DOI: 10.1111/cgf.14362
Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

Volume 40 (2021), Number 5

Delaunay Meshing and Repairing of NURBS Models

X. Xiao[1], P. Alliez[1,2], L. Busé[1,2] and L. Rineau[3]

[1]Inria Sophia Antipolis-Méditerranée, France
[2]Université Côte d'Azur, France
[3]GeometryFactory, France

Abstract

CAD models represented by NURBS surface patches are often hampered with defects due to inaccurate representations of trimming curves. Such defects make these models unsuitable to the direct generation of valid volume meshes, and often require trial-and-error processes to fix them. We propose a fully automated Delaunay-based meshing approach which can mesh and repair simultaneously, while being independent of the input NURBS patch layout. Our approach proceeds by Delaunay filtering and refinement, in which trimmed areas are repaired through implicit surfaces. Beyond repair, we demonstrate its capability to smooth out sharp features, defeature small details, and mesh multiple domains in contact.

CCS Concepts

• ***Mathematics of computing*** $\rightarrow$ ***Mesh generation;***

1. Introduction

CAD models represented by NURBS surface patches often exhibit defects due to inaccurate representations of trimming curves, imperfect modeling processes or format migration. Such defects range from gaps to self-intersections through overlaps and islands. While small-size defects are troubleless for visualization or manufacturing purposes, they hamper the direct generation of volume meshes that are required for simulation. A common approach consists of (1) fixing the CAD model, (2) generating a surface mesh and (3) generating the final volume mesh. This approach may become labor-intensive due to the trial-and-error nature of the fixing process and the iterations required between steps (1) and (2) until the volume mesh generator takes as input a valid surface mesh (both watertight and intersection-free). Streamlining modeling-and-simulation processes requires fully automated approaches capable of repairing while meshing.

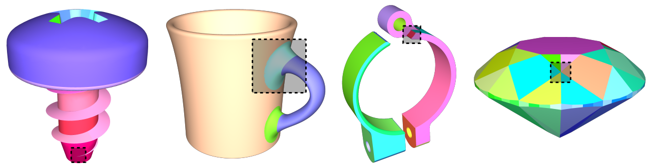
Beyond repair-and-meshing, computational engineering practitioners require additional capabilities closer to modeling: e.g., smoothing sharp features, defeaturing or meshing multiple domains in contact.

1.1. Related work

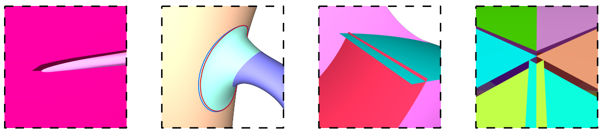
We now review the related work focused on repair and meshing of NURBS models, and discuss the requirements of Delaunay meshing approaches.

Repair. Consistent and accurate CAD models have been a quest for decades [Pie05]. Defects in NURBS models originate from a combination of inaccurate modeling processes, numerical rounding and inaccuracies, migrations between models and data formats and imperfect automated NURBS generation algorithms. Common means to repair proceed through patching or reconstruction via design history or knowledge-guided NURBS [YH06, RPS12]. Commercial solutions such as CADfix or CADdoctor have been proposed to check, repair and translate multi-CAD models, where processes are as automated as possible. Attene et al. [Att10, ACK13] attempt to heal defects locally and generate a watertight triangle mesh. Bischoff and Kobbelt [BK05] combine the local surface repair with volumetric reconstruction following the detection of artifacts in a CAD tessellation, such that a manifold output is always guaranteed. Software tools (e.g., MeshFix, MeshWorks, PolyMender) have been proposed for repairing meshes and polygon soups, with strict robustness requirements for 3D printing [LEM*17]. The quest for valid volume meshing with unconditional robustness culminates with the recent significant advances for tetrahedral meshing in the wild [HZG*18] and its recent efficient variant [HSW*20]. The robustness of the approach is confirmed, though the mesh quality is not unconditionally consistent when large defects are present in the model.

Meshing NURBS models. Mesh generation methods for NURBS models differ depending on whether they proceed in parameter space [BLG00, GBA*17] or directly in 3D. The dependence on the input NURBS patch layout, particularly when meshing each patch in parameter space, can be removed via a process referred to as mesh quilting, which stitches at the mesh level and possibly offers additional virtues for repairing and defeaturing. Virtual topology has been adopted in commercial software tools

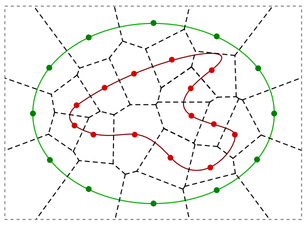


(a) *NURBS models with defects in shaded regions*

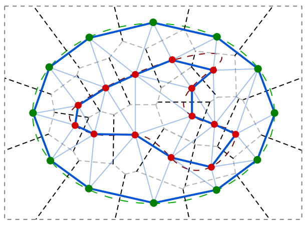


(b) *Close-up views of defects*

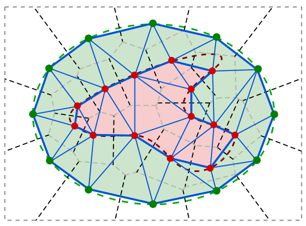
Figure 1: *Four common types of defects in NURBS models. From left to right: gaps, overlaps, intersections and mismatched end points.*



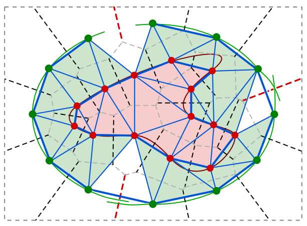
(a) *Sampling and voronoi diagram*



(b) *Boundary triangulation to $\partial\Omega$*



(c) *Domain triangulation to Ω*



(d) *Domain triangulation with boundary defects*

Figure 2: *Delaunay triangulation restricted to 2D multidomain Ω and its boundary $\partial\Omega$. Delaunay triangulation can fail with wrong intersections between Voronoi dual edges and the boundary due to boundary defects (problematic Voronoi edges are depicted in red in (d)).*

(e.g. Ansys Workbench, Abaqus CAE, Siemens NX) to clean up the geometry with virtual topological merging or splitting operations [FCF*08, TSRA17], which provides an alternative approach to removing dependency on input NURBS layout. Most volume mesh generation approaches generate the volume mesh once a valid surface mesh is available [GR09], while the Delaunay filtering and refinement approach interleaves surface and volume meshing in 3D, with full independence to the input patch layout [CDS12]. The Delaunay-based meshing approach thus removes the need for a valid surface mesh as input but requires *consistent* answers to intersection queries either between line queries and the input NURBS surfaces, or between point queries and the 3D domain bounded by the input NURBS surfaces. The resulting 3D mesh is always valid "by design" as results from the filtering of a 3D Delaunay triangulation, even for multiple domains in contact. Nevertheless, Delaunay refinement further requires proper initialization and conservative sizing of mesh elements with respect to the local feature size. The latter measures the distance between surface points and the medial axis of the domain [AB99]. For Delaunay refinement with weighted Voronoi diagrams, surface boundary curves are sampled with protecting balls which are treated as weighted points during Delaunay refinement. The local feature size is then extended to blend with the local gap size of polyhedral domains to determine the ball sizes [CDR07]. A challenge for Delaunay meshing of NURBS models is trimming defects due to the inconsistent representations of trimming curves in CAD models. Figure 1 depicts four common types of defects in a trimmed NURBS model. The presence of trimming defects can derail Delaunay filtering and refinement processes, as illustrated in Figure 2. The protecting-ball approach is further applied to repair defects in NURBS models. The first contribution to such a Delaunay-based meshing for NURBS model has been pioneered by Busaryev et al. [BDL09], by considering all surface boundaries as sharp features and by relying upon a merging process to determine the protecting balls. This approach, however, generates a Delaunay mesh confined by the input NURBS layout, and sharp features are preserved without taking into account their local geometric features.

1.2. Contributions

We seek a robust approach which offers the capability to repair while meshing, without requiring a valid surface mesh as input, and to generate volume meshes which are (1) valid by design and (2) independent of the input NURBS patch layout, without resorting to post-processing steps such as mesh quilting or remeshing. The mesh quality is guaranteed by the Delaunay refinement methodology we adopt, which provides relevant user-defined facet and cell criteria to control the meshing process. Departing from the protecting ball approach which preserves all surface boundaries during Delaunay meshing [BDL09], our approach treats sharp features and smooth boundary connections differently such that the Delaunay mesh elements in smooth regions are fully independent of the NURBS layout. We also provide a thorough analysis of ball sampling and sizing in order to control the element sizes of the final mesh, especially near sharp features. We now summarize the technical contributions which form the framework of our approach for repairing and meshing NURBS models with trimming defects:

- A novel ball generation process is proposed to cover all trimmed regions, distinguishing smooth regions from sharp features. Ball sizes are adaptive to geometric features and defect sizes, and the user-defined criteria are used as prior knowledge when determining ball sizes protecting sharp features (Section 4).
- Given a NURBS model as input, we analyze it in order to identify three types of areas: sharp features to preserve, trimmed areas to repair near trimming curves and defect-free areas away from sharp features and trimming curves. These areas are dealt with differently in order to provide consistent intersection

queries during Delaunay refinement. In particular, the defects are healed via implicit surfaces interpolating the defect-free areas (Section 5).

- We demonstrate the versatility of our framework not just for meshing, but also for smoothing out sharp features, removing small features, and meshing multiple domains. Such operations are especially relevant for computational engineering and simulation practice (Section 6).

2. Background on NURBS models and Delaunay meshing

In this section, we briefly review the design of CAD models with trimmed NURBS surfaces and the challenges of the inherent trimming defects to Delaunay meshing.

2.1. Trimmed NURBS models

In computer-aided design, a solid is often defined via a boundary representation, commonly referred to as B-Rep, i.e. a collection of connected surface elements which define the boundary between the interior and exterior of the solid. NURBS surfaces are commonly used to represent these surface elements for the mathematical soundness and flexibility for free-form modeling.

In a B-Rep CAD model, NURBS surfaces are trimmed in the parameter space to represent arbitrary surface boundaries which are intersection curves between surfaces. In general, the intersection computation between two NURBS surfaces leads to solving a system of nonlinear equations, and the 3D intersection curve is usually not a NURBS curve but a piecewise algebraic curve of a high degree. As a consequence, 3D intersection curves between NURBS surfaces are usually approximated via several methods, e.g. lattice evaluation, recursive subdivision or marching methods [HLQ07]. A subsequent curve fitting scheme is often performed to generate a 3D intersection NURBS curve C which does not lie on either of the two NURBS surfaces in general, except in low-degree surface cases. This 3D intersection curve is also represented in each parameter space of the two intersecting NURBS surfaces as two low-degree B-spline (or NURBS) 2D trimming curves C^{trim}, via curve fitting of points sampled in the parameter space [MH18]. Theses 2D trimming curves can be embedded in 3D by evaluating surfaces along C^{trim}. Consequently, the intersection curve between two NURBS surfaces has three distinct representations in the CAD model, i.e. a 3D curve and two 2D trimming curves. The fact that the images of these three curves do not coincide with each other leads to a variety of trimming defects, as already shown in Figure 1.

2.2. Delaunay meshing

Isotropic Delaunay meshing of an input 3D domain Ω hinges upon the concept of filtering and refining a 3D weighted Delaunay triangulation, whose vertices are located either inside Ω or on the domain boundary $\partial\Omega$. The Delaunay triangulation and the Voronoi diagram of a 3D point set are dual to each other. Specifically, a Voronoi edge is the dual of a tetrahedron facet, and Voronoi vertices are located at tetrahedron circumsphere centers.

