by the intersection of the two cones with a shifted supporting plane. Simple linear time algorithms [Preparata and Hong 1977; Toussaint 1983] (and even sublinear time algorithms [Kirkpatrick and Snoeyink 1995]) are known to determine these polygons' pairs of vertices that span their common (separating) tangent lines. The corresponding pairs of vectors from $\Gamma^0 \times \Gamma^1$ span the sought tangent planes.

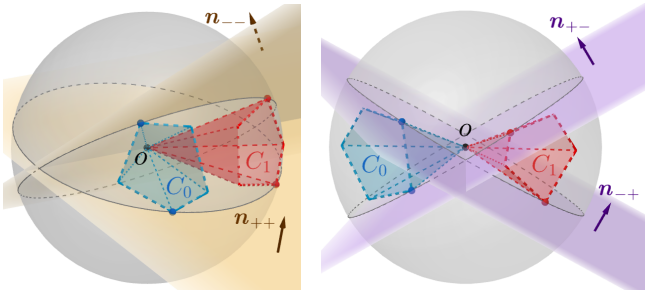

Fig. 5. Left: Outer tangent planes T_{++} and T_{--} (yellow). Right: Inner tangent planes T_{-+} and T_{+-} (purple). The infinite blue and red cones are clipped by a sphere (gray) around origin o for visualization purposes.

4.2 Face Elements

For an orthogonally guardable triangular patch $\mathbf{P} : \{\boldsymbol{p}_{ij}\}$ we construct a Bézier tetrahedron $\mathbf{H} : \{\boldsymbol{p}_{ijk}\}$ as the *face element* of $\mathbf{P}$ by first adopting the control points of $\mathbf{P}$ as control net layer 0 of $\mathbf{H}$, i.e. $\boldsymbol{p}_{ij0} = \boldsymbol{p}_{ij}$ $(i, j \geq 0, i + j \leq n)$. This implies conformance, i.e. the base triangle of $\mathbf{H}$ is exactly $\mathbf{P}$. The other control net layers of $\mathbf{H}$ are carefully placed in such a way that $\mathbf{H}$ is regular by construction.

4.2.1 Construction Idea. The blue and the red vectors in the first layer of $\mathbf{H}$ (layer 0 with vectors Δ^0_{ij0} and Δ^1_{ij0}) define cones C_0 and C_1 (as in Section 4.1.1). Our algorithm to construct $\mathbf{H}$ places the remaining control points (layers 1 and up) in such a way that *all* blue vectors lie in this cone C_0 and *all* red vectors in C_1. Consequently, for any (u, v, w), the first and the second column of the Jacobian matrix $J_{\mathbf{H}}$ are two distinct vectors inside C_0 and C_1, respectively (since they are convex combinations of blue and red vectors in Δ^0 and Δ^1).

The union of all planes through the origin that intersect both C_0 and C_1 define a sub-space which is the locus of all vectors that could

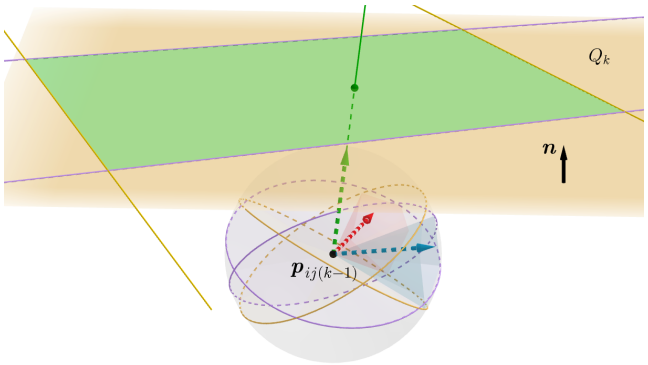

Fig. 6. The four great circles shown on the sphere are the sphere intersections of the four tangent planes T_{++}, T_{--}, T_{-+}, T_{+-}, as shown in Fig. 5, translated to a concrete point $\boldsymbol{p}$. On top the intersection of these with a plane Q_k is shown. If the green vector is chosen to point inside the polygonal green region $g(\boldsymbol{p}) \cap Q_k$, it is right-handed with respect to any pair of vectors from the blue and red cones.

be coplanar with some pair of vectors from C_0 and C_1. Thus, to preclude the zero-valued determinant of the Jacobian, the construction algorithm places the control points such that the third column's vector of the Jacobian never lies inside the above sub-space. The boundary of this sub-space is actually formed by the above defined tangent planes of C_0 and C_1. For a point $\boldsymbol{p}$, let $T_{-+}(\boldsymbol{p})$, $T_{+-}(\boldsymbol{p})$, $T_{++}(\boldsymbol{p})$ and $T_{--}(\boldsymbol{p})$ denote the translations of the oriented tangent planes to point $\boldsymbol{p}$. Then, we denote as $g(\boldsymbol{p})$ the intersection of the four halfspaces on the positive sides of these four planes. Observe that for any point $\boldsymbol{p}'$ in the interior of $g(\boldsymbol{p})$, the (green) vector $\boldsymbol{p}'-\boldsymbol{p}$ (Fig. 6) together with any arbitrary pair of vectors from inside C_0 and C_1 satisfies the nonzero right-hand formation at point $\boldsymbol{p}$.

By this argument, we position the other layers $k>0$ on parallel planes Q_k with normal vectors $\boldsymbol{n}$ as follows. To have proper green vectors $\Delta^2_{ij(k-1)}$, it suffices to position the control points $\boldsymbol{p}_{ijk}$ inside the *green regions* defined by $Q_k \cap g(\boldsymbol{p}_{ij(k-1)})$, as illustrated in Fig. 6. For simplicity, we place all control points of layers $k>0$ inside the common *guard region* $G_{\mathbf{P}} = \bigcap_{ij} g(\boldsymbol{p}_{ij})$.

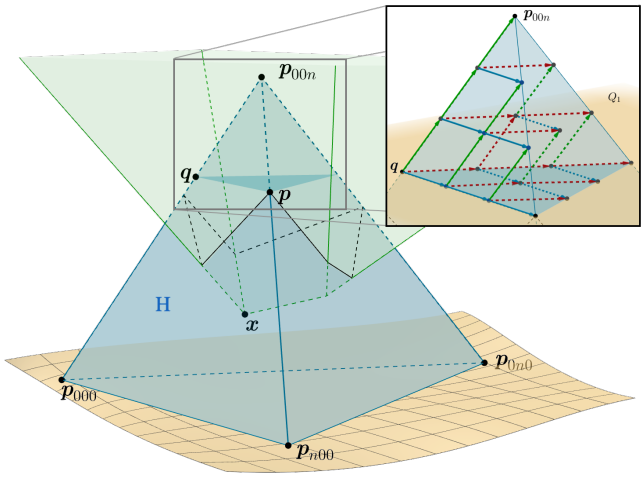

Fig. 7. Placement of the control points for the control net layers $k>0$, above the point $\boldsymbol{q}$. The green polyhedron is the guard region $G_{\mathbf{P}}$.

4.2.2 Construction Algorithm. Algorithm 2 describes the face element construction in detail. For a given orthogonally guardable Bézier triangle $\mathbf{P}$, we first calculate the tangent planes and their normal vectors $\{\boldsymbol{n}_{++}, \boldsymbol{n}_{--}, \boldsymbol{n}_{-+}, \boldsymbol{n}_{+-}\}$. The planes defining the boundaries of all $g(\boldsymbol{p}_{ij})$ $(i,j>0, i+j\le n)$ can be grouped into four families of parallel tangent planes $\{T_{++}(\boldsymbol{p}_{ij})\}$, $\{T_{--}(\boldsymbol{p}_{ij})\}$, $\{T_{-+}(\boldsymbol{p}_{ij})\}$, and $\{T_{+-}(\boldsymbol{p}_{ij})\}$. Due to being parallel, the boundary of $G_{\mathbf{P}}$ is formed by only four *extreme planes* E_{++}, E_{--}, E_{-+}, and E_{+-}, one from each of these families. Concretely, for each family, with common normal vector $\boldsymbol{n}'$, its extreme plane is the one that does not have any control point $\boldsymbol{p}_{ij}$ on its positive side. Then $G_{\mathbf{P}}$ simply is the intersection of the positive (closed) half-spaces of these four extreme planes.

Let $\boldsymbol{x}$ be the lowest point (over the base) within $G_{\mathbf{P}}$ (see Fig. 7). We will position the *tip-point* $\boldsymbol{p}_{00n}$ above $\boldsymbol{x}$ (thus inside $G_{\mathbf{P}}$) but need to make sure that it lies above the base triangle, i.e. within the (infinite) prism M with triangular base $\{\boldsymbol{p}_{000}, \boldsymbol{p}_{n00}, \boldsymbol{p}_{0n0}\}$ and axis in direction of $\boldsymbol{n}$. This is required for the proof of convergence in Section 5. To ensure this, if $\boldsymbol{x}$ is not in the interior of prism M, we define the center line L_c passing through $(\boldsymbol{p}_{000}+\boldsymbol{p}_{n00}+\boldsymbol{p}_{0n0})/3$ in direction of $\boldsymbol{n}$ and replace $\boldsymbol{x}$ by the intersection point of L_c and the boundary of $G_{\mathbf{P}}$ (i.e. the highest intersection point of L_c and the four extreme planes). Afterwards, we position the tip-point $\boldsymbol{p}_{00n}$ inside $G_{\mathbf{P}} \cap M$ by $\boldsymbol{p}_{00n} = \boldsymbol{x} + \mu h_{\boldsymbol{x}} \boldsymbol{n}$, where $h_{\boldsymbol{x}}$ is the height of $\boldsymbol{x}$ over the base plane and μ is a positive constant (that allows for balancing between the size of the resulting element and its distortion). We use $\mu=1$ by default. In the (planar) special case of $h_{\boldsymbol{x}}=0$, we instead set $h_{\boldsymbol{x}}$ to a value proportional to the base triangle area.

Then, by considering the straight side edges $\overline{\boldsymbol{p}_{00n}\boldsymbol{p}}$, $\boldsymbol{p} \in \{\boldsymbol{p}_{000}, \boldsymbol{p}_{n00}, \boldsymbol{p}_{0n0}\}$, we calculate $\boldsymbol{p}$ as the highest intersection point of the side edges with $G_{\mathbf{P}}$ (see Fig. 7). Let Q_1 (the plane that will contain layer 1 of the control net) be the plane with normal $\boldsymbol{n}$ passing through $\boldsymbol{p}$. The intersection of Q_1 and $\overline{\boldsymbol{p}_{00n}\boldsymbol{p}_{000}}$ is the first control point of the second layer of $\mathbf{H}$ which is denoted by $\boldsymbol{q}$. Finally, by a simple uniform distribution as Fig. 7 illustrates, the control points of layer 1 and then the other layers can be positioned, in essence by uniformly subdividing the top portion of the tetrahedron bounded by Q_1.

