Variational Mesh Offsetting by Smoothed Winding Number

Haoran Sun, Shuang Wu, Hujun Bao*, and Jin Huang

***Abstract*—Surface mesh offsetting is a fundamental operation in various applications (*e.g.* shape modeling). Implicit methods that contour a volumetric distance field are robust at handling intersection defects, but it is challenging to apply shape control (*e.g.* preserving sharp features in the input shape) and to avoid undesired topology changes. Explicit methods, which move vertices towards the offset surface (with possible adaptivity), can address the above issues, but it is hard to avoid intersection issues. To combine the advantages of both, we propose a variational framework that takes mesh vertex locations as variables while simultaneously involving a smooth winding-number field associated with the mesh. Under various shape regularizations (*e.g.* sharp feature preservation) formulated on the mesh, the objective function mainly requires that the input mesh lie on the offset contour of the field induced by the resulting mesh. Such a combination inherits the ability to apply flexible shape regularizations from explicit methods and significantly alleviates intersection issues because of the field. Moreover, the optimization problem is numerically friendly by virtue of the differentiability of the field *w.r.t.* the mesh vertices. Results show that we can offset a mesh while preserving sharp features of the original surface, restricting selected parts to quadric surfaces and penalizing intersections.**

***Index Terms*—Surface offsetting, winding number, geometry processing, variational, surface modeling**

I. Introduction

SURFACE offsetting is a fundamental operation in many applications such as manufacturing, modeling, meshing, *etc*. While mathematically, offsetting can be defined as the contour of the signed distance field of a surface, practical applications often demand more nuanced behavior. For example, the standard offsetting definition can be a good choice for collision detection [1]. However, in other scenarios, such as modeling, where designers often first design the surfaces of one side and offset them to obtain solids [2], [3], they may want to preserve the sharp features of the original surface. In general, different applications impose distinct requirements on the offsetting process. Please see Figure 2 and Figure 8 for different varieties of offsetting used in previous methods.

In this paper, we focus on the offsetting of manifold and oriented triangular surface meshes, a prevalent representation for 3D shapes. Offsetting via directly optimizing or moving vertices (with possible remeshing) of the explicit mesh were extensively explored in earlier works [4]–[8]. These methods are intuitive, but may easily produce intersection issues since they ignore the surrounding ambient space. Consequently,

The authors are with State Key Lab of CAD and CG, Zhejiang University, Hangzhou, 310027, China. E-mail: {hrsun, shuangwu2002}@zju.edu.cn, {bao, hj}@cad.zju.edu.cn.
Corresponding author: Hujun Bao.

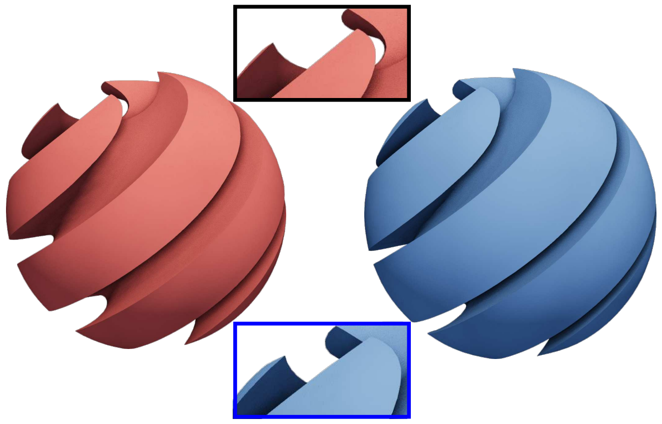


Fig. 1: Our variational approach can offset a 3D surface (a) into a new one (b) with the sharp features in (a) preserved.

a second category based on contouring volumetric implicit fields, typically signed distance fields, has gained popularity due to its inherent intersection-free robustness along with contouring techniques such as Marching Cubes [9]. However, flexibility is generally limited in the contouring procedure. Achieving customizable results or controlling the topology changes is challenging.

To bridge the gap between the flexibility of explicit methods and the robustness of implicit approaches, we propose a novel variational framework for surface offsetting. Our core idea is to leverage the strengths of both representations simultaneously. Similar to previous explicit methods, we treat the vertex locations as the optimization variables, but use the implicit field induced by the output mesh in the optimization objective to incorporate information about the ambient space, thereby helping to resist the intersection issues.

Such a framework requires the implicit field to be tightly bound to the explicit mesh, *i.e.*, differentiable *w.r.t.* the mesh vertex locations. Moreover, the field should have behavior similar to that of a distance field around the mesh and properly encode intersection behavior. A promising candidate is the recently proposed *Gaussian-smoothed winding number (GauWN)* [10], which is the convolution of the winding number field with a Gaussian kernel. However, it is only efficient enough for 2D closed polygons. Thus, we adopt the *regularized Poisson kernel* [11], which has been introduced in graphics to define the *regularized winding number* for the purpose of removing the singularity issues in point cloud geometry in reconstruction applications [12]. In this paper, we prove that the regularized winding number is also equivalent to convolving the winding number with the Gaussian kernel and can be used here.

This hybrid variational framework allows us to perform surface offsetting with unique advantages. By optimizing directly on the mesh, we can easily incorporate customizable shape regularizations, such as terms for preserving sharp features or quadric fitting. Simultaneously, the guidance from the differentiable implicit field ensures strong resistance to self-intersections, even for complicated models or multi-component surfaces undergoing coordinated offsetting. Our method operates without requiring an explicit background grid for the implicit field or additional effort to maintain consistency between the field and the mesh. Instead, it synchronizes the mesh evolution with the underlying smooth field information automatically by construction. We demonstrate that our method can reliably offset meshes with intricate details and open boundaries, while also providing optional control over topological adaptation.

In summary, the main contributions of this work are:

- A novel variational offsetting framework for 3D triangular meshes (including open boundaries) that combines the flexibility of explicit mesh optimization and the robustness inherited from an implicit smoothed generalized winding number field.
- We reverse the common approach of optimizing vertices toward the input mesh field's outer (or inner) contour. Instead, we introduce a more stable scheme in which the inner (or outer) contour of the resulting mesh's induced field is required to fit the input mesh. In this formulation, the ambient information of the resulting mesh becomes available through its field. As a consequence of the optimization, the smoothed winding number field is driven toward a state representing properly embedded surfaces, thereby penalizing intersections.
- We show that a simple shape regularization term suffices to preserve sharp features in the input mesh. An additional example regularization technique (quadric fitting) is also provided for more comprehensive shape control. We also demonstrate the ability to offset multi-component meshes.

II. Related Work

Our work builds upon advances in mesh deformation, offsetting techniques, and the use of implicit fields, particularly the winding number. We position our contribution relative to these areas below.

A. Modeling and deformation

In shape modeling, particularly in shape deformation, surface meshes are arguably the *de facto* representation. Many variational approaches [13]–[15], that employ Laplacian or other discrete differential operators on meshes, offer rich features for regularizing the mesh and its elements, such as feature preservation, high-quality mesh elements, *etc*. Beyond variational approaches, various techniques also exist to evolve a mesh through a flow (mainly for fairing) [16]–[18]. A comprehensive survey for such mesh-only techniques can be found in the book [19]. Our method adopts a flow-based strategy to offset the surface, with each step formulated as a variational problem capable of handling various regularizations.

Numerous methods also exist for modeling shapes using implicit fields. One representative technique is the level set method [20], which constructs and edits a scalar field on a background mesh, then defines the shape boundary as the contour of the scalar field. These methods discretize the field using a background mesh and are often referred to as Eulerian methods [21].