For an input 3D domain Ω, we retain the tetrahedra whose circumcenters are located inside Ω, and the boundary facets have dual Voronoi edges intersecting $\partial\Omega$. Filtering thus requires computing intersections either between 3D points and Ω, or between Voronoi edges and $\partial\Omega$. In addition, a restricted Delaunay triangulation derived from an initial coarse sampling of $\partial\Omega$ must be refined by inserting new vertices, referred to as Steiner points, to satisfy the mesh criteria. The Steiner points are inserted at circumcenters of poorly shaped facets and cells, which are computed as the intersections between Voronoi edges and $\partial\Omega$.

The inevitable trimming defects in NURBS models can jeopardize the Delaunay meshing which relies on reliable and consistent intersection computation with the 3D domain. Due to the existence of defects, the intersection can be either a false negative (i.e., an intersection is missed due to gaps) or dubious points due to surface overlaps and intersections. Wrong intersections hamper the Delaunay filtering to identify valid cells and surface facets and to insert Steiner points during Delaunay refinement. As a result, the defects in a NURBS model may terminate the Delaunay refinement process early or even break the termination guarantees.

3. Overview

3.1. Overall procedure

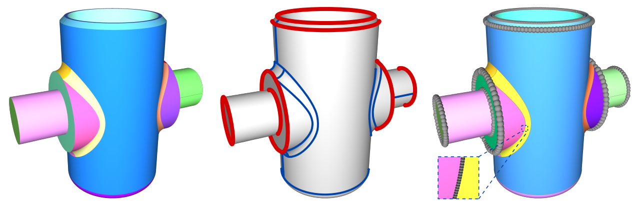
The following pseudo-code Algorithm 1 outlines the proposed algorithm for meshing NURBS models with trimming defects. Each step is discussed in full detail in the following sections.

Algorithm 1 Isotropic meshing of NURBS models with trimming defects.

```
Input: NURBS model, mesh quality parameters
Output: Isotropic tetrahedral mesh
1: Ball covering of trimmed regions
2: Implicit surface approximation inside blending balls
3: Delaunay refinement in three types of subdomains
```

NURBS models require preprocessing before Delaunay meshing, in which the trimmed regions (including the boundaries of surface patches) are singled out through ball covering for separate treatment. Sharp features (creases, corners, cusps) are covered with so-called *protecting* balls as in weighted Delaunay meshing approaches [DL09, CDS12], while other boundary curves are covered with *blending* balls inside which NURBS surfaces are substituted by approximate implicit surfaces. The connection between two adjacent NURBS surfaces is considered smooth when the surface normal directions along the common boundaries deviate by a small angle, where *small* refers to a user-defined angle threshold. As depicted in Figure 3, a NURBS model is divided into three subdomains for Delaunay meshing: vicinity of sharp creases, vicinity of smooth boundary curves and NURBS surfaces away from boundary curves. The trimmed regions processed by ball covering, i.e. the vicinities of boundary curves, are distinguished from defect-free areas.

The Delaunay refinement process differs depending on the three types of subdomains partitioned. In the defect-free subdomain, intersections between Voronoi dual edges and trimmed NURBS surfaces are computed efficiently via a hierarchical bounding volume



(a) *A NURBS model* **(b)** *Subdomains* **(c)** *Balls covering boundary curves*

Figure 3: *A NURBS model is divided into three types of subdomains: sharp creases and corners (red), smooth boundary curves (blue) and NURBS surfaces away from boundary curves (gray). The intersection computation differs among these three types of subdomains.*

Table 1: *User-defined parameters.*

Ball covering	maximum ball size $r_{\max}$ minimum ball size $r_{\min}$ feature angle θ_f
Facet criteria	uniform triangle circumradius F_a smallest triangle angle F_s surface approximation F_δ
Cell criteria	uniform tetrahedron circumradius C_a Tetrahedral radius/edge ratio C_s

hierarchy data structure storing Bézier surface patches, and efficient line-Bézier intersections. The subdomain covered with protecting balls is never queried for intersection during Delaunay refinement. The sizes of protecting balls depend on local feature sizes and user-defined discretization parameters, which later determine the discretization of sharp creases in the final mesh. Inside blending balls, NURBS surfaces are approximated by implicit surfaces defined through points sampled on the NURBS surfaces, such that the defects are *repaired* during Delaunay mesh refinement.

3.2. User parameters

The output mesh quality is controlled by user-defined parameters impacting both the ball covering and the Delaunay refinement steps. Table 1 records the main user-defined parameters required by our meshing algorithm.

In the ball covering process, a feature angle θ_f between the two surface normals is prescribed to classify a sharp or smooth surface connection (set by default to 10°). The ball sizes, which take into account geometric features, are bounded by the prescribed minimum and maximum sizes $r_{\min}$ and $r_{\max}$. This provides the user with control over the discretization of sharp features as well as over the area of blending surfaces inside blending balls. Smaller protecting balls lead to a denser mesh around sharp features, and smaller blending balls give more faithful approximations of trimmed regions. In practice, for a better mesh quality around sharp features, protecting balls are expected to be reasonably large to cover curves with small curvatures and at the same time agree with the mesh parameters in the vicinity, and the largest ball size is restricted to $r_{\max}$. The minimum ball size $r_{\min}$ is mainly assigned to blending balls, in order to fully cover trimmed areas with small enough balls, see Section 4.2 for more details on the determination of ball sizes.

The mesh criteria in the Delaunay refinement are usually user-defined [JAYB15], which determines the facet and cell properties in the mesh. Facet criteria control facet sizes and shapes. Facet sizes can be either uniform or varying with a sizing field, which are related to user-defined parameters on the triangle circumradius F_a and the surface approximation error F_δ. The facet shape is controlled with a prescribed smallest angle F_s of the triangle. The property of tetrahedron cells is related to cell sizes and shapes. The cell size is controlled via the circumsphere radius C_a, and the cell shape is controlled via the radius-edge ratio C_s of the tetrahedron. The parameter for the cell shape should be chosen carefully to ensure the termination of the refinement process.

3.3. Intersection oracle

Computing intersections is central to the Delaunay meshing process, from the initial sampling to Delaunay filtering and refinement. The intersections are abstracted through an oracle which takes different surface representations as input [JAYB15]. For a NURBS model, the oracle considers intersections with trimmed NURBS surfaces and blended implicit surfaces. Different intersection algorithms are activated based on whether the line segment query passes through blending balls.

Trimmed NURBS surfaces. Each intersection query is first computed with untrimmed NURBS surfaces defined over complete parametric domains. For the sake of computational efficiency, untrimmed NURBS surfaces are first decomposed into rational Bézier surfaces and stored into an AABB tree for accelerating intersection queries [ATW20]. Computing intersections between line primitives and Bézier surfaces is performed through recursive subdivision [SIN20]. Since the pre-images of an intersection point are obtained in the complete parametric domain, they are likely to lie outside the trimmed parametric domain bounded by trimming curves and thus should be filtered out. Only intersection points with pre-images inside the trimmed parametric domain are valid. Details on intersection computation with trimmed NURBS surfaces are provided by Shen et al. [SBAD16].

Blending regions. When a line segment query intersects blending balls, it is first divided into sub-segments delineated by the boundaries of blending balls. For sub-segments inside blending balls, the intersection is computed with implicit surfaces that are substituted for trimmed regions. Details on solvers for implicit surface and intersections inside blending balls are provided in Section 5. Other sub-segments outside balls are processed by the above trimmed NURBS surface intersection. The complete set of resulting intersection points is obtained through computing intersections with all sub-segments.

4. Ball covering

All the trimmed regions of a NURBS model are covered with balls as a preprocessing before Delaunay meshing. Preprocessing via ball covering to repair trimming defects has been proposed by Busaryev et al. [BDL09], in which all boundary curves are preserved with protecting balls such that the defects are never queried during Delaunay refinement. By contrast, our approach hinges upon a two-phase ball generation process, with protecting balls covering sharp features and blending balls covering smooth trimmed regions, such that the two different trimmed regions are dealt with differently later during Delaunay refinement.

The complete ball covering of a 3D intersection curve is required for a valid repair, that is, all defects must be contained inside balls. The completeness of ball covering depends on ball sizes and ball sampling processes on the curves. Busaryev et al. proposed to sample protecting balls uniformly on the images of trimming curves, and a subsequent merging operation is performed on the balls to cover the defects in trimmed regions [BDL09]. The ball distribution and sizing heavily depends on this merging operation. It requires an appropriate pairing of balls to merge, which, however, cannot always be guaranteed. To avoid the side effect of the merging operation, we choose to sample balls directly along 3D intersection curves, taking into account local size of defects while determining the ball sizes. In our approach, all the 3D curves are first covered with protecting balls of adaptive sizes. After distinguishing sharp and smooth features, balls covering smooth trimmed regions are then replaced by uniformly sized balls considering defect sizes.

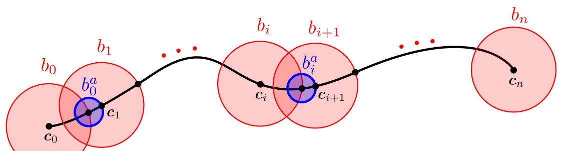
4.1. Sampling

For each intersection curve of two NURBS surfaces, balls are generated iteratively from one end to the other end. We follow the concept of using auxiliary balls introduced by Dey et al. [DL09] to sample balls along a 3D curve. The ball generation begins with placing two end balls b_0 and b_n, with centers at the two end points of the curve. After a ball b_i with the center c_i and the radius r_i is determined, the next ball b_{i+1} is positioned with the aid of an auxiliary ball b_i^a. As depicted by Figure 4a, the auxiliary ball b_i^a is centered at the forward intersection between b_i and the curve, with the radius of $r_i/3$ to allow sufficient separation between centers of adjacent balls. The ball center c_{i+1} is at the forward intersection between b_i^a and the curve.

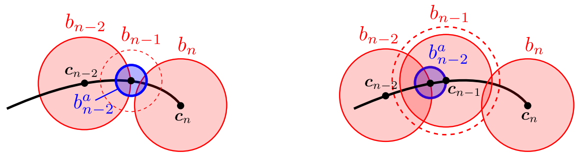
There are two different scenarios when the ball generation reaches the other end of the curve, as depicted in Figure 4b and 4c. In order to guarantee a full covering of balls over the curve and a sufficient overlapping of adjacent balls at the end, the following rules are adopted. In the first scenario when the last auxiliary ball intersects the end ball b_n, the last ball b_{n-1} is positioned at the center of the auxiliary ball b_{n-2}^a, with the radius of $\frac{2}{3}r_{n-2}$. In the second scenario when the ball b_{n-1} intersects the end ball b_n, the ball is replaced with an enlarged size of $\frac{7}{6}r_{n-1}$.

To ensure that the curve is completely covered by balls, adjacent balls must overlap sufficiently. In addition, a ball center should not be contained in another ball, which is required in the weighted Delaunay meshing to preserve surface features covered by protecting balls. According to [DL09], the following sufficient overlap of two balls is enforced during ball generation,

$$\max\{r_i, r_{i+1}\} < ||c_{i+1} - c_i|| \leq r_i + r_{i+1} - \frac{1}{7}\min\{r_i, r_{i+1}\}, \quad (1)$$



(a) *Ball positions determined with auxiliary balls.*



(b) *First end scenario.* **(c)** *Second end scenario.*

Figure 4: *Sampling of balls along a curve.*

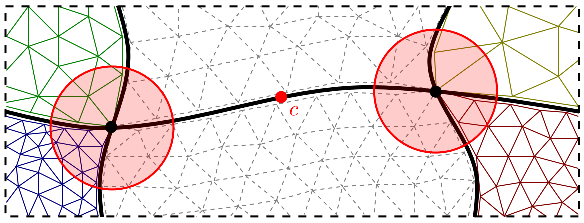
where c_i and c_{i+1} are the two ball centers; the lower limit controls the separation between a ball center and the other ball, and the upper limit controls the minimum overlap between two balls.