Algorithm 2: Face Element Construction

Data: Orthogonally guardable Bézier triangle $\mathbf{P}: \{\boldsymbol{p}_{ij}\}$
Result: Regular face element conforming to $\mathbf{P}$

$\boldsymbol{n} \leftarrow$ normal vector of the base plane of $\mathbf{P}$
calculate tangent planes' normals $N = \{\boldsymbol{n}_{++}, \boldsymbol{n}_{--}, \boldsymbol{n}_{-+}, \boldsymbol{n}_{+-}\}$
calculate extreme planes $\mathbf{E} = \{E_{++}, E_{--}, E_{-+}, E_{+-}\}$
calculate lowest point $\boldsymbol{x}$ of $G_{\mathbf{P}}$
calculate prism M
if $\boldsymbol{x}$ is not in the interior of M **then**
  calculate line L_c
  $\boldsymbol{x} \leftarrow$ highest intersection point of $\{L_c \cap E \mid E \in \mathbf{E}\}$
end
$\boldsymbol{p}_{00n} \leftarrow \boldsymbol{x} + \mu h_{\boldsymbol{x}} \boldsymbol{n}$
$\mathbf{P}_c \leftarrow \{\boldsymbol{p}_{000}, \boldsymbol{p}_{n00}, \boldsymbol{p}_{0n0}\}$
$\boldsymbol{p} \leftarrow$ highest intersection point of $\{E \cap \overline{\boldsymbol{p}_{00n}\boldsymbol{p}} \mid (E, \boldsymbol{p}) \in \mathbf{E} \times \mathbf{P}_c\}$
calculate $\boldsymbol{q}$ based on $\boldsymbol{p}$
adopt control point layer 0 from $\mathbf{P}$
place layers $k>0$ above $\boldsymbol{q}$

The following lemmas show that Algorithm 2 successfully creates a regular conforming face element for any given orthogonally guardable triangular patch.

Lemma 1. *For any orthogonally guardable Bézier triangle $\mathbf{P}$, $G_{\mathbf{P}}$ is non-empty and any line in direction of the normal vector $\boldsymbol{n}$ intersects the boundary of $G_{\mathbf{P}}$ in exactly one point.*

Proof. Since $\mathbf{P}$ is orthogonally guardable, all of the tangent planes $T_{++}, T_{--}, T_{-+}, T_{+-}$ have the normal vector $\boldsymbol{n}$ on their positive sides. This implies that $\boldsymbol{n}_t^\top \boldsymbol{n} > 0$, for all the normal vectors $\boldsymbol{n}_t$ of the boundary planes of $G_{\mathbf{P}}$ since they are parallel to tangent planes. Therefore, any line in direction of the normal vector $\boldsymbol{n}$ intersects the boundary of $G_{\mathbf{P}}$ in exactly one point. □

Lemma 2. *Given an orthogonally guardable Bézier triangle* **P**, *Algorithm 2 creates a regular face element conforming to* **P**.

Proof. As the base face of the constructed tetrahedron is a Bézier triangle equal to **P**, it conforms exactly to **P**. Lemma 1 implies that the lowest point of $G_{\mathbf{P}}$, namely $\boldsymbol{x}$, always exists; it is easily computed (e.g. by a 3D convex hull algorithm or by intersecting multiple triples of extreme planes). The same holds for the intersection of center line L_c and $G_{\mathbf{P}}$. In addition, Lemma 1 ensures that the tip-point $\boldsymbol{p}_{00n}$, shifted up from $\boldsymbol{x}$ along $\boldsymbol{n}$, is inside the guard region $G_{\mathbf{P}}$. By construction, the blue and the red vectors of the layers $k > 0$ (all parallel to base triangle edges) lie inside C_0 and C_1, and the green vectors Δ^2_{ijk} $(i, j, k \geq 0, i + j + k \leq n - 1)$ are such that any convex combination thereof is in right-hand formation with any two vectors in C_0 and C_1. Thus, recalling eq. (3), the determinant of the Jacobian is positive everywhere. □

The latter observation is also made use of to state a sufficient injectivity condition for Bézier tetrahedra by Vavasis [2003].

A further interesting property of the face elements resulting from this construction can be shown—which we will exploit for the construction of edge elements in Section 4.3. Let the three facets of a face element that are not the base facet be called *side facets*.

Lemma 3. *For a face element* $\mathbf{H} : \{\boldsymbol{p}_{ijk}\}$, *two side facets, namely those that contain the control point* $\boldsymbol{p}_{000}$, *are guardable.*

Proof. We consider the facet $\mathbf{T} : \{\boldsymbol{p}'_{ij}\}$ consisting of control points $\{\boldsymbol{p}_{0ij} | i, j \geq 0, i + j \leq n\}$; the argument for the other side facet is analogous. Let blue and red vectors of **T** be respectively $\{\Sigma^0_{ij}\} = \{\boldsymbol{p}'_{(i+1)j} - \boldsymbol{p}'_{ij}\}$ and $\{\Sigma^1_{ij}\} = \{\boldsymbol{p}'_{i(j+1)} - \boldsymbol{p}'_{ij}\}$. Observe that the blue vectors of **T** are a subset of the green vectors of **H**; precisely $\{\Sigma^0_{ij}\} = \{\Delta^2_{0ji}\}$. Analogously, the red vectors $\{\Sigma^0_{ij}\}$ of **T** are a subset of the red vectors of **H**, $\{\Sigma^1_{ij}\} = \{\Delta^1_{0ji}\}$. As Fig. 6 demonstrates, one of the inner tangent planes supports all red and green vectors of **H** while the other one separates them (the same as the outer tangent planes). Therefore, $\{\Sigma^0_{ij}\}$ and $\{\Sigma^1_{ij}\}$ are also supported and separated by the same planes which conclude the guardability of **T**. □

Note that the indexing of the control points of a Bézier triangle is arbitrary in the sense that any of its three corner control points could be labeled p_{00} (which then implies the indexes of all other control points of the control net), i.e. indexes can be rotated into three different *orientations*, without changing its shape. Formally, this corresponds to a simple affine reparametrization of the triangle surface. The notion of (orthogonal) guardability depends on this choice (because the blue and red vectors do).

Definition 5 (Strong Guardability). *A 3D Bézier triangle* **P** *is strongly guardable if it is orthogonally guardable in at least two orientations.*

For a strongly guardable triangle, we can modify the face element construction to take the inner and outer tangent planes of two orientations into account. The intersection region of the resulting eight (instead of four) extreme planes is non-empty (because the base normal is contained for both orientations), so the positioning of control points can be based on this modified guard region instead. For this modification, Lemma 3 can be adapted as follows.

Lemma 4. *For a strongly guardable triangle* **T**, *the above construction that takes two orientations into account constructs a regular conforming face element such that* all *of its side facets are guardable.*

Proof. Applying Lemma 3 for one orientation for which **T** is orthogonally guardable proves the guardability of the two side facets containing the corresponding origin corner. Applying it for the second orientation for which **T** is orthogonally guardable proves guardability of the third. Hence, all side facets are guardable. □

4.3 Edge Element Construction

Face elements cover the curved input triangles. However, their curved edges are still exposed, and the face elements' side facets are curved in general. To achieve the goal of concealing all curvedness, we therefore construct a second type of elements, edge elements, covering these curved facets and the curved edges. They essentially fill the gap sectors between face elements. They are designed such that their exposed facets are planar.

On a high level, we distinguish between *inner edges* and *boundary edges* of the input mesh, and apply different constructions. At an inner edge e, there are $k \geq 2$ incident patches ($k = 2$ at manifold

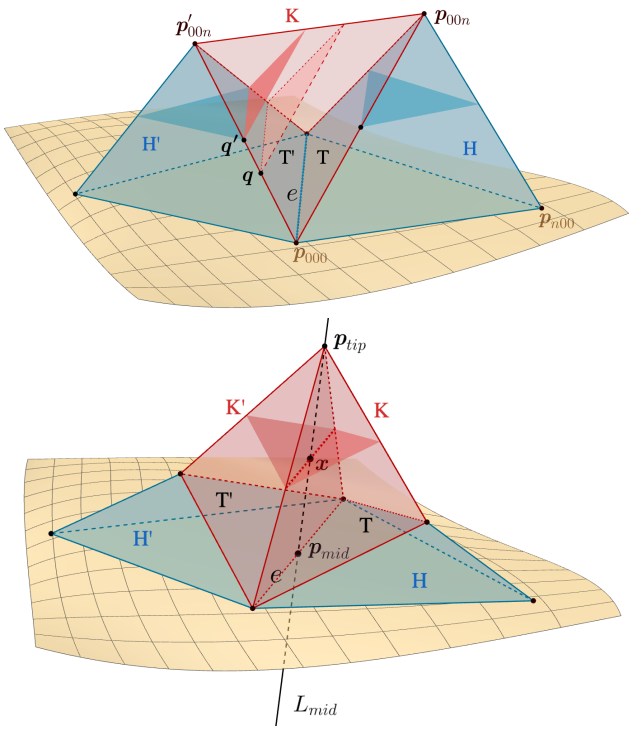

Fig. 8. Illustration of inner edge elements construction. Top: a case where a single tetrahedron is sufficient. Bottom: a case where a fan of two tetrahedra is constructed. Note that a case of straight edge e is shown here for clarity; in general, $e, \mathbf{T}, \mathbf{T}'$ are curved. The exposed triangles of the red tetrahedra $\mathbf{K}, \mathbf{K}'$, by contrast, are always planar by construction.

edges), dividing the space around the edge into k sectors. Each sector contains two face elements that are incident at the edge. We call such a pair *adjacent*. Let $\mathbf{H}$ and $\mathbf{H}'$ be two such adjacent face elements at e. We construct either one or two coupled edge elements between $\mathbf{H}$ and $\mathbf{H}'$ (as illustrated in Fig. 8 by red tetrahedra). We repeat this for the further pairs of adjacent face elements surrounding the edge to complete the construction of the edge elements of e. In the case that e is a boundary edge, four to six coupled edge elements are created (red and purple tetrahedra in Fig. 9). These cases are spelled out in the following.

