

Efficient Offset Mesh Generation with Preserved Features and Boundaries

CANJIA HUANG, Xiamen University, China

JUAN CAO, Xiamen University, China

BAILIN DENG, Cardiff University, United Kingdom of Great Britain and Northern Ireland

JIANGBO HUANG, ZWSOFT CO., LTD., China

GE YIN, Sujing Interactive Technology CO., LTD., China

ZHONGGUI CHEN, Xiamen University, China

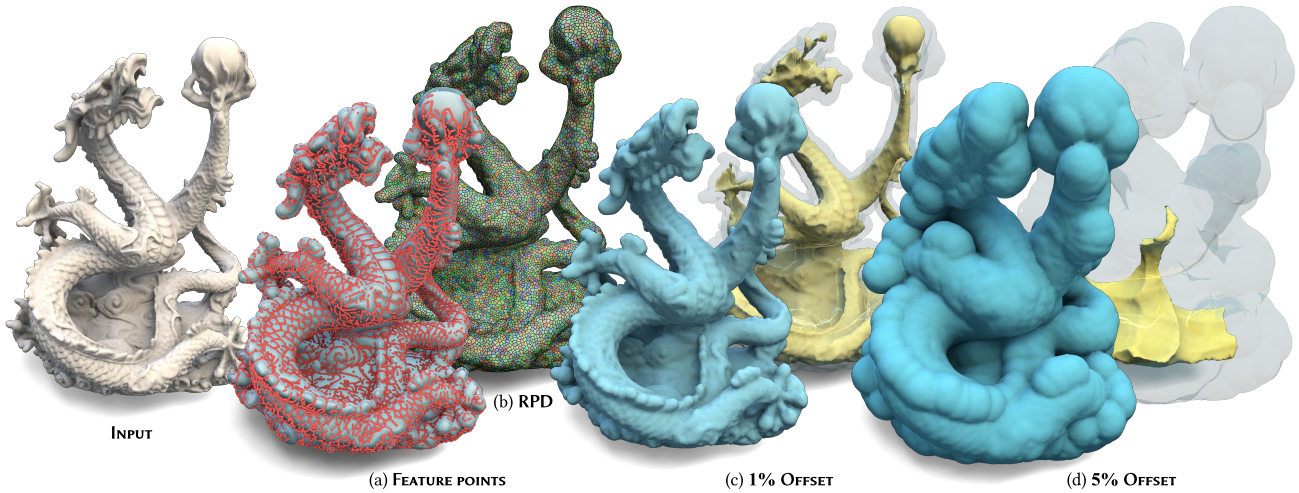


Fig. 1. This article presents an efficient point-based framework for generating offset meshes. Our method reconstructs predominantly topologically correct offset meshes that preserve features and optionally open boundaries by computing feature points on the offset surface (a) and directly constructing the restricted power diagram (RPD) on the offset surface (b). It supports simultaneous generation of outward and inward offsets at varying distances (c)–(d).

Mesh offsetting plays a pivotal role in CAD/CAM applications. This article introduces an efficient method for generating high-precision, predominantly topologically correct offset meshes from triangular meshes, employing direct offset and point representation. By offsetting sampled points of the input triangular mesh along multiple normals, we create

points on a superset of the offset surface. Subsequent feature enhancement captures and enhances features on the point set, facilitating the tailored restricted power diagram-based reconstruction of the offset surface and features. Benchmarking on public datasets, along with comprehensive comparisons with state-of-the-art methods, demonstrates the accuracy, effectiveness, and robustness of our approach. Experimental results showcase its capability to handle diverse models and support arbitrary offset distances, producing predominantly topologically correct offset meshes with small approximation errors while preserving sharp features and open boundaries—even in challenging regions such as thin plates and thin tubular structures.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**.

Additional Key Words and Phrases: Geometry processing, direct offsetting, point-based representation, feature-preserving, boundary-preserving

ACM Reference Format:

Canjia Huang, Juan Cao, Bailin Deng, Jiangbo Huang, Ge Yin, and Zhonggui Chen. 2027. Efficient Offset Mesh Generation with Preserved Features and Boundaries. *ACM Trans. Graph.* 46, 1, Article 10 (February 2027), 20 pages. <https://doi.org/10.1145/3832781>

1 Introduction

An offset operation expands or contracts an object by the same distance. When applied to a solid, an offset operation can be viewed as

The research was supported by National Natural Science Foundation of China (Nos. 62272402, 62372389), Natural Science Foundation of Fujian Province (No. 2024J01513243), and the Cardiff-Xiamen Collaboration Fund between Cardiff University and Xiamen University.

Authors' Contact Information: Canjia Huang, Xiamen University, Xiamen, Fujian, China; e-mail: huangcanjia0214@gmail.com; Juan Cao (corresponding author), Xiamen University, Xiamen, Fujian, China; e-mail: juancao@xmu.edu.cn; Bailin Deng, School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom of Great Britain and Northern Ireland; e-mail: DengB3@cardiff.ac.uk; Jiangbo Huang, ZWSOFT CO., LTD., Guangzhou, Guangdong, China; e-mail: huangjiangbo@zwcad.com; Ge Yin, Sujing Interactive Technology CO., LTD., Guangzhou, Guangdong, China; e-mail: yinge@sujingtech.com; Zhonggui Chen, Xiamen University, Xiamen, Fujian, China; e-mail: chenzhonggui@xmu.edu.cn.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2027 Copyright held by the owner/author(s).

ACM 0730-0301/2027/02-ART10

<https://doi.org/10.1145/3832781>

a specific instance of the Minkowski sum or difference [Rossignac and Requicha 1986], achieved by merging the solid with a sphere whose radius is the offset distance. This operation is widely used in various fields, including path planning [Farouki et al. 1995], collision detection [Teschner et al. 2005], and boundary layer meshing [Dyedov et al. 2009]. In contexts such as **numerical control (NC)** machining, offset compensation is crucial for various cutters to ensure a gouge-free tool path [Kim and Yang 2005]. Preserving the integrity of the offset boundary is also critical: in NC machining it prevents unnecessary compensation and shank–material interference [Hwang 1992], and in rapid prototyping it helps correct shape deviations caused by nonuniform shrinkage [Koc and Lee 2002; Qu and Stucker 2003].

Although the offset operation is conceptually straightforward, generating offset meshes with accurate geometry, high mesh quality, and reliable boundary behavior remains a significant challenge. The resulting shape depends intricately on both the input geometry and the offset distance, and different offsets of the same model can exhibit markedly different topology (e.g., Figure 1(c)–(d)), posing challenges for obtaining offset meshes that preserve features and maintain correct topology. As a result, existing offsetting techniques diverge widely in behavior and robustness. For example, the grid-based methods [Leung and Wang 2017; Wang 2018] are limited by voxel resolution, especially near sharp features and thin structures; adaptive refinement [Wang et al. 2024; Zint et al. 2023] alleviates this at the cost of substantial computation. Wrapping approaches [Portaneri et al. 2022] often fail to preserve sharp features, while multi-source distance-field methods [Wang et al. 2024] retain features but produce low-quality cut meshes. Methods based on Minkowski sums [Lien 2008] struggle with self-intersections, making them difficult to deploy robustly.

In this article, we propose a robust and efficient offset method capable of accurately handling arbitrary offset distances, preserving sharp features on the offset surface, maintaining predominantly correct topology—exhibiting geometric proximity and topological equivalence to the ground truth across most regions—and accurately preserving the open boundaries of the input model’s offset. Direct offset methods typically offset each simplex element of the input mesh, with intersections between offset surfaces of different elements generating sharp features. Resolving these self-intersections, however, is often computationally challenging and inefficient. In contrast, our approach uses a point-based representation of the offset surface, enabling efficient detection and trimming of self-intersection regions while preserving key features; see Figure 1(a). We further introduce a tailored **restricted power diagram (RPD)**-based reconstruction method that adaptively captures offset surface details and robustly computes the RPD directly on the offset surface; see Figure 1(b), to reconstruct an offset mesh with predominantly correct topology. The specific contributions of this paper are as follows:

- We employ a point-based representation to facilitate offset mesh generation, capturing the potentially complex topology of the offset surface without explicitly handling intersections among offset surfaces of different simplex elements. The vertices of the generated offset mesh lie precisely on the offset surface, ensuring small approximation errors in the resulting offsets.

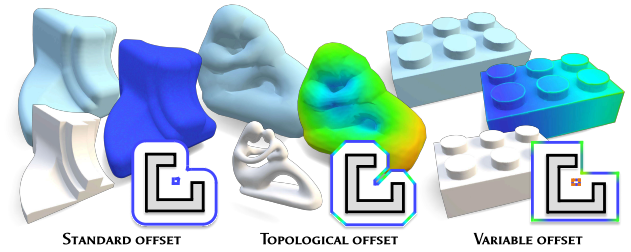


Fig. 2. Varieties of offset surfaces. From left to right (with bluer colors indicating smaller offset errors): the standard offset, where the distance from the offset surface to the input mesh is constant and user specified; the topological offset [Zint et al. 2025], which admits a continuous bijective mapping to the infinitesimal offset surface; and the variable offset [Cao et al. 2023], in which the offset distance varies across input facets to preserve features of the original model, rather than features of the offset surface.

- We introduce a projection-and-averaging algorithm to enhance features on offset points, placing feature points along lines and corners within all identified potential feature regions. Compared with conventional feature enhancement methods based on normal deviation, our algorithm more effectively handles feature corners and shallow features.
- We develop a feature-preserving surface reconstruction method for offset mesh generation, employing a topology-aware subdivision scheme to robustly compute the RPD on the implicit offset surface, enabling the extraction of a resultant mesh via a companion method that is homeomorphic to the offset surface. This approach, together with our point-based representation and feature enhancement, also allows rapid post-processing to preserve clean open boundaries, which is unachievable by existing methods.

The remainder of this paper is structured as follows: Section 2 reviews related work on offset operations; Section 3 presents the key insights and an algorithm overview; Section 4 details offset point generation; Section 5 describes feature enhancement; Section 6 introduces the RPD-based feature-preserving reconstruction; and Sections 7 and 8 present experiments, comparisons, and conclusions with future work.

2 Related Work

In this section, we review existing methods for computing offset surfaces, categorizing them based on their fundamental approaches. Several alternative offset operations have been proposed, such as non-manifold offset [Lee 1999, 2009] for sheet or wireframe models, topological offset [Zint et al. 2025] ensuring consistent topology with an infinitesimal offset, and variable offset [Cao et al. 2026, 2023; Sun et al. 2025; Zhao et al. 2026] allowing different distances for each part of the model. The distinctions among these varieties of offset methods are illustrated in Figure 2. Since these methods differ from the definitions used in our work and other comparative methods, we do not discuss them further.

Distance-field-based offset methods. The offset surface ensures that every point has the same minimal distance to the original mesh, making the conventional offsetting process equivalent to an isosurface extraction in the input model’s distance field. Several

implicit surface reconstruction algorithms have been developed to achieve this goal. Breen et al. [2000] converted CSG models into distance volumes and then extracted offset surfaces based on the distance values on a grid using the marching cube method. Due to grid resolution constraints, numerous redundant computations are introduced, particularly in non-feature regions, to ensure feature accuracy. To enhance computational efficiency, Varadhan and Manocha [2004] used an adaptive sub-division algorithm to generate a volumetric grid at an appropriate resolution, while Pavić and Kobbelt [2008] employed an octree to mitigate grid-related overhead. Additionally, Zint et al. [2023] and Wang et al. [2024] utilized an adaptive octree, selectively refining only in regions where the offset topology is complex. In contrast to volumetric methods, Portaneri et al. [2022] refined a coarse tetrahedral mesh to approximate the distance field of the input mesh; however, this wrapping approach may introduce significant errors in concave regions of the offset surface.

To preserve sharp features, Dai et al. [2024] enhanced the wrapping algorithm by directly inserting the feature skeleton into the initial coarse tetrahedral mesh. Liu and Wang [2011] employed a modified dual contouring algorithm to extract intersection-free offset surfaces from a narrow band distance field. Zint et al. [2023] utilized a remeshing algorithm to enhance geometric precision in features, while Wang et al. [2024] performed incremental half-plane cutting within the tetrahedral cells to construct the feature formation of multi-source distance fields generated from input triangles. Most methods based on conventional distance field definition either treat open boundaries as defective holes or produce hollow closed models [Liu and Wang 2011; Portaneri et al. 2022]. While Wang and Chen [2013] achieved thickening by adding cross-section regions and inner-shell regions in the distance field. In CAD/CAM applications that necessitate offsetting on zero-volume models while satisfactorily preserving boundaries, existing methods become inadequate.

Ray-based offset methods. The **ray-based representation (ray-rep)** of a shape is obtained by computing the intersection of a set of rays with the given shape, and it is stored using some ray-based data structures. Due to its straightforward computational nature and low storage cost, ray-rep is employed by some offset methods to compute and store results. Hartquist et al. [1999] pointed out that ray-rep can simplify the implementation of offset, sweeps, and Minkowski operations. Chen et al. [2020] used a two-stage sweeping algorithm to compute surface offsets in the dixel data structure [Van Hook 1986] that avoids volumetric distance field computations. Methods based on ray-based data structures are generally simple to implement and highly parallel, making them well-suited for computation on GPUs. For example, Wang and Manocha [2013] used **Layered Depth Images (LDI)** [Shade et al. 1998] to organize the samples of the boundary into structured points and compute their offsets on GPUs. The previously mentioned ray-based data structures, which only store depth values without surface normals, may lead to unavoidable aliasing errors in regions surrounding sharp features. Inspired by LDI, Wang et al. [2010] introduced **Layered Depth-Normal Image (LDNI)**, which stores points with normals, facilitating an efficient transformation from the ray-rep into a polygonal mesh surface that preserves sharp

features. Chen and Wang [2011] constructed an LDNI point-based offset model that allows for efficient filtering of invalid points. Ray-based offset methods yield voxel representations, necessitating an additional step for conversion into other representations, such as B-rep and mesh surfaces, to facilitate downstream applications.

Direct offset methods. Direct offset methods directly dilate each component of the input geometry to obtain an offset result [Kim and Yang 2004], or compute the Minkowski sum of the input and a sphere with a radius equal to the offset distance [Hachenberger 2007]. A challenging aspect involves removing self-intersections and invalid segments situated within the region bounded by the offset in the offset superset. Wang [1996] presented a method to calculate intersection curves of offsets of two parametric surfaces, but it can deal with global self-intersections only. Seong et al. [2006] identified both local and global self-intersections by computing the zero-set of polynomial equations for the offset surface. These regions were subsequently trimmed by projecting the zero-set into an appropriate parameter space. Jung et al. [2004] and Zaharescu et al. [2011] considered removing self-intersections from the raw offset triangular mesh and then stitching the remaining triangles into a manifold mesh. Campen and Kobbelt [2010a, b] introduced a technique for the robust extraction of a polygonal mesh representing the outer boundary of the offset superset. The method ensures exact computations at self-intersections; however, maintaining mesh quality near intersection regions remains a challenge. Despite attempts to minimize unnecessary computations, these methods still demand a relatively prolonged computing time.

Point-based offset methods. Point-based offset methods represent offset surfaces using discrete points, avoiding the initial meshing step and thus simplifying computations compared to traditional offset and trim self-intersection approaches, which are often computationally expensive. For example, point-based representations enable efficient filtering of invalid points as an alternative to trimming [Lien 2007, 2008]. Calderon and Boubekeur [2014] further introduced a morphological analysis framework for point clouds, demonstrating the advantages of point-based over voxel-based methods in feature representation. However, such methods may lose small details and sharp features. To mitigate this, Adams and Dutré [2003] resampled intersections to better capture sharp edges, while Chen et al. [2005b, a] generated intersection points where normals were inconsistent. Jun et al. [2002] and Kim et al. [2005] proposed curve-based scanning to detect feature points at junctions of adjacent curve segments. Meng et al. [2018] optimized movable points toward features, and Cai et al. [2025, 2024] respectively established frameworks for point cloud feature identification and enhancement based on winding numbers and normals. Nonetheless, converting unstructured point representations into meshes for downstream applications remains challenging, as critical surface information can be lost, hindering accurate topology and feature preservation. Ye et al. [2024] achieved watertight and feature-preserving reconstructions based on optimal transport. Zhao et al. [2026] employed power diagrams to approximate variable-radius offset surfaces, but this requires dense point sampling and incurs high computational cost. This work addresses these issues by introducing an efficient method to reconstruct high-quality meshes from point-based offset

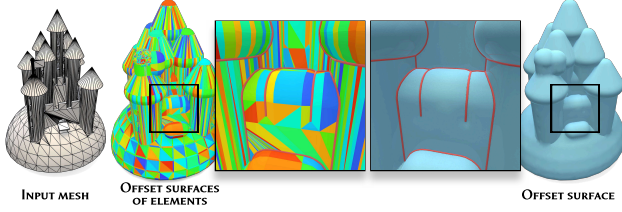


Fig. 3. The offset surface of the input mesh is the union of trimmed offset surfaces from its simplex elements (distinct colors). Features (with sharp ones marked as red curves) arise from intersections between these element offsets. Direct construction produces many intricate intersections; after tessellation, 936,102 of 945,210 triangles intersect.

representations, ensuring faithful preservation of essential geometric details and features.