The combination of these two approaches is not new, particularly in simulation contexts. In fluid simulation, surface-tracking methods [22] excel at capturing thin features but face significant robustness challenges when handling topological changes. To address this, implicit fields are often employed to correct the mesh [23]–[25]. However, our approach only maintains a surface representation without requiring a background mesh (except for an AABB tree for acceleration).

B. Offset and its related techniques

Offsetting a mesh can be viewed as a special modeling and deformation task. Contouring a (signed) distance field by marching cubes [9] is possibly the most intuitive and straightforward way. However, it is well known that marching cubes is not well-suited for handling sharp features. Dual marching cubes [26] was proposed to handle sharp features, but it may produce non-manifold meshes. Recently, methods have been proposed to improve contour extraction specifically for offsetting tasks. For example, [27] proposes an interesting idea: once the topology of the mesh is correct, remeshing can be applied to reduce the geometrical error of the offsetting, so that the resolution of spatial discretization can be bounded. [28] is based on another observation: the difficulties of contouring the signed distance field lie in the lack of sufficient information about the distance field for the entire shape. To address this, they propose to process each shape primitive individually before merging the results.

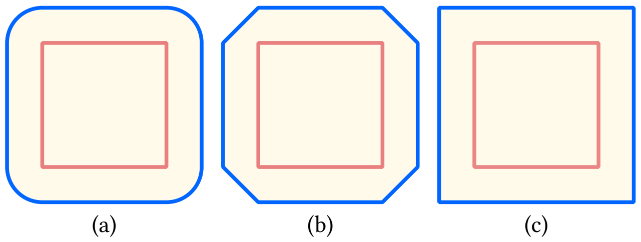


Fig. 2: **Varieties of offsetting of the red shape.** (a): fillet-like offsetting (standard definition, volumetric contouring based methods). (b): chamfer-like offsetting ([8]). (c): original-feature-preserved offsetting (Solidify modifier of Blender [29]).

While such field-based (or background-mesh based) methods can be relatively robust, they are mainly suitable for equal-distance offsetting and are difficult to extend to accommodate customizable shape requirements. A very recent work [30] is different: it can generate varying-distance offsets, but it

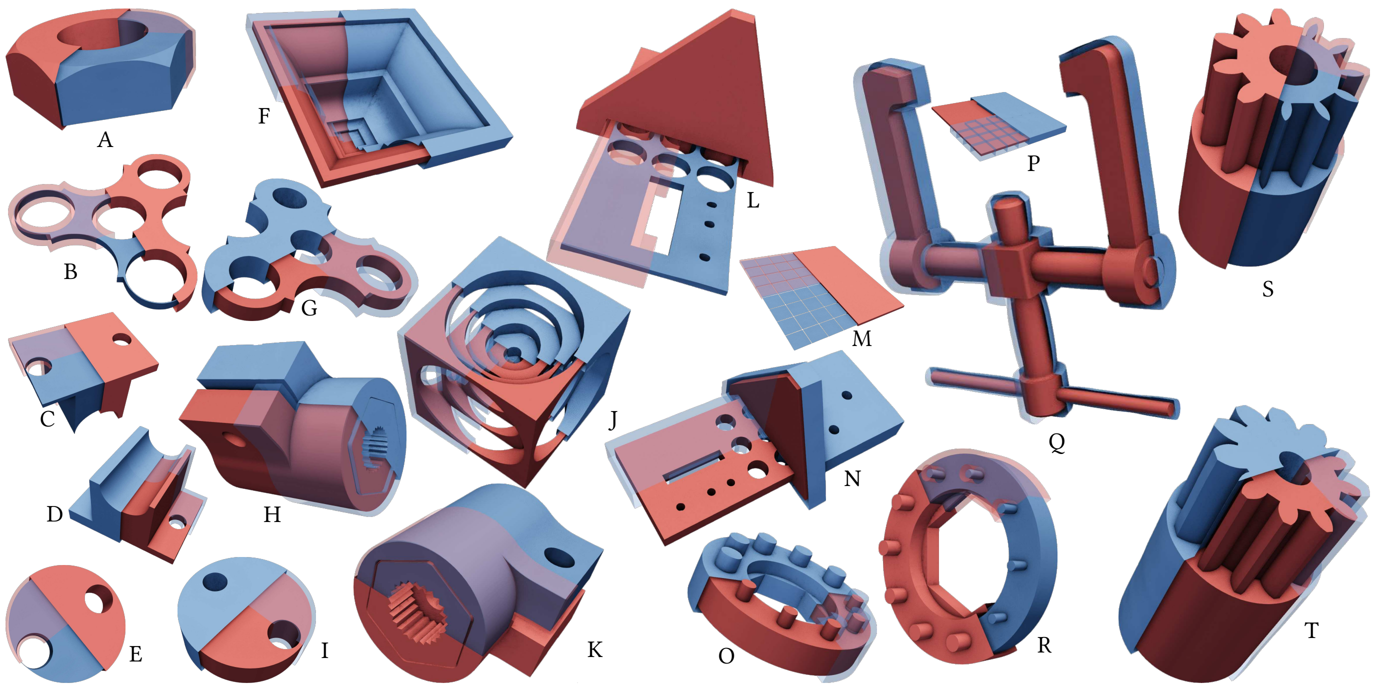


Fig. 3: **The gallery of our offsetting results.** *Red*: input surfaces; *Blue*: offset surfaces.

still does not have the ability to apply various shape regularizations (*e.g*. preserving input sharp features). On the other hand, surface mesh-based methods are more flexible at this point. For example, [8] can generate chamfer-like offsetting instead of standard fillet-like offsetting by performing mesh operations such as vertex relocation and splitting. The popular modeling package Blender [29] also provides tools with non-standard offsetting behaviors, including an "even thickness Solidify modifier" that can preserve surface features in a way similar to our method, demonstrating the demand for flexible offsetting behaviors from designers. For 2D polygons, there are straight-skeleton-based methods [31], [32] that can generate offset curves with similar behaviors in the inward direction. However, since explicit mesh representations lack inherent awareness of the ambient space, handling self-intersections (embedding issues) is challenging. Consequently, methods like [33] are often employed as a post-processing step to resolve intersections, but such pure mesh-based methods are still not robust enough. Other alternative techniques like IPC [34], [35] may also be used, especially considering that the normal flow results in [35] show some similar effects of offsetting. However, it is unclear how to define a proper energy to drive the flow so that the mesh vertices get the desired offset surface. It should be noticed that the normal flow in [35] is regularized by Laplacian smoothing, so the distance between the moved vertex and the input mesh may not be the specified flow speed multiplying time step even in a single step. By taking a mesh as the primary representation, our method gains the ability to perform customizable offsetting, but a distance field-like implicit field induced by the mesh is also used to enhance robustness.

There are also some techniques related to offsetting, such as shell construction [36] and boundary layer generation [37], but they generally emphasize different requirements: desired mesh structure (*e.g*. only containing prism elements) and quality of the elements in the region bounded by the input mesh and the offset one. Our target application, mesh offsetting, has no such requirement, but focuses more on distance control, shape regularization and intersection-free (*i.e*., globally intersection-free instead of just local bijection). For example, [36] may result in self-intersections though the shell map is still bijective locally. A post-processing of intersection detection and experimental shrinking may alleviate the issue, but it may change the shape of the top surface in an undesired way.