4.3.1 Inner Edge Elements. Assume two adjacent face elements $\mathbf{H} : \{\boldsymbol{p}_{ijk}\}$ and $\mathbf{H}' : \{\boldsymbol{p}'_{ijk}\}$ and let their adjacent side facets be denoted by $\mathbf{T}$ and $\mathbf{T}'$, respectively (see Fig. 8 top). By Lemma 4 we know that the triangles $\mathbf{T}$ and $\mathbf{T}'$ are guardable. Thus, we first check whether a single edge element $\mathbf{K}$, filling the gap, can be constructed by considering either $\mathbf{T}$ or $\mathbf{T}'$ as the base triangle of $\mathbf{K}$. Assuming $\mathbf{T}$ as the base, we first compute the guard region $G_{\mathbf{T}}$ and then determine whether $\boldsymbol{p}'_{00n}$ lies inside $G_{\mathbf{T}}$. If $\boldsymbol{p}'_{00n} \in G_{\mathbf{T}}$ we compute the $\boldsymbol{q}$-point of $\mathbf{K}$, analogous to the face element construction. Let us denote the first point of the second layer of $\mathbf{T}'$ (on the edge $\overline{\boldsymbol{p}_{000}\boldsymbol{p}'_{00n}}$) by $\boldsymbol{q}'$. If $\boldsymbol{q}'$ lies above $\boldsymbol{q}$ (further from $\boldsymbol{p}_{000}$ along the edge $\overline{\boldsymbol{p}_{000}\boldsymbol{p}'_{00n}}$) we use $\boldsymbol{q}'$ as $\boldsymbol{q}$ in the routine that places the control points of $\mathbf{K}$ (layers $k > 0$) uniformly as described for the face elements construction. Otherwise, if $\boldsymbol{p}'_{00n} \notin G_{\mathbf{T}}$ or $\boldsymbol{q}'$ is not above $\boldsymbol{q}$, we repeat the same procedure by assuming $\mathbf{T}'$ as the base triangle of $\mathbf{K}$.

Algorithm 3: Inner Edge Elements Construction

```
Data: Non-boundary edge e
Result: One or two regular edge elements (for one sector),
        conforming to e and to the adjacent face elements, or
        'uncoverable'
T, T' ← incident face element sides of e
for each S ∈ {T, T'} do
    calculate G_S, q, q', p_tip with respect to base S
    if p_tip ∈ G_S and q' is higher than q then
        q ← q'
        adopt control point layer 0 of tetrahedron K from S
        place layers k > 0 of K above q
        return {K}
    end
end
compute line L_mid : (p_mid, n_mid)
compute the set E of all extreme planes of T and T'
if n_e^⊤ n_mid > 0 for all normal vectors n_e of the planes in E then
    x ← highest intersection point of {L_mid ∩ P | P ∈ E}
    p_tip ← x + μ h_x n_mid
    compute q-points of K and K' w.r.t. p_tip
    adopt control point layers 0 from T and T', respectively
    place layers k > 0 of K, K' above their respective q-points
    return {K, K'}
end
return 'uncoverable'
```

If a valid single edge element cannot be constructed this way, we instead attempt to construct two coupled edge elements $\mathbf{K}$ and $\mathbf{K}'$ as Fig. 8 bottom illustrates. Let $\boldsymbol{n}_{\mathbf{T}}$ and $\boldsymbol{n}_{\mathbf{T}'}$ be the normal vectors of the base triangles of $\mathbf{T}$ and $\mathbf{T}'$ and $\boldsymbol{p}_{\text{mid}}$ be the mid point of the shared edge e. We define the orthogonal bisector line L_{mid} in direction of $\boldsymbol{n}_{\text{mid}} = \boldsymbol{n}_{\mathbf{T}} + \boldsymbol{n}_{\mathbf{T}'}$ passing through the point $\boldsymbol{p}_{\text{mid}}$. If this line intersects each guard region boundary, $\partial G_{\mathbf{T}}$ and $\partial G_{\mathbf{T}'}$, in only one point, we compute $\boldsymbol{x}$ as the highest (the farthest from $\boldsymbol{p}_{\text{mid}}$) of these two points. Note that this is the lowest point in $G_{\mathbf{T}} \cap G_{\mathbf{T}'}$ in direction $\boldsymbol{n}_{\text{mid}}$ from $\boldsymbol{p}_{\text{mid}}$. Then, similar to the face element construction, we set the (shared) tip point $\boldsymbol{p}_{\text{tip}} = \boldsymbol{x} + \mu h_{\boldsymbol{x}} \boldsymbol{n}_{\text{mid}}$. Finally, we compute the q-points (second layer positions) of $\mathbf{K}$ and $\mathbf{K}'$ and complete the layers (using the same method used in the face element construction). Note that to test whether each of the guard regions $G_{\mathbf{T}}$ and $G_{\mathbf{T}'}$ is intersected by line L_{mid} in the desired way, it is sufficient to check whether $\boldsymbol{n}_e^{\top} \boldsymbol{n}_{\text{mid}} > 0$ for all normal vectors $\boldsymbol{n}_e$ of extreme planes.

Algorithm 3 summarizes the above construction method for one sector. We run it for all sectors (two in case of a manifold inner edge) to attempt to create the edge elements of e. If unsuccessful (i.e. neither the single element construction, nor the coupled element construction is applicable for some sector), the algorithm reports that e is uncoverable, requesting subdivision of an involved patch. It will succeed after sufficient subdivision (Lemma 10).

4.3.2 Boundary Edge Elements. Now, the only remaining curved parts to be covered are those facets of face elements that contain the boundary edges. Fig. 9 top shows the sketch of our construction for a given boundary edge e with its only incident triangle $\mathbf{T}$. To construct the edge elements, we define an auxiliary Bézier triangle $\mathbf{S}$, opposite of $\mathbf{T}$ across e, such that two edges of $\mathbf{S}$ are straight while

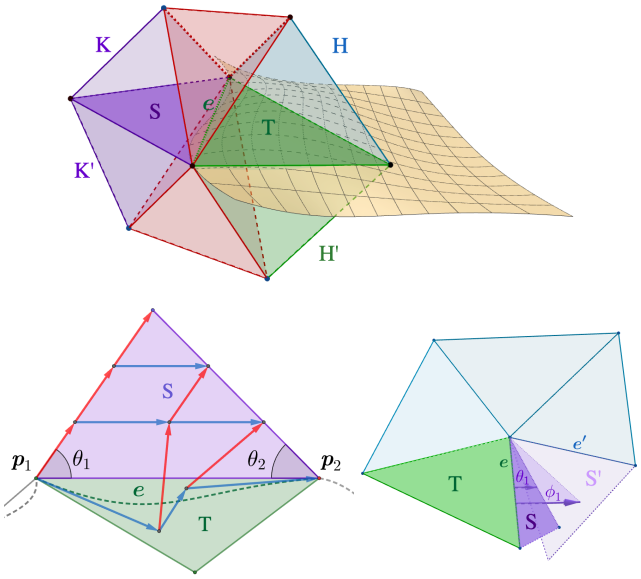

Fig. 9. Top: Illustration of boundary edge elements construction. Middle: Bottom left: Control point positioning to make $\mathbf{S}$ conform to the boundary edge e. Bottom right: Inner angle restriction for $\mathbf{S}$, to avoid intersection with other incident elements.

its third edge conforms to e. If $\mathbf{S}$ is not guardable, subdivision is requested. If it is guardable, we first construct two Bézier tetrahedra $\mathbf{K}$ and $\mathbf{K}'$ conforming to $\mathbf{S}$ by applying the face element construction (Section 4.2) to $\mathbf{S}$ from both sides (purple tetrahedra). This reduces the problem to an inner edge element construction problem, i.e. we can now apply the construction of Section 4.3.1 to e to create the tetrahedra in between $\mathbf{K}$ and the face element of $\mathbf{T}$ (red tetrahedra). The fan of red and purple tetrahedra then forms the set of edge elements of boundary edge e.

The main challenge here is to construct a well-shaped auxiliary triangle $\mathbf{S}$ that does not hinder termination of the overall method. We define $\mathbf{S}$ on the base plane of $\mathbf{T}$ and need to choose the inner angles θ_1, θ_2 of $\mathbf{S}$ at endpoints $\boldsymbol{p}_1, \boldsymbol{p}_2$ of e (as illustrated in Fig. 9 bottom) sufficiently small. Otherwise, the auxiliary triangle could (invariant to subdivision) intersect another triangle incident at $\boldsymbol{p}_i$ (Fig. 9 middle left), or the neighboring auxiliary triangle (Fig. 9 middle right). We therefore choose each angle θ_i to be smaller than half of ϕ_i (e.g. $\theta_i = \frac{1}{3}\phi_i$), the angle towards the closest intersection line of the base plane with the supporting plane of a triangle incident at $\boldsymbol{p}_i$ (or towards the closest edge incident at $\boldsymbol{p}_i$ contained in the base plane). We furthermore limit $\theta_i \leq \pi/3$.

We then raise the degree to the global order n, and adopt the control points of e for the first layer of control points along e to make $\mathbf{S}$ conform to e (Fig. 9 bottom). Note that $\mathbf{S}$ may be non-guardable. In this case and also in the case that the gaps cannot be filled by the inner edge element construction (red tetrahedra), the algorithm reports that e is uncoverable, requesting subdivision of $\mathbf{T}$. Algorithm 4 summarizes the method.

Algorithm 4: Boundary Edge Element Construction

```
Data: Boundary edge e
Result: Four to six regular edge elements, conforming to e and to
        the incident triangle's face elements, or 'uncoverable'
construct auxiliary triangle S on e
if S is orthogonally guardable then
    K, K' ← construct face elements for both sides for S
    K_1 ← call Algorithm 3 for one side of e
    K_2 ← call Algorithm 3 for other side of e
    if K_1 ≠ 'uncoverable' and K_2 ≠ 'uncoverable' then
        return {K, K'} ∪ K_1 ∪ K_2
    end
end
return 'uncoverable'
```

The following lemmas conclude the regularity of the inner and boundary edge elements, and show the guardability of their planar facets.