3 Insight

The offset of a surface $\mathcal{S}$ can be defined as the boundary of its Minkowski sum (or difference) with a ball whose radius equals the offset distance d , or equivalently, as the isosurface of its unsigned distance field. Mathematically, it can be defined as

$$\mathcal{O}_d(\mathcal{S}) = \{\mathbf{p} \in \mathbb{R}^3 \mid \text{dist}(\mathbf{p}, \mathcal{S}) = d\}, \quad (1)$$

where $\text{dist}(\mathbf{p}, \mathcal{S})$ is the unsigned distance from the point $\mathbf{p}$ to the surface $\mathcal{S}$. For a polygonal mesh $\mathcal{M}$, the offset surface $\mathcal{O}_d(\mathcal{M})$ is the union of different parts generated by its simplex elements (i.e., vertices, edges, and facets), as illustrated in Figure 3. The part $\tilde{\mathcal{O}}_d(\tau)$ for a simplex element τ is a subset of the offset surface $\mathcal{O}_d(\tau)$ for τ , because a point on $\mathcal{O}_d(\tau)$ may have a distance shorter than d to other directly incident mesh elements and thus do not belong to the overall offset surface $\mathcal{O}_d(\mathcal{M})$; see Figure 4 for an example. In the following, we call $\tilde{\mathcal{O}}_d(\tau)$ the *trimmed offset surface* for τ . Directly computing the subset of $\mathcal{O}_d(\tau)$ that belongs to $\mathcal{O}_d(\mathcal{M})$ is challenging due to potential global intersections with offset surfaces of other elements. Representing each trimmed offset surface explicitly requires Boolean operations between surfaces and half-spaces. Converting these intersecting surfaces into a clean, application-ready mesh also requires computationally intensive steps such as mesh repair, Boolean operations, volume meshing, and removal of internal artifacts [Guo and Fu 2024; Levy 2025], since each element produces its own trimmed offset surface, greatly increasing the overall complexity (Figure 3).

To address these challenges, we propose a point-based framework rooted in the concept of direct offset that efficiently represents and processes the numerous self-intersections arising from offsets of distinct elements. Since features on the offset surface originate from these intersections (Figure 3), we incorporate a feature recognition and enhancement strategy to preserve them. Our method accepts any polygonal mesh as input, along with a user-defined unsigned offset distance, without assumptions on manifoldness, self-intersections, or mesh quality (even a triangle soup; see Figure 5). The output is a 2-pseudo-manifold triangular mesh that faithfully captures the offset surface while preserving sharp features. The approach consists of three main stages:

- (1) **Offset points generation:** Offset points are generated for each simplex element to represent its trimmed offset surface.

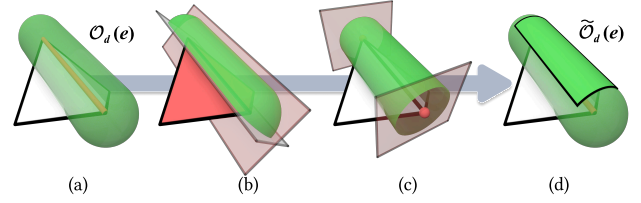


Fig. 4. The trimmed offset surface of an edge e in a triangle mesh. (a) The offset surface at distance d from an isolated edge $\mathcal{O}_d(e)$ forms a capsule-shaped surface (green); (b,c) portions of this surface contain points whose distance to other mesh elements is less than d and are therefore trimmed by planes (red, illustrative) defined by directly incident elements (two facets and two vertices, highlighted in red); and (d) the resulting edge trimmed offset surface $\tilde{\mathcal{O}}_d(e)$ after removing the trimmed regions.

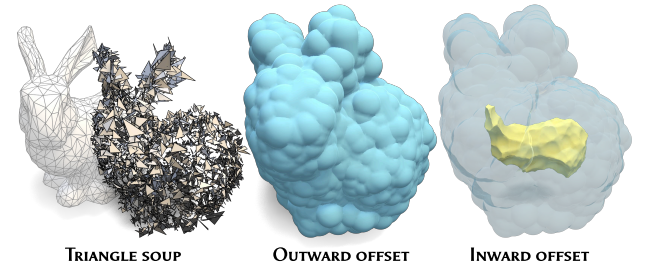


Fig. 5. Outward and inward offset results at a distance of 5% of the diagonal length on the triangle soup generated from the Bunny model.

The union of these points comprehensively covers the input mesh's offset surface (Figure 6(a)).

- (2) **Feature enhancement:** Potential feature regions are identified, and feature points are emphasized to preserve sharp details (Figure 6(b)).
- (3) **Offset mesh reconstruction:** A tailored RPD-based algorithm connects the offset point set into a mesh, ensuring accurate geometry and preservation of sharp features (Figure 6(c)), and the open boundaries of the input mesh can optionally be recovered.

Figure 6 illustrates the pipeline of our method, with detailed explanations of each stage provided in the following sections.

4 Offset Points Generation

In this section, we present a strategy to represent the offset surface of an input mesh $\mathcal{M}$ using a point set that provides a comprehensive sampling of the offset surface. Following the definition in [Lien 2007], a point set $\mathcal{P}$ is an r -covering of a surface $\mathcal{S}$ if for every point $\mathbf{s} \in \mathcal{S}$ there exists a point $\mathbf{p} \in \mathcal{P}$ such that $\|\mathbf{s} - \mathbf{p}\|_2 < r$. Our goal is to construct such an r -covering of the offset surface by generating points on the trimmed offset surfaces of the mesh elements.

For smooth surfaces, offset points are obtained by displacing surface samples by a distance d along the local normal. For piecewise-linear polygonal meshes, however, normals are constant only within facets and undefined along edges and at vertices, requiring special treatment. In the following, we consider the offset point computation for an unsigned offset distance d for different types of simplex elements.

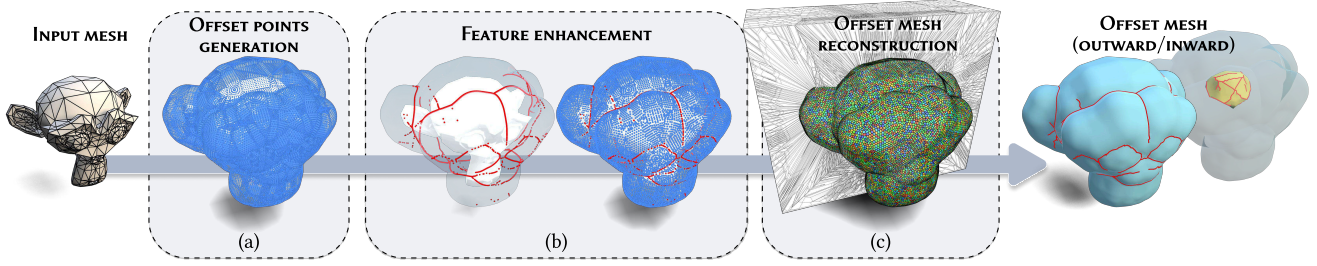


Fig. 6. Algorithm pipeline. (a) Starting from an input polygonal mesh, offset points (blue) are generated from each simplex element to adequately cover the offset surface; (b) feature points (red) are generated along feature lines and corners within identified potential feature regions, followed by optimization of the point distribution; and (c) the RPD is computed on the implicit offset surface, and a tailored mesh extraction algorithm is applied to obtain the final outward (blue) and inward (yellow) offset meshes.

For each facet f , to achieve shape-adaptivity and mitigate over-sampling on poorly shaped triangles, we construct a set of concentric similar triangles with vertex displacements bounded by r , sample their edges with spacing r , and offset each sample by $\pm d$ along the normal, yielding a locally r -covering set of offset points (Figure 7(a)).

For each edge e , its trimmed offset surface corresponds to a cylindrical segment. Since the normal is not uniquely defined along e , we adopt a “one point, multiple normals” (1vN) strategy. Specifically, for a boundary edge, multiple normals are assigned to each point on e by interpolating between $\mathbf{n}_f$ and $-\mathbf{n}_f$ with uniformly spaced angular intervals θ . The interpolation is performed via **spherical linear interpolation (Slerp)** [Shoemake 1985] around the axis aligned with e , with the interval set to $\theta = 2 \arcsin(\min\{\frac{r}{2d}, 1\})$, ensuring that adjacent offset points are spaced by r . The edge e is then uniformly sampled at spacing r , and each sample is offset by a distance d along all interpolated normals to generate the edge’s offset samples, as shown in Figure 7(b). For a 2-manifold interior edge in $\mathcal{M}$, its trimmed offset surface constitutes only a segment of the semi-cylindrical surface (rightmost in Figure 7(b)). In this case, the multiple normals are interpolated only between the normals $\mathbf{n}_{f_1}$ and $\mathbf{n}_{f_2}$ of the two adjacent facets.

Similarly, for each vertex v , multiple normals are assigned and the vertex is offset along these normals to generate sampling points on its trimmed offset surface. For a boundary vertex, Slerp interpolation is first performed between the vectors perpendicular to adjacent edges e_1 and e_2 in the facet plane, using uniform angular intervals θ . The resulting interpolated vectors serve as axes for a second Slerp between $\mathbf{n}_f$ and $-\mathbf{n}_f$, producing the set of normals at v ; see Figure 7(c). For a 2-manifold interior vertex in $\mathcal{M}$, when the incident edges have dihedral angles that are either all greater than π or all less than π (i.e., all incident edges are either all concave or all convex with respect to the orientation of $\mathcal{M}$), the vertex normals are obtained by interpolating between the multiple normals of its incident edges and the vertex’s initial normal, computed as the average of the incident edge multiple normals, as shown in the rightmost of Figure 7(c).

For non-manifold edges and vertices, as well as vertices whose incident edges have dihedral angles that are neither all greater than π nor all less than π , we duplicate them on adjacent triangles and offset them as boundary elements to ensure sufficient sampling thereon (Figure 8). The above sampling strategy produces an

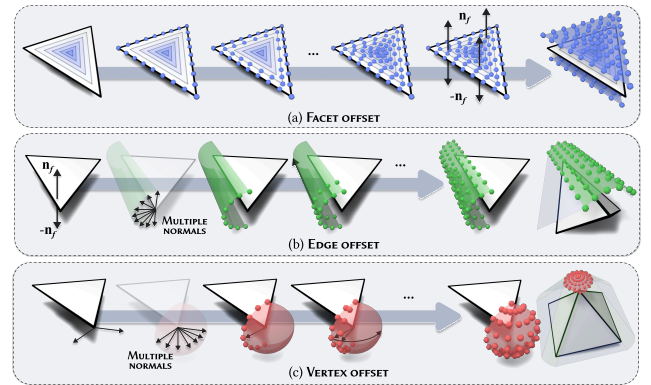


Fig. 7. Generation of offset points. (a-c) demonstrate the r -covering sampling strategy for trimmed offset surface of a facet, an edge, and a vertex, respectively.

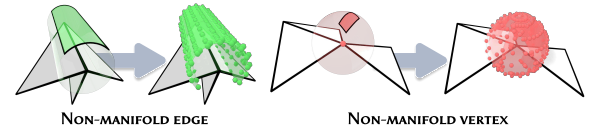


Fig. 8. Offset sampling for non-manifold edges and vertices. Duplicating and offsetting these elements yields sufficiently dense coverage of their trimmed offset surfaces.

r -covering for each element’s trimmed offset surface; consequently, the union of all samples $\mathcal{P}$ forms an r -covering of the complete offset surface (see Section A of the Supplementary Material for proof). The samples preserve the local geometry of their originating elements—planar for facets, cylindrical for edges, and spherical for vertices.

5 Feature Enhancement

The approach presented in Section 4 produces a set $\mathcal{P}$ of offset points that provide an r -covering of the offset surface. However, there might be an insufficient number of points in $\mathcal{P}$ that exactly lie on the features of the offset surface, thereby hindering subsequent explicit reconstruction from recovering these features. Therefore, to enhance features in the reconstructed offset surface, we detect regions that are likely to contain features (Section 5.1), and place

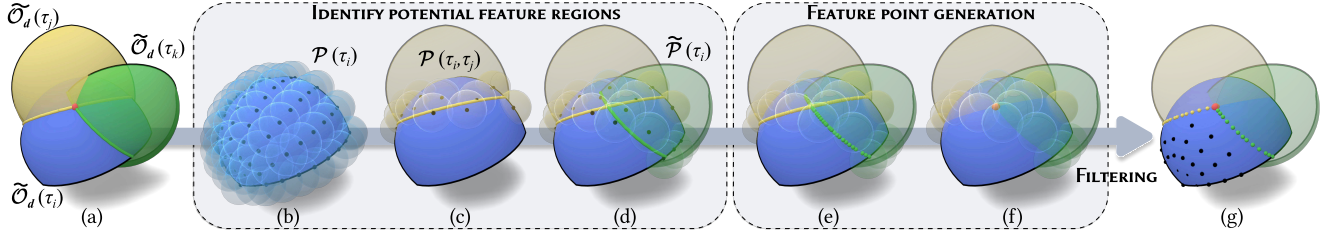


Fig. 9. Feature enhancement pipeline. (a) The trimmed offset surface $\tilde{\mathcal{O}}_d(\tau_i)$ (blue) intersects two other trimmed offset surfaces, producing feature lines and feature corners on $\tilde{\mathcal{O}}_d(\tau_i)$; (b) the point set $\mathcal{P}(\tau_i)$ r -covers $\tilde{\mathcal{O}}_d(\tau_i)$; (c) points whose r -balls intersect another trimmed offset surface $\tilde{\mathcal{O}}_d(\tau_j)$ (yellow) form the subset $\mathcal{P}(\tau_i, \tau_j)$; (d) all points whose r -balls intersecting other trimmed offset surface constitutes $\tilde{\mathcal{P}}(\tau_i)$; (e, f) feature points are generated along feature lines and at feature corners, corresponding to the identified potential feature lines (d) and their intersections, respectively; and (g) points not lying on the final offset surface are filtered out, and the distribution of offset points is improved to facilitate subsequent mesh reconstruction.

additional points along feature lines and corners (Section 5.2). The overall feature enhancement pipeline is illustrated in Figure 9.

5.1 Identify Potential Feature Regions

We first consider identifying regions on the trimmed offset surface of a single element that may contain features. The trimmed offset surface generated by each simplex element is smooth in its interior; therefore, feature lines and corners arise from intersections between trimmed offset surfaces of different elements (Figure 3). Although directly computing these intersections is difficult, our point-based representation efficiently localizes their regions, enabling feature point generation that guides subsequent reconstruction to preserve mesh features. Specifically, if the neighborhood of an offset point contains points generated from other elements, their corresponding offset surfaces likely intersect. These neighborhoods are thus identified as the potential feature regions.

For each simplex element τ_i (facet, edge, or vertex) of the input mesh, the generated offset points $\mathcal{P}(\tau_i)$ form an r -covering of its corresponding trimmed offset surface $\tilde{\mathcal{O}}_d(\tau_i)$. According to our offset points generation strategy, for every $\mathbf{p} \in \mathcal{P}(\tau_i)$, the open ball $B(\mathbf{p}, r)$ centered at $\mathbf{p}$ with radius r satisfies $B(\mathbf{p}, r) \cap \tilde{\mathcal{O}}_d(\tau_i) \neq \emptyset$; see Figure 9(b). Moreover, the intersection between $\tilde{\mathcal{O}}_d(\tau_i)$ and the trimmed offset surface $\tilde{\mathcal{O}}_d(\tau_j)$ of another element τ_j is r -covered by a subset $\mathcal{P}(\tau_i, \tau_j) \subseteq \mathcal{P}(\tau_i)$ as illustrated in Figure 9(c), which can be defined as

$$\mathcal{P}(\tau_i, \tau_j) = \{\mathbf{p} \in \mathcal{P}(\tau_i) \mid B(\mathbf{p}, r) \cap \tilde{\mathcal{O}}_d(\tau_j) \neq \emptyset\}.$$

Since the trimmed offset surface $\tilde{\mathcal{O}}_d(\tau_i)$ may intersect with trimmed offset surfaces of other elements, we define the union point set:

$$\tilde{\mathcal{P}}(\tau_i) = \bigcup_{j \neq i} \mathcal{P}(\tau_i, \tau_j),$$

which is a subset of $\mathcal{P}(\tau_i)$ and r -covers the entire intersection region on $\tilde{\mathcal{O}}_d(\tau_i)$; see Figure 9(d). In other words, the union of the r -balls centered at points in $\tilde{\mathcal{P}}(\tau_i)$ covers the intersection region on $\tilde{\mathcal{O}}_d(\tau_i)$, referred to as the *feature region* of $\tilde{\mathcal{O}}_d(\tau_i)$. Directly extracting $\tilde{\mathcal{P}}(\tau_i)$ from $\mathcal{P}(\tau_i)$ is computationally expensive, as it requires traversing all trimmed offset surfaces that may intersect $\tilde{\mathcal{O}}_d(\tau_i)$. Instead, we first extract a superset $\tilde{\mathcal{P}}^+(\tau_i) \supseteq \tilde{\mathcal{P}}(\tau_i)$, which can be identified efficiently. The detailed procedure is described below.