C. Implicit field and the winding number

A core requirement for the implicit field within our variational framework is efficient *differentiability w.r.t.* mesh vertex locations. While the study of implicit fields is a rich area, the differentiability *w.r.t.* mesh vertices is not the main focus of most existing work. For instance, the (signed) distance field has been successful in the contouring-based methods, but it is known to be only C^0 continuous and not smooth for piecewise linear surfaces [38], [39]. Its gradient *w.r.t.* the vertex locations even changes sparsity pattern at points with more than one nearest point on the surface. Recognizing the non-smoothness issue, various methods have been proposed to construct smoother implicit fields (*e.g*. [39]–[41]). However, efficient evaluation of the gradient *w.r.t.* the vertex locations is generally not the goal when they are designed. One notable exception is the *implicit moving least squares field (IMLS)* [42], which can be viewed as a smoothed SDF and has been employed in recent work [43] for collision handling with such differentiability properties used. But there is another important requirement for the implicit field in our application: the ability of *handling intersections*. Because SDFs are generally considered only well-defined for intersection-free surfaces [44], this limitation forces [43] to rely on specialized carving techniques to address self-intersection issues in collision applications.

Another widely used field is the winding number, which can be viewed as the generalization of the indicator field with a well-defined behavior for intersecting boundaries. It has been successfully extended to 3D meshes, including those with open boundaries as the *generalized winding number (GWN)* [45], [46]. Applications in 3D geometry, such as robust tetrahedral mesh generation [47], [48], have benefited greatly from these advancements. The vanilla winding number field is piecewise constant for closed boundaries, which means it cannot provide distance information. Additionally, the gradient is either zero or infinite. In response, [10] proposed convolving the field with a Gaussian kernel for useful *distance information* near the boundary and differentiability *w.r.t.* both the query point locations and vertex locations. They found efficient numerical methods for this Gaussian-smoothed winding number in the 2D closed boundary cases, using an extended form of the divergence theorem. However, it is complicated and challenging to extend that extended divergence theorem-based technique to 3D cases.

Through a different approach, the *regularized Poisson kernel* [11] was initially proposed to handle the singular integrals on the surface. It was recently introduced to graphics for defining *regularized winding number* [12]. They use the properties of the regularized Poisson kernel to avoid singularity when evaluating the field in point-cloud settings for surface reconstruction applications. We will later show that this formulation, which allows for efficient computation, is equivalent to convolving the GWN with a Gaussian kernel. It can thus be used as a signed distance field approximately near the boundary, which nicely matches our requirements.

III. Framework of Variational Mesh Offsetting

Let $F(\cdot; V)$ be a distance-like scalar field induced by a mesh surface $S(V)$ containing vertices $V = \{\mathbf{v}_i\}$, where its gradient near the mesh faces is similar to a signed distance field (up to a constant scaling). An intuitive way to offset a surface is to advect the vertices along the gradient of F:

$$\dot{\mathbf{v}}_i = o_i \nabla_{\mathbf{v}_i} F(\cdot; V), \tag{1}$$

where $o_i \in \mathbb{R}$ stands for the speed of offsetting, and $\nabla_{\mathbf{v}_i}$ is the gradient operator at $\mathbf{v}_i$. Methods that move the vertices along the normal work similarly, which just uses the signed distance field of the current mesh as F. If o_i is the same for all vertices, this can be approximately viewed as moving the vertices towards the o-contour of the input mesh's implicit field $F(\cdot; V)$ in a single step. [10] takes this viewpoint, and shows a variational way to advect the vertices $V^{(k+1)}$ towards the o-contour described by $\{o_i\}$ in the previous field $F^{(k)} = F(\cdot; V^{(k)})$:

$$\min_{V^{(k+1)}} \sum_i \left(F^{(k)}(\mathbf{v}_i^{(k+1)}) - \left[F^{(k)}(\mathbf{v}_i^{(k)}) - o_i \right] \right)^2. \tag{2}$$

Similar to methods that use only the mesh representation, Equation (2) has limitations in intersection handling as well. Though it involves an implicit field (*i.e.*, smoothed winding number) here, this field only characterizes the ambient information of the previous mesh with vertices $\{\mathbf{v}_i^{(k)}\}$ instead of the advected one with vertices $\{\mathbf{v}_i^{(k+1)}\}$. As acknowledged in their paper [10], it is required to utilize the smoothed winding number $F^{(k+1)} = F(\cdot; V^{(k+1)})$ of the advected mesh to prevent intersections. Moreover, it just asks $\mathbf{v}_i^{(k+1)}$ to lie on the o-contour of $F^{(k)}$. Without sufficient regularization, $\mathbf{v}_i^{(k+1)}$ is largely under-constrained, which results in a noisy mesh (see Figure 4 for an illustration), even though the vertices do lie on the o-contour. That is why an additional regularization term $\left\| V^{(k+1)} - V^{(k)} \right\|_2^2$ is indispensable in [10] for such a forward scheme.

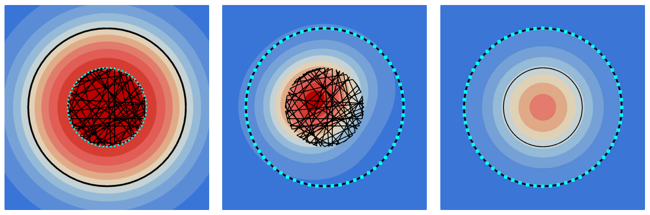


Fig. 4: **Under-constrained forward scheme *vs.* more stable backward scheme for inward offsetting of a black circle.** The background displays the smoothed winding number field with distance-like property associated with the input surface in (a) and the result surfaces in (b, c). (a): A severely distorted mesh can also minimize the forward scheme energy because it just asks vertices of *output* mesh (cyan) to reach the target contour of the *input* surface's implicit field. (c): On the contrary, the backward scheme asks the contour of *output* surface's implicit field to align precisely with the query points on the *input* surface (cyan). As shown in (b), the distorted mesh in (a) does not minimize the backward scheme energy.

A. Backward scheme

To address this issue, we propose advecting the mesh vertices in a backward scheme: the o-contour of the advected mesh should fit the previous mesh well:

$$\min_{V^{(k+1)}} \int_{\mathbf{s} \in \mathcal{S}(V^{(k)})} \left(\left[F^{(k+1)}(\mathbf{s}) - o(\mathbf{s}) \right] - F^{(k)}(\mathbf{s}) \right)^2 \mathrm{d}\mathbf{s}, \tag{3}$$

where $F^{(k+1)}$ is the implicit field induced by the advected mesh containing vertices $V^{(k+1)}$. To make a clear comparison to Equation (2), we temporarily take vertices in the previous mesh as the quadrature points $\mathbf{s}$ to approximate the above integral as

$$\min_{V^{(k+1)}} \sum_i \left(\left[F^{(k+1)}(\mathbf{v}_i^{(k)}) - o_i \right] - F^{(k)}(\mathbf{v}_i^{(k)}) \right)^2. \tag{4}$$

This formula looks similar to the forward scheme in Equation (2), but there are significant differences between them. In the forward scheme, the field $F^{(k)}$ is fixed, and only the query points $V^{(k+1)}$ are updated in this field. In the backward scheme, the query points $V^{(k)}$ are fixed, and we update the field $F^{(k+1)}$ by optimizing the advected vertices $V^{(k+1)}$. As demonstrated in Section V-A, the backward scheme is much more stable than the forward one when using only a few quadrature points (*i.e.*, triangle barycenters). Besides, when using the following Gaussian-smoothed winding number, the backward scheme implicitly penalizes intersections, as explained in the following section.