Lemma 5. *Edge elements constructed according to Algorithm 3 or Algorithm 4 are regular.*

Proof. The argument for regularity of the single edge element constructed by following Algorithm 3 is similar to that for face elements: Since the control points of the layers $k > 0$ are inside the guard region G (for any of the base triangles $\mathbf{T}$ or $\mathbf{T}'$), the right-hand formation holds for any convex combinations of the blue, red and green vectors. For the second case, where there are two coupled edge elements, $\boldsymbol{x}$ is a point on the boundary of $G_{\mathbf{T}} \cap G_{\mathbf{T}'}$ (since it is the highest intersecting point). Then, because L_{mid} can cross $G_{\mathbf{T}} \cap G_{\mathbf{T}'}$ at most once ($G_{\mathbf{T}}$ and $G_{\mathbf{T}'}$ individually partition L_{mid} into two parts), the tip-point $\boldsymbol{p}_{\text{tip}}$ lies inside $G_{\mathbf{T}} \cap G_{\mathbf{T}'}$. The rest of the argument is analogous to the regularity argument for the face elements. For boundary edge elements, regularity follows because the output consists of tetrahedra that are created with the face element construction method or the inner edge element construction method. □

Lemma 6. *The planar facets of the edge elements are guardable and the guard region G for these facets is the entire half-space defined by the base plane.*

Proof. Each of the planar facets of the edge elements consists of parallel blue vectors and a set of red vectors that are defined with the points of two consecutive layers (upward) as Fig. 10 illustrates. Thus, the conical hull C_0 of the blue vectors (which is a ray in this case) is separate from the conical hull C_1 of the red vectors and they are supported by a plane as shown in Fig. 10. Moreover, all of the blue and the red vectors of the base triangle are coplanar which implies that the inner and outer tangent planes are the same as the base plane at any control point $\boldsymbol{p}_{ij}$. Therefore, the guard region G is the half-space bounded by the base plane. □

4.4 Straight Element Construction

Having covered the curved input patches with face and edge elements, the domain yet to be meshed is now bounded by a piecewise planar (triangulated) surface. Constrained tetrahedralization of this domain using standard mesh generation methods, e.g. [George et al. 2003; Si 2015], gives a set of straight tetrahedra. Order n control nets then need to be assigned to them, to make them properly join their adjacent elements. The control points for each of the straight tetrahedra that is not (face or edge) adjacent to any of the edge elements are placed uniformly, such that the defined map is linear. The control point placement in a straight tetrahedron $\mathbf{H}$, however, can be *restricted* by adjacent edge elements on the shared edges or facets; the uniformly placed control points would lead to a non-C^0 situation on these edges or faces. In such elements the control points are therefore set differently.

Assume for a moment that $\mathbf{H}$ is restricted on one facet only. Consider this facet as the base of $\mathbf{H}$ (i.e. layer 0 of the control points). The control points imposed on this shared facet by the neighboring element form a guardable triangle and, due to planarity, the guard

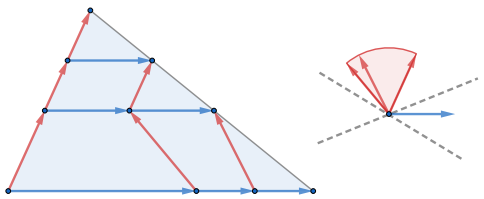

Fig. 10. Blue and red vectors of a planar facet of an edge element.

region G is the entire half-space formed by the base plane according to Lemma 6. Uniform layers $k > 0$ thus form a regular tetrahedron over this base layer: All of the green vectors are contained in the base half-space, which implies that the constructed element is regular. We can thus safely adopt the neighbor's control points on the shared facet, adjusting the otherwise uniform distribution, yielding a regular (non-linear but straight-sided) element.

Now assume that **H** is restricted on one edge only. We adopt the respective control points and complete the control points of layer 0 (on an adjacent facet) uniformly, again creating a guardable base (a special case of Fig. 10) for **H**. Analogously to the above case, regularity then is obtained by positioning all other layers uniformly.

In any other case (**H** restricted on more than one facet or on other edge configurations), we first split **H** into subtetrahedra (using a 1:4 or 2:6 split, introducing an additional vertex at the tetrahedron center or a facet center), reducing any configuration to those handled by the above control point placement rules. This is spelled out in Algorithm 5.

Algorithm 5: Straight Elements Construction

Data: Polyhedral domain **D**, bounded by planar guardable Bézier triangles $\{\mathbf{T}_i\}$
Result: Mesh of conforming straight-sided Bézier tetrahedra filling **D**, conforming to the Bézier triangles $\{\mathbf{T}_i\}$

```
tetrahedralize(D)
for each facet F between straight tetrahedra do
    if F is restricted on 2 edges then
        perform 2:6-split on F
    end
end
for each straight tetrahedron H do
    if H is restricted on >1 facet or >1 non-adjacent edges then
        perform 1:4-split on H
    end
end
for each straight tetrahedron H do
    set control points of H
end
```

Lemma 7. *The straight elements that are constructed with the above method are regular.*

4.5 Input Subdivision

If the input patches are such that they do not enable the construction of all face and edge elements right away, or imply overlapping elements, our method employs local subdivision. To preserve conformance, we do not subdivide patches in isolation, but take their adjacency in the input mesh into account. This can be done by performing *red-green refinement* [Bank and Weiser 1985]. All initial triangular patches start out as *red* patches. As Fig. 11 left illustrates, the subdivision of a red triangular patch **T** yields four red sub-triangles. Then, for every adjacent patch $\mathbf{C}_i$ of **T**, if $\mathbf{C}_i$ is red, we "hang" the corresponding new vertex (by a hanging edge shown in dashed green) to the opposite vertex in $\mathbf{C}_i$ and mark all these adjacent patches as *green*. If a green patch is to be subdivided

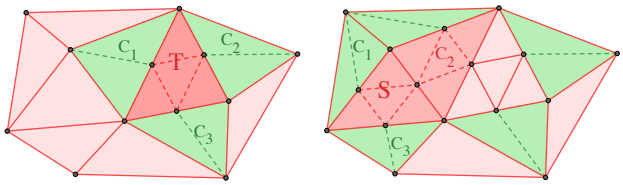

Fig. 11. Illustration of red-green refinement for subdivision of the input.

in a later step of our method (as an adjacent or as the main patch to be subdivided), the hanging edge is removed and a proper split into four sub-triangles is performed. Fig. 11 right shows the case for $\mathbf{C}_1$ when we subdivide the triangle **S**. The control points for the sub-triangles, such that they together define the same original surface, can be obtained by de Casteljau's algorithm [Prautzsch et al. 2013, §10.4].

As an alternative, *longest edge refinement* with recursive propagation [Rivara 2009] may be used. Both schemes guarantee that the interior angles of the elements are bounded from below (and consequently from above) under repeated subdivision. We require this property to guarantee termination; it is exploited in Lemma 8. The former scheme leads to less propagation of refinement (beyond the patch of interest), the latter scheme is easier to implement (avoiding temporary green subdivisions).

4.6 Disjointness Test

By our construction method, the input patches are covered by a set of curved face and edge elements. The boundary of the union of all these elements is piecewise planar since each of their inner (potentially curved) facets has an adjacent element that conforms to this facet by construction, and each outer facet (of edge elements) is planar by construction. This implies that this union is equal to the union of the of these elements' linear versions (straight tetrahedra defined by the four corner control points). A check for overlap between (face and edge) elements can therefore be carried out by checking for overlap of their linear versions.

To check disjointness, we therefore perform pairwise intersection tests between straight tetrahedra. Two tetrahedra are considered overlapping if their intersection is not empty, not one of their vertices, not one of their edges, and not one of their facets. For efficiency, of course spatial search data structures (e.g. an axis-aligned bounding box tree) can be employed, instead of naively performing the intersection test for all pairs.

4.7 Numerics

When all computations in an algorithm use basic arithmetic operators only, numbers stay in the rationals, enabling implementing the algorithm using an exact rational number type (such as `mpq` [Granlund et al. 2019]) for numerical reliability. This is the case for our method—there are two spots in the above routines, however, where this is not obvious:

In the face element and edge element constructions, unit normal vector $\boldsymbol{n}$ is made use of. Normalization requires non-rational square roots. However, we only need a unit normal in one expression, $h_{\boldsymbol{x}}\boldsymbol{n}$,

with height $h_{\boldsymbol{x}} = (\boldsymbol{x}-\boldsymbol{a})^\top \boldsymbol{n}$, where $\boldsymbol{a}$ is a point on the base triangle or edge. For a non-unit normal $\hat{n}$, we can rewrite $h_{\boldsymbol{x}}\boldsymbol{n} = \frac{1}{\hat{n}^\top\hat{n}}(\boldsymbol{x}-\boldsymbol{a})^\top \hat{n}\hat{n}$, i.e. a unit normal is not actually required to evaluate this expression. As the normal appears squared in this expression, only its (rational) squared length is needed.

Second, in the auxiliary triangle construction for boundary edge elements, we need to choose the triangle's third corner point such that angles $\theta_i < \frac{1}{2}\phi_i$. To achieve this without using trigonometric functions, we normalize the edge vector of edge e and the direction vector of the closest intersection line *in the 1-norm*. From the resulting vectors $\boldsymbol{d}_1, \boldsymbol{d}_2$ (which have a length difference of at most a factor of $\sqrt{3}$) we can then compute $2\boldsymbol{v}_1 + \boldsymbol{v}_2$, a vector closer to $\boldsymbol{d}_1$ than the bisector of $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$, defining a valid edge direction for the auxiliary triangle.

5 TERMINATION

The following lemmas support the proof of the overall algorithm's termination. Conceptually, we need to show that, after sufficient subdivision (using a regular subdivision scheme, such as red-green refinement or recursive longest edge refinement, cf. Section 4.5) of the input patches, eventually:

- all patches become (strongly) guardable,
- all patch edges become coverable,
- overlaps between face and edge elements vanish.

5.1 Guardable Triangles under Subdivision

Lemma 8. *Under repeated regular subdivision of a regular Bézier triangle* $\mathbf{T}$*, a sub-triangle's blue and red cones* C_0 *and* C_1 *converge to rays. The angle between these two rays is bounded by a positive constant from below.*

Proof. The sub-triangle's control points converge to the triangle surface under subdivision [Prautzsch et al. 2013, §11.4]. The rate of convergence is quadratic in the diameter of the subtriangle. Due to regularity of subdivision (i.e. boundedness of the sub-triangles' inner angles) the diameter goes to zero. Hence, due to regularity of the triangle, control vectors converge to be coplanar as the diameter goes to zero. More specifically, blue vectors Γ^0 converge to be parallel, and red vectors Γ^1 to be parallel [Li et al. 2012], hence the cones C_0 and C_1 converge to degenerate cones (i.e. rays). The angle between these rays is larger than some positive constant due to the lower bound on the sub-triangles' inner angles in the parameter domain in combination with the bounded angle distortion of the map $\mathbf{T}$ that is applied to it, which is due to its regularity. □

Lemma 9. *Under repeated regular subdivision of a regular Bézier triangle* $\mathbf{T}$*, a sub-triangle* $\mathbf{T}'$ *becomes strongly guardable and the guard region* G *converges to the half-space bounded by the base plane of* $\mathbf{T}'$*.*

Proof. According to Lemma 8, under repeated regular subdivision, cones C_0 and C_1 of sub-triangle $\mathbf{T}'$ converge to rays. These are tangent to the base plane of $\mathbf{T}'$, thus perpendicular to its normal. The cones therefore become supported and separated by two planes containing the normal, thus satisfying the guardability conditions (Definition 3) as well as, because this argument holds for any orientation of the control points, the strong guardability conditions (Definition 5). Furthermore, the inner and outer tangent planes spanned by C_0 and C_1 converge to the base plane as these cones converge to rays in the base plane. As the control points converge to the surface, thus to the base plane, also the (translated) extreme planes converge to the base plane. Therefore, the guard region G, defined as the intersection of these extreme planes, converges to the half-space bounded by the base plane. □

Remark: In contrast to the notion of guardability of curves and the manner of curve subdivision used in the 2D setting [Mandad and Campen 2020a], in our 3D setting guardability is not monotonous under subdivision: When a 1:2 split is applied (as is required in both subdivision schemes discussed above), the sub-triangles of a guardable triangle can be non-guardable; for a 1:4 split this is not the case. Nevertheless, after a sufficient number of subdivision steps, guardability is achieved and maintained in potential further steps.