4.2. Adaptive sizing

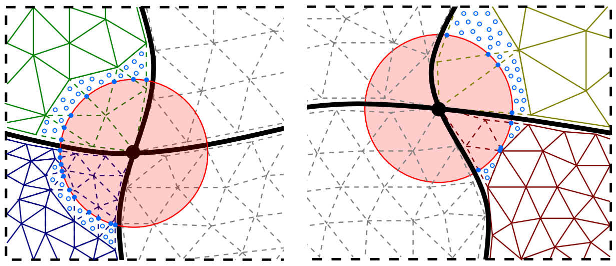
The ball size is determined considering the ball connectivity and the desired mesh approximation. A ball should intersect sufficiently with its adjacent balls while not intersecting non-adjacent balls. Departing from the ball refinement approach proposed by Dey and Levine [DL09] to ensure the separation between non-adjacent balls, we achieve ball separation by measuring several *feature distances* at ball centers, including the distance to non-adjacent surfaces and the local feature size of the curve. Such an adaptive sizing also considers the approximation accuracy of sharp features. In the following, we use the *ball curve* to denote the curve covered by a ball.

4.2.1. Preprocessing of NURBS surfaces

To avoid the complexity of direct computations on NURBS surfaces, feature distances are computed on a tessellation of NURBS surfaces which is obtained by simply mapping 2D Delaunay meshes in parametric domains to 3D Euclidean space. Trimming curves are discretized with a good approximation relying on a faithful mapping to a set of 3D Bézier curves. Based on the fact that a 2D Bézier curve can be mapped into a 3D Bézier curve faithfully if the curve lies on a tensor-product Bézier surface, we follow the methodology presented by Lasser et al. [LB95]. To summarize, the Bézier equivalents of trimming curves are divided by the knot spans of the decomposed Bézier surfaces of the NURBS surface such that each Bézier curve segment is restricted within one Bézier surface. Each trimming curve is thus exactly mapped into a set of 3D Bézier curves. The simple tessellation of NURBS surfaces with a controlled surface approximation can therefore be obtained straightforwardly, and the triangles are stored into an AABB tree [ATW20] for accelerating subsequent queries.



(a) *Ghost triangles (dashed gray) in adjacent surfaces.*



(b) *Additional ghost triangles (dashed) overlapped with end spheres and extra sampling points (blue dots).*

Figure 5: *Computing the surface separation distance at a ball center.*

4.2.2. Surface separation distance $d^{\mathrm{s}}_{\mathrm{sep}}$

In order to ensure the separation between balls on different curves, the ball size considers the shortest distance between the ball center and surfaces non-adjacent to the ball curve. The shortest distance is computed by traversing the AABB tree created in the preprocessing. The triangles belonging to surfaces adjacent to the ball curve are tagged as *ghost* in the AABB tree, see Figure 5a, and these *ghost* triangles are intentionally ignored during tree search. In addition, for in-between balls on the curve, only triangles that do not intersect the two end balls are considered. However, it can be risky to get an undesired large separation distance if triangles intersecting the two end balls are excluded. We solve this issue with a hybrid scheme depicted in Figure 5b by considering dense sampling points inside these extra ghost triangles but outside end balls, including intersection points between ghost triangles and end balls. In general, the intersection points between the additional ghost triangles and balls are critical. The shortest distance between the ball center and the sampling points is then compared with the distance query with the AABB tree. The separation distance takes the smaller value.

4.2.3. Curve separation distance $d^{\mathrm{c}}_{\mathrm{sep}}$

Apart from the separation of any two balls on different curves, non-adjacent balls on the same curve should not intersect as well. This separation distance is estimated by increasing the ball size gradually until it intersects with a non-adjacent curve segment. The intersection is computed with discretized curves obtained in the NURBS preprocessing considering a prescribed approximation error. The process of gradually growing balls is illustrated in Figure 6a. The curve separation distance is therefore obtained by measuring the distance between the ball center and the non-adjacent line segment intersected.

4.2.4. Curve approximation $d^{\mathrm{c}}_{\mathrm{approx}}$

Sizes of protecting balls should also take into account the user-defined mesh quality for a good approximation of sharp features in the final mesh. As the centers of protecting balls are treated as weighted vertices in the Delaunay meshing process, the approximation of sharp features depends on the line segments connecting protecting ball centers. The deviation of the line segments from the curve is restricted within a prescribed edge approximation error δ^{c}, which can be controlled by increasing the ball size r_i gradually, as in the process of estimating the edge separation distance, until it reaches the edge approximation error δ^{c}, see Figure 6b. The distance $d^{\mathrm{c}}_{\mathrm{approx}}$ is the largest ball size that satisfies the edge approximation error. The ball size takes two thirds of the distance obtained, i.e. $r_i = \frac{2}{3} d^{\mathrm{c}}_{\mathrm{approx}}$. This guarantees that the line segment connecting two adjacent balls deviates from the curve within δ^{c}. The ball growing process terminates when either the curve separation or the curve approximation is satisfied. In addition, the curve curvature κ is considered in the curve approximation, that is, $d^{\mathrm{c}}_{\mathrm{approx}}$ is also bounded by the curvature radius $1/\kappa$ to avoid a large ball in the region with a high curvature.

4.2.5. Trimming error ε

A 3D intersection curve has a trimming curve in each of its two adjacent surfaces. The images of the two trimming curves in 3D space are not identical in general, and such a discrepancy leads to trimming errors. In order to estimate the trimming error around a ball center, we compute at the ball center the normal plane to the curve and its intersection points with the images of two trimming curves. The trimming error is estimated as the largest distance among the ball center and the two intersection points. The trimming error indicates a lower bound for the ball size, and the sufficient overlapping between adjacent balls enforced with (1) in general ensures that the trimming error can be fully covered along a curve. The complete covering of trimming errors can always be guaranteed by checking if the intersections between a ball and the images of two trimming curves lie inside the adjacent balls.

4.2.6. Sizing formula

In addition to the feature distances described above, a ball size is often bounded by the user-defined maximum size $r_{\max}$, which is necessary for the control of sharp feature discretization. Hence, the ball size is determined by the following formula:

$$r = \min\{\frac{1}{3}\min\{d^{\mathrm{s}}_{\mathrm{sep}}, d^{\mathrm{c}}_{\mathrm{sep}}, 2d^{\mathrm{c}}_{\mathrm{approx}}\}, r_{\max}\}, \tag{2a}$$

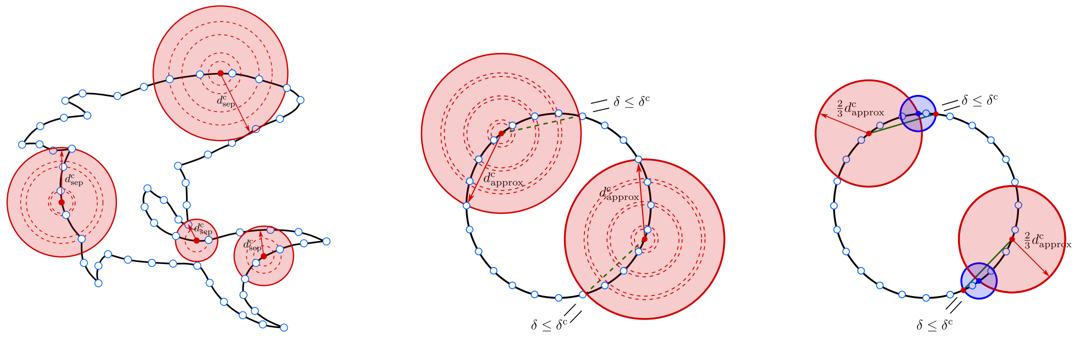
$$r \geq 2\varepsilon. \tag{2b}$$

Blending balls have uniform sizes given by the minimum size $r_{\min}$ and the trimming error ε, that is,

$$r = \max\{r_{\min}, 2\varepsilon\}. \tag{3}$$

As a final step, the ball size obtained must satisfy the sufficient overlap between adjacent balls (1), i.e.

$$r \in \{r|\text{sufficient overlap}\}. \tag{4}$$



(a) *Curve separation distance* **(b)** *Curve approximation* **(c)** *Guarantee on curve approximation error with line segments*

Figure 6: *Determination of ball sizes considering curve separation distance and curve approximation with ball growing method. The ball size increment depends on the curve discretization, and each dashed circle represents an increment step.*

4.3. Algorithms

The ball generation for each curve is a two-phase process: adaptive-sized balls for sharp feature curves and uniform-sized balls for smooth feature curves. Before the uniform-sized balls for smooth feature curves are generated, it is necessary to identify whether a curve delineates a sharp feature. The angle criterion is used for detecting sharp features, that is, the curve segment inside a ball is regarded as a sharp feature if the angle between surface normals inside the ball is greater than a user-defined angle threshold. The surface normals are evaluated at sampling points on boundary curves. All balls covering smooth features are then replaced with uniform-sized balls. Finally, after all curves are covered with balls, balls at the curve ends should be merged. The overall algorithm for generating balls covering curves is outlined as follows in Algorithm 2 and the algorithm for generating balls on a single curve is provided in Appendix A.

Algorithm 2 Ball covering of trimmed NURBS models

```
Input: Trimmed NURBS model
Output: A set of balls $\{\mathcal{B}_i\}$ covering all intersection curves $\{\mathcal{C}_i\}$
  for all curves do
      - Generate adaptive-sized balls (Appendix A) using sizing formula (2)
      - Distinguish balls covering sharp feature
      - Replace balls covering smooth feature curve with uniform-sized ones (Appendix A) using sizing formula (3)
  end for
  Merge end balls and update ball connection information.
```

5. Repair of trimming defects

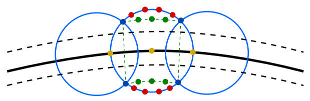
Trimming defects covered with protecting balls and blending balls are repaired in different ways. The defects are in fact neglected inside protecting balls in the Delaunay refinement, as they are in the regions that are never interrogated. The preservation of sharp features with protecting balls has been studied for the Delaunay meshing [BDL09, DL09]. In our implementation, the approximation of the sharp feature depends on the adaptive-sized protecting balls which are controlled with surface features and user-defined parameters, as described in Section 4.

In order to repair trimming defects inside blending balls, e.g. non-watertightness, non-manifold, self-intersection, etc., which jeopardize the Delaunay meshing process, we propose a new blending technique to approximate and blend the surfaces covered by blending balls in such a way that implicit surfaces are substituted for the original NURBS surfaces inside blending balls. A line segment query emitted during Delaunay refinement is divided into sub-segments after computing its intersection with boundaries of blending balls. The implicit intersection is then computed for sub-segments inside blending balls.

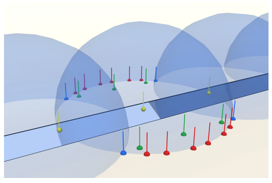
5.1. Blending surfaces

Given discrete points in space, Duchon [Duc77] introduced an interpolating spline based on the minimization of a semi-norm generalizing the Fourier transform of the thin-plate bending energy. The 1D cubic spline and 2D thin-plane spline are two special cases of the Duchon's interpolating spline. We adopt its 3D Hermite interpolation variant which interpolates the values and gradients at points. This interpolating spline was used by Huang et al. [HCJ19] for surface reconstruction from a set of sampling points, in which it is demonstrated that the interpolating scheme generates reasonable implicit surface with sparse and non-uniform sampling points, and is resilient to sampling imperfections. In our setting, Duchon's interpolating spline is a relevant choice as reliable sampling points and their surface normals can be obtained readily around blending balls.

Using Duchon's implicit interpolant, the implicit surfaces inside blending balls interpolate sampling points and surface normals. In order to trade approximation accuracy for computational cost, a handful of sampling points are strategically selected on NURBS surfaces in four regions as depicted in Figure 7, which are blending ball boundaries, intersections between adjacent balls, truncated cone boundaries and 3D intersection curves, such that the implicit surface fits the blending region with satisfactory accuracy using a small number of sampling points.