B. Smoothed winding number as the implicit field

Besides the distance-like property, to make the problem easy to solve, it is better to use a field that can be effectively differentiated *w.r.t.* the vertices. The Gaussian-smoothed winding number W_σ in [10] matches the requirements, and is a good candidate for the implicit field. It is defined as:

$$W_\sigma(\mathbf{q};\mathcal{S}(V)) = (w*G_\sigma)(\mathbf{q}) = \int_{\mathbb{R}^3} w\,(\mathbf{y})\,G_\sigma(\mathbf{q}-\mathbf{y})\,\mathrm{d}\mathbf{y}, \quad (5)$$

where w is the winding number field of the surface $\mathcal{S}(V)$; G_σ is the Gaussian kernel. For a detailed analysis of the distance-like behavior of its spatial gradient near the boundary, we refer the reader to [10, Sec. 4.5].

Compared to other distance-like fields, some of which may be even differentiable as well, there is a crucial advantage to using this field: being a convolution of the winding number field, it remains well-defined for intersecting surfaces. As discussed in Section V-B, properly using it can eliminate the need for additional collision handling during optimization. However, if one uses fields that are only well-defined for intersection-free surfaces, it becomes necessary to include additional intersection handling to keep the shapes generated during the optimization free of intersections.

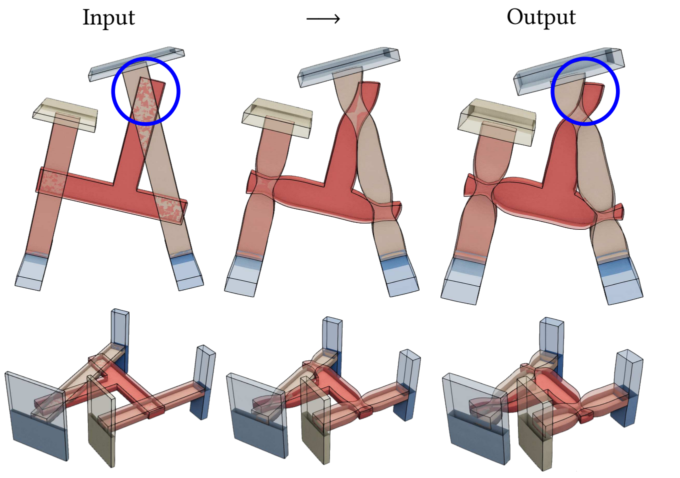


Fig. 5: **Robustness when intersections happen.** Being able to tolerate intersections is crucial for a robust offset method. For the input shape having severe intersections, our method can still offset it along with the effect of (locally) shrinking the intersected regions (*e.g.* the regions in blue circle).

To utilize this ability of the smoothed winding number, one could follow [10] to add a penalty to the backward scheme Equation (4). However, such a penalty is generally not necessary in our method, which greatly improves efficiency. In the following, we show that the carefully designed backward scheme implicitly penalizes intersections.

For an intersection-free surface $\mathcal{S}$, any point $\mathbf{s} \in \mathcal{S}$ must satisfy $\lim_{\epsilon\to 0^+} w(\mathbf{s}+\epsilon\mathbf{n}) = 0$ and $\lim_{\epsilon\to 0^-} w(\mathbf{s}+\epsilon\mathbf{n}) = 1$ [10]. After convolution smoothing, these values in the smoothed winding number field are around 0.5, but crucially still within the range $[0,1]$. Conversely, near intersection regions, the winding number values deviate significantly from this range, taking values such as $\cdots,-2,-1,+2,+3,\cdots$. The optimization in Equation (3) requires that field values $W_\sigma^{(k+1)}(\mathbf{s})$ at input surface points $\mathbf{s} \in \mathcal{S}^{(k)}$ (which are near the output surface $\mathcal{S}^{(k+1)}$ in such a geometry flow) should equal $W_\sigma^{(k)}(\mathbf{s}) + o(\mathbf{s})$. By appropriately selecting o, the target value can be kept within $[0,1]$ (see Section A.4 in the supplementary material). Thus, the optimization effectively penalizes field values outside this range, pushing them back toward valid values and consequently reducing intersections.

Though the backward scheme cannot guarantee that the resulting mesh to be intersection free, its ability of penalizing intersections helps to resolve minor intersections possibly occurring in the deformation process (see Figure 7). The stress test in Figure 5 further demonstrates the robustness of our method when the shapes have serious intersections.

C. Shape regularization for sharp feature preserving

Now we get an objective function in Equation (3) (denoted as E_W) to offset the mesh, which also incorporates intersection handling. By virtue of mesh representation, one can apply various mesh-based shape regularizations E_R easily to match the requirements arising from different applications:

$$\min_{V^{(k+1)}} E_W(V^{(k+1)}) + E_R(V^{(k+1)}), \quad (6)$$

Unlike the forward scheme in [10], it is not necessary to apply strong "smoothness" or "stability" regularizations to overcome the under-constrained issue, so the resulting shape can be more tightly controlled by E_R without being significantly affected by additional regularizations.

Here we propose a simple shape regularization that preserves sharp features of the input surface: For each pair of adjacent faces f_i and f_j sharing an edge, we add the following term to make the angle between the normals of the two faces close to the input one:

$$E_{\text{sharp}}(V^{(k+1)}) = \sum_{\langle f_i, f_j\rangle} \left(\left\langle \mathbf{n}_i^{(1)}, \mathbf{n}_j^{(1)} \right\rangle - \left\langle \mathbf{n}_i^{(k+1)}, \mathbf{n}_j^{(k+1)} \right\rangle \right)^2 . \quad (7)$$

IV. Numerical method

In this section, we will discuss how to compute the smoothed winding number and implement the offsetting algorithm. Further details can be found in Section A in the supplementary material.

A. Computation of the smoothed winding number

It is not practical to compute the smoothed winding number field W_σ directly using the convolution definition in Equation (5). An extended divergence theorem is used in [10] to transform the convolution into a boundary integral for 2D closed polygon curves. However, this strategy cannot be easily extended to 3D triangular meshes, and is infeasible for open boundaries.

But fortunately, the Gaussian smoothed winding number corresponds exactly to the boundary integral of the regularized Poisson kernel [11], which has been used in point cloud reconstruction under the name regularized winding number [12]. To make this point clear, we start from the definition of generalized winding number [45], [46] of a surface $\mathcal{S} \subset \mathbb{R}^3$:

$$w\left(\mathbf{q}\right) = \int_{\mathcal{S}} \nabla_{\mathbf{x}} \mathcal{G}(\mathbf{q}, \mathbf{x}) \cdot \mathbf{n} \, \mathrm{d}\mathbf{x}, \tag{8}$$

where $\mathbf{q}$ is the query point, $\mathcal{G}$ is the Green's function of the Laplace operator and $\mathbf{n}$ is the unit normal vector.

Substituting the definition of $w\left(\mathbf{y}\right)$ (Equation (8)) into Equation (5) yields:

$$W_{\sigma}\left(\mathbf{q}\right) = \int_{\mathbb{R}^3} G_{\sigma}(\mathbf{q}-\mathbf{y}) \left[\int_{\mathcal{S}} \nabla_{\mathbf{x}} \mathcal{G}(\mathbf{y}, \mathbf{x}) \cdot \mathbf{n} \, \mathrm{d}\mathbf{x} \right] \mathrm{d}\mathbf{y}. \tag{9}$$

By changing the order of integration, we have

$$\begin{aligned} W_{\sigma}\left(\mathbf{q}\right) &= \int_{\mathcal{S}} \mathbf{n} \cdot \underline{\left[\int_{\mathbb{R}^3 \setminus \{\mathbf{x}\}} G_{\sigma}(\mathbf{q}-\mathbf{y}) \nabla_{\mathbf{x}} \mathcal{G}(\mathbf{y}, \mathbf{x}) \, \mathrm{d}\mathbf{y} \right]} \mathrm{d}\mathbf{x} \\ &=: \int_{\mathcal{S}} \mathbf{n} \cdot \underline{X(\mathbf{q}, \mathbf{x})} \, \mathrm{d}\mathbf{x}. \end{aligned} \tag{10}$$

This derivation transforms the volumetric convolution (Equation (5)) into a boundary integral over the surface $\mathcal{S}$. The core task now is to evaluate the vector kernel $X(\mathbf{q}, \mathbf{x})$, which represents the convolution of the Green's function gradient $\nabla_{\mathbf{x}} \mathcal{G}(\mathbf{y}, \mathbf{x})$ with the Gaussian kernel $G_{\sigma}(\mathbf{q}-\mathbf{y})$ *w.r.t.* $\mathbf{y}$. The following theorem (proved in Section B in the supplementary material) shows the RWN field [12] integrated from the regularized Poisson kernel [11] is equivalent to the Gaussian convolved generalized winding number field.