5.2 Coverable Edges under Subdivision

Lemma 10. *Under repeated regular subdivision of incident regular non-degenerate Bézier triangles, inner edges as well as boundary edges become coverable.*

Proof. Triangles incident on an edge become strongly guardable by Lemma 9. A side facet of a face element constructed on these is guardable according to Lemma 4 and its blue and red vectors converge to be coplanar as its one curved edge converges to be straight. This implies that the guard region G of a side facet converges to the half-space bounded by the side facet's base plane. For an inner edge, the angle between two side facets sharing this edge converges to be less than 2π and positive by the non-degenerate assumption, hence the angle between the mid-line L_{mid} and the side facets converges to be less than π and positive. The latter implies that the side facets' guard regions $G_{\mathbf{T}}$ and $G_{\mathbf{T}'}$ will eventually partition L_{mid} into two parts, such that two edge elements covering the inner edge can be constructed. Regarding boundary edges: The edges of a sub-triangle converge to straight segments since the blue and the red vectors converge to be parallel (Lemma 8). Thus, the auxiliary triangle $\mathbf{S}$, constructed with bounded inner angles, becomes guardable. Then the (purple) elements conforming to these auxiliary triangles are constructible; in analogy to the arguments regarding face elements (Lemma 11) their guard regions G converge to the base plane and their base angles to zero. The remaining (red) elements become constructible by the above argument for inner edge elements. In conclusion, after sufficient regular subdivision, edge elements can be constructed for all edges, i.e. all edges become coverable. □

5.3 Vanishing Overlaps under Subdivision

For a straight-sided tetrahedron, let *base angles* be the three inner angles between a designated base facet and its side facets.

Lemma 11. *Under repeated regular subdivision of a regular Bézier triangle* $\mathbf{T}$*, a guardable sub-triangle's face element converges to a straight-sided tetrahedron with base angles zero.*

Proof. The edges of the sub-triangle converge to straight segments since the blue and the red vectors converge to be parallel (Lemma 8). Due to convergence of the boundary of guard region

G to the base plane (Lemma 9), the points $\boldsymbol{x}$ or $\boldsymbol{x}'$ used in the face element construction converge to the interior of the sub-triangle, in such a way that the base angles of the tetrahedron spanned by the sub-triangle corners and $\boldsymbol{x}$ or $\boldsymbol{x}'$ converge to zero. As the distance between the tip point $\boldsymbol{p}_{00n} = \boldsymbol{x}(\boldsymbol{x}') + \mu h_{\boldsymbol{x}} \boldsymbol{n}$ and $\boldsymbol{x}$ or $\boldsymbol{x}'$, respectively, converges to zero at the same rate, the base angles of the face element likewise converges to zero. □

Lemma 12. *Under repeated regular subdivision of triangles (satisfying the input assumptions of Section 3.2) whose face elements and adjacent edge elements cause overlaps, overlaps vanish.*

Proof. Under repeated regular subdivision of a regular triangle $\mathbf{T}$, a sub-triangle's face elements' base angles converge to zero (Lemma 11), thus each face element in its entirety converges to $\mathbf{T}$. Similarly, each edge element converges to the union of its edge and the adjacent faces, though this argument requires a case distinction, spelled out below. This then allows us to conclude that all elements (the face elements and the inner and boundary edge elements) converge locally to the input patches; as these are assumed to be non-intersecting (cf. Section 3.2), this eventually resolves all global overlaps.

The inner edge elements converge to the union of their two defining patches since the tip-points of their face elements converge to their patches (Lemma 11) and the mid-point p_{mid} converges to the shared edge. For red boundary edge elements, the same is true as they are constructed in the same manner. For purple boundary edge elements, note that the inner angles of each auxiliary triangle $\mathbf{S}$ are bounded from above and below by construction, by a constant independent of the subdivision level of its defining edge. Hence, the base angles of a purple boundary edge element, generated by the face element construction on an auxiliary triangle, converge to zero by an argument analogous to Lemma 11. As $\mathbf{S}$ converges to its defining edge under subdivision of the edge, so does the purple boundary edge element. Finally, note that the angles of auxiliary triangles are chosen such that they do not overlap with those of adjacent boundary edges, excluding also such local overlaps. □

Note that Lemma 10 and Lemma 12 assume that *all* involved triangles are repeatedly subdivided. For efficiency, Algorithm 1, when encountering a non-coverable edge or overlapping elements, does not aggressively subdivide all involved triangle patches right away, but more parsimoniously only the one with the tallest face element. As the height of face elements converges to zero under subdivision (Lemma 11), it is ensured that all other involved patches will eventually be subdivided as well in further iterations if necessary.

5.4 Termination & Success

Combining Lemmas 2 and 8 to 11 immediately leads to the following conclusion about Algorithm 1:

Theorem 1. *The proposed 3D Bézier Guarding algorithm terminates and yields non-overlapping elements.*

Finally, using Lemmas 2, 5 and 7 and recalling that conformance is by construction for all elements, the following can be concluded:

Theorem 2. *The proposed 3D Bézier Guarding algorithm generates a conforming higher-order mesh with regular tetrahedral elements.*

6 EVALUATION

Having formally shown the correctness of the algorithm, we wish to in addition empirically validate our implementation thereof and evaluate its behavior. To this end we apply it to various sets of input instances generated by means of randomization—so as to cover a wide range of configurations, including highly unrealistic ones, as a form of stress test—as well as to some example models. Note that there is no comparable prior method with regularity and conformance guarantee that would naturally lend itself to comparison. We therefore focus on demonstrating the general behavior of the proposed method in the following.

6.1 Datasets

For testing purposes we create and use the following datasets.

Random Triangle Soups:

- Set A: 1000 example input instances, each consisting of a single curved cubic triangle, created by randomly picking a template (Fig. 12) and perturbing each control point by a random displacement vector (within 10% of the triangle's bounding box size) and verifying that the resulting triangle is valid (regular and injective).
- Set B: 100 example input instances, each containing a random selection of 10 of the above 1000 randomly perturbed triangles, each one of which is randomly rotated, scaled, and positioned within a bounding box, while preventing mutual intersection.
- Set C: 10 example input instances, each containing a random selection of 100 of the template triangles, each one of which is randomly rotated, scaled, and positioned within a bounding box, while preventing mutual intersection.

These sets contain instances with isolated triangles only, i.e. there are no inner edges. Note, however, that due to being non-guardable or due to their implied face or edge elements overlapping, subdivision will occur, creating subtriangles and therefore inner edges. This means that all parts of the algorithm, including those responsible for inner edge elements, come into play in experiments on these datasets.

Fig. 12. Template Bézier patches used as basis for randomized stress test dataset generation.

Table 1. Result statistics for the various randomized datasets and the example models. Reported are the numbers of input triangles and their number after subdivision, the numbers of generated face, (inner and boundary) edge, and straight elements, the run time (in exact numerics mode) to perform the construction of all face and edge elements, to adaptively subdivide the patches, to test elements for intersection, and to generate the straight mesh part. As can be seen, the algorithm always succeeds in generating all the required curved elements and they are always regular. Only the final straight mesh generation part (delegated to TetGen in our implementation) can fail in extreme scenarios (e.g. 8 instances of Set B and 3 of Set C) due to numerical limits (last column).

	Triangles		Elements				Time (s)					Success		
	#T in	#T out	#FE	#IEE	#BEE	#SE	guard	subdiv	itest	TetGen	total	guard	regular	total
min	1	1	2	0	12	56	0.2	0.0	0.4	0.0	0.6			
Set A avg	1	116.5	232.9	329.8	79.9	1387.8	16.9	0.1	42.0	0.1	59.1	100%	100%	99.5%
max	1	4228	8456	12461	952	41980	609.4	2.6	1563.2	3.3	2178.5			
min	10	405	810	1046	600	10133	67.9	0.2	156.0	0.2	224.3			
Set B avg	10	1741.9	3483.8	4966.8	1056.5	32876.4	212.2	1.2	537.6	1.3	752.3	100%	100%	92%
max	10	6666	13332	19478	2153	121536	752.3	5.3	1968.2	8.6	2734.4			
min	100	10087	20174	28102	8725	290151	1284.4	6.8	3355.5	15.3	4662.0			
Set C avg	100	18139.2	36278.4	51758.2	10893.2	335060.7	2125.4	13.1	6122.8	28.7	8290.0	100%	100%	70%
max	100	27499	54998	79370	12982	358049	3096.4	18.5	9375.0	54.0	12543.9			
min	1000	1026	2052	3419	0	11476	30.4	0.1	289.1	0.3	319.9			
Set D avg	1000	1244.5	2489.0	4009.2	0	13424.0	89.0	0.2	398.9	0.6	488.7	100%	100%	100%
max	1000	1742	3484	5432	0	17939	318.9	1.1	632.7	4.8	957.5			
Roof	12	1812	1812	2723	0	8771	190.1	2.2	449.0	0.5	641.8	100%	100%	100%
RoundedCube	12	144	144	216	0	693	12.5	0.1	10.5	0.0	23.1	100%	100%	100%
EdgyDonut	16	3152	3152	4728	0	14898	350.7	3.8	1031.9	0.9	1387.3	100%	100%	100%
Retinal	1168	1330	1330	1995	0	6736	35.2	0.1	166.1	0.2	201.6	100%	100%	100%
Hand	1300	3132	3132	4704	0	15354	104.9	0.9	554.6	0.6	661.0	100%	100%	100%
NoisyCube	642	3740	3740	5649	0	18534	150.2	1.3	829.1	0.8	981.4	100%	100%	100%
Sculpture	816	2130	2130	3199	0	10260	76.7	0.7	415.5	0.3	493.2	100%	100%	100%
Gyroid	10166	11934	11934	17901	0	59829	265.2	1.4	6276.3	5.9	6548.8	100%	100%	100%
HalfTunnel	1568	3136	6272	4608	384	17331	82.7	0.5	385.3	0.6	469.1	100%	100%	100%
YasIsland	1475	1686	3372	4802	1026	20669	93.8	0.1	350.2	0.6	444.7	100%	100%	100%
Beetle	2118	6586	13172	19075	3047	73200	667.9	3.9	2015.5	8.2	2695.5	100%	100%	100%

Nevertheless, the subtriangle mesh created through implied subdivision is of a special kind (G^1 continuous). To also challenge the implementation with more general configurations, we create random test instances consisting of input meshes whose elements are only C^0, i.e. that contain creases between adjacent triangles:

Random Closed Meshes:

- Set D: 100 input instances, each consisting of an initially linear mesh (Fig. 14, 1000 triangles), degree-elevated to a random degree n within $[2, 5]$, and each control point perturbed by a random vector (within $20\%/n$ of the triangle's bbox size).