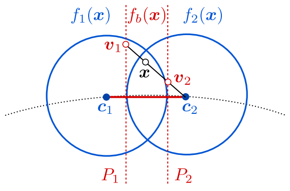


(a) *Sampling points*

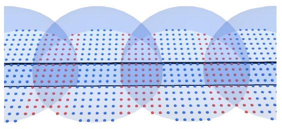


(b) *Surface normals*

Figure 7: *Sampling points and their surface normals for constructing the interpolating implicit surface. Red surface points are on the ball boundary; blue points at intersections between adjacent balls; green points on the truncated cone inside the ball; yellow points on the 3D intersection curve.*



(a) *Blending function*



(b) *Implicit surface visualization*

Figure 8: *Continuous implicit surfaces between adjacent surfaces with a blending function. The implicit surface is visualized by sampling, with blue surface points inside individual balls and red surface points inside two adjacent balls evaluated with blending function.*

The polyharmonic spline with the triharmonic radial basis function (RBF) $\phi(x,y) = ||y-x||^3$ is considered to interpolate sampling points on surfaces. Given points $p_i \in \mathbb{R}^3 (i = 1,2,\cdots,m)$ on a NURBS surface and their surface normals n_i, an implicit surface interpolating the points and their normals is defined as follows,

$$f(x) = \sum_j^m \alpha_j \phi(x,p_j) + \sum_j^m \beta_j^{\mathrm{T}} D_y \phi(x,p_j) + \gamma^{\mathrm{T}} x + \eta \qquad (5)$$

with the unknown coefficient vectors $\alpha \in \mathbb{R}^m$, $\beta \in \mathbb{R}^{3m}$, $\gamma \in \mathbb{R}^3$ and $\eta \in \mathbb{R}$; $D_y\phi(\cdot,\cdot)$ denotes the first derivative of triharmonic RBF w.r.t. the second variable.

The unknown coefficient vectors are obtained by considering the interpolation of point coordinates and normals as well as additional orthogonality conditions, which are

$$f(p_i) = 0, \quad \bigtriangledown f(p_i) = n_i \quad \text{(interpolation)}, \qquad (6)$$

$$\sum_i^m \alpha_i p_i + \sum_i \beta_i = \mathbf{0}, \quad \sum_i^m \alpha_i = 0 \quad \text{(orthogonality)}. \qquad (7)$$

The linear equation system obtained from the above conditions is expressed in the following matrix form,

$$\mathbf{Ku} = \mathbf{F} \qquad (8)$$

with

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_0 & \mathbf{K}_1 \\ \mathbf{K}_1^{\mathrm{T}} & \mathbf{0} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \eta \end{pmatrix} \quad \text{and} \quad \mathbf{F} = \begin{pmatrix} 0 \\ n \\ 0 \\ 0 \end{pmatrix}. \qquad (9)$$

Details on entries of matrix $\mathbf{K}$ are provided in Appendix B.

5.2. Implicit surface interrogation

The implicit function (5) has a zero value at the intersection point between a line segment and the implicit surface defined. If the sub-segment inside a blending ball intersects the implicit surface, it is well assumed that there is a single intersection since the sub-segment clipped by the blending ball is very short and it becomes increasingly normal to the surface as the Delaunay refinement proceeds. We hence resort to bisection to compute the intersection. In the case of multiple intersections between the sub-segment and the implicit surface, it is enough to insert one of the intersection points in Delaunay refinement.

Since the implicit surfaces are computed individually inside each blending ball, they are not supposed to be continuous across adjacent balls, i.e. the implicit surfaces inside the overlapping domain of two adjacent balls are different. To overcome the hurdle of the non-continuity of the implicit surfaces across two adjacent balls, the following blending function is proposed to construct a continuous transition from one implicit surface to the other,

$$f_b(x) = (1-w) f_1(x) + w f_2(x), \qquad (10)$$

where f_1 and f_2 are implicit functions in two adjacent balls (centered at c_1 and c_2 respectively, as illustrated in Figure 8a), w is the weight given by

$$w = \frac{||x-v_1||}{||v_2-v_1||} \qquad (11)$$

with v_1 and v_2 the intersections between the line c_2-x and two planes P_1 and P_2 orthogonal to the direction c_2-c_1.

Hence, in each bisection iteration when computing the intersection with implicit surfaces, if the point lies in the overlapping domain of adjacent balls it is evaluated with the blending function (10), and the implicit function (5) is used for points in one blending ball exclusively.

Since NURBS surfaces are approximated with implicit surfaces inside blending balls, the discrepancy between the two representations is inevitable, though small enough to be negligible, near the balls. For enhancing the robustness of the intersection oracle, both the NURBS intersection and the implicit intersection are computed in the transition region near the ball boundary, and the implicit intersection takes priority if both intersections exist. In order to enforce the watertightness in the transition region, we introduced sufficient interpolating points located strategically around blending balls as illustrated in Figure 7. In practice, the extreme case of a dual Voronoi edge passing through the tiny gap around a blending ball does not hinder the Delaunay meshing process, because it may only occur in the first few steps of Delaunay refinement depending on the initial Delaunay sampling, and Voronoi edges tend to be normal to the surface as the refinement proceeds.

6. Results

A selection of examples are shown in the following to demonstrate the capability of our approach to simultaneously repair and mesh NURBS models with trimming defects and its added values to NURBS defeaturing. More examples are given in the appendix. The general Delaunay refinement framework of the CGAL library is used in our implementation [JAYB15], and the SISL library [SIN20] is used for the spline intersection computation.

6.1. Parameterization-free meshing

The parameterization-free meshing is demonstrated with a mechanical part as shown in Figure 9a. Sharp features are covered with protecting balls, and other smooth trimmed regions are covered with blending balls. The edge approximation error is considered for protecting balls, and blending balls have uniform sizes. Figure 9b shows the Delaunay surface mesh with a prescribed surface approximation error. The distribution of mesh elements is not constrained by the layout of trimmed NURBS surfaces, which is a key feature of our approach. All the sharp boundary curves are preserved and discretized with protecting ball centers. However, the weighted Delaunay meshing using protecting balls has an inherent issue of generating excessively small elements between sharp features subtending small angles. We resort to local remeshing for these cases [FTB16, TFTB20] to replace the ill-shaped elements. Figure 9c shows the comparison of the mesh elements in the cusp region before and after remeshing. The mesh quality is reported in Appendix C.

Our method can be extended to multidomain meshing straightforwardly, and different mesh quality criteria can be prescribed in domains. The multidomain meshing of turbines is given in Appendix D.

6.2. Defect repair

As depicted in Figure 1, four common types of defects exist in a trimmed NURBS model, including gaps, overlaps, intersections and mismatched end points. The NURBS model in Figure 10 is considered for the repair of surface overlaps. The blending balls are adaptive to the overlap size to ensure the complete covering of the defects, see Figure 10b. Inside blending balls, implicit surfaces are computed to bridge the two adjacent surfaces. The overlap defect is repaired in the Delaunay mesh controlled by the surface approximation error criterion, as shown in Figure 10d, and the mesh elements cross the overlap region in the NURBS model.

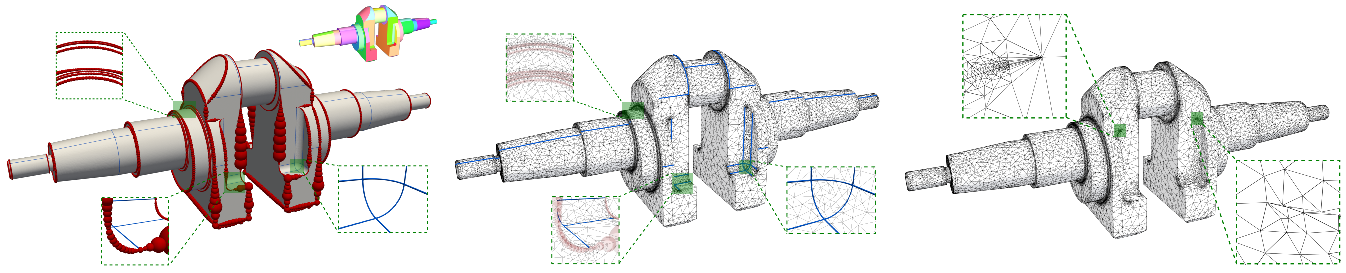
Figure 11 shows that multiple types of defects are repaired systematically with our approach, see Figure 1b for the close-up view of the defects, which includes gaps, intersection and mismatched ends between adjacent surfaces. The defects in sharp features are covered with protecting balls such that the defect regions are not interrogated in the Delaunay refinement and thus repaired automatically in the Delaunay mesh. The defects in the smooth trimmed regions are covered with blending balls. The two adjacent surfaces are bridged with implicit surfaces inside blending balls as shown in Figure 11c, such that the implicit surfaces are considered when the defect regions are interrogated during Delaunay meshing. Figure 11d shows the Delaunay mesh around the defect regions repaired with protecting and blending balls. The sharp features are repaired and preserved in the final mesh by connecting protecting ball centers, and other defect regions are meshed with crossing elements.

A watertight surface mesh is usually required by most NURBS meshers to generate a volume mesh. However, this requirement may not be guaranteed for some immoderately bad CAD designs. Figure 12a shows a NURBS model with tapering surfaces penetrating between surfaces, with unobserved defects around the tips. Gmsh [GR09] cannot mesh this non-watertight case with small element sizes around penetrating tips, leaving the tapering surfaces unmeshed (see Figure 12b) and thus preventing from generating a volume mesh. The Delaunay mesh in Figure 12c is generated with our approach, in which the narrow tips are covered with protecting balls such that the intricate details due to a bad design are neglected in the meshing. The protecting balls are obtained with the sizing formula presented in Section 4.2. It is feasible to replace the small sized protecting balls with larger ones, in order to improve the mesh quality in the defect regions.

It is worth mentioning that a robust tetrahedral meshing algorithm *TetWild* proposed recently by Hu et al. [HZG*18] can generate a tetrahedral mesh from a defect-laden surface mesh. Though it is not devised specifically for NURBS meshing, it provides another means of obtaining the tetrahedral mesh from an initial NURBS tessellation. The algorithm has been demonstrated unconditionally robust for general defects. One needs to select proper parameters in order to get a volume mesh with consistent quality. We compared with TetWild and its efficient variant *FTetWild* [HSW*20] to generate a volume mesh from a tessellation which maintains trimming defects of a NURBS model (see Section 4.2.1). Figure 13 shows the diamond tessellation that challenges TetWild to generate a mesh with consistent quality, though different parameters are selected. Gaps can be robustly filled but may yield artifacts such as bumps in the mesh. FTetWild is faster and generates more consistent mesh qualities than TetWild for the diamond model, though we can still observe missing tetrahedra in the mesh as depicted in Figure 13e. Figure 14 compares TetWild with our method on a NURBS model with multiple defects including large gaps, overlaps and intersections. We observe that the mesh quality given by TetWild with default parameters is not consistent in similar regions. By contrast, our method provides a more flexible and consistent control on the mesh quality while repairing defects.

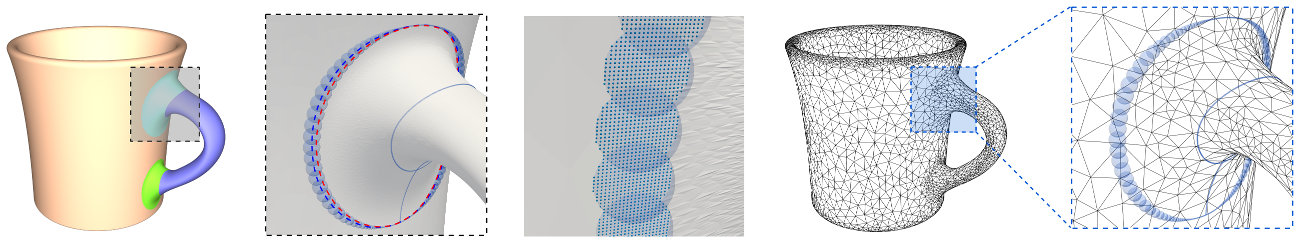
6.3. Comparison with Busaryev et al.