Theorem 1. $X(\mathbf{q}, \mathbf{x})$ *has the analytical form of the regularized Poisson kernel [11] used by the regularized winding number (RWN) [12]:*

$$\frac{\mathbf{q}-\mathbf{x}}{2\pi r^3} (r g_{\sigma}(r) - \Phi_{\sigma}(r)), \tag{11}$$

where $r = \|\mathbf{q}-\mathbf{x}\|$, g_{σ} *is the 1-dimensional Gaussian kernel, and* Φ_{σ} *is the error function* $\Phi_{\sigma}(r) = \int_0^r g_{\sigma}(t) \, \mathrm{d}t = \frac{1}{2} \mathrm{erf}\left(\frac{r}{\sqrt{2}\sigma} \right)$.

As a consequence, the integral Equation (10) can be computed by applying a simple numerical quadrature method on the boundary.

Chen *et al.* [12] used this smooth field, but did not provide a formulation of the gradient *w.r.t.* the mesh vertices. Following the idea of [10], we give an efficient way to compute this derivative in Section A.1 in the supplementary material.

B. Offsetting algorithm and energy discretization

In the optimization, we discretize the energy Equation (3) as

$$\min_{V^{(k+1)}} \sum_{\mathbf{s}_i \in \mathcal{S}(V^{(k)})} |\mathbf{s}_i| \left(\left[W_{\sigma}^{(k+1)}(\mathbf{s}_i) - o_i \right] - W_{\sigma}^{(k)}(\mathbf{s}_i) \right)^2. \tag{12}$$

$\mathbf{s}_i$ and $|\mathbf{s}_i|$ denote the barycenter of i-th triangle in $S(V^{(k)})$ and the area of the triangle respectively. The number of $\mathbf{s}_i$ is roughly twice the number of $\mathbf{v}_i$ in Equation (4).

In all the following results, the weight w_{sharp} in the total regularization term $E_R = w_{\text{sharp}} E_{\text{sharp}}$ is $10^5 \overline{|f|}$ where $\overline{|f|}$ is the average triangle area at the k-th iteration. Of course, one can use a more accurate quadrature method for better accuracy. In this paper, we just use the simple one in Equation (12).

ALGORITHM 1: Offsetting algorithm

```
Input: $\mathcal{S}(V^{(1)})$;      // the initial surface
Input: $k_{\max}$ ; // the number of offsetting
       steps
$k \leftarrow 0$;
while $k < k_{\max}$ do
    $V^{(k+1)} \leftarrow$
      $\arg\min_{V^{(k+1)}} E_W(V^{(k+1)}) + E_R(V^{(k+1)})$;
    $V^{(k+1)} \leftarrow$ tangential relaxation$(V^{(k+1)})$;
    $k \leftarrow k + 1$;
end
```

The whole offsetting pipeline is summarized in Algorithm 1. The nonlinear optimization problem in each step is solved by the Levenberg-Marquardt method (see Section A.2 in the supplementary material). After each step, we perform a tangential relaxation to keep the mesh elements in a decent quality. Note that this relaxation tries to preserve the current shape. Otherwise, the resulting shape may deviate from the desired one specified by the shape regularization. For instance, replacing the tangential relaxation by common mesh smoothing will smooth the sharp features in an unexpected way. Please refer to Section A.5 in the supplementary material for the details.

V. Justifications and Performance

In this section, we will show some experiments to justify our technical choices and evaluate the performance. All experiments regarding timing were conducted on a desktop computer with an AMD Ryzen™ 9 5900X processor running at 4.4 GHz. The smoothed winding number and its derivatives are computed using 20 threads parallelization.

A. Comparison of backward and forward schemes

As discussed before, the forward scheme Equation (2) is not stable due to the under-constrained issue. One may think that it can be easily alleviated by using more quadrature points, but the following experiment shows that this approach is not very effective. In Figure 6, we use triangle barycenters as the

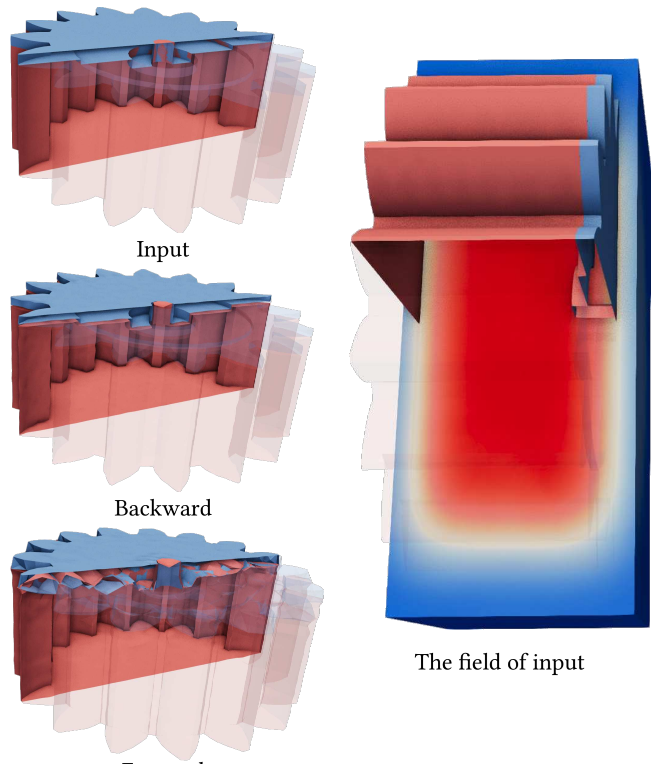


Fig. 6: **Comparison of the forward scheme and the backward scheme.** The input surface consists of two contacted components. Both schemes are applied for inward offsetting which should separate the two components. The shape regularization is deliberately omitted to highlight their fundamental differences. The results clearly demonstrate the superior stability of the backward scheme.

quadrature points in both the forward and backward schemes. Though the quadrature points are nearly doubled compared to using mesh vertices, the forward scheme is still unstable, which produces noisy shapes, while the backward scheme has no such artifacts. The results of the forward scheme may improve by using many more quadrature points, but this obviously increases the computational cost. More importantly, the forward scheme brings intersections, whereas the backward scheme successfully generates an offset surface without any intersections.

B. Rationale for selecting the smoothed winding number

For the distance-like field F, there could be different choices as long as it is efficient enough for optimization. It would be better to use a field capable of intersection handling. In this section, we provide a comparison of the smoothed winding number and the smoothed signed distance field IMLS [42], which was recently used for collision handling [43].