Furthermore, we make use of the following non-random input models (see Fig. 13) for testing and demonstration purposes:

Models:

- Roof (Fig. 1): 6 rectangular spline patches of bi-degree 3, converted to 12 Bézier triangles of degree 6.
- RoundedCube: 6 rectangular spline patches of bi-degree 2, converted to 12 Bézier triangles of degree 4.
- EdgyDonut: 8 rectangular spline patches of bi-degree 3, converted to 16 Bézier triangles of degree 6.
- Retinal: 1168 triangular patches of degree 3.
- Hand: 1300 triangular patches of degree 3.
- NoisyCube: 642 triangular patches of degree 3.
- Sculpture: 816 triangular patches of degree 3.
- Gyroid: 10166 triangular patches of degree 3.
- HalfTunnel: 1568 triangular patches of degree 3, boundary.
- YasIsland: 1475 triangular patches of degree 3, boundary.
- Beetle: 2118 triangular patches of degree 3, boundary.

Fig. 13. Some piecewise polynomial input surface models used for testing.

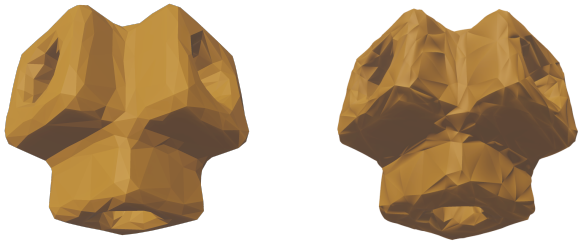

Fig. 14. Left: Linear template mesh used as basis for Set D. Right: One example perturbed higher-order version thereof.

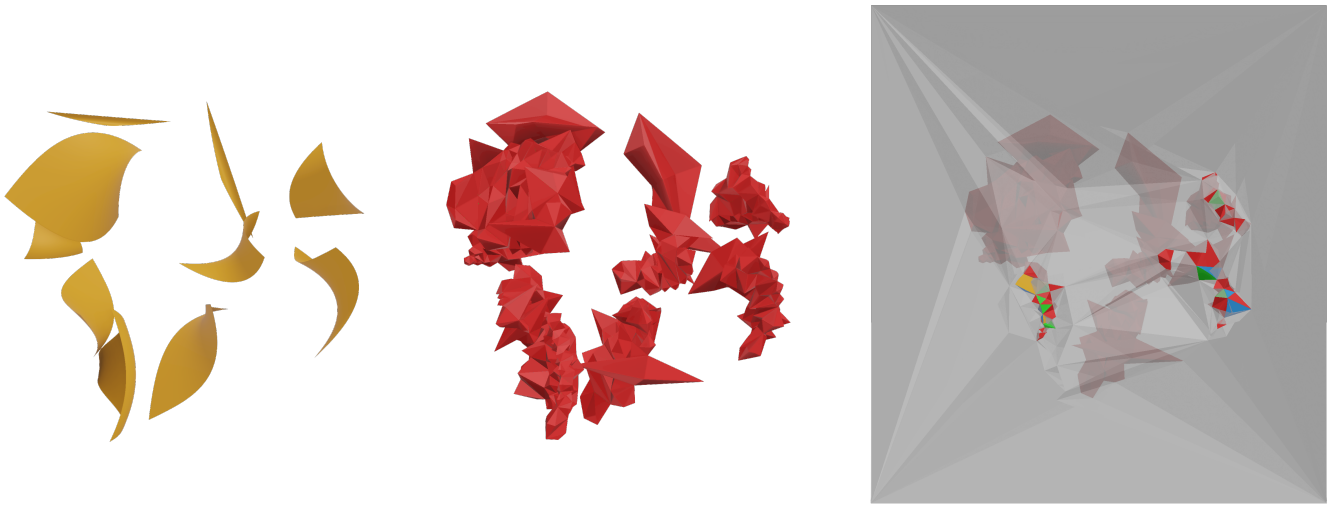

Fig. 15. Left: One example instance from Set B. Center: face and edge elements constructed for this. Notice that only edge elements (red) are visible, because they completely cover the (non-planar) sides of the face elements. Right: Sliced view of the complete result mesh (including semi-transparent straight elements in gray, filling a prescribed bounding box), revealing some face elements (blue and green).

6.2 Statistics

We apply our implementation of the proposed method to all instances of all datasets described in Section 6.1. Statistics (avg, min, max) per dataset are reported in Table 1. This includes the numbers of face elements, inner and boundary edge elements, and straight elements generated, as well as the time taken by the parts of the algorithm. It can be seen that our algorithm succeeded in generating all the required face and edge elements on all input instances, and their disjointness and regularity were verified (using the routine described in appendix A) in all cases.

Note that, in particular for some of the randomized stress test instances, there can be extreme configurations (e.g. very narrow passages) in the input. This can trigger many iterations of subdivision and thereby cause large numbers of small elements. Nevertheless, our algorithm is able to successfully construct all the required face and edge elements even for such (rather unrealistic) extreme cases. This is due to our implementation making use of exact rational arithmetic to avoid any numerical issues (cf. Section 4.7)—with one exception: We chose to delegate the straight mesh generation part to TetGen [Si 2015], which assumes standard double precision. In some of the geometrically intricate configurations a few elements get so small that TetGen is unable to handle them as boundary constraints. This is the (sole) reason for the $< 100\%$ total success rates that can be observed in the last column of Table 1 for some of the randomized datasets.

Fig. 15 shows an example result from Set B, Fig. 16 an example result from Set C, and Fig. 17 an example result from Set D. In Figures 18 and 19 the results for two of the non-random example models are shown. Fig. 20 illustrates the effect of parameter μ: A small choice leads to rather flat face elements, with a tendency of

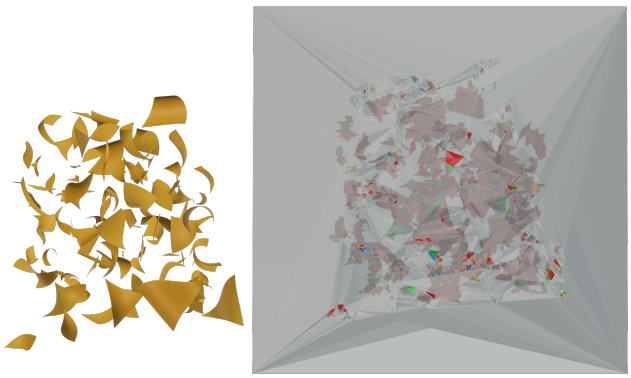

Fig. 16. One example result from the instances of Set C. On the right a sliced view of the complete result mesh is shown (including semi-transparent straight elements in gray, filling a prescribed bounding box).

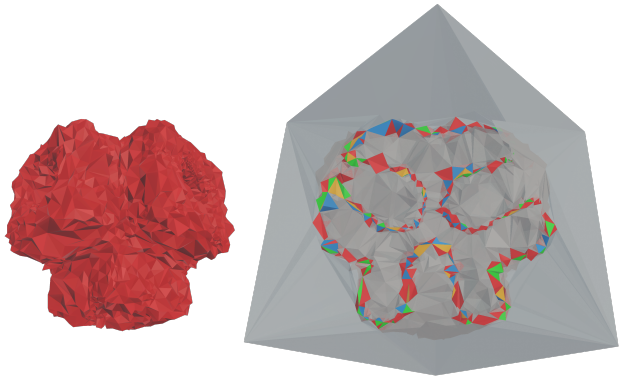

Fig. 17. One example result from the instances of Set D. On the right a sliced view of the complete result mesh is shown (including semi-transparent straight elements in gray, filling a prescribed bounding box).

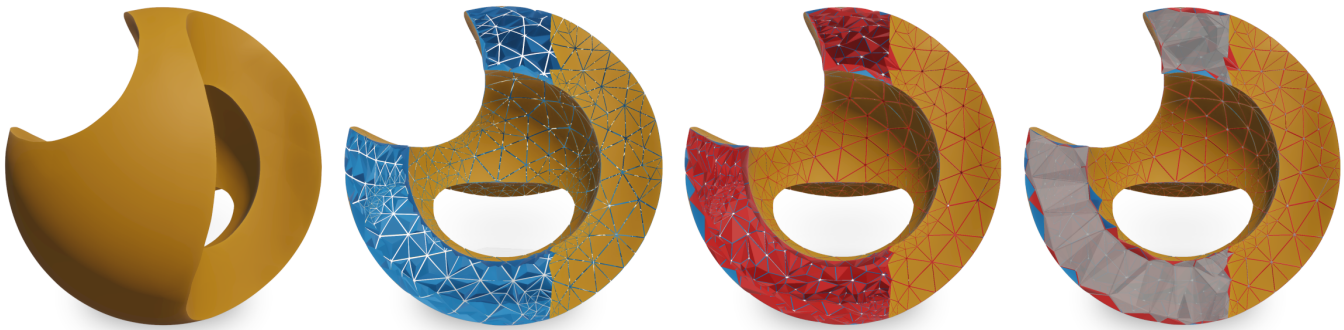

Fig. 18. Left: Input surface model Sculpture. Center: face and edge elements constructed for this. Right: Sliced view of the complete result mesh (including semi-transparent straight elements in gray).

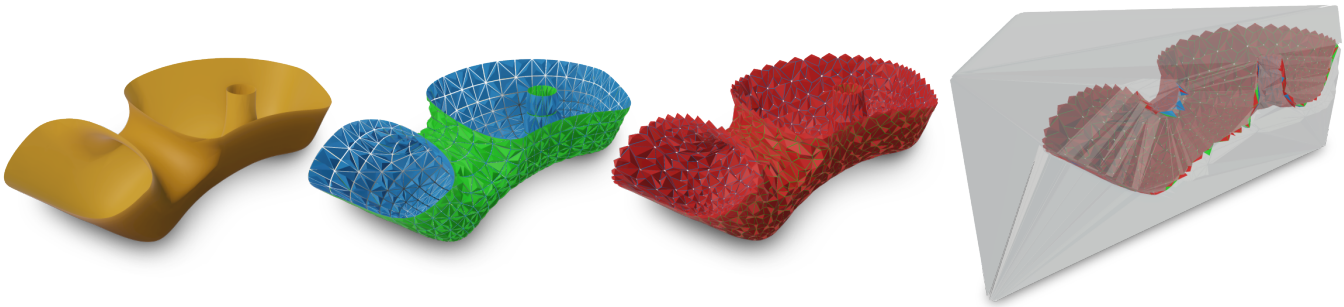

Fig. 19. Left: Input surface model HalfTunnel. Center: face and edge elements constructed for this. Right: Sliced view of the complete result mesh (including semi-transparent straight elements in gray).

triggering less subdivision due to less intersections, a larger choice to taller face elements of often nicer aspect ratio, with a tendency of leading to somewhat denser meshes.