Though protecting balls have also been utilized by Busaryev et al. [BDL09] to repair defects in NURBS models, our approach differs substantially and offers additional benefits. Figure 15 compares the Delaunay mesh obtained with the two approaches. As discussed in Section 4, the sizes of protecting balls are determined adaptively based on geometric features, avoiding unnecessarily small or large ball sizes due to the shrinking and merging processes proposed by Busaryev et al. Moreover, the use of blending balls to repair defects in smoothly connected trimmed regions removes the dependence of mesh elements on the NURBS layout.



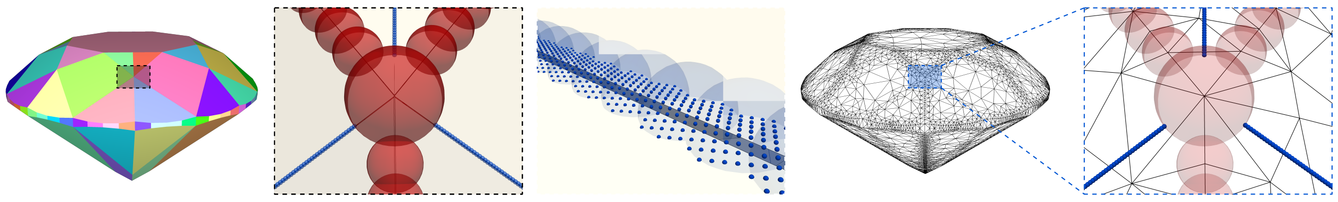
(a) *Protecting and blending balls* **(b)** *Mesh with surface approximation* **(c)** *Mesh elements in cusp regions*

Figure 9: *Parameterization-free meshing with sharp feature preservation. The inherent issue of Delaunay mesh in cusp regions is solved with remeshing. In (c): Left - ill-shaped elements in the cusp region; right - remeshed elements with sharp features preserved.*



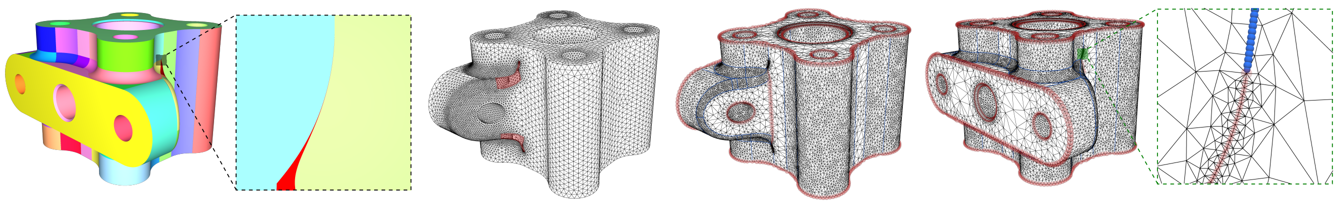
(a) *NURBS model* **(b)** *Blending balls covering overlaps* **(c)** *Implicit surface* **(d)** *Delaunay mesh repairing overlaps*

Figure 10: *Repair of overlap between NURBS surfaces. The boundary curves of the overlapping surfaces are depicted in (b). Implicit surfaces inside blending balls are visualized with point sampling in (c).*



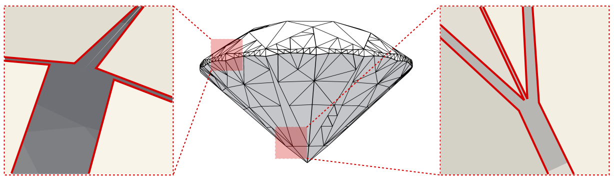
(a) *NURBS model* **(b)** *Balls covering defects* **(c)** *Implicit surface sampling* **(d)** *Delaunay mesh*

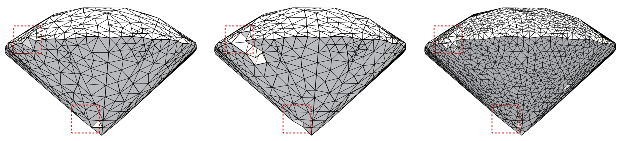
Figure 11: *Repair of multiple types of defects, including gaps, intersections and end mismatch.*

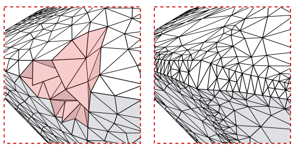


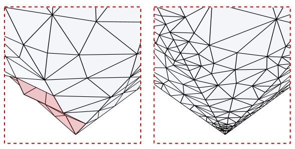
(a) *NURBS model with a tapering surface* **(b)** *Gmsh result with missing elements (in red regions)* **(c)** *Delaunay mesh* **(d)** *Close-up view at the penetrating tip*

Figure 12: *Repair of a non-watertight NURBS model (from the ABC library) near penetrating tips.*


(a) *Defects of gaps in the tessellation of the NURBS model.*


(b) *Mesh generated by TetWild with parameters $L = 0.05, e = 0.001$ (left), $L = 0.05, e = 0.0002$ (middle) and $L = 0.01, e = 0.001$ (right).*


(c) *Close-up view compared with our method (right)*


(d) *Close-up view compared with our method (right)*

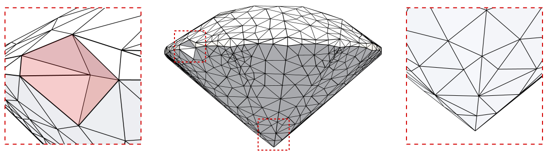
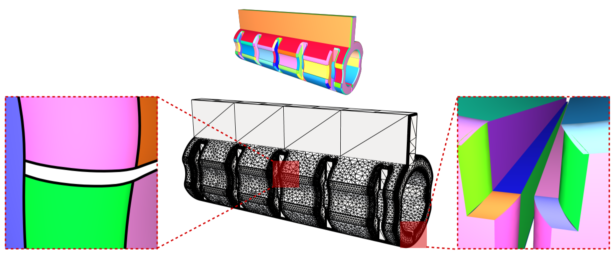
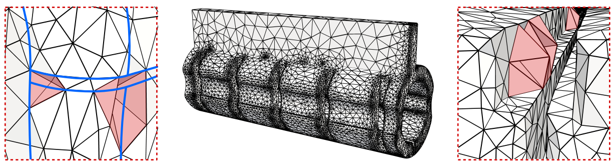

(e) *Mesh generated by FTetWild with $L = 0.05, e = 0.0002$.*

Figure 13: *Defects in Diamond model challenge TetWild and fTetWild to generate a mesh with consistent quality in some regions (highlighted in red in (c), (d) and (e) for visualization purpose). L and e are ratios of the bounding box, controlling the target edge length and the envelop size, respectively.*


(a) *Triangulated tessellation of a NURBS model (from the ABC library) that contains large gaps, overlaps and intersections.*


(b) *Tetrahedral mesh obtained with TetWild (L and e are default values, 0.05 and 0.001 respectively), with either missing or extra tetrahedra in some regions (highlighted in red for visualization purpose).*

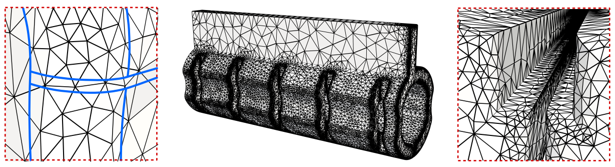

(c) *Defects are repaired in the mesh and mesh quality is consistent with our method.*

Figure 14: *Multiple defects of large gaps, overlaps and intersections pose a challenge to TetWild in some regions without tuning meshing parameters.*

In addition to the defect repair, blending balls can also be leveraged for removing small features in a NURBS model. Such an operation has been explored in [BDL09] Section 8 but by using protecting balls, which leads to unnecessary constraints in the final mesh. The small geometric details in Figure 16a are covered with protecting balls and preserved with small element sizes, see Figure 16b and 16d. To remove the small geometric details, blending balls are used to cover the target regions containing small details (see Figure 16c). The geometric details are replaced with implicit surfaces inside blending balls and thus neglected automatically during Delaunay refinement and defeatured in the final mesh as shown in Figure 16e. In Appendix D, another application of blending spheres to smooth out sharp features is demonstrated.

6.4. Performance

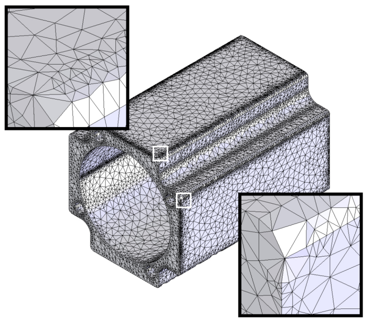
The timings of generating the Delaunay meshes presented in the examples are reported in Table 2. The computation was performed on a laptop computer with Intel(R) Xeon(R) CPU E5-2630 v4 2.20GHz. The running time depends on the complexity of the NURBS models and on the output mesh density (number of facets and tetrahedra).

Table 2: *Running time (in seconds) of ball generation and Delaunay meshing with balls and mesh statistics.*

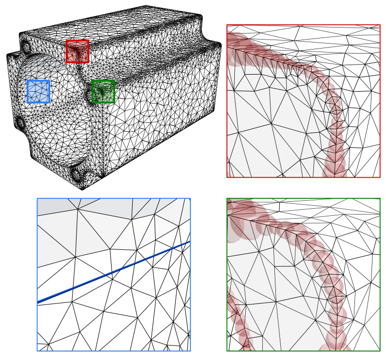
Model	ball cover	meshing	#balls	#facets	#tetrahedra
Fig. 9	137.1	262.2	20228	34134	100049
Fig. 10	23.7	44.3	3239	12936	23046
Fig. 11	5.8	18.4	22312	12492	25410
Fig. 12	102.6	426.6	27037	67956	138655
Fig. 14	11.1	1355.1	13801	62920	120820
Fig. 15	6.0	139.2	24932	35412	67339
Fig. 16	4.7	124.8	5287	26928	52327

6.5. Limitations

Since ball sizes are related to trimming errors in order to fully cover all the defects (see sizing formula (2) and (3)), immoderate defect sizes require large balls which put the proposed approach at risk



(a) *Delaunay mesh with Busaryev's method (figure from [BDL09]).*



(b) *Delaunay mesh with our method.*

Figure 15: *Comparison with Busaryev's method. Sharp features are preserved considering geometric features and mesh elements near smooth connections, and are not constrained by the NURBS layout.*

when defect sizes are comparable to surface dimensions. Large protecting balls can swallow entire surface features, leading to undesired mesh quality in the repaired region, and they may break the ball separation rule, which can derail or terminate the Delaunay meshing process. Such cases with visually large defects require resorting to CAD repair as a preprocessing step.

7. Conclusions

We have presented a method for simultaneously repairing and meshing NURBS models with trimming defects, through providing a consistent interface to a Delaunay-based isotropic simplicial mesh generation framework. Defects such as gaps and overlaps due to trimming are healed through substituting local NURBS surfaces by implicit surfaces defined within a set of healing balls covering the defects. The ball covering process is guaranteed to fully cover all trimmed regions. It echoes the ball sampling process that has been proved to satisfy the requirement for weighted Delaunay meshing [DL09]. The implicit surface using the Duchon's interpolating spline provides a tight approximation to the original NURBS surface. The blending property of the above implicit surfaces also provides us with a means to smooth out sharp features and defeature small details. We also contribute an algorithm for estimating the local feature size of NURBS surfaces and sharp features. Such a local feature size is used for sizing the above healing balls. We demonstrate the robustness and versatility of our method: the output volume meshes are valid by design (without overlaps or inverted elements), independent of the input NURBS patch layout and parameterization, and conforming at interfaces between multiple domains in contact.