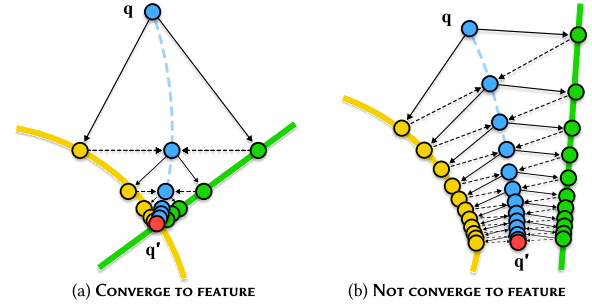


Fig. 10. Iterative projection (solid arrows) and averaging (dashed arrows) of an initial point $\mathbf{q}$ between two trimmed offset surfaces (yellow and green) either converges to a feature point $\mathbf{q}'$ at their intersection (a), or fails to do so (b).

For a point $\mathbf{p} \in \tilde{\mathcal{P}}(\tau_i)$, its r -ball intersects with one or more trimmed offset surfaces. According to Equation (1), a necessary condition for such an intersection is that the distance from the point $\mathbf{p}$ to the corresponding original element of the trimmed offset surface is less than $d + r$. Therefore, for each point $\mathbf{p} \in \mathcal{P}(\tau_i)$, we locate all facets, edges, and vertices of the input mesh within a distance of $d + r$. If any elements other than the one associated with $\tilde{\mathcal{O}}_d(\tau_i)$ are found, the r -ball of $\mathbf{p}$ may intersect with their trimmed offset surfaces, and we include $\mathbf{p}$ into a point set $\tilde{\mathcal{P}}^+(\tau_i)$. The union of the r -balls centered at points in $\tilde{\mathcal{P}}^+(\tau_i)$ defines the *potential feature region* of $\tilde{\mathcal{O}}_d(\tau_i)$, which fully contains the feature region, i.e., $\tilde{\mathcal{P}}(\tau_i) \subseteq \tilde{\mathcal{P}}^+(\tau_i) \subseteq \mathcal{P}(\tau_i)$, although some r -balls may not actually participate in intersections (see Figure 10(b)). These cases will be further analyzed in Section 5.2.1. Furthermore, if $\text{dist}(\mathbf{p}, \mathcal{M}) < d - r$, the r -ball centered at $\mathbf{p}$ lies entirely inside the space enclosed by the offset surface of the input mesh $\mathcal{M}$. In this case, identifying or generating feature points inside such an r -ball is unnecessary, as no intersections occur on the resulting offset mesh. This pre-filtering step effectively eliminates a large number of irrelevant offset points.

5.2 Feature Points Generation

Given the point set $\tilde{\mathcal{P}}^+(\tau_i)$ that r -covers the potential feature region of the trimmed offset surface $\tilde{\mathcal{O}}_d(\tau_i)$, we aim at generating feature points that accurately represent the features on $\tilde{\mathcal{O}}_d(\tau_i)$.

Directly computing intersections between $\tilde{\mathcal{O}}_d(\tau_i)$ and its neighboring trimmed offset surfaces is computationally expensive and numerically unstable. Instead, we traverse $\tilde{\mathcal{P}}^+(\tau_i)$ and apply an iterative averaged-projection algorithm, initialized from a point $\mathbf{q}$ near each $\mathbf{p} \in \tilde{\mathcal{P}}^+(\tau_i)$, which rapidly converges to the feature and enables accurate and robust feature extraction (Figure 9(e)–(f)).

5.2.1 Generation Along Feature Lines. Since feature points lie at the intersection of $\tilde{\mathcal{O}}_d(\tau_i)$ and $\tilde{\mathcal{O}}_d(\tau_j)$, the point $\mathbf{q}$ should be simultaneously close to both surfaces. We therefore iteratively update $\mathbf{q}$ by averaging its projections onto the two surfaces, denoted $\text{proj}_{\tilde{\mathcal{O}}_d(\tau_i)}\mathbf{q}$ and $\text{proj}_{\tilde{\mathcal{O}}_d(\tau_j)}\mathbf{q}$, respectively, as follows:

$$\mathbf{q}^{(k+1)} = \frac{1}{2} \left(\text{proj}_{\tilde{\mathcal{O}}_d(\tau_i)}\mathbf{q}^{(k)} + \text{proj}_{\tilde{\mathcal{O}}_d(\tau_j)}\mathbf{q}^{(k)} \right), \quad (2)$$

where $\mathbf{q}^{(k)}$ denotes the position of $\mathbf{q}$ after the k th iteration of the process. In this way, $\mathbf{q}$ rapidly converges to a point $\mathbf{q}'$, with the iteration terminating when $\|\mathbf{q}^{(k+1)} - \mathbf{q}^{(k)}\|_2$ falls below a sufficiently small threshold (set to 10^{-10} in our experiments). If $\mathbf{q}'$ lies on both trimmed offset surfaces, i.e., $\tilde{\mathcal{O}}_d(\tau_i)$ and $\tilde{\mathcal{O}}_d(\tau_j)$ intersect (Figure 10(a)), and the normal difference at $\mathbf{q}'$ on the two surfaces exceeds θ_ϵ (set to 8° by default), we accept $\mathbf{q}'$ as a feature point. Otherwise (Figure 10(b)), we discard $\mathbf{p}$ from $\tilde{\mathcal{P}}^+(\tau_i)$.

Projecting a point onto a trimmed offset surface might seem straightforward, as it essentially reduces to projecting onto the underlying geometry (a plane, a cylindrical surface, or a spherical surface corresponding respectively to the offset surface of a facet, an edge, or a vertex). However, the offset surface associated with each mesh element can be trimmed by the offset surfaces of adjacent elements (Figure 4). As a result, a naïve projection may place the projected point outside the valid portion of the trimmed offset surface, leading to inaccurate feature point localization. To guarantee a correct projection, we delineate for each mesh element a region within which projecting a point directly onto the underlying geometry ensures it properly lies on the corresponding trimmed offset surface. Following the definition of fundamental offset-geometry elements in [Chen et al. 2025], we introduce the associated *offset region* for each element below.

For a facet f of the polygonal mesh, its offset region is defined as

$$\begin{aligned} \mathcal{R}(f) = \{ \mathbf{p} \in \mathbb{R}^3 \mid & \\ & (\text{perp}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) - \mathbf{p}) \cdot (\text{perp}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) - \mathbf{v}_3) > 0, \\ & (\text{perp}(\mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_2) - \mathbf{p}) \cdot (\text{perp}(\mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_2) - \mathbf{v}_2) > 0, \\ & (\text{perp}(\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_1) - \mathbf{p}) \cdot (\text{perp}(\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_1) - \mathbf{v}_1) > 0 \}, \end{aligned}$$

where $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$ are the points corresponding to the vertices of facet f ; $\text{perp}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ computes the perpendicular foot for $\mathbf{v}_3$ on the line defined by $\mathbf{v}_1, \mathbf{v}_2$; see Figure 11(a). Thus, the offset region of a facet depends solely on itself, whereas the offset region of an edge e additionally involves its incident

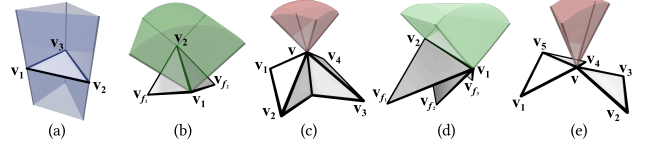
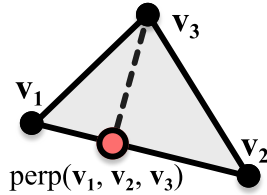


Fig. 11. Offset regions of different simplex elements in the input mesh. (a-e): facet, manifold edge, manifold vertex, non-manifold edge, and non-manifold vertex, respectively.

facets and is defined as

$$\begin{aligned} \mathcal{R}(e) = \{ \mathbf{p} \in \mathbb{R}^3 \mid & \\ & (\mathbf{p} - \mathbf{v}_1) \cdot (\mathbf{v}_2 - \mathbf{v}_1) > 0, \\ & (\mathbf{p} - \mathbf{v}_2) \cdot (\mathbf{v}_1 - \mathbf{v}_2) > 0, \\ & (\mathbf{p} - \text{perp}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_{f_i})) \cdot (\text{perp}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_{f_i}) - \mathbf{v}_{f_i}) \geq 0, \\ & i = 1, \dots, n \}, \end{aligned}$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the points corresponding to the endpoints of edge e , $f_i (i = 1, \dots, n)$ are the facets incident to e , and $\mathbf{v}_{f_i}$ is the point corresponding to the vertex of facet f_i opposite to e ; see Figure 11(b). For a vertex v , the offset region is defined with respect to all its incident facets:

$$\mathcal{R}(v) = \{ \mathbf{p} \in \mathbb{R}^3 \mid (\mathbf{p} - \mathbf{v})(\mathbf{v} - \mathbf{v}_i) \geq 0, i = 1, \dots, n \},$$

where $\mathbf{v}_i (i = 1, \dots, n)$ are the points corresponding to all the vertices adjacent to the vertex v , with $\mathbf{v}$ being the point corresponding to v ; see Figure 11(c). Note that the above definitions impose no constraints on the relationships among these incident elements. Therefore, these definitions are applicable not only to elements on 2-manifold meshes but also extend to non-manifold meshes and meshes with open boundaries, as exemplified in Figure 11(d)–(e).

When $\mathbf{q}$ lies inside the offset region of an element, it can be directly projected onto the underlying geometry to obtain a point on the trimmed offset surface. To ensure that the averaged-projection process in Equation (2) converges to the correct features, we restrict $\mathbf{q}$ to the active space $\mathcal{R}(\tau_i) \cap \mathcal{R}(\tau_j)$, where $\mathcal{R}(\tau_i)$ and $\mathcal{R}(\tau_j)$ denote the offset regions of the two elements. Since an offset region is an intersection of half-spaces, point inclusion is easy to test; moreover, as both offset regions are convex, a line search ensures that all projections remain within this domain.

Note that the final position of a converged point depends on its initial location. To ensure adequate coverage of the target features, we initialize by randomly sampling $k = 4$ spatial points $\mathbf{q}$ within the convex space $\mathcal{R}(\tau_i) \cap \mathcal{R}(\tau_j)$.

5.2.2 Generation at Feature Corners. Each feature corner is uniquely determined by the intersection of three trimmed offset surfaces. To compute these points, for each point $\mathbf{p} \in \tilde{\mathcal{P}}^+(\tau_i)$, two neighboring points $\mathbf{p}_j$ and $\mathbf{p}_k$ belonging to distinct elements are selected from $\tilde{\mathcal{P}}^+(\tau_i)$, with their corresponding trimmed offset surfaces $\tilde{\mathcal{O}}_d(\tau_j)$ and $\tilde{\mathcal{O}}_d(\tau_k)$, which potentially intersect. A similar iterative procedure is then applied to generate the feature point at the corner of $\tilde{\mathcal{O}}_d(\tau_i)$, $\tilde{\mathcal{O}}_d(\tau_j)$, and $\tilde{\mathcal{O}}_d(\tau_k)$, updating the spatial point $\mathbf{q}$ (initialized as $\mathbf{p}$) as

$$\mathbf{q}^{(k+1)} = \frac{1}{3} \left(\text{proj}_{\tilde{\mathcal{O}}_d(\tau_i)}\mathbf{q}^{(k)} + \text{proj}_{\tilde{\mathcal{O}}_d(\tau_j)}\mathbf{q}^{(k)} + \text{proj}_{\tilde{\mathcal{O}}_d(\tau_k)}\mathbf{q}^{(k)} \right).$$

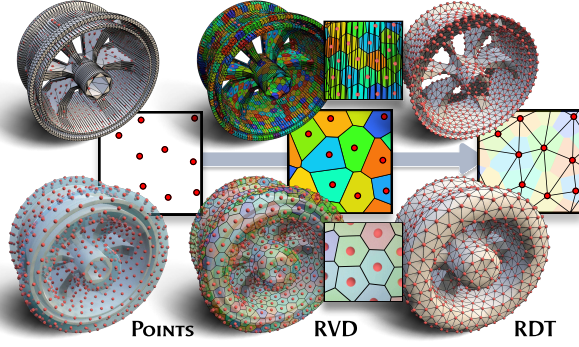


Fig. 12. Mesh reconstruction/remeshing via RVD. Top: the RVD and its dual RDT are computed directly on the mesh by clipping triangles with the half-spaces defining the Voronoi diagram. Bottom: on the offset surface, a mature and robust scheme for computing the RVD and extracting the corresponding RDT is still lacking.

Finally, the converged point $\mathbf{q}'$ is checked to determine whether it lies simultaneously on all three surfaces, confirming whether it represents the desired feature corner.

5.3 Offset Point Filtering

Up to this stage, we obtain a set of points $\mathcal{P}$ that initially lie on the trimmed offset surfaces of their respective elements. However, global self-intersections may cause a significant portion to deviate from the true offset surface of the input mesh. To facilitate the subsequent generation of the final offset mesh, these redundant points are removed. Specifically, we construct an AABB tree of the input mesh to perform fast distance query from each offset point to the mesh. Offset points whose distance to the input mesh is less than $d - \epsilon$ are discarded, where $\epsilon = 10^{-10}$ is a tolerance. To avoid skinny triangles, points that are closer to other points than $0.1r$ are further removed. Within each local region, retained points are prioritized in the order of feature corners, feature lines, and non-feature points. Following [Xu et al. 2022], non-feature points within the r -neighborhood of feature points are additionally swept to enhance connectivity. During filtering, a precomputed kd-tree of offset points is used to accelerate neighborhood queries. Figure 9(e) shows the results of offset point filtering.

6 Offset Mesh Reconstruction

With the well-distributed and feature-aware offset point set generated in the previous section, we reconstruct a triangular mesh using the RPD [Basselin et al. 2021]. The **restricted Voronoi diagram (RVD)** [Edelsbrunner and Shah 1994] and RPD are widely used in mesh reconstruction [Boltcheva and Lévy 2017; Wang et al. 2022; Xu et al. 2022] and remeshing [Pellerin et al. 2014; Wen et al. 2025; Yan et al. 2014, 2009]. They are defined as the intersection of the three-dimensional Voronoi or power diagram of a point set with an input surface, from which a triangulation can be extracted via the dual structure: the **restricted Delaunay triangulation (RDT)** or **restricted regular triangulation (RRT)**, as shown in Figure 12. To ensure topologically correct triangulations, the RVD must satisfy the *Topological Ball Property* [Edelsbrunner and Shah 1994], i.e., each k -dimensional Voronoi face intersecting the surface $\mathcal{S}$ forms

a $(k - 1)$ -dimensional topological ball. Although extremely dense sampling guarantees this property, it is impractical, and existing methods for explicit surfaces often fail in thin or sharp-feature regions, leading to incorrect topology.

In our setting, the RPD must be computed for points on an implicit offset surface. Prior approaches, including heuristic point insertion in SDF-RVD [Hou et al. 2022] and Voronoi cell classification [Liu et al. 2025], are not robust across diverse geometric conditions and often fail near sharp features. Accurately computing the RVD/RPD and extracting a topologically correct mesh thus remains an open challenge. To achieve robust and reliable offset surface reconstruction, we propose a tailored method for computing the RPD on the offset surface (Section 6.1) and a complementary technique to extract the resulting mesh from the computed RPD in such challenging cases (Section 6.2).

6.1 RPD Computation on Offset Surface

In this section, we first describe the computation of the RVD of the offset point set $\mathcal{P}$ on the offset surface. The Voronoi diagram of $\mathcal{P}$, computed via the duality between DT and VD, comprises a set of Voronoi cells, each corresponding to an offset point. The RVD computation can be decomposed into determining the intersection of each Voronoi cell with the offset surface:

$$\mathbf{Vor}(\mathbf{p})|_{\mathcal{O}_d(\mathcal{M})} := \mathbf{Vor}(\mathbf{p}) \cap \mathcal{O}_d(\mathcal{M}),$$

where $\mathbf{Vor}(\mathbf{p})$ denotes the bounded Voronoi cell of $\mathbf{p} \in \mathcal{P}$, a convex polytope restricted within a large bounding box of the offset surface. Unlike the explicit case, the **restricted Voronoi cell (RVC)** cannot be obtained by simply clipping the offset surface with the cell's bounding planes [Yan et al. 2009], as the offset surface is implicitly defined. Leveraging the convexity of Voronoi cells, each cell can be decomposed into a set of tetrahedra. We first determine the intersection of an individual tetrahedron with the offset surface—a process further reducible to edge and facet intersections. Finally, the results from all constituent tetrahedra are aggregated to determine the intersection between the Voronoi cell and the offset surface.