Signed distance fields are well-defined only for intersection-free surfaces, so optimization frameworks that rely on them typically require complicated handling mechanisms such as interior-point methods (*i.e.*, explicit collision detection and resolution). The IMLS [42] smoothed from the SDF, while having better differentiability, inherits this drawback from the signed distance field. As shown in Figure 7, the IMLS field cannot identify intersection regions, and therefore does not provide a proper gradient to push penetrated parts away. In contrast, the smoothed winding number, which combines distance information with the winding number, gives the desired gradient. Furthermore, the implicit moving least squares field generates spurious contours even with slight intersections, necessitating explicit collision detection mechanisms. The smoothed winding number elegantly overcomes these limitations, making it more suitable and easier to use for offsetting tasks with its robustness in handling intersections.

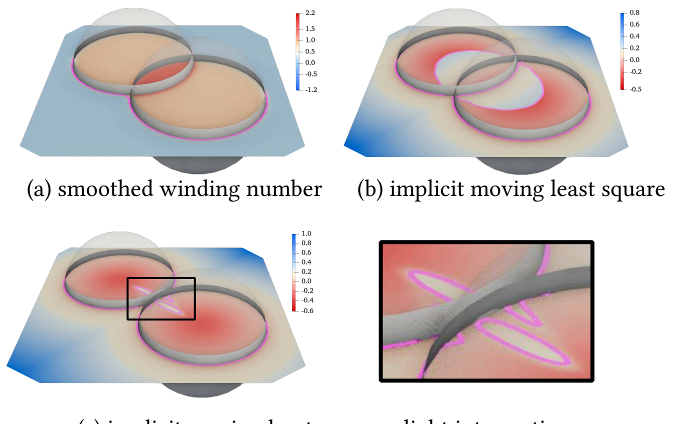
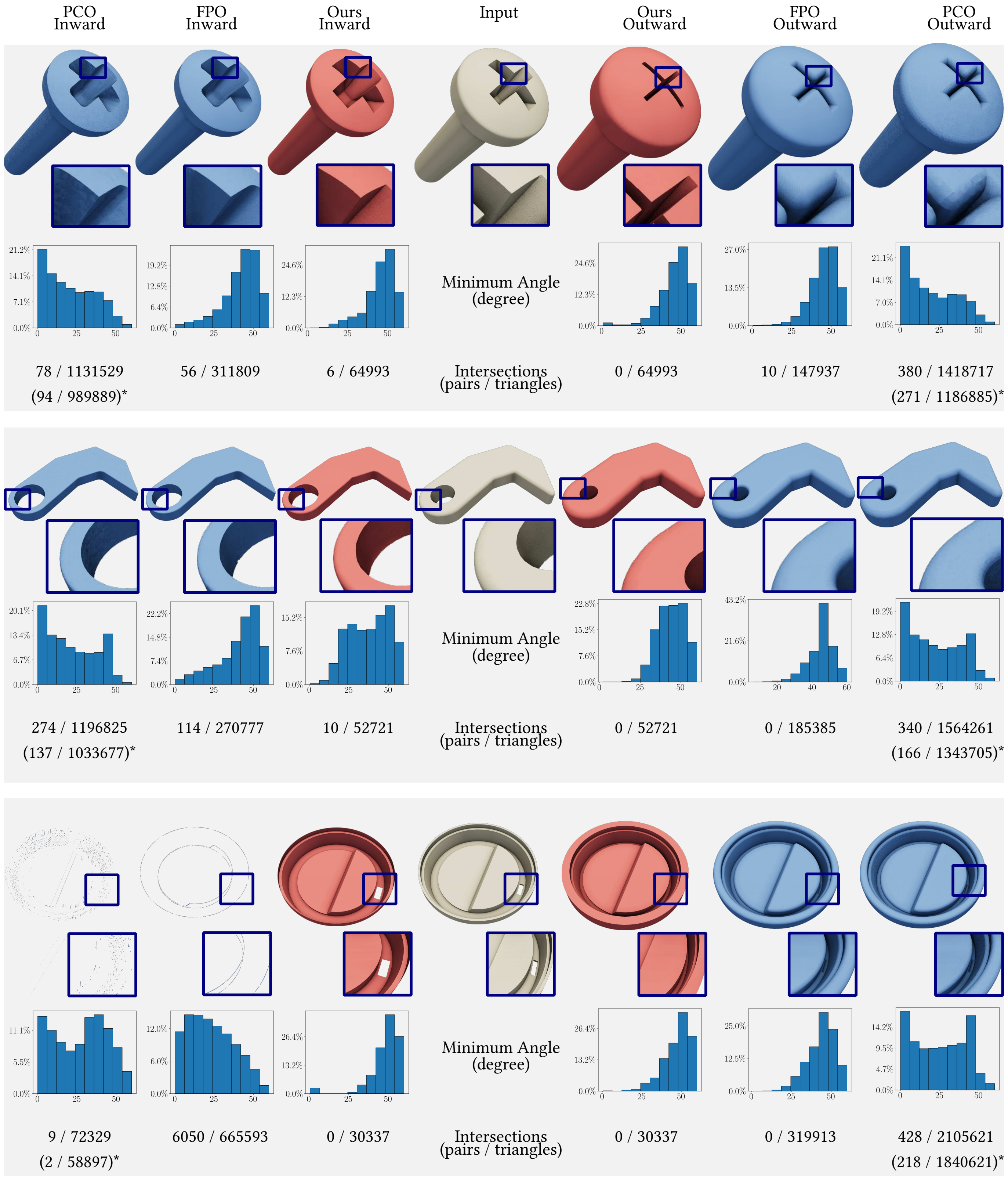

(a) smoothed winding number
(b) implicit moving least square
(c) implicit moving least square, slight intersection

Fig. 7: **Behaviors of different implicit fields.** As a smoothed SDF, the result from implicit moving least squares [42] is not well-defined when there are intersections (even very slight ones), and will introduce spurious contours. However, the smoothed winding number can still give a reasonable field for such regions.

C. Performance of offsetting

The computation of the smoothed winding number and the sparse matrix decomposition in the Levenberg-Marquardt optimization constitute the most computationally intensive components of our method (see Table I). Additional performance measurements for the offsetting results presented in Figure 3 are provided in Table 1 in the supplementary material. As our approach is based on a geometry flow, the computational time directly correlates with the offsetting distance, which determines the number of required steps. Our experiments reveal small differences between results obtained using a few large offset steps and those using many small steps (see Figure 9). It also implies that the deviation from the expected offset accumulates slowly over iterations. As possible future work, one could measure the distance between the current mesh and the input one, and apply an additional offset step with a proper offsetting speed o_i for each vertex $\mathbf{v}_i$ to compensate. Consequently, we fixed the expected offsetting parameter $\hat{o}$ to 0.2 across all other results presented in this paper (see Section A.4 in the supplementary material).



*: The official implementation of PCO will generate nearly coincident vertices. To reduce its influence, we also report the number of intersection pairs after merging vertices whose distance is within 1‰ of the diagonal length of the bounding box using the `vtkCleanPolyData` filter of VTK [49].

Fig. 8: **Comparison with volumetric-based methods.** The recent volumetric-based methods FPO [27] and PCO [28] focus on standard offsets, and try to preserve sharp features of the contour. PCO better preserves the features, whereas FPO generates smoother normal distributions. Sharply different from their results, our method preserves the sharp features in the input surface, which is difficult for volumetric-based methods. The number of intersected triangle pairs is reported by TetGen [50]. PCO will output polygon meshes instead of triangle meshes. Therefore, we perform a triangulation with the `vtkDataSetTriangleFilter` filter of VTK [49] before evaluating the intersection and element quality.