The proposed approach is mainly limited by exceedingly large defect sizes such that balls covering the defects are larger than surface dimensions. Consequently, the entire surface feature may be swallowed by large protecting balls, and the blending balls may fail to heal the defects as non-adjacent surfaces are covered. The robustness of the implicit surface interrogation near the transition region is currently enhanced with a workaround for extremely rare cases of dual Voronoi edges almost tangential to surfaces and passing through gaps between implicit surfaces and original NURBS surfaces. In addition, the current mesh-and-repair algorithm provides insufficient feedback to users about the set of possible parameters for automatic defeaturing in the repair process. There is also an intrinsic issue of Delaunay-based approaches in which mesh elements are distorted and ill-shaped near sharp features subtending small angles. Resorting to a remeshing post-processing is a solution [FTB16], but having a single integrated algorithm would be more elegant.

Our Delaunay-based mesh-and-repair algorithm has guaranteed to generate valid isotropic simplicial meshes for NURBS models. As future work, we would like to extend the approach to anisotropic meshing and higher-order mesh elements [FAB*18], which are relevant for numerical analysis. In addition, we wish to complement our meshing algorithm by a scale-space analysis stage, providing users with a preview on the whole spectrum of possible operations on NURBS models in terms of repair, defeaturing and smoothing.

References

[AB99] Amenta N., Bern M.: Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry 22*, 4 (1999), 481–504.

[ACK13] Attene M., Campen M., Kobbelt L.: Polygon mesh repairing: An application perspective. *ACM Computing Surveys 45*, 2 (2013).

[Att10] Attene M.: A lightweight approach to repairing digitized polygon meshes. *The visual computer 26* (2010), 1393–1406.

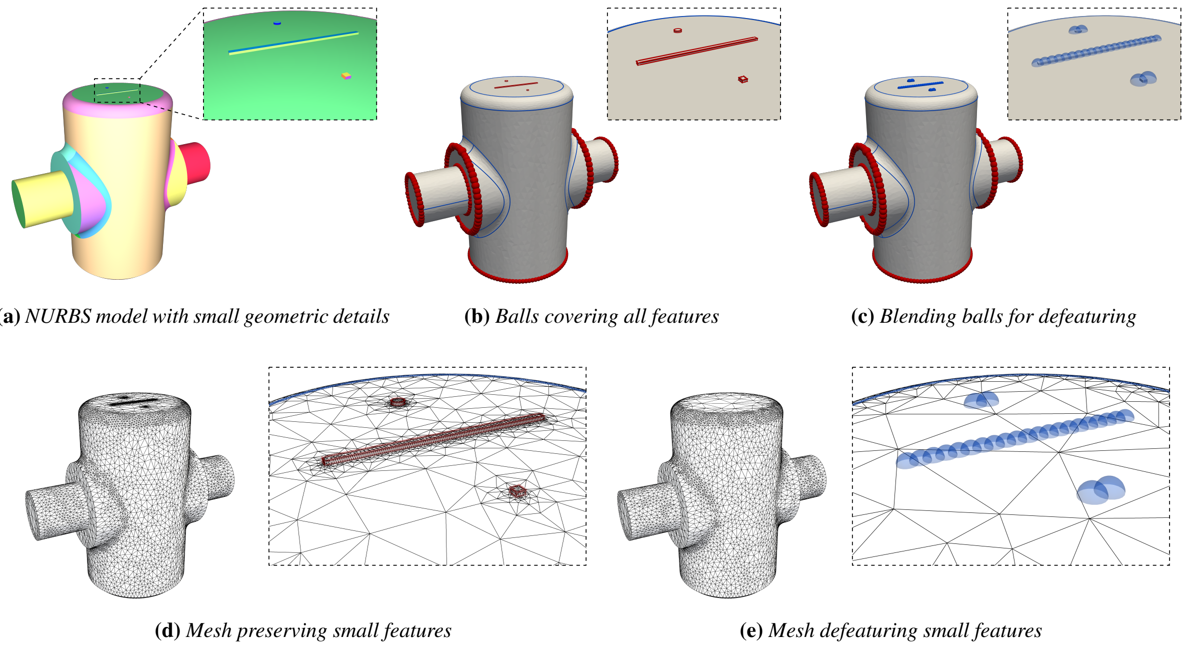
[ATW20] Alliez P., Tayeb S., Wormser C.: 3D fast intersection and distance computation. In *CGAL User and Reference Manual*, 5.2 ed. CGAL Editorial Board, 2020. URL: https://doc.cgal.org/5.2/Manual/packages.html#PkgAABBTree.

[BDL09] Busaryev O., Dey T. K., Levine J. A.: Repairing and meshing imperfect shapes with delaunay refinement. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (2009), pp. 25–33.

[BK05] Bischoff S., Kobbelt L.: Structure preserving cad model repair. *Computer Graphics Forum 24*, 3 (2005), 527–536. doi:https://doi.org/10.1111/j.1467-8659.2005.00878.x.

[BLG00] Borouchaki H., Laug P., George P.-L.: Parametric surface meshing using a combined advancing-front generalized delaunay approach. *International Journal for Numerical Methods in Engineering 49*, 1-2 (2000), 233–259.

[CDR07] Cheng S.-W., Dey T. K., Ramos E. A.: Delaunay refinement for piecewise smooth complexes. In *Proceedings of the eighteenth*


(a) *NURBS model with small geometric details* **(b)** *Balls covering all features* **(c)** *Blending balls for defeaturing*

(d) *Mesh preserving small features* **(e)** *Mesh defeaturing small features*

Figure 16: *Small features in NURBS model are removed in the final mesh with blending balls.*

annual ACM-SIAM symposium on Discrete algorithms (2007), pp. 1096–1105.

[CDS12] Cheng S.-W., Dey T. K., Shewchuk J.: *Delaunay Mesh Generation*, 1st ed. CRC Press, 2012.

[DL09] Dey T. K., Levine J. A.: Delaunay meshing of piecewise smooth complexes without expensive predicates. *Algorithms 2*, 4 (2009), 1327–1349.

[Duc77] Duchon J.: Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive theory of functions of several variables*. Springer, 1977, pp. 85–100.

[FAB*18] Feng L., Alliez P., Busé L., Delingette H., Desbrun M.: Curved optimal delaunay triangulation. *ACM Transactions on Graphics 37*, 4 (2018), 16.

[FCF*08] Foucault G., Cuillière J.-C., François V., Léon J.-C., Maranzana R.: Adaptation of cad model topology for finite element analysis. *Computer-Aided Design 40* (2008), 176–196.

[FTB16] Faraj N., Thiery J.-M., Boubekeur T.: Multi-material adaptive volume remesher. *Computers & Graphics 58* (2016), 150–160.

[GBA*17] George P. L., Borouchaki H., Alauzet F., Laug P., Loseille A., Marcum D., Maréchal L.: *Mesh Generation and Mesh Adaptivity: Theory and Techniques*. American Cancer Society, 2017, pp. 1–51.

[GR09] Geuzaine C., Remacle J.-F.: Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering 79*, 11 (2009), 1309–1331.

[HCJ19] Huang Z., Carr N., Ju T.: Variational implicit point set surfaces. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–13.

[HLQ07] Hasle G., Lie K.-A., Quak E.: *Geometric modelling, numerical simulation, and optimization*. Springer, 2007.

[HSW*20] Hu Y., Schneider T., Wang B., Zorin D., Panozzo D.: Fast tetrahedral meshing in the wild. *ACM Trans. Graph. 39*, 4 (2020).

[HZG*18] Hu Y., Zhou Q., Gao X., Jacobson A., Zorin D., Panozzo D.: Tetrahedral meshing in the wild. *ACM Trans. Graph. 37*, 4 (2018).

[JAYB15] Jamin C., Alliez P., Yvinec M., Boissonnat J.-D.: Cgalmesh: a generic framework for delaunay mesh generation. *ACM Transactions on Mathematical Software (TOMS) 41*, 4 (2015), 1–24.

[LB95] Lasser D., Bonneau G.-P.: Bézier representation of trim curves. In *Geometric modelling*. Springer, 1995, pp. 227–242.

[LEM*17] Livesu M., Ellero S., Martínez J., Lefebvre S., Attene M.: From 3d models to 3d prints: An overview of the processing pipeline. *Computer Graphics Forum 36*, 2 (May 2017), 537–564.

[MH18] Marussig B., Hughes T. J.: A review of trimming in isogeometric analysis: challenges, data exchange and simulation aspects. *Archives of computational methods in engineering 25*, 4 (2018), 1059–1127.

[Pie05] Piegl L. A.: Knowledge-guided computation for robust cad. *Computer-Aided Design and Applications 2*, 5 (2005), 685–695.

[RPS12] Rajab K., Piegl L. A., Smarodzinava V.: Cad model repair using knowledge-guided nurbs. *Engineering with Computers 29* (2012), 477–486.

[SBAD16] Shen J., Busé L., Alliez P., Dodgson N.: A line/trimmed nurbs surface intersection algorithm using matrix representations. *Computer Aided Geometric Design 48* (2016), 1–16.

[SIN20] SINTEF S.: The sintef spline library, 2020. URL: https://github.com/SINTEF-Geometry/SISL.

[TFTB20] Tournois J., Faraj N., Thiery J.-M., Boubekeur T.: Tetrahedral remeshing. In *CGAL User and Reference Manual*, 5.2 ed. CGAL Editorial Board, 2020. URL: https://doc.cgal.org/5.2/Manual/packages.html#PkgTetrahedralRemeshing.

[TSRA17] Tierney C. M., Sun L., Robinson T. T., Armstrong C. G.: Using virtual topology operations to generate analysis topology. *Computer-Aided Design 85* (2017), 154–167.

[YH06] Yang J., Han S.: Repairing cad model errors based on the design history. *Computer-Aided Design 38*, 6 (2006), 627 – 640.

Appendix A: Algorithms for ball covering

Algorithm 3 Ball covering of a curve.

```
Input: A 3D curve $\mathcal{C}$, two end balls $b_0$ and $b_n$
Output: A set of balls $\mathcal{B}$
 1: $\mathcal{B} \leftarrow \mathcal{B} \cup \{b_0\}$
 2: $b_i \leftarrow b_0$
 3: while $b_i \cap b_n = \emptyset$ do
 4:     $b_i^a(c_i^a, r_i^a)$ // compute the auxiliary ball
 5:     if $b_i^a \cap b_n = \emptyset$ then
 6:         $b_{i+1}(c_{i+1}, r_{i+1})$ // compute the next ball
 7:         $\mathcal{B} \leftarrow \mathcal{B} \cup \{b_{i+1}\}$
 8:     else // the first ending scenario
 9:         $b_i^a \leftarrow b_i^a(c_i^a, \frac{2}{3}r_i)$ // enlarge the radius to $\frac{2}{3}r_i$
10:         $b_{i+1} \leftarrow b_i^a$
11:         $\mathcal{B} \leftarrow \mathcal{B} \cup \{b_{i+1}\}$
12:         break
13:     end if
14:     $i \leftarrow i+1$
15: end while
16: if $b_i \cap b_n \neq \emptyset$ and $i > 0$ then // the second ending scenario
17:     $b_i \leftarrow b_i(c_i, \frac{7}{6}r_i)$ // enlarge the radius to $\frac{7}{6}r_i$
18: end if
19: $\mathcal{B} \leftarrow \mathcal{B} \cup \{b_n\}$
```

Appendix B: Computing implicit surfaces

The interpolation condition at each sampling point is expressed as

$$f(p_i) = \sum_j \alpha_j \phi(p_i, p_j) + \sum_j \beta_j^{\mathrm{T}} D_y \phi(p_i, p_j) + \gamma^{\mathrm{T}} p_i + \eta = 0, \quad (12)$$

$$\nabla f(p_i) = \sum_j \alpha_j D_y \phi(p_i, p_j) + \sum_j \beta_j^{\mathrm{T}} D_y^2 \phi(p_i, p_j) + \gamma = n_i. \quad (13)$$