6.1.1 Edge/Offset Surface Intersection. Based on the sphere tracing algorithm [Hart 1996], we compute the intersections between an edge (treated as a spatial line segment) and the offset surface. The classical sphere tracing algorithm finds the first intersection between a ray and a distance field's zero isosurface by iteratively advancing from the ray's origin along the ray according to the local distance value; see Figure 13(a). In our case, an edge may intersect the offset surface multiple times, and the offset surface is not explicitly available. To resolve this, we reformulate the offset surface as the zero isosurface of the field $|\text{dist}(\mathbf{p}, \mathcal{M}) - d|$. Although this field differs from $\text{dist}(\mathbf{p}, \mathcal{O}_d(\mathcal{M}))$, they coincide on the zero set, and the inequality

$$|\text{dist}(\mathbf{p}, \mathcal{M}) - d| \leq \text{dist}(\mathbf{p}, \mathcal{O}_d(\mathcal{M})),$$

ensures that it behaves as a *conservative* distance function [Schott et al. 2025] (see Section A of the Supplementary Material for proof). Sphere tracing can therefore be applied reliably along the edge to detect intersections.

To obtain more than one intersection, note that while Hou et al. [2022] trace from both endpoints (Figure 13(c2)), their approach fails when more than two intersections are present. To ensure that

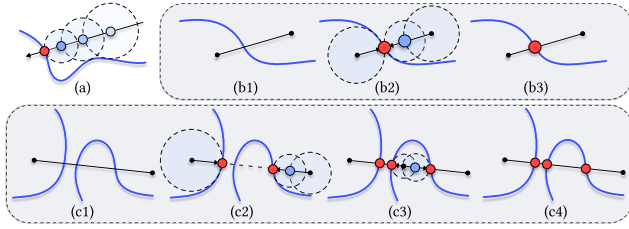


Fig. 13. Edge/Offset surface intersection computation (schematic). (a) The intersections (red) between a ray and the offset surface (blue curve) are computed using sphere tracing; (b1)–(b3) if the intersections traced inward from both edge endpoints coincide, the edge intersects the offset surface at that location; (c1)–(c4) otherwise, adaptive edge splitting is applied, and intersections on the sub-edges are recomputed iteratively until all intersections are found.

all intersections are identified, we adopt an adaptive edge-splitting strategy, which split the edge at the midpoint between two intersection points and performs sphere tracing from the midpoint toward both intersection points, see Figure 13(c3). This process is repeated iteratively, ensuring that each resulting sub-edge contains at most one intersection point. Intersection points closer than a small threshold are treated as a single intersection. During tracing, the cumulative traversal length is monitored, and the process terminates once it exceeds the edge length.

6.1.2 Facet/Offset Surface Intersection. We next address the computation of intersections between a triangle and the offset surface. We first apply the edge/offset surface intersection algorithm described above independently to each of its three edges to obtain the corresponding intersection points. A sufficiently small triangle is intersection-free if none of its edges intersect the offset surface. Otherwise, because the offset surface is watertight and predominantly 2-manifold in practical cases (even when the input is non-manifold or contains other defects), edge/offset-surface intersection points typically occur in pairs in non-degenerate configurations. If exactly two edges each contain a single intersection point, the intersection between the triangle and the offset surface is topologically a segment connecting these two points, as illustrated in Figure 14(a). Such triangles are referred to as *topologically determinable*.

If any edge contains more than one intersection point, the intersection topology becomes ambiguous due to multiple possible configurations, as shown in Figure 14(b). In this case, the edge/offset surface intersection algorithm recursively splits the edge into sub-edges and computes intersections until each sub-edge contains exactly one intersection point. This guarantees that the resulting sub-triangles have a well-defined topology; see Figure 14(b). This process resolves the intersection topology between the offset surface and the original triangle and produces a set of *cut segments* that are topologically equivalent to the true intersection.

6.1.3 Tetrahedron/Offset Surface Intersection. A tetrahedron intersects the offset surface if at least one of its edges intersects it. For such a tetrahedron, if all four of its triangular facets are either intersection-free or topologically determinable, a closed *cut loop* can be traced across the facets. This loop defines a *cut facet*, which can be triangular or quadrilateral, representing the intersection

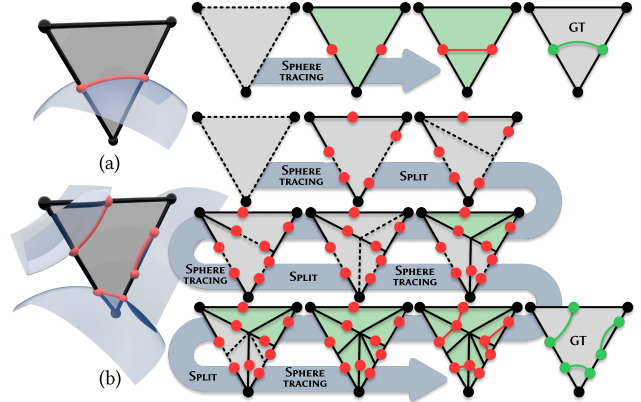


Fig. 14. Computation of facet/offset surface intersections. (a) A simple case where exactly two edges each intersect the offset surface at a single point; and (b) a more complex case resolved through iterative edge splitting and edge/offset surface intersection tests to determine the local intersection topology. Topologically determinable (sub-)triangles are shown in green. Black solid segments denote line segments processed by sphere tracing (Section 6.1.1), while dashed black segments indicate those not processed. Red vertices mark the computed intersection points, and red segments depict the resulting cut segments, which are topologically equivalent to the true intersection curves.

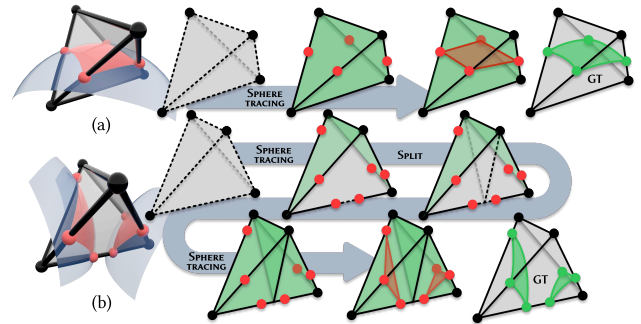


Fig. 15. Computation of tetrahedron/offset surface intersections. (a) A case where all four facets are topologically determined; and (b) a more complex case resolved through iterative splitting and edge/offset surface intersection tests to establish the local intersection topology. Topologically determinable (sub-)tetrahedra are highlighted in green. Red segments and surfaces denote the computed cut loops and cut facets, respectively. The remaining elements are consistent with those Figure 14.

between the tetrahedron and the offset surface (Figure 15(a)). Tetrahedra of this type are referred to as *topologically determinable*.

If any facet is not topologically determinable, the intersection topology becomes ambiguous. Following the facet/offset surface intersection procedure, the tetrahedron is recursively refined into sub-tetrahedra until all resulting pieces are topologically determinable (Figure 15(b)). The cut facets of all determinable sub-tetrahedra are then assembled to form the cut facet of the original tetrahedron. The resulting composite consists of triangular and quadrilateral patches whose boundary forms the cut loop and is topologically equivalent to the true intersection with the offset surface; see Figure 15.

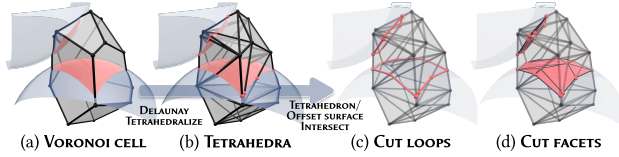


Fig. 16. Computation of Voronoi-cell/offset-surface intersections. (a) A Voronoi cell (a convex polytope) is first tessellated into tetrahedra (b). For each tetrahedron, we compute its intersection with the offset surface (Section 6.1.3). Combining these per-tetrahedron intersections yields the boundaries of the overall intersection, called cut loops (red segments in (c)), and their discretized approximation, the cut facets (red facets in (d)).

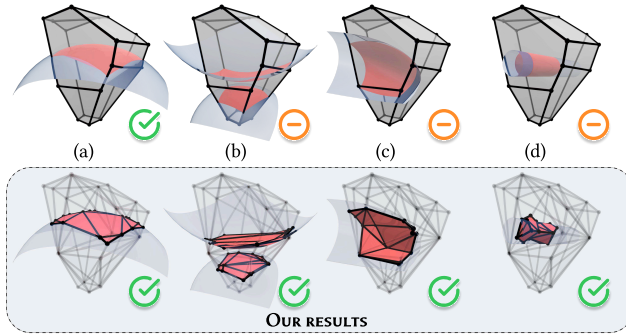


Fig. 17. Voronoi cells and intersections with the offset surface. (a) A cell clearly partitioned by intersections between Voronoi edges and the offset surface; (b)–(d) three representative ambiguous configurations where the Voronoi cell intersects a thin shell, a thin gap, and a thin tubular structure, respectively. Our method robustly resolves all cases and produces topologically correct intersections (red cut facets shown at the bottom).

6.1.4 Voronoi Cell/Offset Surface Intersection. To compute the intersections between a Voronoi cell and the offset surface, we decompose the cell into tetrahedra and apply the algorithm described in the previous section to each tetrahedron. Specifically, the Voronoi cell, as a convex polytope, is tetrahedralized using the Delaunay tetrahedralization of its vertices. To ensure continuity among the cut loops and facets generated within the cell, all tetrahedra sharing a split edge are subdivided simultaneously. After processing, each tetrahedron becomes topologically determinable, and aggregating their intersection results yields the topology of the intersection of the Voronoi cell and the surface.

The cut loops (each consisting of a sequence of connected segments arranged in an end-to-end manner) of the Voronoi cell can be obtained by tracing the cut segments on the boundary facets; see Figure 16. The union of cut facets from all tetrahedra provides a topologically consistent approximation of the RVC. These cut facets are grouped according to connectivity to form multiple connected components; for convenience, each connected component is hereafter referred to as an RVC of the Voronoi cell. Note that the number of cut loops does not necessarily equal the number of RVCs. This relationship will later assist in determining the correct topology of the local offset surface in Section 6.2.

6.1.5 Analysis in Challenging Regions. Figure 17(a) illustrates Voronoi cells in which intersections between Voronoi edges and the offset surface clearly delineate the cells. In such cases, the RVC can

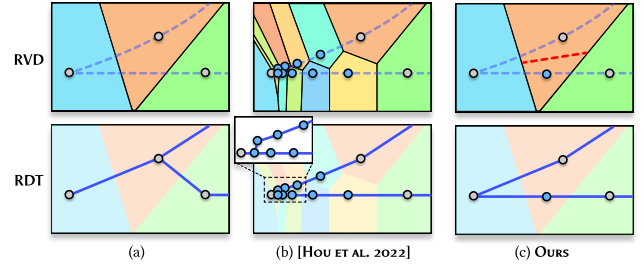
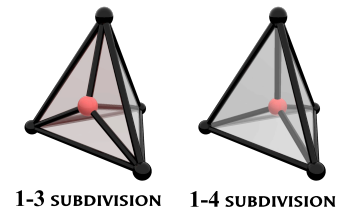


Fig. 18. Comparison of RVD-based methods near sharp features. (a) RVD-based methods often produce non-manifold structures near sharp features due to violations of the “one site, one region” property; (b) Hou et al. [2022] heuristically mitigate this issue by iteratively inserting additional points (blue), but termination is not guaranteed and non-manifold structures may still occur; in contrast, (c) our approach captures the intended topology within Voronoi cells in such challenging regions (cf. Figure 17(b) and (c)) and directly constructs correct results using a tailored mesh extraction scheme (detailed in Section 6.2).

be approximated by a polygon formed by connecting an ordered set of intersection points. However, as shown in Figure 17(b)–(d), there exist configurations in which these intersections ambiguously delineate the cells. To address this ambiguity, Hou et al. [2022] proposes an iterative strategy that inserts additional points at the intersection of the line connecting the Voronoi site and the cell centroid with the convex hull of the cell. This heuristic, however, does not reliably resolve configurations such as those in Figure 17(c)–(d) and often fails near sharp features (Figure 18), with no guarantee of termination.

In contrast, as shown in Figure 17, our algorithm robustly and efficiently handles all configurations encountered in practice. Only in extremely rare, deliberately constructed extreme variants of Figure 17(d), where offset samples are very sparse and the surface contains narrow tubular regions or extremely sharp corners, may all tetrahedron edges fail to intersect the offset surface, resulting in an empty RVC and the loss of extremely small offset branch. To resolve this issue, we can adaptively subdivide tetrahedra within each Voronoi cell: 1–4 subdivision is applied to tetrahedra with large inscribed-sphere radii, and 1–3 subdivision is applied to tetrahedra with large facet inscribed-circle radii (see inset), until the inscribed-sphere radii fall below a prescribed threshold. After processing all cells, we obtain a topologically correct approximate RVD on the offset surface, without losing branches larger than the prescribed size, where each Voronoi cell is associated with its corresponding cut loop and RVC.



6.1.6 Remarks. To better preserve features in subsequent reconstruction steps, we compute the RPD of the offset point set by assigning higher weights to feature points, which better aligns the dual connectivity with feature lines (Figure 19). In our experiments, feature points are empirically assigned a weight of r , while non-feature points receive $0.3r$. To prevent unbounded Voronoi cells along the outer boundary, the Voronoi diagram is restricted within

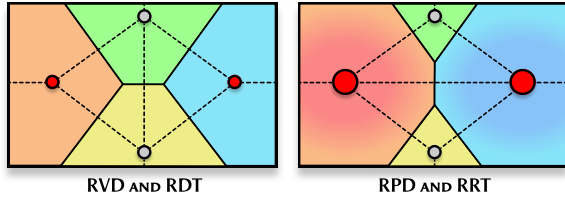


Fig. 19. A 2D schematic illustrates that RPD, computed by assigning higher weights to feature points (marked in red), exhibits dual connectivity (dash lines) more favorable for recovering feature lines compared to the RVD.

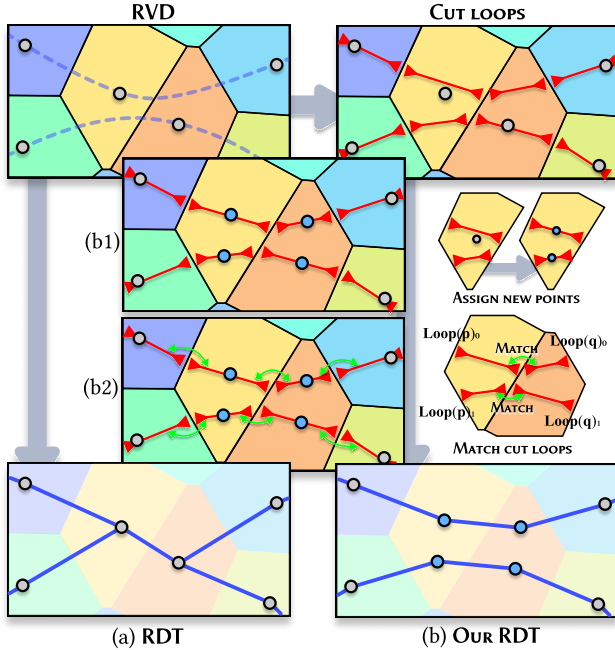


Fig. 20. Handling disconnected RVCs in RVD cells. Cells that violate the “one site, one region” property can produce disconnected RVCs, leading to non-manifold structures in the dual RDT (a). We assign new points (blue) to each RVC within such cells (b1) and reconstruct their connectivity by matching adjacent cut loops (red) (b2), yielding a correct reconstruction in these challenging regions (b).

a large bounding box. A symbolic perturbation is also applied to Voronoi vertices to avoid degenerate cases. Since RVC computation for each Voronoi cell is independent, the entire RVD computation can be efficiently parallelized.