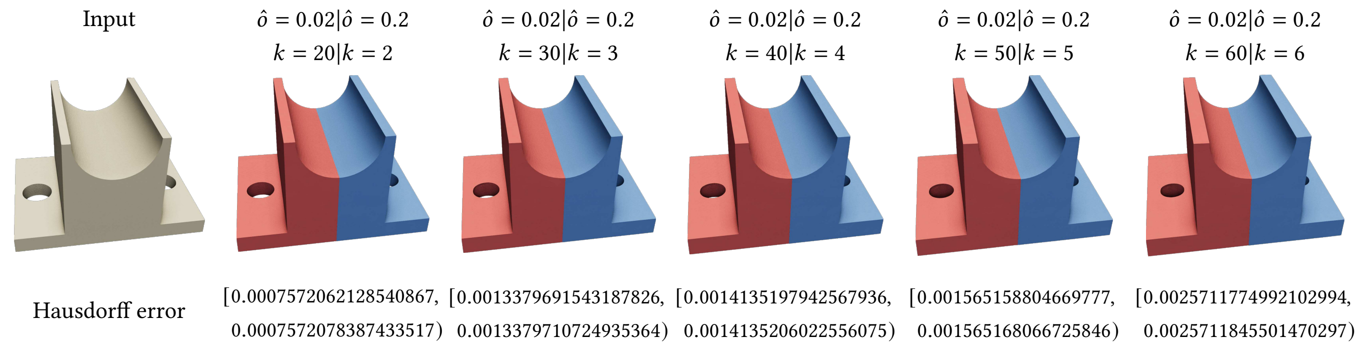


Fig. 9: **Large offset steps and small steps.** A few steps of large offset (blue) can make the results similar to those from many steps of small ones (red). “Hausdorff error” is the lower bound and upper bound of the two-sided Pompeiu-Hausdorff distance normalized by the bounding box diagonal length, reported by [51]. The error between the two results increases slowly with the offsetting iterations, but can be compensated by an addition step of offsetting.

k	Number of Evaluation			Time / s			
	V	G	F	V	G	F	Step
1	2 398 970	9 372 720	20	29.11	0.26	21.92	63.09
2	2 845 290	11 158 000	24	33.62	0.31	26.28	74.51
3	2 956 870	11 604 320	25	34.93	0.33	27.24	77.10
4	2 510 550	9 819 040	21	29.45	0.28	23.34	65.50
5	2 845 290	11 157 925	24	33.15	0.32	26.11	73.50
6	2 956 870	11 604 242	25	34.36	0.33	27.10	76.30
7	2 845 290	11 157 925	24	32.39	0.32	26.19	73.09
8	2 956 870	11 604 242	25	33.54	0.33	30.17	78.72
9	2 845 290	11 157 925	24	32.55	0.32	26.35	73.79
10	2 510 550	9 818 974	21	28.71	0.28	27.12	68.72

TABLE I: **The cost of each offset step in case S of Figure 3.** Left: “V” and “G” are the number of smoothed winding number value computations and the gradient (*w.r.t.* a vertex) computations respectively, while “F” is the number of matrix factorizations in the Levenberg-Marquardt method (*i.e.*, number of iterations in each offset step). Right are the total time for “V”, “G”, “F” and the whole step respectively.

VI. Results

Here, we provide some results to demonstrate the advantages of our method along with comparisons to representative offsetting methods. Some extensions of our method are at the end of this section.

A. *Comparison with volumetric-based offsetting ([27], [28])*

A standard offset surface can be viewed as a contour of a signed distance field. Volumetric field-based offsetting approaches (*e.g.*, FPO [27] and PCO [28]) are widely used for this purpose. These methods also talk about “feature preservation”: to preserve the sharp features in the contour of the distance field (not the sharp features in the input shape, see Figure 8). The concept of feature preservation in our method is different. Our approach ensures that the offset surface maintains the sharp features present in the original surface, or even certain algebraic surface properties (*e.g.*, quadric surfaces, see Section VI-D1). This type of feature preservation is particularly valuable in industrial design applications. To evaluate our method, we tested it on meshed CAD models randomly sampled from the ABC dataset [52], with results presented in Figure 3. All experiments used a Gaussian kernel parameter of $\sigma = 0.005$, with models normalized such that the major axis of the bounding box equals 1.

Unfortunately, in the experiments, all of FPO, PCO and ours generated self-intersected meshes. As we have described, our method can *penalize* intersections but cannot *prevent* intersections (see Section VII for discussion about factors that may lead to intersections). On the other hand, while a basic volumetric-based method that only performs marching cubes can guarantee the output surfaces to be intersection-free, we found the official implementations of these more advanced methods generate self-intersected meshes. The element quality (minimal angle) of the meshes is also reported in Figure 8, which shows that our method can generate meshes with similar element quality as FPO.

Unlike classical offsetting problems, to keep the sharp features of the input surface, the offsetting distance is not uniform. The distribution of these offsetting distances is illustrated in Figure 10, showing generally uniform distances in smooth regions while varying appropriately near sharp features.

Another important feature of our method is that it maintains homeomorphism between the original and offset surfaces, whereas typical volumetric field-based methods may alter the mesh connectivity and even surface topology. This homeomorphism preservation is particularly beneficial for applications such as padding layer (or boundary layer) generation (see Section VI-D2).

B. *Comparison with pure explicit offsetting (Blender [29])*

Among offsetting techniques based on explicit mesh representation, the built-in Solidify modifier in Blender [29] (a popular open-source software for 3D modeling) shares some similarities with our method. This modifier directly displaces mesh vertices to create the offset surface. Furthermore, the modifier includes an option (“Constraints” thickness mode) that aims to preserve sharp features of the original surface. However, the Solidify modifier operates as a purely explicit method, lacking inherent awareness of the ambient space. Consequently, it is likely to generate intersections for non-trivial shapes. Common failure cases include regions with

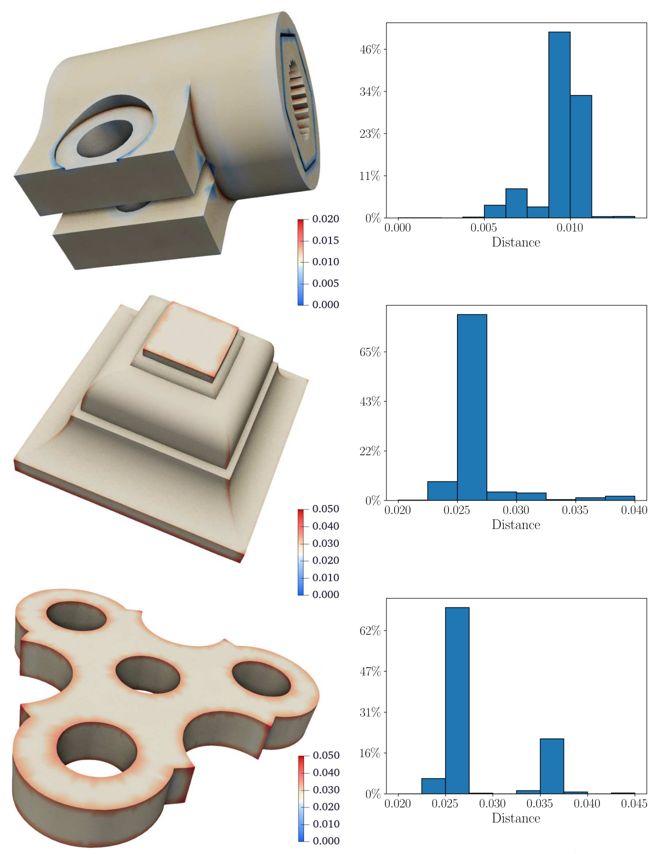


Fig. 10: **Distance distribution.** The distances between the vertices of the offset surface and the input surface are not constant due to the requirement of feature preservation, but the distances in smooth regions are generally uniform and dominate the distribution.