Hence, the matrix $\mathbf{K}$ in Section 4.1 is computed via

$$\mathbf{K}_0 = \begin{pmatrix} \mathbf{k}_{00} & \mathbf{k}_{01} \\ \mathbf{k}_{10} & \mathbf{k}_{11} \end{pmatrix} \quad \text{and} \quad \mathbf{K}_1 = \begin{pmatrix} \mathbf{k}_{02} & \mathbf{k}_{03} \\ \mathbf{k}_{12} & \mathbf{0} \end{pmatrix}, \quad (14)$$

where

$$\mathbf{k}_{00} = \{\phi(p_i, p_j)\}_{m \times m}, \quad \mathbf{k}_{01} = \{D_y^{\mathrm{T}} \phi(p_i, p_j)\}_{m \times m}, \quad (15)$$

$$\mathbf{k}_{10} = \{D_y \phi(p_i, p_j)\}_{m \times m}, \quad \mathbf{k}_{11} = \{[D_y^2 \phi(p_i, p_j)]^{\mathrm{T}}\}_{m \times m}, \quad (16)$$

and

$$\mathbf{k}_{02} = \{p_i^{\mathrm{T}}\}_{m \times 1}, \quad \mathbf{k}_{03} = \{1\}_{m \times 1}, \quad \mathbf{k}_{12} = \{\mathbf{I}_3\}_{m \times 1}. \quad (17)$$

Appendix C: Mesh quality measures

Figure 17 gives the mesh quality of the mechanical part with respect to cell and facet shapes, measured by dihedral angles of tetrahedra and the ratios of smallest edge and cicumradius of tetrahedra and triangles. The ratios for tetrahedra are scaled by $\sqrt{6}/4$ such that a regular tetrahedron (with four equilateral triangular faces) has a ratio of 1. Similarly, the ratios for triangles are scaled by $\sqrt{3}/3$ such that an equilateral triangle has a ratio of 1.

The two meshes in Figure 17a and 17b are generated with the same parameters except the cell sizes. The meshing parameters (see Table 1 in Section 3.1) are $r_{\max} = 5$, $r_{\min} = 0.2$, $\theta_f = 10°$, $F_s = 22°$, $F_\delta = 0.1$, $C_s = 2$. The cell sizes are 5 and 1, respectively. The mesh quality has been optimized with techniques implemented in CGAL.

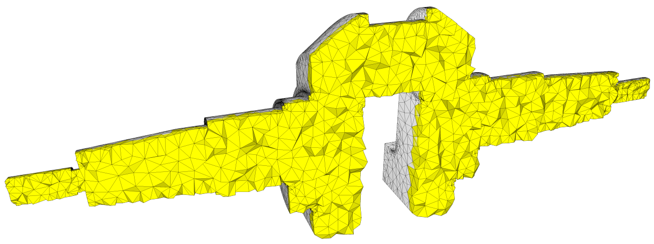
Appendix D: Extended applications

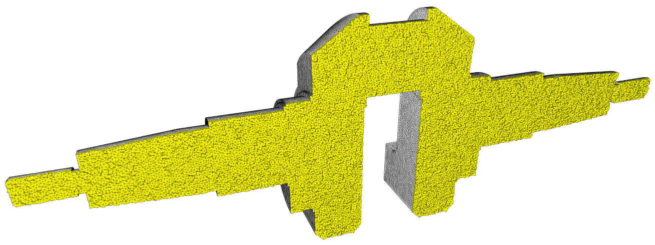
We extend our approach to Delaunay meshing of multiple domains. The multidomain meshing is a key preprocessing step for the numerical analysis of fluid dynamics and aerodynamics, which usually requires volume meshes of both the objects and the ambient environment (e.g. fluid and air). We consider the meshing of NURBS turbine models immersed in a spherical envelope as shown in Figure 18b. The Delaunay mesh in the multiple domains is depicted with different colors in Figure 18c. The mesh quality can be user-tuned in a flexible way to set different mesh criteria for each domain. In this example, cell sizes are uniform outside turbines, except the bounding sphere region of the turbine colored in yellow (see Figure 18c). The volume meshes of the turbines are shown in Figure 18d. Two turbines are meshed with different surface approximation errors and cell sizes. The meshes of different domains conform at their interfaces.

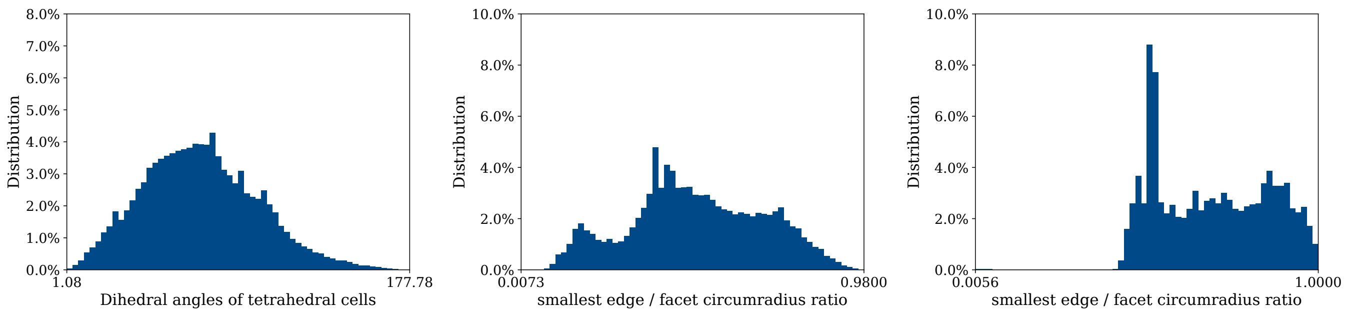
Another possible application of blending balls is to smooth out sharp features. The protecting balls can be switched to be blending ones such that the sharp features are approximated with smooth implicit surfaces bridging the two adjacent surfaces. Figure 19 shows the comparison of sharp features covered with protecting balls and blending balls, where the sharp features are preserved inside protecting balls and smoothed out inside blending balls.

Appendix E: Additional examples

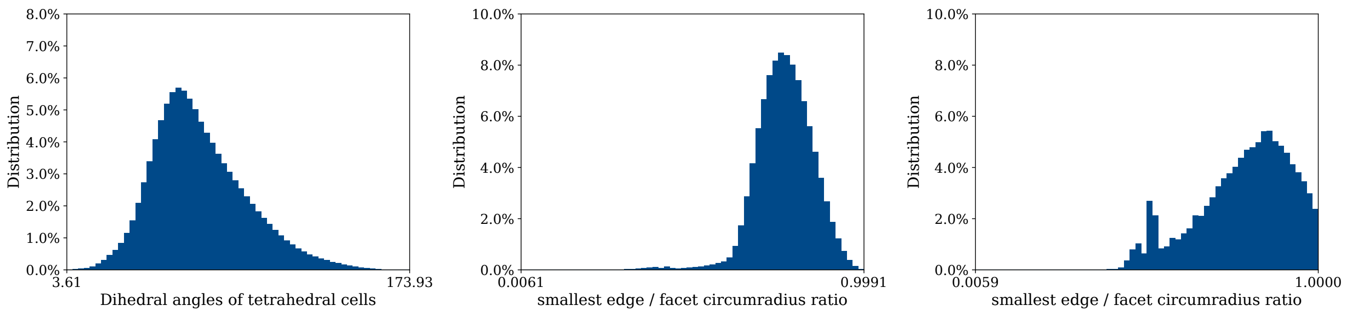
Figures 20 to 25 show the Delaunay meshes of STEP models in Visionair repository. Figure 26 is a complex mechanical part containing a number of small geometric details. Apart from trimming defects, the model also suffers from topological defects of non-closed or self-intersected trimming curves. We detected and solved these specific defects in the preprocessing of NURBS models by reconstructing piecewise linear trimming curves. Short sharp feature edges lead to locally dense mesh, and this is one thing that the algorithm is to be improved to consider a merging operation of short sharp edges.


(a) *Delaunay mesh with coarse cells.*


(b) *Delaunay mesh with dense cells.*

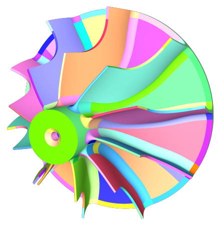


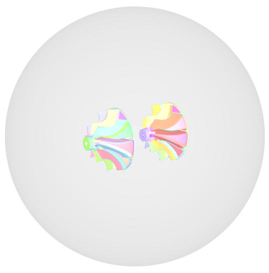
(c) *Mesh quality of mesh (a). The ratio of 1 represents a regular tetrahderon cell or an equilateral triangle facet.*

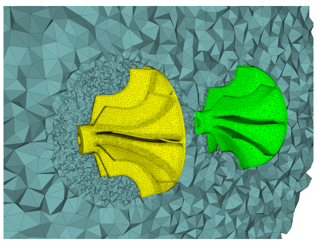


(d) *Mesh quality of mesh (b). The ratio of 1 represents a regular tetrahderon cell or an equilateral triangle facet.*

Figure 17: *Mesh qualities measured by cell and facet shapes.*


(a) *NURBS turbine model*


(b) *Spherical envelope*


(c) *Multidomain mesh*

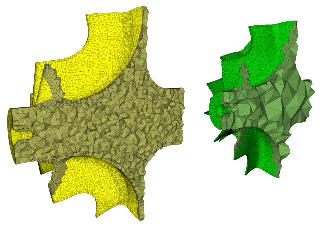

(d) *Turbine volume meshes with different mesh criteria*

Figure 18: *Multidomain meshing of turbines inside a spherical envelope.*

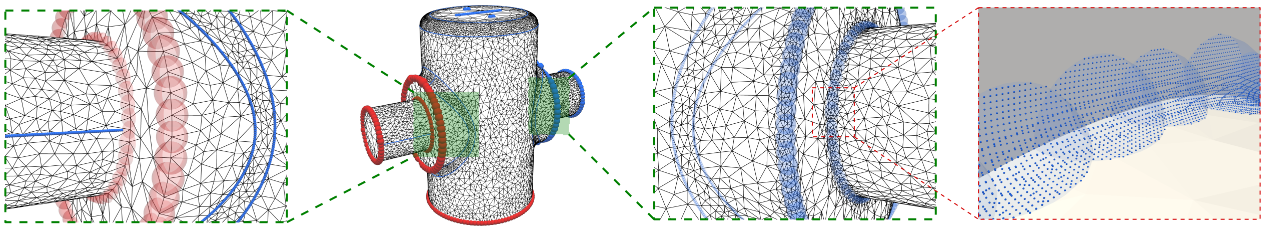


Figure 19: *Sharp features are smoothed out with blending balls. Left to right: 1) sharp features preserved with protecting balls; 2) protecting balls can be switched to blending balls to cover sharp features; 3) smoothed sharp features; 4) visualization of implicit surfaces by sampling inside blending balls.*