6.2 Resulting Mesh Extraction

Once the RVD satisfies the “one site, one region” property, meaning that each site dominates a region topologically equivalent to a disk [Wang et al. 2020], the final mesh can be directly extracted from the dual RDT. However, near thin-plate structures or sharp features, an RVD cell may consist of multiple disconnected RVCs. Directly extracting the dual in such cases can lead to incorrect connectivity in the resulting RDT, as illustrated in Figure 20(a). Previous work has addressed this issue by inserting additional sites [Hou et al. 2022; Yan et al. 2009] or by locally modifying the RVD construction

algorithm [Yan et al. 2014], but such strategies are unnecessarily expensive. We propose a simpler and more efficient method that derives the correct triangulation directly from the RVD computed on the offset surface. The core idea is to assign a point to each region and establish the mesh connectivity by leveraging the RVD-aware adjacency information obtained in Section 6.1. For clarity, we distinguish between “sites”, which define the Voronoi diagram, and “points”, which serve as vertices of the resulting mesh. These terms are equivalent in the process described in Section 6.1, where the Voronoi diagram is computed directly from the offset points; however, additional points are subsequently introduced to facilitate the derivation of the correct triangulation.

For each site $\mathbf{p} \in \mathcal{P}$, the RVD cell generated by our method consists of a set of cut loops $\text{Loop}(\mathbf{p})_i$ (with cardinality n_L , each forming a closed loop of connected segments) together with a set of RVCs (with cardinality n_R , each composed of connected triangular and quadrilateral facets). Each cut loop serves as the boundary of exactly one RVC; thus, every cut loop is uniquely associated with a single RVC. To enforce the “one site, one region” requirement, we assign exactly one point to each RVC. When $n_R = 1$, the original Voronoi site $\mathbf{p}$ is assigned to that RVC. When $n_R > 1$, we instead assign one point to each RVC, computed as the centroid of its boundary points (see Figure 20(b1)).

Since a Voronoi cell may therefore contain multiple assigned points, establishing their exact connectivity requires matching the cut loops between adjacent RVD cells. A segment of a cut loop $\text{Loop}(\mathbf{p})_i$ lies on a facet of the Voronoi cell $\text{Vor}(\mathbf{p})$. Since each Voronoi facet is dual to a Delaunay edge, we can identify the neighboring Voronoi cell $\text{Vor}(\mathbf{q})$ that shares the facet. If the RVD cell of $\mathbf{q}$ contains multiple cut loops, the corresponding cut loop $\text{Loop}(\mathbf{q})_j$ is identified by

$$\text{Loop}(\mathbf{q})_j = \arg \min_j \text{dist}(\text{Loop}(\mathbf{p})_i, \text{Loop}(\mathbf{q})_j),$$

where $\text{dist}(\text{Loop}(\mathbf{p})_i, \text{Loop}(\mathbf{q})_j)$ denotes the minimum distance between all pairs of segments in the two loops. Once all cut loops are matched, connectivity among the assigned points follows directly: two points are connected if their corresponding RVC boundary loops are matched (Figure 20(b2)). Based on this point connectivity, a triangular mesh can be robustly extracted from the RVD computed on the offset surface (Figure 20(b)).

To ensure the validity of the extracted mesh, Wang et al. [2022] summarized the manifoldness conditions required for the RVD; if these conditions hold, the induced triangulation is guaranteed to be a 2-manifold. They also enumerated four possible violation cases (Figure 21(a)):

Case I: A site dominates disconnected RVCs.

Case II: An RVC has two or more boundary curves.

Case III: A site dominates the whole surface, or its RVC has only one or two neighboring RVCs.

Case IV: The common boundary between two adjacent RVCs consists of two or more curved segments.

Although the RVD computed on the offset surface differs slightly from the standard setting, our method robustly handles all four cases (Figure 21(c)). Case I is resolved naturally by assigning one point to each disconnected component. For RVCs violating Cases II-IV, we directly use the facets of the RVC to construct the local

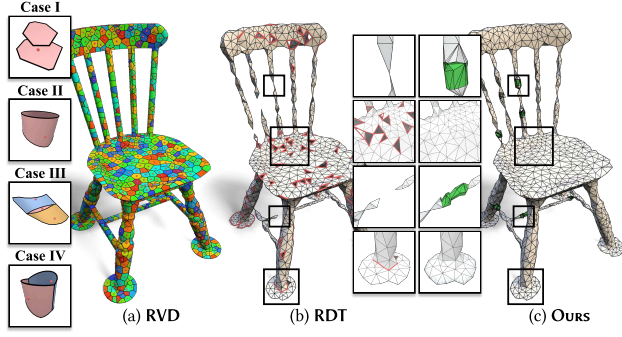


Fig. 21. Typical problematic RVD configurations. (a) Four RVD configurations, where RVCs associated to the same site (red point) are highlighted using identical colors. These cases may produce non-manifold geometries (highlighted in red in (b)) and topological errors in the dual RDT; Our method resolves these cases by inserting additional points and stitching the surrounding triangulation using the computed cut facets (green) (c).

mesh and then merge it with the surrounding triangulation (see Section B.1 of the Supplementary Material for details). Because each RVC is topologically equivalent to the intersection of the offset surface with its corresponding Voronoi cell, the resulting mesh consistently preserves features and topological fidelity.

6.2.1 Acceleration Strategy. The triangulation extraction process shows that determining the intersection topology with the offset surface does not require executing the full algorithm of Section 6.1 for every Voronoi cell. Because each Voronoi cell satisfies $n_R \leq n_L$, if a single cut loop can be extracted from its boundary, the corresponding RVC can be obtained directly from the polygon formed by that loop, which fully determines the topology of the intersection between the cell and the offset surface. In this case, each polygonal facet of the Voronoi cell needs only to satisfy two conditions: each edge intersects the offset surface at most once, and exactly two edges intersect it. For such configurations, processing a Voronoi cell reduces to computing segment/surface intersections, significantly lowering the computational cost. In practice, under 1% of cells, typically in complex regions, need the full procedure; the rest are efficiently handled using the simplified method.

6.3 Post-Processing for Preserving Open Boundaries

Similar to the definition of a thickened solid in [Wang and Chen 2013], for an input surface $\mathcal{S}$ with open boundaries $\partial\mathcal{S}$ and a signed distance d , the boundary-preserving offset surface $\mathcal{O}_d(\mathcal{S})$ is defined as

$$\mathcal{O}'_d(\mathcal{S}) = \{\mathbf{p} \in \mathbb{R}^3 \mid \text{signed_dist}(\mathbf{p}, \mathcal{S}) = d, \text{proj}_{\mathcal{S}}\mathbf{p} \in \mathcal{S} \setminus \partial\mathcal{S}\},$$

$$\mathcal{O}_d(\mathcal{S}) = \overline{\mathcal{O}'_d(\mathcal{S})},$$

where $\text{proj}_{\mathcal{S}}\mathbf{p}$ is the projection of $\mathbf{p} \in \mathbb{R}^3$ onto $\mathcal{S}$, and $\overline{\mathcal{O}'_d(\mathcal{S})}$ denotes the closure of $\mathcal{O}'_d(\mathcal{S})$. For clarity, we refer to this operator as the *single-sided* offset, while the offset defined in Equation (1) as is referred to as the *two-sided* offset. By definition, the single-sided offset surface is a subset of the two-sided offset surface.

The single-sided offset for surfaces with open boundaries is required in many practical scenarios. For example, when computing offsets for meshes representing solid boundaries [Kim and Yang

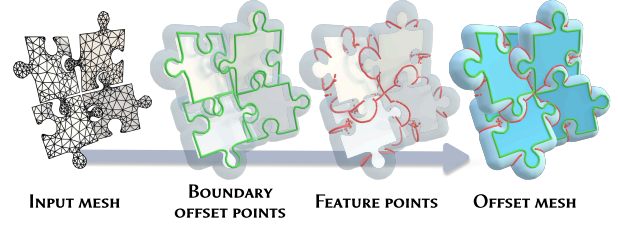


Fig. 22. Single-sided offset boundary extraction. The boundary consists of offsets of the input open edges and intersections between offset surfaces of different elements. We capture these using dense sampling and offsetting of boundary edges (green points) and the feature-point generation algorithm described in Section 5.2 (red points), enabling accurate boundary recovery through facet filtering.

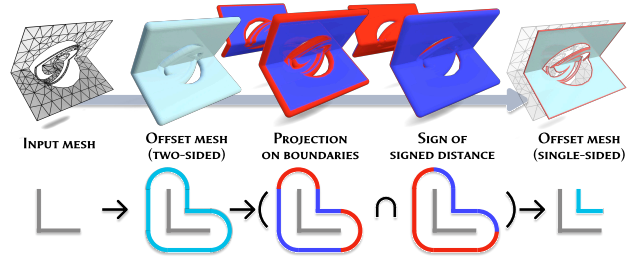


Fig. 23. Pipeline for extracting a single-sided offset result (with its 2D schematic). Starting from the two-sided offset, we extract the single-sided result by filtering facets whose centroid projections lie on the original open boundary (red) and by selecting the desired sign of the signed distance (blue for positive, red for negative). The remaining facets form the final single-sided offset surface.

2005; Yau and Lee 2004] or open-boundary meshes extracted from designed surface regions [Wang and Chen 2013], preserving boundary integrity helps reduce unnecessary compensation and prevents interference between the tool shank and the workpiece [Hwang 1992]. Similarly, in rapid prototyping and manufacturing [Koc and Lee 2002], accurate single-sided boundary offsets are essential for compensating errors caused by non-uniform shrinkage [Qu and Stucker 2003].

After reconstructing the two-sided offset mesh using our previous method, obtaining the single-sided offset is straightforward, since its boundaries are preserved in the two-sided result. By integrating dense sampling and offsetting of input boundary edges (Section 4) with the proposed feature point generation method (Section 5.2), our approach automatically captures all open boundaries of the offset surface; see Figure 22. We then remove facets not belonging to the single-sided offset: we project each triangle centroid onto the input mesh, discard facets whose projections lie on open boundaries, and retain only those on the “one side”, determined by the sign of the signed distance via the angle-weighted pseudonormal [Baerentzen and Aanaes 2005], as illustrated in Figure 23.

In contrast, existing offset methods do not naturally produce single-sided results. For comparison, we filter vertices with distances below the offset distance for the thickening results [Wang 2018], and apply our post-processing strategy to two-sided offsets from **feature-preserving offset (FPO)** [Zint et al. 2023] and **precision-controllable offset (PCO)** [Wang et al. 2024]. As shown

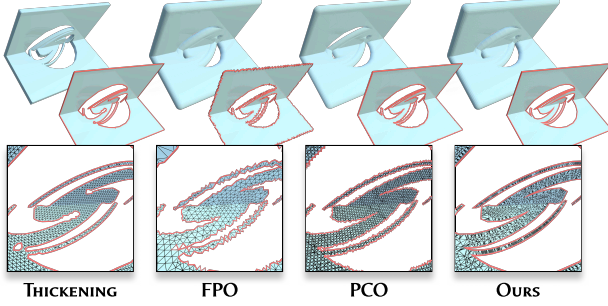


Fig. 24. Comparison of single-sided offset results derived from different methods. Thickening [Wang 2018] generates a thickening model, from which we extract a single-sided offset by filtering out facets that do not satisfy the offset definition. PCO [Wang et al. 2024] computes an iso-surface of the signed distance field. FPO [Zint et al. 2023] and our method both yield two-sided offsets, and their single-sided results are obtained using the post-processing described in Section 6.3. These methods are included to illustrate representative behaviors on open-boundary inputs.

in Figure 24, our method generates cleaner and more accurate open boundaries. The thickening method gives a rough approximation of boundaries due to its voxelized representation, while both FPO and PCO produce insufficient points along open boundaries, resulting in noticeable artifacts.

7 Experiments

7.1 Experimental Setting

Our method is implemented in C++. All experiments, are performed on a laptop with an 8 Core 3.2 GHz (AMD Ryzen 7 5800H) CPU, 16 GB RAM, and an NVIDIA GeForce RTX 3060 GPU. In the comparative experiments, all parallelizable methods are run with maximum parallelism, and the GPU-based methods are run on the laptop GPU. Experiments are conducted on the complete Thingi10K dataset [Zhou and Jacobson 2016]. All models are normalized by the diagonal length of their bounding boxes, and the length measurements are expressed as a percentage of the diagonal length. The entire Thingi10K dataset includes 10,000 models, many of which contain self-intersections, non-manifold vertices and edges, and open boundaries.

We adopt various metrics to evaluate the quality of the resulting offset meshes. Our primary concern is the error of the offset mesh, which is critical for controlling geometric errors in components and impacting the quality of machined products [Bohez 2002; Jha and Kumar 2003]. We use the metric of average approximation error (denoted by E_{ave}) to quantify the relative error of the offset meshes, defined as the average distance of a mesh to the offset surface relative to the given offset distance d as follows:

$$E_{ave} = \frac{1}{m} \sum_{i=1}^m \left| \frac{\|\mathbf{q}_i - \text{proj}_{\mathcal{M}} \mathbf{q}_i\|_2 - |d|}{|d|} \right|,$$

where m is the number of sampling points. To evaluate the accuracy of the offset surface's curvature, we employ the average normal deviation [Zint et al. 2023] (denoted by σ_{ave}) to measure the consistency

of the normals with the input mesh, which is computed as follows:

$$\sigma_{ave} = \frac{1}{m} \sum_{i=1}^m \angle(\mathbf{q}_i - \text{proj}_{\mathcal{M}} \mathbf{q}_i, \mathbf{n}_{\Delta}(\mathbf{q}_i)),$$

where $\mathbf{n}_{\Delta}(\mathbf{q}_i)$ is the normal of the triangle from which the point $\mathbf{q}_i$ is sampled. Smaller values in the approximation errors and normal deviations indicate a more accurate approximation to the offset.

In the comparative experiments, we propose R_{topo} , a Hausdorff-distance-based metric that calculates the proportion of the offset result within a specific threshold of the ground truth to approximately quantify topological fidelity, defined as follows:

$$R_{topo} = \frac{1}{m} \left(\#\{\mathbf{q}_i \mid \frac{\|\mathbf{q}_i - \text{proj}_{\mathcal{O}_d(\mathcal{M})} \mathbf{q}_i\|_2}{|d|} < \epsilon\} \right),$$

where $\mathcal{O}_d(\mathcal{M})$ is the exact offset surface, and $\epsilon = 1.0\%$ is the tolerance for accounting for the different errors that may arise in the results generated by the different methods. As the value of R_{topo} increases, the topological approximation of the results to the ideal surface becomes more accurate. Details descriptions of this metric are provided in Section C of the Supplementary Material.

As a reference, we also include the average minimum angle (denoted by G_{min}), the proportion of skinny triangles (denoted by R_{skinny}), and the average mean-ratio (denoted by G_{Δ}), as the metric for assessing the quality of the mesh. In particular, we consider a triangle as a skinny triangle if its minimum angle is less than 5° . For a triangle T with side lengths a, b, c and area S_T , the mean-ratio G_{Δ} of the triangle is computed as

$$G_{\Delta}(T) = \frac{4\sqrt{3}S_T}{a^2 + b^2 + c^2}.$$

For subsequent analysis and machining processes, a mesh with as few skinny triangles as possible is preferred, and it is desirable for all triangles to be close to equilateral, meaning that G_{min} approaches $\pi/3$ and G_{Δ} approaches 1.

7.2 Influence of Parameters and Ablation Studies

Our method involves two independent adjustable parameters: d to regulate the distance between the offset surface and the input mesh; r to control the covering density of the offset surface, affecting the level of detail and the minimum feature size to be preserved. Our method supports offsets at arbitrary distances, as illustrated in Figure 25, with the corresponding quantitative data presented in Table S1 (where figures and tables numbered with the prefix “S” are presented in the Supplementary Material). As r decreases, the number of vertices in the offset mesh increases, leading to a higher density of offset points and enabling the preservation of smaller-scale features. All vertices in the final mesh lie precisely on the offset surface, making it a piecewise-linear approximation of the offset surface. Increasing the vertex count reduces approximation errors but also increases computational time. Even when r is very large and the offset samples become sparse, the proposed topology-aware reconstruction reliably produces an offset mesh. Experimental results with different values of r are shown in Figure 26 and Table S2. In our experiments, we default to setting $r = 1.0\%$.