Fig. 21 demonstrates the method's behavior on a non-manifold input configuration, formed by a prescribed curved interface inside a domain to which the mesh is asked to conform.

Runtime. Compared to modern linear tetrahedral meshing methods, the time taken to generate a conforming higher-order mesh with our method is certainly much higher (see Table 1). Note that other recent methods for the higher-order case [Feng et al. 2018; Jiang et al. 2021] likewise report run times up to the range of hours.

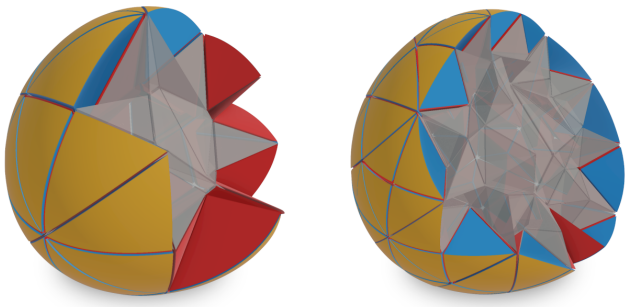

Fig. 20. Effect of parameter μ on RoundedCube. Left: $\mu = 0.1$. Right: $\mu = 1.0$.

Our research implementation performs intersection tests between tentative face and edge elements in a simplistic manner, does not exploit obvious parallelization options, and indiscriminately uses exact rational arithmetic [Granlund et al. 2019] throughout. Tests with using a standard double precision representation for all control points throughout the algorithm (while keeping all predicates and intermediate constructions exact) already showed a speed up by a factor of around 6×—while not causing irregularity for any of our dataset instances. We therefore believe there to be ample chances for improvement through further engineering efforts.

7 CONCLUSION

We have presented a method to generate tetrahedral meshes consisting of polynomial elements that are regular *and* conforming by construction. The polynomial order can freely be chosen, based on the application scenario. Key components are explicit construction

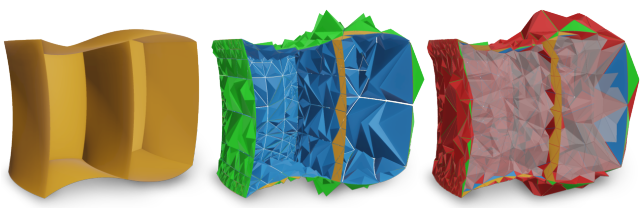

Fig. 21. Cut-away views of an example result for a non-manifold model.

rules for the various types of required mesh elements together with a systematic subdivision procedure that ensures the applicability of these constructions. It is a first step into the field of guaranteeing approaches for the 3D higher-order setting, but also points to various interesting questions and challenges for future work.

7.1 Limitations & Future Work

Element Quality. Our focus herein is entirely on mesh *validity*, in terms of regularity and conformance. Mesh *quality*, in terms of element distortion, can be arbitrarily low (see Table 2), and the resulting meshes are certainly not ideal candidates right away for, e.g., simulation purposes. One option to address this aspect is the application of mesh optimization in a post-process, of geometric kind (adjusting control points) or additionally of topological kind (incrementally modifying mesh connectivity). Our method effectively provides a feasible starting point for this, and the optimization can be constrained to preserve regularity and conformance. A few works have considered such higher-order tetrahedral mesh optimization already [Liu et al. 2021; Jiang et al. 2021; Dobrev et al. 2019] but this field deserves further attention. Another option could be the incorporation of lower bounds on quality directly into the element construction rules [Mandad and Campen 2021].

Parsimony. Related to this mesh quality aspect is the aspect of parsimony. Guardability is a sufficient but not a necessary condition for the existence of a regular conforming element. Hence, part of the performed subdivision typically is superfluous, leading to unnecessarily dense meshes in some regions. Finding tighter conditions is an interesting direction. Alternatively, a reduction of mesh complexity can also be addressed as part of the post-process mesh optimization discussed above, using, e.g., conformance-preserving face or edge collapses [Liu et al. 2021].

Numerics. When implementing the proposed method using standard floating point numbers with limited precision, there can of course be numerical inaccuracies that break the formal convergence guarantee. This is avoided when using, as discussed in Section 4.7, exact arithmetic, as we did in our implementation. Nevertheless, it is likely that downstream applications wish to operate on meshes represented using standard floating point numbers. Naive conversion (using truncation or rounding of control point coordinates) from rational to floating point numbers may lead to tiny degeneracies or inversions—though this is not the case for any of the 1220 instances from our datasets. In 16 instances, however, there is some *near*-degeneracy that the floating point based straight element generation using TetGen cannot handle (cf. last column of Table 1). Hence, while not a major problem, this numerical aspect deserves further attention in future work. Note that this is related to the above mesh quality improvement direction, because highly distorted near-degenerate elements constitute the highest risk in this regard.

Table 2. Quality statistics over the face and edge elements, averaged over all output meshes of each dataset. The scaled Jacobian and mean ratio shape quality measure [Gargallo-Peiró et al. 2015a] are reported; for both the ideal value is 1.0, while 0.0 would indicate degenerate elements. Let us remark that the choice of a higher parameter value μ would lead to higher quality; for $\mu = 3$ the scaled Jacobians increase by around an order of magnitude.

	Shape Quality			Scaled Jacobian		
	min	max	avg	min	max	avg
Set A	0.027	0.671	0.247	0.0052	0.114	0.037
Set B	0.0018	0.802	0.223	0.00039	0.177	0.038
Set C	0.000029	0.818	0.201	0.000016	0.187	0.038
Set D	0.0037	0.650	0.156	0.0069	0.281	0.076

Straight Meshing. For the generation of the straight-sided elements we employed TetGen [Si 2015], using its boundary preservation option to yield a mesh conforming to the other elements (i.e. without refinement along the boundary). While we did not encounter problems with this, Jiang et al. [2021] report their observation that using this option seems to reduce robustness. Either way, reliable conforming linear meshing with boundary preservation is conceptually possible [George et al. 2003]; the reliability of existing implementations of course is another question.

Trimmed Patches. Besides triangular/rectangular Bézier patches and B-spline surfaces, a further common variant for smooth surface representation are *trimmed* versions thereof. The trimmed parameter domains can be partitioned conformingly into higher-order triangles, using 2D Bézier Guarding [Mandad and Campen 2020a]. Composing these 2D Bézier triangles with the patch function yields a set of 3D Bézier triangles, as illustrated in Fig. 22, that together form the same surface as the trimmed patch (see [Lasser 2008, Theorem 1] for the rectangular case, [DeRose 1988, Theorem 4.1] for the triangular case). These could then be taken as input. The degree of these will be rather high, though (e.g. up to 18 in the case of cubic trimming curves in a bi-cubic patch). While this is inevitable if the resulting tetrahedral mesh is desired to conform to the 3D trimmed patch boundaries exactly, for many practical purposes approximate lower-order solutions [Du et al. 2021] are likely more reasonable.

Beyond Polynomials. Finally, an extension from polynomial elements to rational elements would be worthwhile from a practical perspective. In this way, beyond NUBS also NURBS (non-uniform *rational* B-splines) could be supported as input surfaces to be conformed to exactly. Inspiration may be taken from recent work that

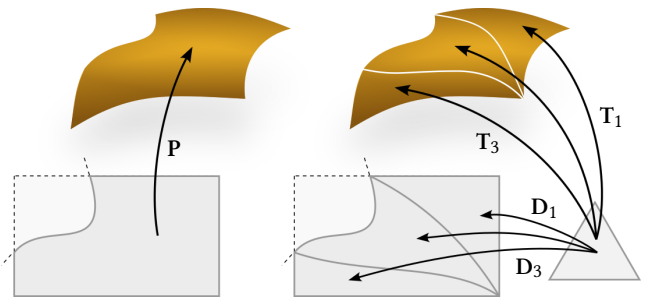

Fig. 22. Left: A trimmed tensor product Bézier patch $\mathbf{P} = \sum \mathbf{p} B^m B^n$ in $\mathbb{R}^3$ with its rectangular parameter domain trimmed by a polynomial curve of degree o. Right: Triangulation of the trimmed domain by 2D Bézier triangles $\mathbf{D}_i = \sum \mathbf{d} B^o$ gives rise to 3D Bézier triangles $\mathbf{T}_i = \mathbf{P} \circ \mathbf{D}_i = \sum \mathbf{t} B^{(m+n)o}$ that tile the trimmed patch, with control points $\mathbf{t}$ depending on $\mathbf{d}$ and $\mathbf{p}$.

addresses curved 2D triangle mesh generation conforming to rational curves [Khanteimouri et al. 2022; Yang et al. 2022].

ACKNOWLEDGMENTS

The authors thank Steffen Hinderink for his support in dataset preparation. This work was funded by the Deutsche Forschungsgemeinschaft (DFG) - 451286978; 456666331.

REFERENCES

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

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

Randolph E Bank and Alan Weiser. 1985. Some a posteriori error estimators for elliptic partial differential equations. *Mathematics of computation* 44, 170 (1985), 283–301.

C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* 22, 4 (1996), 469–483.

Adam W Bargteil and Elaine Cohen. 2014. Animation of deformable bodies with quadratic bézier finite elements. *ACM Trans. Graph.* 33, 3 (2014).

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

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

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

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

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

Tony D DeRose. 1988. Composing Bézier Simplexes. *ACM Trans. Graph.* 7, 3 (1988), 198–221.

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

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

Veselin Dobrev, Patrick Knupp, Tzanio Kolev, Ketan Mittal, and Vladimir Tomov. 2019. The Target-Matrix Optimization Paradigm for High-Order Meshes. *SIAM Journal on Scientific Computing* 41, 1 (2019).

Xiaoxiao Du, Gang Zhao, Wei Wang, Mayi Guo, Ran Zhang, and Jiaming Yang. 2021. Triangular and Quadrilateral Bézier Discretizations of Trimmed CAD Surfaces and Its Application to the Isogeometric Analysis. *Computer-Aided Design & Application* 18, 4 (2021), 738–759.

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

José Marıa Escobar, Eduardo Rodrıguez, Rafael Montenegro, Gustavo Montero, and José Marıa González-Yuste. 2003. Simultaneous untangling and smoothing of tetrahedral meshes. *Comput. Methods Appl. Mech. Eng.* 192, 25 (2003), 2775–2787.