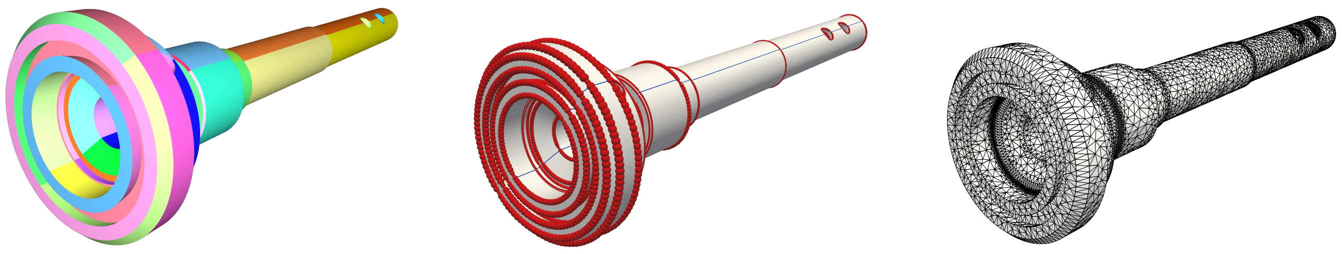


Figure 20: *Axle*

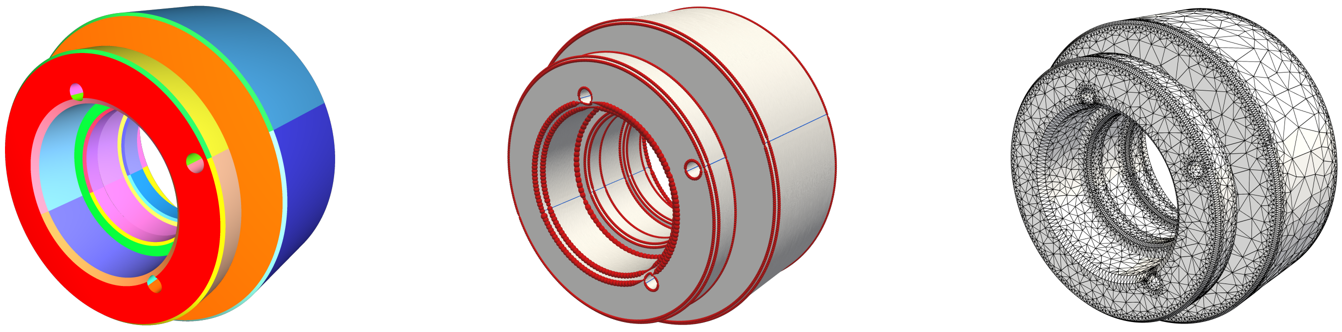


Figure 21: *CoolingDown*

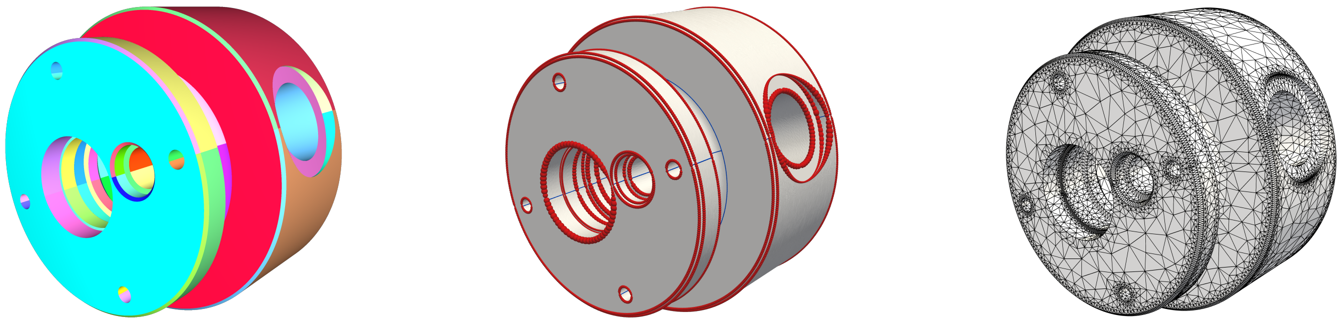


Figure 22: *CoolingUp*

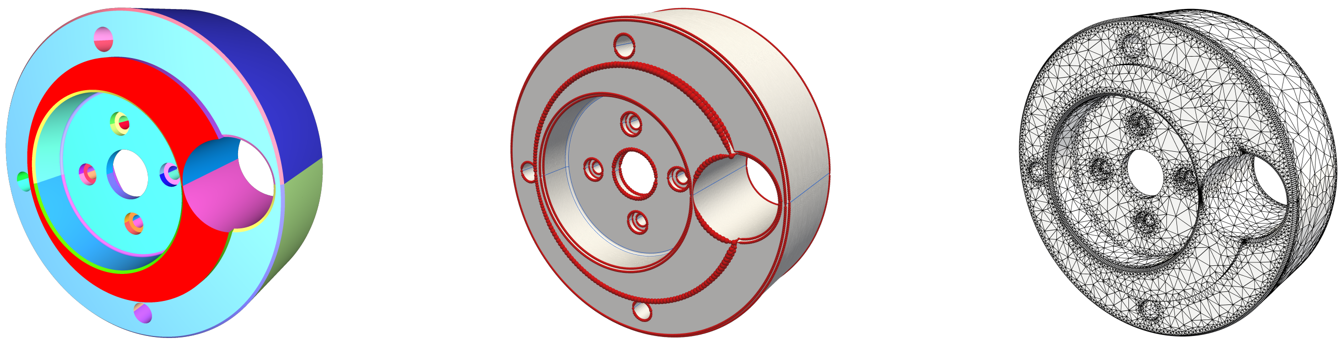


Figure 23: *CoolingDown Second*

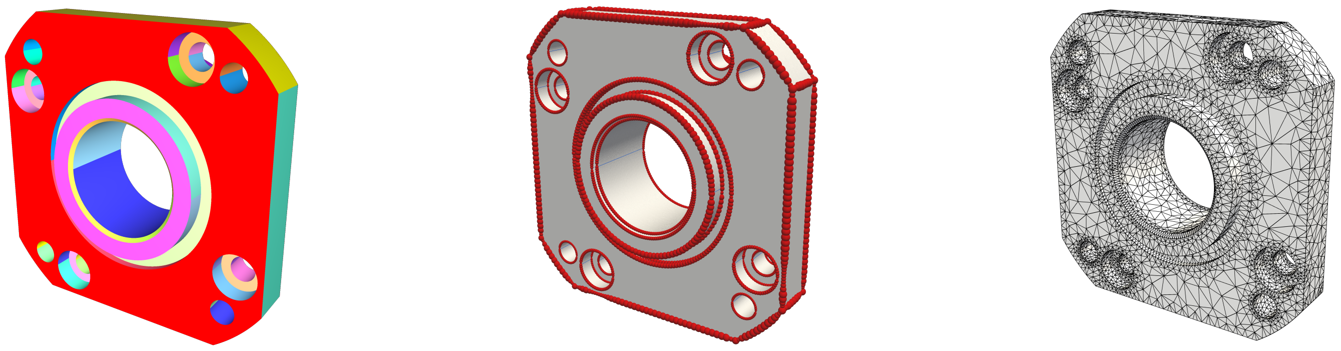


Figure 24: *CoverFront*

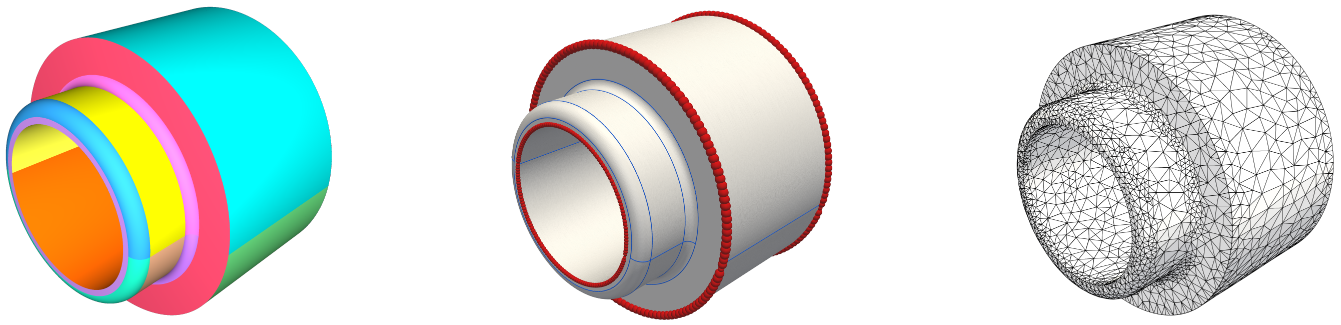


Figure 25: *Stator*

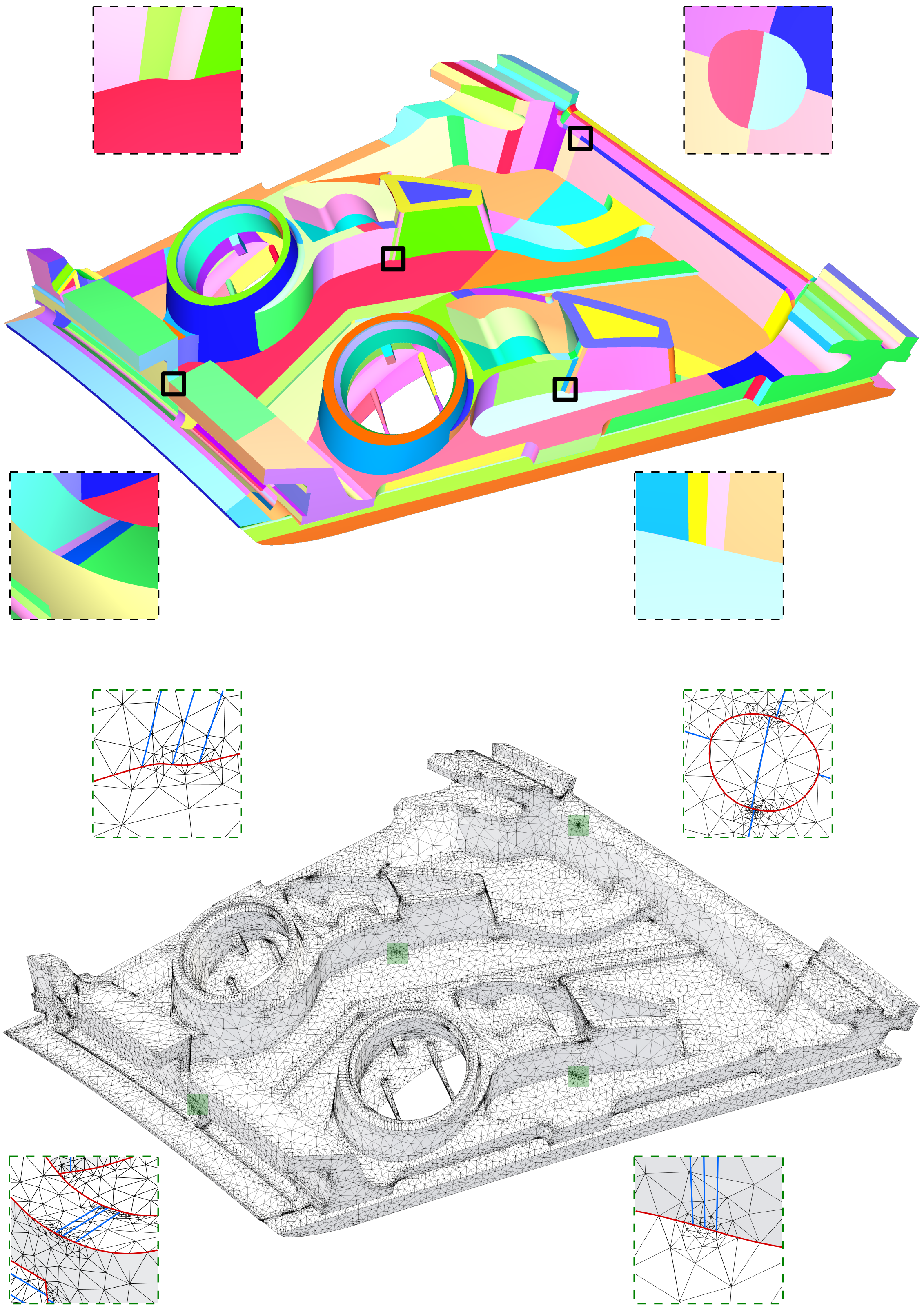


Figure 26: *A complex mechanical part (609 NURBS surfaces) and its Delaunay surface mesh. Apart from trimming defects, the model also suffers from topological defects of trimming curves, see Appendix D for more details. The locally dense mesh is due to short sharp feature curves, as shown in close-up views.*