In the feature enhancement stage, two preset parameters, θ_{ϵ} and k , control the retained feature sharpness and the sampling density along feature lines, respectively (Figure 28). These parameters are

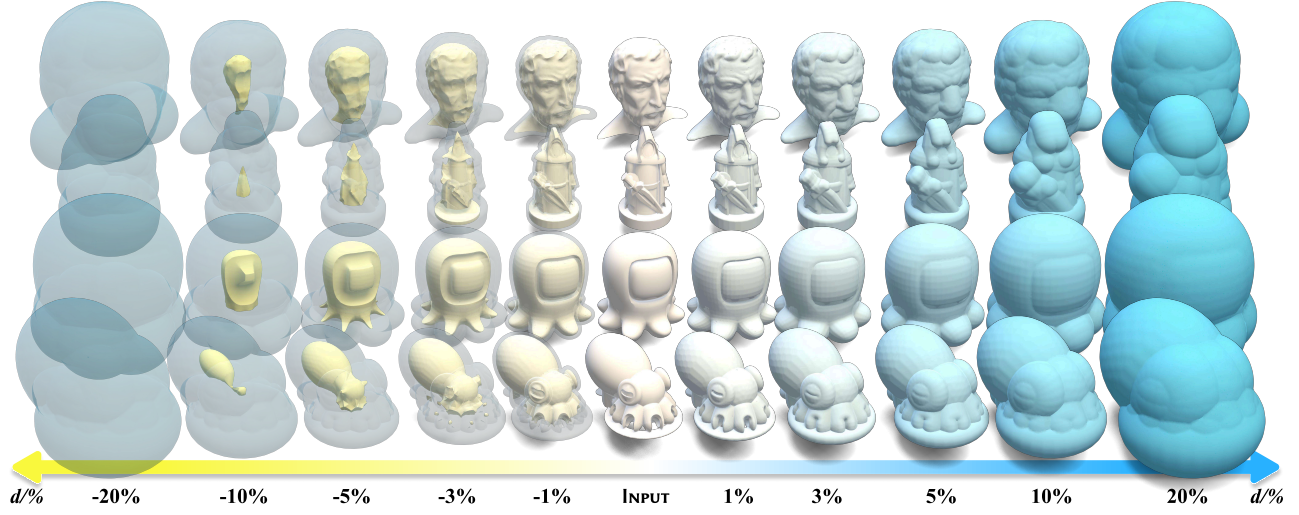


Fig. 25. Results at different offset distances for models from the Thingi10K dataset (top to bottom: 98480, 116877, 740253, and 942120) with $r = 1.0\%$. Since our method produces a two-sided offset, the inward offset is extracted from the interior of the result. Corresponding quantitative data are reported in Table S1.

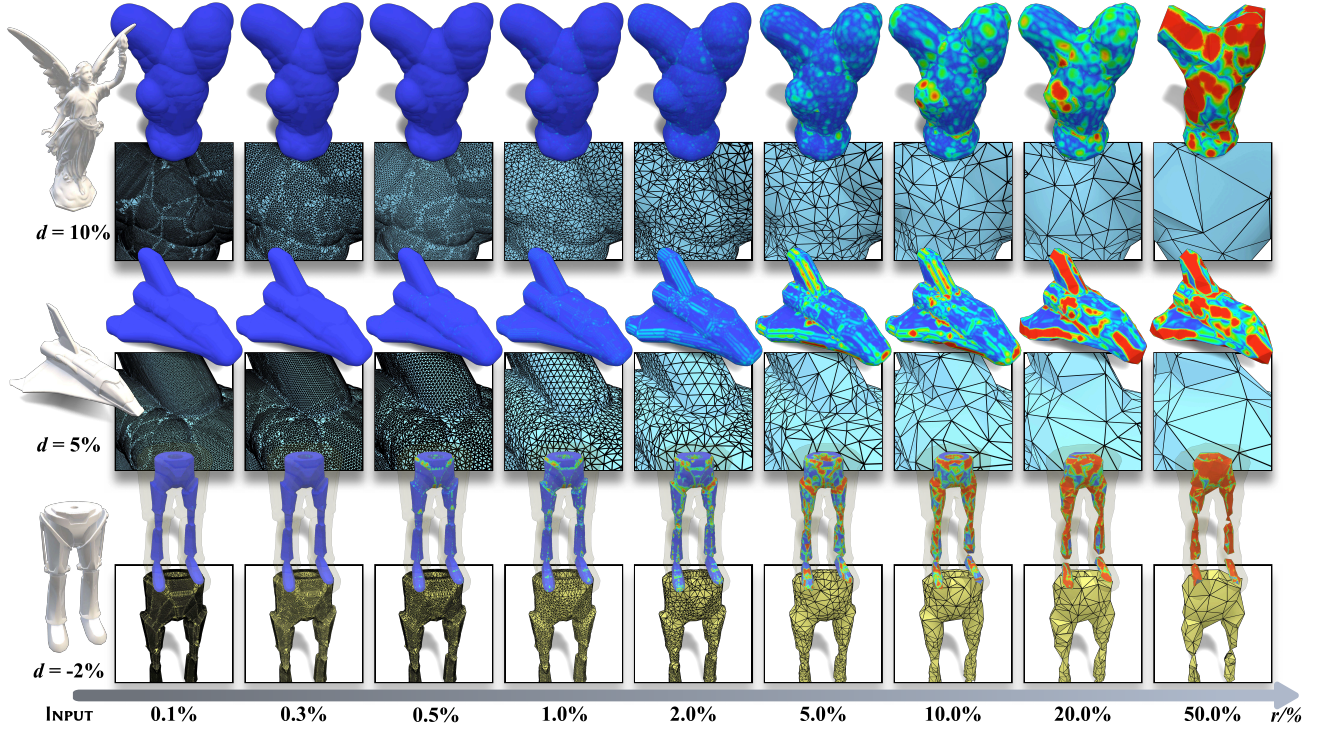


Fig. 26. Offset results at different covering densities for models from the Thingi10K dataset (top to bottom: 252119, 255238, 110372). Larger offset errors are indicated by colors closer to red. Corresponding quantitative results are reported in Table S2.

not sensitive to precise values and can be flexibly chosen in practice; in all our experiments, we set them to $\theta_\epsilon = 8^\circ$, $k = 4$, respectively. We generate feature points both along feature lines and at feature corners, and assign them different priorities during subsequent filtering; omitting corner feature points leads to incorrect reconstruction of these features; see the comparison with and without feature enhancement in Figure 27(c).

General RVD/RPD-based reconstruction methods are not directly applicable to connecting offset points into a mesh, as they require a base surface for diagram computation. To address this limitation, Boltcheva and Lévy [2017] approximate the base surface using a local disk per point, while Xu et al. [2022] employ the Poisson reconstruction of the points [Kazhdan et al. 2006] as the base surface. However, inaccuracies in these approximations

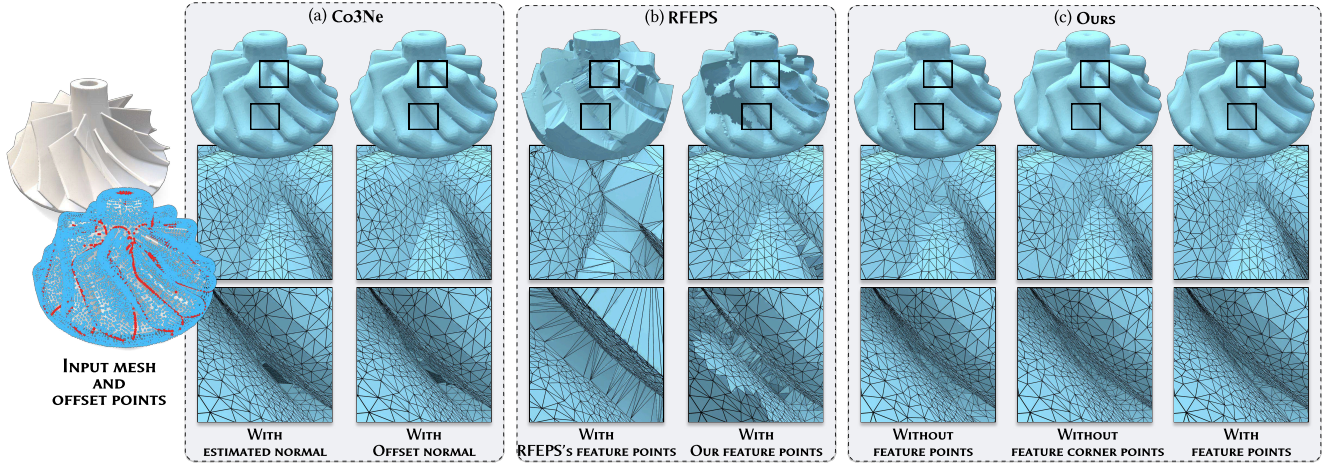


Fig. 27. Comparison of RVD/RPD-based reconstruction methods on feature-enhanced offset points generated by our method (model 43149, Thingi10K). (a) Co3Ne [Boltcheva and Lévy 2017], which requires points with normals as input, is evaluated using both estimated normals and precise normals computed as projection vectors onto the input mesh; in both cases, holes appear near thin-plate structures; (b) RFEPS [Xu et al. 2022], which can generate feature points, is evaluated using both its own and our feature points; however, the underlying Poisson reconstruction fails near sharp features, producing reconstruction errors and many non-orientable facets; and (c) Our explicit reconstruction method correctly reconstructs the geometry from offset points alone but incompletely recovers sparse features; with only feature-line points and without feature-corner points, feature corners are missing; after full feature enhancement, features are effectively preserved.

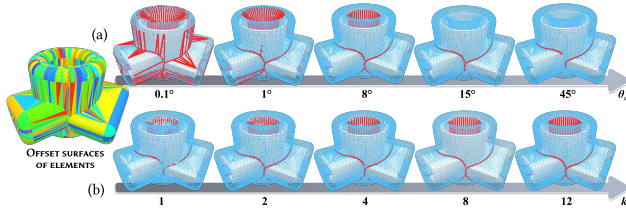


Fig. 28. Impact of parameters θ_e and k on feature enhancement (model 296805, Thingi10K). The left image shows the union of offset surfaces of all simple elements; boundaries between differently colored regions indicate features. (a) With $k = 4$, increasing θ_e filters out shallow features, retaining only sharper ones; (b) with $\theta_e = 8^\circ$, larger k yields denser feature samples and more complete coverage along feature lines.

Table 1. Average Computation Time (in seconds) for Each Stage of Our Pipeline on the Full Thingi10K Dataset, together with the Corresponding Percentage of the Total Runtime, Under Three Different Parameter Configurations

Offset Params.	Preproc.	Offset Pts. Gen.	Feature Enh.		Reconst.	Total
			Feature Pts. Gen.	Filtering		
$d = 5\%$ $r = 1\%$	0.013 (0.412%)	0.024 (0.761%)	1.262 (40.038%)	0.263 (8.344%)	1.598 (50.698%)	3.152
$d = 10\%$ $r = 1\%$	0.013 (0.387%)	0.041 (1.221%)	1.337 (39.827%)	0.409 (12.183%)	1.565 (46.619%)	3.357
$d = 5\%$ $r = 0.5\%$	0.013 (0.188%)	0.071 (1.026%)	2.338 (33.801%)	0.832 (12.028%)	3.633 (52.523%)	6.917

lead to reconstruction errors for thin structures and sharp features, even when accurate offset normals and feature points are provided, as shown in the right subfigures of Figure 27(a)–(b). In contrast, our tailored method computes the RPD directly on the offset

surface, which naturally captures local topological structures while remaining efficient, thereby enabling correct reconstruction with feature preservation (Figure 27(c)).

7.3 Performance

We evaluate our method on the entire Thingi10K dataset, successfully generating valid offset meshes for all models except four invalid ones in the dataset (models 286163, 81313, and 74463 contain no valid geometry, and model 49911 cannot be parsed). To comprehensively assess the performance of our method under different parameter settings, we conduct evaluations using three configurations: $d = 5\%$, $r = 1\%$; $d = 10\%$, $r = 1\%$; and $d = 5\%$, $r = 0.5\%$. Table 1 reports the average proportion of time consumed by each stage of the pipeline, while Figure S4 provides comprehensive statistics of several key evaluation metrics across the entire dataset.

7.4 Comparisons

We compare our method against several advanced offsetting techniques, including PCO [Wang et al. 2024], FPO [Zint et al. 2023], alpha wrapping [Portaneri et al. 2022], thickening [Wang 2018], LDNI [Leung and Wang 2017], Minkowski sum [Lien 2008]. To ensure optimal performance, all compared methods are configured with their default parameters whenever applicable. For alpha wrapping, the α parameter is set to 1.0% of the model's diagonal length. LDNI and thickening are both run at resolutions of 128, with LDNI executed on the GPU. The Minkowski-sum method uses a 64-segment spherical kernel.

Feature preservation. Figure 29 provides a detailed comparison of feature preservation across different methods. While the Minkowski sum method preserves features, it generates numerous intersecting triangles that require further processing. Thickening

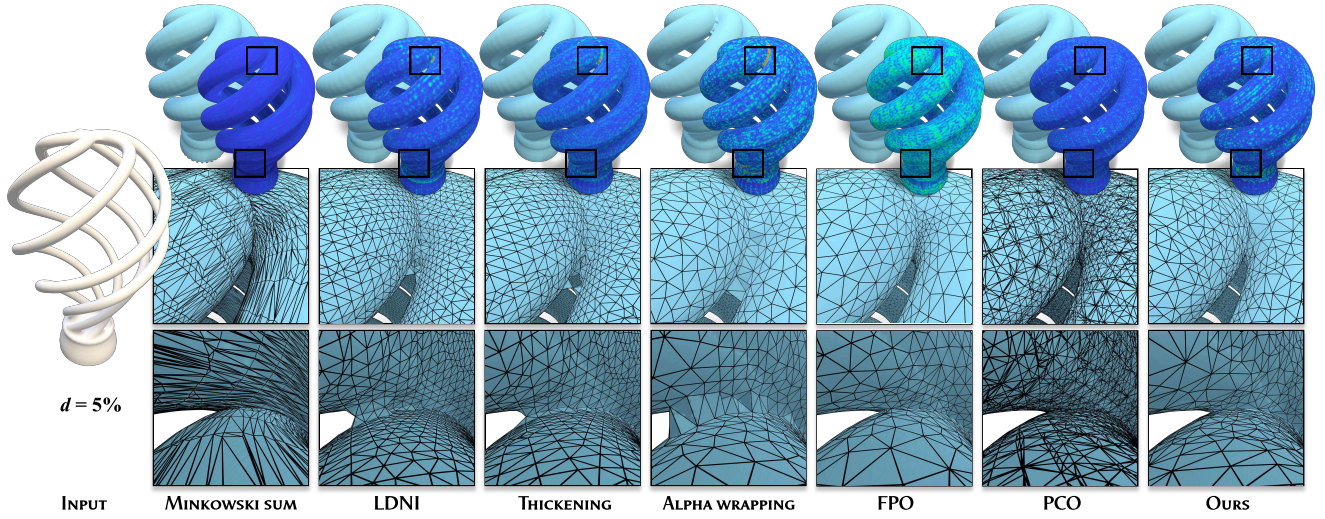


Fig. 29. Feature preservation comparison for offset results (model 39084, Thingi10K). Offset errors are color-coded, with red indicating larger errors. Quantitative results are reported in Table S3.

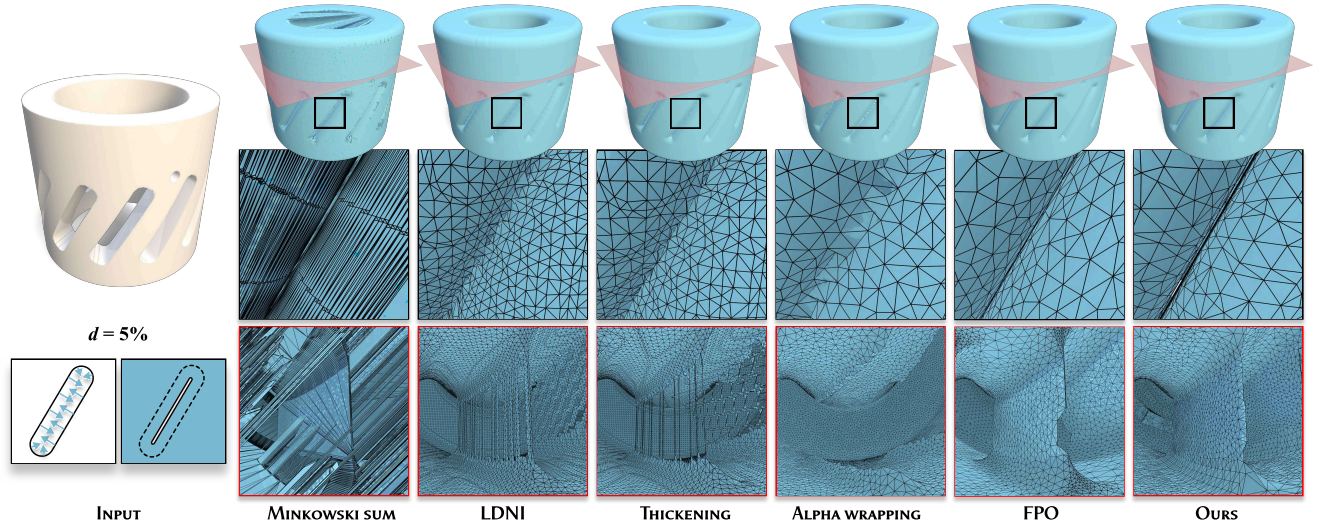


Fig. 30. Comparison of offset results with thin-plate structures (model 57810, Thingi10K). The input mesh consists of planes separated by approximately twice the offset distance, producing an extremely thin plate after offsetting (gap $\approx 0.1\%$ of the model diagonal; see lower-left schematic). Because the gap is far below the resolution of voxel-based LDNI and thickening, as well as the α parameter in alpha wrapping, these methods exhibit fragmentation or structure loss. Adaptive octree-based methods require extensive refinement, resulting in high computational cost (6226.030 s for FPO) and excessive memory usage (PCO failed under a 16 GB limit and is therefore omitted). In contrast, our topology-aware reconstruction recovers the structure in 11.445 s. Quantitative results are reported in Table S3.

and LDNI often generate coarse or blurred geometry in feature-rich areas, while alpha wrapping does not preserve sharp features. FPO improves mesh quality in these regions through local optimization operators, but this comes at a substantial computational cost. PCO can retain feature lines but produces poor mesh quality near sharp features, resulting in many skinny triangles. In contrast, our method reconstructs feature lines both efficiently and accurately, maintaining high fidelity even at sharp features. Additional comparisons are provided in Figure S5, and the corresponding quantitative results are reported in Table S3.