Leman Feng, Pierre Alliez, Laurent Busé, Hervé Delingette, and Mathieu Desbrun. 2018. Curved Optimal Delaunay Triangulation. *ACM Trans. Graph.* 37, 4 (2018).

Zachary Ferguson, Pranav Jain, Denis Zorin, Teseo Schneider, and Daniele Panozzo. 2023. High-Order Incremental Potential Contact for Elastodynamic Simulation on Curved Meshes. In *ACM SIGGRAPH 2023 Conference Proceedings*. Article 77.

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

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

Abel Gargallo-Peiró, Xevi Roca, Jaime Peraire, and Josep Sarrate. 2015a. Distortion and quality measures for validating and generating high-order tetrahedral meshes. *Engineering with Computers* 31, 3 (2015), 423–437.

Abel Gargallo-Peiró, Xevi Roca, Jaime Peraire, and Josep Sarrate. 2015b. Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *Int. J. Numer. Methods Eng.* 103, 5 (2015), 342–363.

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

Paul-Louis George, Houman Borouchaki, and Eric Saltel. 2003. 'Ultimate'robustness in meshing an arbitrary polyhedron. *Internat. J. Numer. Methods Engrg.* 58, 7 (2003), 1061–1089.

Ronald N Goldman and Daniel J Filip. 1987. Conversion from Bézier rectangles to Bézier triangles. *Computer-Aided Design* 19, 1 (1987), 25–27.

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

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

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

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

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

Zhongshi Jiang, Ziyi Zhang, Yixin Hu, Teseo Schneider, Denis Zorin, and Daniele Panozzo. 2021. Bijective and Coarse High-Order Tetrahedral Meshes. *ACM Trans. Graph.* 40, 4 (2021).

Amaury Johnen, J-F Remacle, and Christophe Geuzaine. 2013. Geometrical validity of curvilinear finite elements. *J. Comput. Phys.* 233 (2013), 359–372.

P Khanteimouri, M Mandad, and M Campen. 2022. Rational Bézier Guarding. *Computer Graphics Forum* 41, 5 (2022), 89–99.

David Kirkpatrick and Jack Snoeyink. 1995. Computing common tangents without a separating line. In *Workshop on Algorithms and Data Structures*. Springer, 183–193.

Dieter Lasser. 2008. Triangular subpatches of rectangular Bézier surfaces. *Computers & Mathematics with Applications* 55, 8 (2008), 1706–1719.

J. Li, T. J. Peters, and J. A. Roulier. 2012. Angular Convergence during Bézier Curve Approximation. *arXiv:1210.2686* (2012).

Zhong-Yuan Liu, Jian-Ping Su, Hao Liu, Chunyang Ye, Ligang Liu, and Xiao-Ming Fu. 2021. Error-bounded Edge-based Remeshing of High-order Tetrahedral Meshes. *Computer-Aided Design* 139 (2021), 103080.

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

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

Manish Mandad and Marcel Campen. 2020a. Bézier Guarding: Precise Higher-Order Meshing of Curved 2D Domains. *ACM Trans. Graph.* 39, 4, Article 103 (2020).

Manish Mandad and Marcel Campen. 2020b. Efficient piecewise higher-order parametrization of discrete surfaces with local and global injectivity. *Computer-Aided Design* 127 (2020), 102862.

Manish Mandad and Marcel Campen. 2021. Guaranteed-quality higher-order triangular meshing of 2D domains. *ACM Trans. Graph.* 40, 4 (2021).

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

Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. 2008. Interactive physically-based shape editing. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*. 79–89.

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

Fariba Mohammadi and Suzanne M Shontz. 2021. A direct method for generating quadratic curvilinear tetrahedral meshes using an advancing front approach. *Proc. of the 29th International Meshing Roundtable* (2021).

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

Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2015. Air meshes for robust collision handling. *ACM Trans. Graph.* 34, 4 (2015), 133.

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

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

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

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

Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. 2013. *Bézier and B-spline techniques.* Springer Science & Business Media.

Franco P. Preparata and Se June Hong. 1977. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM* 20, 2 (1977), 87–93.

Ramsharan Rangarajan and Adrián J. Lew. 2014. Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes. *Int. J. Numer. Methods Eng.* 98, 4 (2014), 236–264.
Maria-Cecilia Rivara. 2009. Lepp-bisection algorithms, applications and mathematical properties. *Applied Numerical Mathematics* 59, 9 (2009), 2218–2235.
Xevi Roca, Abel Gargallo-Peiró, and Josep Sarrate. 2011. Defining quality measures for high-order planar triangles and curved mesh generation. In *Proc. Int. Meshing Roundtable*. 365–383.
Eloi Ruiz-Gironés, Abel Gargallo-Peiró, Josep Sarrate, and Xevi Roca. 2017. An augmented Lagrangian formulation to impose boundary conditions for distortion based mesh moving and curving. *Procedia Engineering* 203 (2017), 362–374.
Eloi Ruiz-Gironés, Josep Sarrate, and Xevi Roca. 2016. Generation of Curved High-order Meshes with Optimal Quality and Geometric Accuracy. *Procedia Engineering* 163 (2016), 315 – 327.
Ruben Sevilla, Luke Rees, and Oubay Hassan. 2016. The generation of triangular meshes for NURBS-enhanced FEM. *Int. J. Num. Methods Engrg.* 108, 8 (2016), 941–968.
Mark S. Shephard, Joseph E. Flaherty, Kenneth E. Jansen, Xiangrong Li, Xiaojuan Luo, Nicolas Chevaugeon, Jean-François Remacle, Mark W. Beall, and Robert M. O'Bara. 2005. Adaptive mesh generation for curved domains. *Appl. Numer. Math.* 52, 2 (2005), 251–271.
Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2 (2015).
Stefan Suwelack, Dimitar Lukarski, Vincent Heuveline, Rüdiger Dillmann, and Stefanie Speidel. 2013. Accurate surface embedding for higher order finite elements. In *Proc. 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 187–192.
Thomas Toulorge, Christophe Geuzaine, Jean-François Remacle, and Jonathan Lambrechts. 2013. Robust untangling of curvilinear meshes. *J. Comput. Phys.* 254 (2013), 8 – 26.
Thomas Toulorge, Jonathan Lambrechts, and Jean-François Remacle. 2016. Optimizing the geometrical accuracy of curvilinear meshes. *J. Comput. Phys.* 310 (2016), 361–380.
Godfried T Toussaint. 1983. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, Vol. 83. A10.
Michael Turner, Joaquim Peiró, and David Moxey. 2018. Curvilinear mesh generation using a variational framework. *Computer-Aided Design* 103 (2018), 73–91.
Stephen Vavasis. 2003. A Bernstein-Bézier Sufficient Condition for Invertibility of Polynomial Mapping Functions. arXiv:cs/0308021 [cs.NA]
Z.J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H.T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. 2013. High-order CFD methods: current status and perspective. *Int. J. Numerical Methods in Fluids* 72, 8 (2013), 811–845.
Zhong Q. Xie, Ruben Sevilla, Oubay Hassan, and Kenneth Morgan. 2013. The generation of arbitrary order curved meshes for 3D finite element analysis. *Computational Mechanics* 51, 3 (2013), 361–374.
Jing Xu and Andrey N. Chernikov. 2014. Automatic Curvilinear Quality Mesh Generation Driven by Smooth Boundary and Guaranteed Fidelity. *Procedia Engineering* 82 (2014), 200 – 212.
Jinlin Yang, Shibo Liu, Shuangming Chai, Ligang Liu, and Xiao-Ming Fu. 2022. Precise High-order Meshing of 2D Domains with Rational Bézier Curves. *Computer Graphics Forum* 41, 5 (2022).
Milos Zlamal. 1973. The finite element method in domains with curved boundaries. *Int. J. Numer. Methods Eng.* 5, 3 (1973), 367–373.

A REGULARITY TEST

To verify correctness, we test all generated elements for conformance and regularity. While conformance testing is a simple matter of comparing control points, certifying regularity is more intricate. The Jacobian determinant of a degree n Bézier tetrahedron $\boldsymbol{H}$, see (2), is a polynomial, of degree $\hat{n} = 3(n-1)$, thus can be expressed in the Bernstein basis (i.e. as a Bézier tetrahedron itself) [Johnen et al. 2013]:

$$\det(J_{\mathbf{H}}) = \sum_{i+j+k\leq\hat{n}} d_{ijk} B^{\hat{n}}_{ijk}(u,v,w).$$

In generalization of the analogous expression for the triangle case [Mandad and Campen 2020b], the coefficients d_{ijk} can be calculated as follows:

$$d_{ijk} = c_n \sum_{\substack{|r|=|s|=|t| \\ r+s+t=(i,j,k,l)}} \frac{i!\,j!\,k!\,l!}{r!\,s!\,t!} \begin{vmatrix} \Delta^0 x_r & \Delta^1 x_r & \Delta^2 x_r \\ \Delta^0 y_s & \Delta^1 y_s & \Delta^2 y_s \\ \Delta^0 z_t & \Delta^1 z_t & \Delta^2 z_t \end{vmatrix},$$

where:

- c_n is a (here irrelevant) positive constant depending on n,
- r, s and t are quadruple index vectors; i.e. $r = (r_1, r_2, r_3, r_4)$,
- $|r| = r_1 + r_2 + r_3 + r_4$ and $r! = r_1!r_2!r_3!r_4!$,
- $l = \hat{n} - i - j - k$,
- $\Delta^0 x_r = x(\Delta^0_{r_1 r_2 r_3})$ where $x(\boldsymbol{v})$ denotes the x-coordinate of vector $\boldsymbol{v}$. Analogously for y_s, z_t, Δ^1 and Δ^2.

Due to the non-negativity of the Bernstein basis, we have that $\min \det J_{\mathbf{H}} \geq \min(d_{ijk})$, providing us with a lower bound. Due to the interpolation of the extremal (corner) coefficients, we have that $\min \det J_{\mathbf{H}} \leq \min(d_{000}, d_{00\hat{n}}, d_{0\hat{n}0}, d_{\hat{n}00})$, providing us with an upper bound [Johnen et al. 2013]. If the *lower* bound is positive, we can conclude that $\boldsymbol{H}$ is regular. Conversely, if the *upper* bound is non-positive, we can conclude that $\boldsymbol{H}$ is not regular. If neither is the case, the Bézier tetrahedron $\det(J_{\mathbf{H}})$ can be (virtually) subdivided using de Casteljau's algorithm [Prautzsch et al. 2013, §10.4], so as to tighten the bounds per sub-tetrahedron [Johnen et al. 2013]. This can continue recursively until at least one sub-tetrahedron certifies non-regularity, until all sub-tetrahedra certify regularity, or (for practical purposes) until a maximum subdivision limit is reached, in which case the regularity check is inconclusive (and reports non-regularity to err on the conservative side).