Challenging-case comparison. Due to potentially large geometric and topological discrepancies between the offset surface and the input mesh, offsetting may introduce unexpected features absent from the original geometry, posing significant challenges for offset algorithms. These challenges are particularly evident when reconstructing offsets for three common cases: thin plate structures (Figure 30), highly sharp features (Figure 31), and thin tubular structures (Figure 32). The Minkowski sum method may yield visually plausible results but leaves many intersecting triangles unresolved, while alpha wrapping requires extremely small alpha

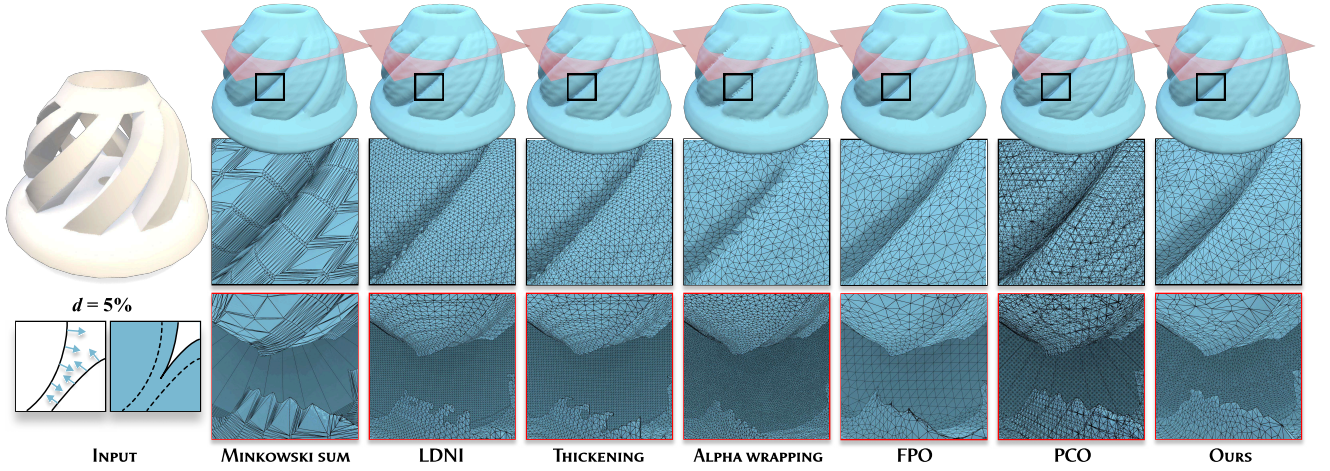


Fig. 31. Comparison of offset results with highly sharp features (model 79279, Thingi10K). Uneven mesh gaps induce extremely sharp offset features (see lower-left schematic). Minkowski sum yields accurate geometry but contains many self-intersections, while LDNI, thickening, and alpha wrapping exhibit fragmentation or feature loss. FPO and PCO require 619.638 s and 37.608 s, respectively, compared to 3.775 s for our method. Quantitative results are reported in Table S3.

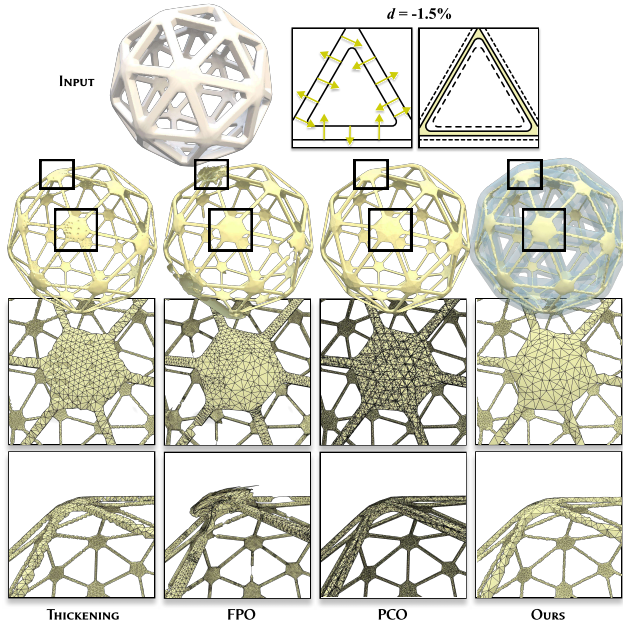


Fig. 32. Comparison of methods on a hollow cage-shaped model (model 509313, Thingi10K). An inward offset produces a structure of thin tubular elements (see the upper-right 2D schematic). The Minkowski sum and LDNI methods fail, and alpha wrapping does not support inward offsets; thus, results are omitted. The thickening method produces non-manifold artifacts due to resolution limits. Adaptive octree-based methods require heavy refinement—FPO takes 2239.354 s with local errors, while PCO yields low-quality meshes. In contrast, our method computes correct outward and inward offsets in 8.668 s. Quantitative results are reported in Table S3.

values, incurring high computational cost. Voxel-based methods (LDNI and Thickening), limited by fixed voxel resolution, often produce fragmented results. Adaptive octree-based methods (FPO and PCO) can capture such structures but require substantial time and memory resources to achieve acceptable quality. In particu-

lar, the Minkowski sum and LDNI methods fail on thin tubular structures, and alpha wrapping does not support inward offsets. In contrast, our method directly produces predominantly topologically correct results within a comparatively short time. Corresponding quantitative results are reported in Table S3.

Quantitative comparison. We evaluate all methods on the same 100 models randomly selected from the Thingi10K dataset under three offset-distance regimes: large ($d = 10\%$), medium ($d = 5\%$), and small ($d = 1\%$). Besides the default settings, we test our method with $r = 0.5\%$, the alpha-wrapping method with $\alpha = 0.5\%$, and LDNI and thickening at a resolution of 256. For the remaining methods, default parameters are used: the Minkowski sum method has no tunable parameters, and the results of FPO and PCO are largely insensitive to parameter choices. Since the thickening method produces outputs that include the input mesh and cross-sectional regions [Wang 2018], and our method also generates inward offsets, these extra parts are removed to ensure consistent outputs before metric evaluation. Statistics are reported in Table 2. We observe that our method is robust, with no failure cases, similar to the alpha wrapping and FPO methods.

In terms of runtime, our method is the fastest among all CPU-based approaches; even at the refined scale of $r = 0.5\%$, it remains faster than most alternatives and outperforms FPO and PCO by one to two orders of magnitude. At small offset distances ($d = 1\%$), the runtime is slightly higher than that of thickening at resolution 128, as our method generates both outward and inward offsets, forming a thin-shell structure that requires additional processing. Accuracy metrics such as E_{ave} and σ_{ave} depend largely on the number of output vertices and facets, since the offset mesh is a piecewise-linear approximation of the offset surface—output meshes with more vertices and facets inherently yield higher precision; under comparable complexity, our method consistently outperforms others. At $d = 5\%$, its E_{ave} and σ_{ave} are only marginally inferior to those of FPO, which dedicates substantial time in optimizing the output mesh by adaptively distributing vertices to reduce vertex and facet counts and

Table 2. Quantitative Comparison on 100 Randomly Selected Thingi10K Models

Offset distance	Method	#Failure ¹	Output complexity		Time (s) ↓	E_{ave} ↓	σ_{ave} ↓	G_{min} ↑	G_{Δ} ↑	R_{skinny} ↓	R_{topo} ↑
			#Vertices	#Facets							
$d = 10\%$	Minkowski sum	41	20,480	40,551	37.566	0.312%	1.253°	22.364°	0.518	17.480%	99.138%
	LDNI ₁₂₈ (GPU)	0	13,343	26,678	1.217	0.614%	2.677°	41.190°	0.858	0.093%	68.572%
	LDNI ₂₅₆ (GPU)	6	53,956	107,897	1.979	0.311%	1.506°	41.506°	0.862	0.040%	80.731%
	Thickening ₁₂₈	1	37,320	74,300	3.471	0.222%	1.824°	40.751°	0.854	0.075%	98.554%
	Thickening ₂₅₆	1	153,982	307,394	11.707	0.263%	1.270°	40.830°	0.855	0.040%	99.198%
	Alpha wrapping _{$\alpha=1.0\%$}	0	20,875	41,747	4.939	0.152%	2.267°	47.005°	0.923	0.000%	96.796%
	Alpha wrapping _{$\alpha=0.5\%$}	0	83,840	167,678	19.554	0.049%	1.326°	47.062°	0.924	0.000%	98.387%
	FPO	0	5,377	10,751	112.797	0.267%	2.327°	44.774°	0.904	0.021%	77.285%
	PCO	32	143,831	287,100	608.862	0.068%	1.052°	28.273°	0.630	11.227%	N/A ³
	Ours _{$r=1.0\%$}	0	27,691	55,378	3.406 ²	0.117%	1.764°	43.014°	0.886	0.003%	98.096%
	Ours _{$r=0.5\%$}	0	72,482	144,960	10.756 ²	0.050%	1.240°	47.007°	0.928	0.008%	98.857%
$d = 5\%$	Minkowski sum	42	31,969	62,454	35.711	0.566%	1.429°	29.790°	0.592	19.627%	98.687%
	LDNI ₁₂₈ (GPU)	0	10,792	21,567	1.178	1.110%	3.865°	41.032°	0.853	0.143%	73.071%
	LDNI ₂₅₆ (GPU)	18	69,891	42,524	1.811	0.514%	2.164°	41.470°	0.859	0.082%	87.540%
	Thickening ₁₂₈	1	36,064	71,804	3.324	0.230%	2.186°	40.793°	0.851	0.107%	97.535%
	Thickening ₂₅₆	3	148,750	296,975	14.189	0.178%	1.316°	40.927°	0.853	0.048%	99.159%
	Alpha wrapping _{$\alpha=1.0\%$}	0	17,333	34,672	4.823	0.396%	3.423°	46.704°	0.923	0.000%	94.781%
	Alpha wrapping _{$\alpha=0.5\%$}	0	69,812	139,628	18.848	0.120%	1.866°	47.015°	0.924	0.000%	98.070%
	FPO	0	12,550	25,102	140.233	0.246%	1.970°	43.880°	0.893	0.018%	94.546%
	PCO	21	171,265	342,060	107.295	0.163%	1.495°	27.242°	0.608	12.583%	N/A ³
	Ours _{$r=1.0\%$}	0	23,986	47,972	3.129 ²	0.296%	2.417°	42.261°	0.876	0.003%	97.150%
	Ours _{$r=0.5\%$}	0	59,558	119,113	7.226 ²	0.133%	1.662°	46.394°	0.921	0.005%	98.834%
$d = 1\%$	Minkowski sum	68	40,926	78,636	27.510	0.606%	0.941°	17.888°	0.412	37.442%	88.250%
	LDNI ₁₂₈ (GPU)	0	34,147	68,235	1.367	7.274%	8.515°	39.744°	0.832	0.259%	64.598%
	LDNI ₂₅₆ (GPU)	33	131,148	262,069	2.643	3.693%	4.283°	40.961°	0.851	0.079%	89.317%
	Thickening ₁₂₈	2	33,189	66,121	3.145	1.039%	3.283°	40.525°	0.845	0.159%	84.293%
	Thickening ₂₅₆	2	134,087	267,751	8.921	0.296%	1.734°	41.095°	0.850	0.082%	85.842%
	Alpha wrapping _{$\alpha=1.0\%$}	0	13,991	28,007	5.895	3.038%	5.792°	46.756°	0.922	0.000%	82.134%
	Alpha wrapping _{$\alpha=0.5\%$}	0	56,595	113,220	19.021	0.834%	3.120°	46.881°	0.923	0.000%	88.461%
	FPO	0	104,503	209,053	450.323	0.145%	0.928°	44.411°	0.901	0.011%	84.795%
	PCO	9	241,267	482,341	41.402	1.925%	4.654°	25.567°	0.575	14.715%	N/A ³
	Ours _{$r=1.0\%$}	0	21,090	42,195	4.486 ²	1.387%	3.169°	40.687°	0.863	0.008%	87.511%
	Ours _{$r=0.5\%$}	0	48,345	96,715	8.833 ²	0.781%	2.424°	44.760°	0.904	0.005%	89.628%

¹ If a method fails to produce a valid result for an input model under a given parameter setting, it is considered a failure. Failure modes vary across methods: the Minkowski sum method may return an empty result; the LDNI method may become unresponsive; and the thickening and PCO methods may crash, with PCO often failing due to out-of-memory errors.

² The recorded time accounts for the simultaneous generation of both outward and inward offset meshes, when applicable.

³ We use the PCO results as the reference surface for computing the R_{topo} metric. Consequently, reporting R_{topo} for PCO itself is not meaningful, as it always equals 100%.

Greener colors indicate better performance; redder colors indicate worse.

specifically refining the σ_{ave} metric, whereas our method maintains a uniform vertex distribution. Despite using only lightweight point filtering, our method achieves superior mesh quality compared to most baselines, second only to alpha wrapping, which fails to preserve sharp features. Since the exact offset surface is typically unavailable, we use PCO as the reference for R_{topo} . PCO employs an adaptive octree to capture fine-scale structures while ensuring complete surface coverage by explicitly accounting for the offsets of individual mesh elements. Except for deliberately constructed cases that lead to non-manifold offset results, it is empirically observed to yield topologically correct results in practice. Our method shows high topological consistency with PCO. Additional statistics are provided in Table S4.

7.5 Extensions

In addition, our method naturally supports other morphological operators—such as opening and closing—by composing two offset

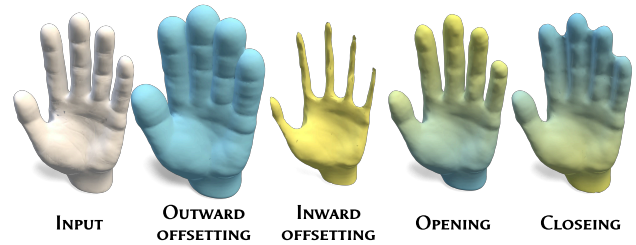


Fig. 33. Combined operators using our method on model 444166 from the Thingi10K dataset. The opening operation is performed by first executing an inward offset followed by an outward offset with the same offset distance, while the closing operation is performed by first executing an outward offset followed by an inward offset with the same offset distance.

operations of equal magnitude in opposite directions, as illustrated in Figure 33. Because the offset points produced by our method

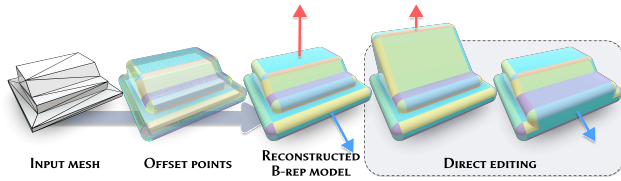


Fig. 34. Segmented point clouds efficiently generated by our method, well suited for B-rep reconstruction and downstream applications such as direct editing.

retain accurate element information, the resulting point cloud can be partitioned into segments with consistent element labels, enabling straightforward reconstruction of a B-rep representation of the offset model. This, in turn, facilitates downstream B-rep operations such as direct editing, as demonstrated in Figure 34.

8 Conclusion and Future Work

In this article, we present a novel direct offset method capable of efficiently generating high-precision offset meshes at arbitrary distances (both positive and negative) from the input triangular mesh. Our method produces offset results with predominantly correct topology, preserving features, and is capable of retaining the open boundaries of the original model. The key to achieving both accuracy and efficiency lies in the adoption of a point-based representation during offset computation, along with tailored feature enhancement and reconstruction algorithms designed to produce outputs with proper topology and desired sharp features. Extensive experiments further validate the advantages of our method.

While our method produces visually plausible results with only minor geometric deviations, it ensures combinatorially 2-manifold outputs without explicitly guaranteeing the absence of facet intersections. Nevertheless, owing to the topology-aware RPD-based reconstruction, the incidence of intersecting facets remains very low. Across the entire Thingi10K dataset, only 9.891% of outputs exhibit self-intersections, involving merely 0.0012% of all facets. Comparative analysis further indicates that the proportion of intersecting facets is substantially lower than that of most competing methods, averaging approximately 0.0005% of all facets. Among the compared approaches, only alpha wrapping, which is based on carving a tetrahedralization, guarantees intersection-free meshes, albeit at the cost of losing sharp features. Statistics and worst-case examples regarding facet intersections are detailed in Table S5 and Figure S6. Our method consists of several modules and shows good compatibility with other related technologies. As future work, we plan to develop simplification and reconstruction schemes tailored for offset operations, with the goal of reducing approximation errors and ensuring the correctness of the generated meshes.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions.

References

Bart Adams and Philip Dutré. 2003. Interactive boolean operations on surfel-bounded solids. In *Proceedings of the ACM SIGGRAPH 2003 Papers*. Association for Computing Machinery, New York, NY, USA, 651–656.

- J. A. Baerentzen and H. Aanaes. 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics* 11, 3 (2005), 243–253.
- J. Basselin, L. Alonso, N. Ray, D. Sokolov, S. Lefebvre, and B. Lévy. 2021. Restricted power diagrams on the GPU. *Computer Graphics Forum* 40, 2 (2021), 1–12.
- E. L. J. Bohez. 2002. Compensating for systematic errors in 5-axis NC machining. *Computer-Aided Design* 34, 5 (2002), 391–403.
- Dobrina Boltcheva and Bruno Lévy. 2017. Surface reconstruction by computing restricted Voronoi cells in parallel. *Computer-Aided Design* 90 (2017), 123–134.
- David E. Breen, Sean Mauch, and Ross T. Whitaker. 2000. 3D scan-conversion of CSG models into distance, closest-point and colour volumes. In *Volume Graphics*, M. Chen, A. E. Kaufman, and R. Yagel (Eds.). Springer London, London, 135–158. DOI: https://doi.org/10.1007/978-1-4471-0737-8_8.
- Shuxian Cai, Juan Cao, Bailin Deng, and Zhonggui Chen. 2025. Feature line extraction based on winding number. *Graphical Models* 141 (2025), 101296.
- Shuxian Cai, Yuanyan Ye, Juan Cao, and Zhonggui Chen. 2024. FACE: Feature-preserving CAD model surface reconstruction. *Graphical Models* 136 (2024), 101230.
- Stéphane Calderon and Tamy Boubekeur. 2014. Point morphology. *ACM Transactions on Graphics* 33, 4, (2014), 13 pages.
- Marcel Campen and Leif Kobbelt. 2010a. Exact and robust (Self-)intersections for polygonal meshes. *Computer Graphics Forum* 29, 2 (2010), 397–406.
- Marcel Campen and Leif Kobbelt. 2010b. Polygonal boundary evaluation of minkowski sums and swept volumes. *Computer Graphics Forum* 29, 5 (2010), 1613–1622.
- Hongyi Cao, Gang Xu, Renshu Gu, Jinlan Xu, Timon Rabczuk, Xiaoyu Zhang, Yuzhe Luo, and Xifeng Gao. 2026. Feature-preserving offset meshing. *ACM Transactions on Graphics* 45, 3 (2026), 18 pages.
- Hongyi Cao, Gang Xu, Renshu Gu, Jinlan Xu, Xiaoyu Zhang, and Timon Rabczuk. 2023. A Parallel Feature-preserving Mesh Variable Offsetting Method with Dynamic Programming. arXiv:2310.08997. Retrieved from <https://arxiv.org/abs/2310.08997>.
- Anka He Chen, Jerry Hsu, Ziheng Liu, Miles Macklin, Yin Yang, and Cem Yuksel. 2025. Offset geometric contact. *ACM Transactions on Graphics* 44, 4 (2025), 21 pages.
- Yong Chen and Charlie C. L. Wang. 2011. Uniform offsetting of polygonal model based on Layered Depth-Normal Images. *Computer-Aided Design* 43, 1 (2011), 31–46.
- Yong Chen, Hongqing Wang, David W. Rosen, and Jarek Rossignac. 2005b. Filletting and rounding using a point-based method. In *Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 533–542.
- Yong Chen, Hongqing Wang, David W. Rosen, and Jarek Rossignac. 2005a. A point-based offsetting method of polygonal meshes. *ASME Journal of Computing and Information Science in Engineering* 5, 3 (2005), 221–235.
- Zhen Chen, Daniele Panozzo, and Jérémie Dumas. 2020. Half-space power diagrams and discrete surface offsets. *IEEE Transactions on Visualization and Computer Graphics* 26, 10 (2020), 2970–2981.
- Jiayi Dai, Yiqun Wang, and Dong-Ming Yan. 2024. Feature-preserving shrink wrapping with adaptive alpha. *Computer Aided Geometric Design* 111, C (2024), 102321.
- Volodymyr Dyedov, Daniel R. Einstein, Xiangmin Jiao, Andrew P. Kuprat, James P. Carson, and Facundo del Pin. 2009. Variational generation of prismatic boundary-layer meshes for biomedical computing. *International Journal for Numerical Methods in Engineering* 79, 8 (2009), 907–945.
- Herbert Edelsbrunner and Nimish R. Shah. 1994. Triangulating topological spaces. In *Proceedings of the 10th Annual Symposium on Computational Geometry*. Association for Computing Machinery, New York, NY, USA, 285–292.
- R.T. Farouki, T. Koenig, K. A. Tarabanis, J. U. Korein, and J. S. Batchelder. 1995. Path planning with offset curves for layered fabrication processes. *Journal of Manufacturing Systems* 14, 5 (1995), 355–368.
- Jia-Peng Guo and Xiao-Ming Fu. 2024. Exact and efficient intersection resolution for mesh arrangements. *ACM Transactions on Graphics* 43, 6 (2024), 14 pages.
- Peter Hachenberger. 2007. Exact Minkowski sums of polyhedra and exact and efficient decomposition of polyhedra in convex pieces. In: *Algorithms – ESA 2007*, Lars Arge, Michael Hoffmann, and Emo Welzl (Eds.). Springer, Berlin, Heidelberg, 669–680.
- John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545.
- E. E. Hartquist, J. P. Menon, K. Suresh, H. B. Voelcker, and J. Zagajac. 1999. A computing strategy for applications involving offsets, sweeps, and Minkowski operations. *Computer-Aided Design* 31, 3 (1999), 175–183.
- Wenjuan Hou, Chen Zong, Pengfei Wang, Shiqing Xin, Shuangmin Chen, Guozhu Liu, Changhe Tu, and Wenping Wang. 2022. SDF-RVD: Restricted voronoi diagram on signed distance field. *Computer-Aided Design* 144, C (2022), 103166.
- J. S. Hwang. 1992. Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer-Aided Design* 24, 12 (1992), 667–676.
- Binit Kumar Jha and Anjani Kumar. 2003. Analysis of geometric errors associated with five-axis machining centre in improving the quality of cam profile. *International Journal of Machine Tools and Manufacture* 43, 6 (2003), 629–636.
- Cha-Soo Jun, Dong-Soo Kim, and Sehyung Park. 2002. A new curve-based approach to polyhedral machining. *Computer-Aided Design* 34, 5 (2002), 379–389.
- Wonhyung Jung, Hayong Shin, and Byoung K. Choi. 2004. Self-intersection removal in triangular mesh offsetting. *Computer-Aided Design and Applications* 1, 1–4 (2004), 477–484.

- Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing*. Eurographics Association, Goslar, DEU, 61–70.
- Dong-Soo Kim, Cha-Soo Jun, and Sehyung Park. 2005. Tool path generation for clean-up machining by a curve-based approach. *Computer-Aided Design* 37, 9 (2005), 967–973.
- Dong-Yoon Lee Kim and Min-Yang Yang. 2004. Offset triangular mesh using the multiple normal vectors of a vertex. *Computer-Aided Design and Applications* 1, 1–4 (2004), 285–291.
- Su-Jin Kim and Min-Yang Yang. 2005. Triangular mesh offset for generalized cutter. *Computer-Aided Design* 37, 10 (2005), 999–1014.
- Bahattin Koc and Yuan-Shin Lee. 2002. Non-uniform offsetting and hollowing objects by using biarcs fitting for rapid prototyping processes. *Computers in Industry* 47, 1 (2002), 1–23.
- Sang Hun Lee. 1999. Offsetting operations on non-manifold boundary representation models with simple geometry. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications* (Ann Arbor, Michigan, USA) (SMA'99). Association for Computing Machinery, New York, NY, USA, 42–53.
- Sang Hun Lee. 2009. Offsetting operations on non-manifold topological models. *Computer-Aided Design* 41, 11 (2009), 830–846.
- Yuen-Shan Leung and Charlie C. L. Wang. 2017. LDNI-DLL: A Dynamic-Link Library for Solid Modeling on GPUs. Retrieved from https://mewangcl.github.io/Projects/GPUBasedLDNI_DLLProj.htm
- Bruno Levy. 2025. Exact predicates, exact constructions and combinatorics for mesh CSG. *ACM Transactions on Graphics* 44, 5 (2025), 27 pages.
- Jyh-Ming Lien. 2007. Point-Based Minkowski sum boundary. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*. 261–270.
- Jyh-Ming Lien. 2008. Covering Minkowski sum boundary using points with applications. *Computer Aided Geometric Design* 25, 8 (2008), 652–666.
- Longdu Liu, Hao Yu, Jiong Guo, Shiqing Xin, Shuangmin Chen, and Changhe Tu. 2025. Direct extraction of high-quality and feature-preserving triangle meshes from unsigned distance functions. *Computers & Graphics* 132 (2025), 104335.
- Shengjun Liu and Charlie C. L. Wang. 2011. Fast intersection-free offset surface generation from freeform models with triangular meshes. *IEEE Transactions on Automation Science and Engineering* 8, 2 (2011), 347–360.
- Wenlong Meng, Shuangmin Chen, Zhenyu Shu, Shi-Qing Xin, Hongbo Fu, and Changhe Tu. 2018. Efficiently computing feature-aligned and high-quality polygonal offset surfaces. *Computers & Graphics* 70 (2018), 62–70.
- Darko Pavić and Leif Kobbelt. 2008. High-resolution volumetric computation of offset surfaces with feature preservation. *Computer Graphics Forum* 27, 2 (2008), 165–174.
- Jeanne Pellerin, Bruno Lévy, Guillaume Caumon, and Arnaud Botella. 2014. Automatic surface remeshing of 3D structural models at specified resolution: A method based on Voronoi diagrams. *Computers & Geosciences* 62, 1 (2014), 103–116.
- Cédric Portaneri, Mael Rouxel-Labbé, Michael Hemmer, David Cohen-Steiner, and Pierre Alliez. 2022. Alpha wrapping with an offset. *ACM Transactions on Graphics* 41, 4, Article 127 (jul 2022), 22 pages.
- Xiuzhi Qu and Brent Stucker. 2003. A 3D surface offset method for STL-format models. *Rapid Prototyping Journal* 9, 3 (2003), 133–141.
- Jaroslav R. Rossignac and Aristides A. G. Requicha. 1986. Offsetting operations in solid modelling. *Computer Aided Geometric Design* 3, 2 (1986), 129–148.
- Hugo Schott, Theo Thonat, Thibaud Lambert, Eric Guérin, Eric Galin, and Axel Paris. 2025. Sphere carving: Bounding volumes for signed distance fields. *ACM Transactions on Graphics* 44, 4 (2025), 13 pages.
- Joon-Kyung Seong, Gershon Elber, and Myung-Soo Kim. 2006. Trimming local and global self-intersections in offset curves/surfaces using distance maps. *Computer-Aided Design* 38, 3 (2006), 183–193.
- Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. 1998. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery, New York, NY, USA, 231–242.
- Ken Shoemake. 1985. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.* 19, 3 (1985), 245–254.
- Haoran Sun, Shuang Wu, Hujun Bao, and Jin Huang. 2026. Variational mesh offsetting by smoothed winding number. *IEEE Transactions on Visualization and Computer Graphics* 32, 2 (2026), 1668–1681.
- M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, et al. 2005. Collision detection for deformable objects. *Computer Graphics Forum* 24, 1 (2005), 61–81.
- Tim Van Hook. 1986. Real-time shaded NC milling display. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery, New York, NY, USA, 15–20.
- G. Varadhan and D. Manocha. 2004. Accurate Minkowski sum approximation of polyhedral models. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*. 392–401.
- Charlie Wang and Yong Chen. 2013. Thickening freeform surfaces for solid fabrication. *Rapid Prototyping Journal* 19, 6 (2013), 395–406.
- Charlie C. L. Wang. 2018. Thickening: A Library for General Mesh Offsetting. Retrieved August 4, 2026 from <https://mewangcl.github.io/Projects/MeshThickeningProj.htm>
- Charlie C. L. Wang, Yuen-Shan Leung, and Yong Chen. 2010. Solid modeling of polyhedral objects by Layered depth-normal images on the GPU. *Computer-Aided Design* 42, 6 (2010), 535–544.
- Charlie C. L. Wang and Dinesh Manocha. 2013. GPU-based offset surface computation using point samples. *Computer-Aided Design* 45, 2 (2013), 321–330.
- Lei Wang, Xudong Wang, Pengfei Wang, Shuangmin Chen, Shiqing Xin, Jiong Guo, Wenping Wang, and Changhe Tu. 2024. PCO: Precision-controllable offset surfaces with sharp features. *ACM Transactions on Graphics* 43, 6 (2024), 16 pages.
- Pengfei Wang, Zixiong Wang, Shiqing Xin, Xifeng Gao, Wenping Wang, and Changhe Tu. 2022. Restricted delaunay triangulation for explicit surface reconstruction. *ACM Transactions on Graphics* 41, 5 (2022), 20 pages.
- Pengfei Wang, Shiqing Xin, Changhe Tu, Dongming Yan, Yuanfeng Zhou, and Caiming Zhang. 2020. Robustly computing restricted Voronoi diagrams (RVD) on thin-plate models. *Computer Aided Geometric Design* 79 (2020), 101848.
- Yu Wang. 1996. Intersection of offsets of parametric surfaces. *Computer Aided Geometric Design* 13, 5 (1996), 453–465.
- Huibiao Wen, Guilong He, Rui Xu, Shuangmin Chen, Shiqing Xin, Zhenyu Shu, Taku Komura, Jieqing Feng, Wenping Wang, and Changhe Tu. 2025. Feature-preserving mesh repair via restricted power diagram. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers'25)*. Association for Computing Machinery, New York, NY, USA, Article 150, 11 pages.
- Rui Xu, Zixiong Wang, Zhiyang Dou, Chen Zong, Shiqing Xin, Mingyan Jiang, Tao Ju, and Changhe Tu. 2022. RFEPs: Reconstructing feature-line equipped polygonal surface. *ACM Transactions on Graphics* 41, 6 (2022), 15 pages.
- Dong-Ming Yan, Guanbo Bao, Xiaopeng Zhang, and Peter Wonka. 2014. Low-resolution remeshing using the localized restricted voronoi diagram. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1418–1427.
- Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. 2009. Isotropic remeshing with fast and exact computation of restricted voronoi diagram. *Computer Graphics Forum* 28, 5 (2009), 1445–1454.
- C.-M. Chuang Yau and Y.-S. Lee. 2004. Numerical control machining of triangulated sculptured surfaces in a stereo lithography format with a generalized cutter. *International Journal of Production Research* 42, 13 (2004), 2573–2598.
- Yuanyan Ye, Yubo Wang, Juan Cao, and Zhonggui Chen. 2024. Watertight surface reconstruction method for CAD models based on optimal transport. *Computational Visual Media* 10, 5 (2024), 859–872.
- Andrei Zaharescu, Edmond Boyer, and Radu Horaud. 2011. Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 4 (2011), 823–837.
- Zihan Zhao, Pengfei Wang, Minfeng Xu, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. 2026. OffsetCrust: Variable-radius offset approximation with power diagrams. *IEEE Transactions on Visualization and Computer Graphics* 32, 9 (2026), 7729–7741.
- Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A dataset of 10,000 3D-printing models. arXiv:1605.04797. Retrieved from <https://arxiv.org/abs/1605.04797>
- Daniel Zint, Zhouyuan Chen, Yifei Zhu, Denis Zorin, Teseo Schneider, and Daniele Panozzo. 2025. Topological offsets. *ACM Trans. Graph.* 44, 4 (2025), 19 pages.
- D. Zint, N. Maruani, M. Rouxel-Labbé, and P. Alliez. 2023. Feature-preserving offset mesh generation from topology-adapted octrees. *Computer Graphics Forum* 42, 5 (2023), e14906.

Received 3 February 2026; revised 23 June 2026; accepted 2 July 2026