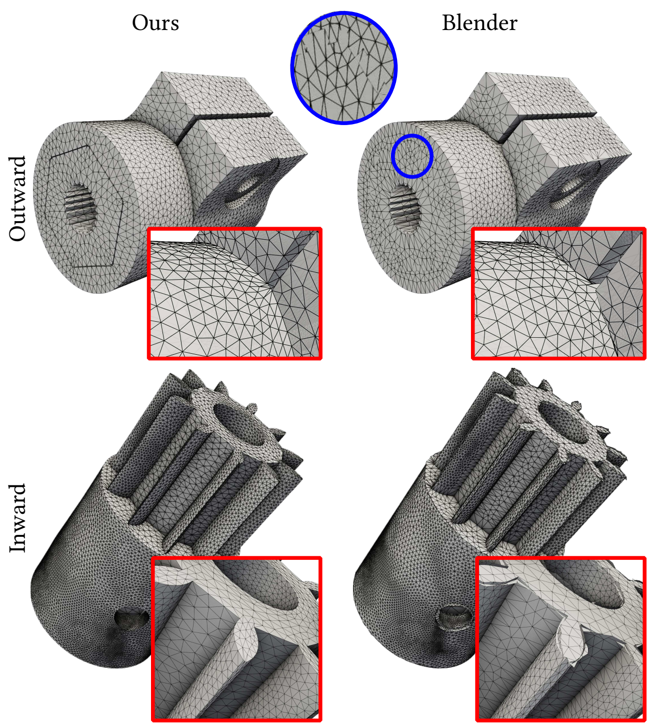


Fig. 11: **Comparison with Blender [29].** Blender's Solidify modifier, similar to our method, can preserve features of the original surface during offsetting (using "Constraints" thickness mode). However, as a purely explicit method lacking ambient space awareness, it is prone to generate self-intersections on non-trivial shapes. For instance, it brings defects for closely placed surface parts (Top, blue circle), outward offsetting at the concave corners (Top, red frame) or inward offsetting at the convex corners (Bottom, red frame). But our method handles these cases robustly because of its ambient space awareness obtained from the implicit field guidance.

closely spaced parts, outward offsets at the concave corners, and inward offsets at the convex corners (see Figure 11). Typically, fixing these issues requires extensive manual cleanup. In contrast, our method, guided by the differentiable smoothed winding number field, inherently incorporates spatial information, which handles such challenging cases automatically.

A recent preprint [53] and its former version [54] also find the offsetting positions using explicit mesh representation and can generate similar results to ours with the initial features preserved. We compared with the method PFP [54], which provides publicly released code. As shown in Figure 1 in the supplementary material, PFP fails to preserve the input sharp features in this experiment.

C. Offsetting multipart, intersected and open boundary surfaces

Some shapes are composed of multiple separated parts. Our method can easily offset such shapes as a whole and still keep the parts separate by virtue of the surface representation and winding number (see Figure 12). Of course, one can also fix a few parts and just offset the others, though such results are not provided here. Such a behavior of coordinated offsetting is very difficult to achieve for volumetric field-based methods and pure explicit ones.

Non-closed surfaces themselves are useful in many applications. For example, offsetting non-closed surfaces is a common task in industrial design where designers may first design a surface and then offset it to obtain a solid model. This task is not trivial for volumetric field-based methods. However, our method can handle this task naturally, as the smoothed winding number can be applied on open surfaces as shown in Figure 13.

D. Extensions

We also explore the potential of our method in other scenarios.

1) Additional shape regularization – quadric surface fitting: Using the explicit surface representation enables us to apply various shape regularizations flexibly and easily. With this ability, one can try to regularize a part of the surface as planar, cylinder, Bézier surface, or other parametric ones. Here, we provide an example of shape regularization other than the one for sharp feature preserving in Section III-C: fitting the vertices to a quadric surface (see Figure 14).

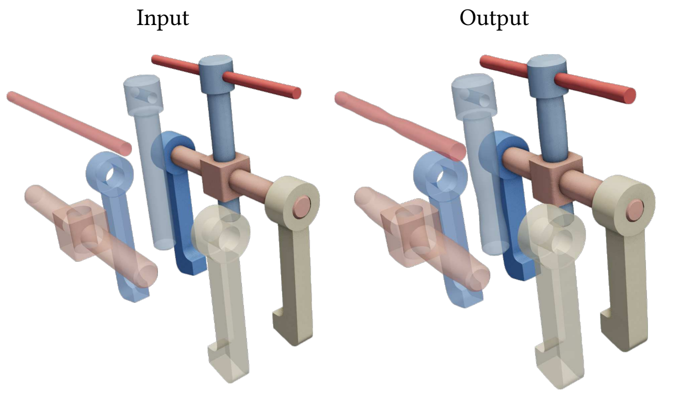


Fig. 12: **Coordinated offsetting a model with multiple parts assembly.** The input surface contains five connected components, contacted at the initial states. Our method offsets all the parts together coordinately. We show the disassembled model translucently.

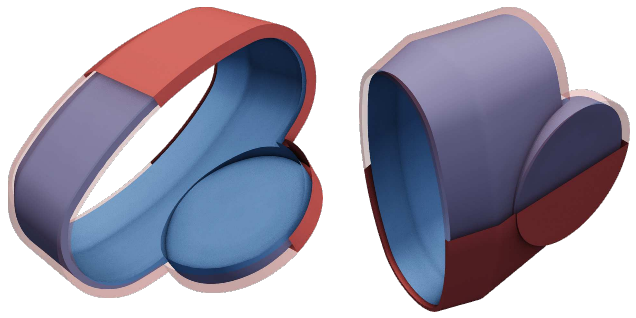


Fig. 13: **Offsetting an open boundary surface.** The generalized winding number has a contour smoothly extending the input surface. Its gradient direction aligns with the surface normal at the open boundary. The smoothed one has similar behavior and provides a reasonable direction. Therefore, our method can directly offset open boundary surface. *Red*: input surface. *Blue*: offset surface.

We first decompose the mesh surface into patches $\{P_i\}$, which can be parts from the input CAD model. Then, all the vertices in each patch are fitted to a quadric surface:

$$\mathbf{v}^\top A\mathbf{v} + 2\mathbf{v}^\top B + C = 0, \tag{13}$$

where $A \in \mathbb{R}^{3\times3}, B \in \mathbb{R}^3, C \in \mathbb{R}$ are the parameters of the quadric surface. As A is symmetric, there are in total 10 parameters $q_i \in \mathbb{R}^{10}$ for each patch P_i:

$$A_i = \begin{bmatrix} q_{i,1} & q_{i,2} & q_{i,3} \\ q_{i,2} & q_{i,4} & q_{i,5} \\ q_{i,3} & q_{i,5} & q_{i,6} \end{bmatrix}, \quad B_i = \begin{bmatrix} q_{i,7} \\ q_{i,8} \\ q_{i,9} \end{bmatrix}, \quad C_i = q_{i,10}. \tag{14}$$

We can then define a series of energy terms as the shape regularizations:

Quadric position fitting energy. The energy term for fitting the vertices to the quadric surface:

$$E_{\text{qpos}} = \sum_{P_i} \sum_{\mathbf{v}_j \in P_i} \left\| \mathbf{v}_j^\top A_i \mathbf{v}_j + 2\mathbf{v}_j^\top B_i + C_i \right\|^2. \tag{15}$$

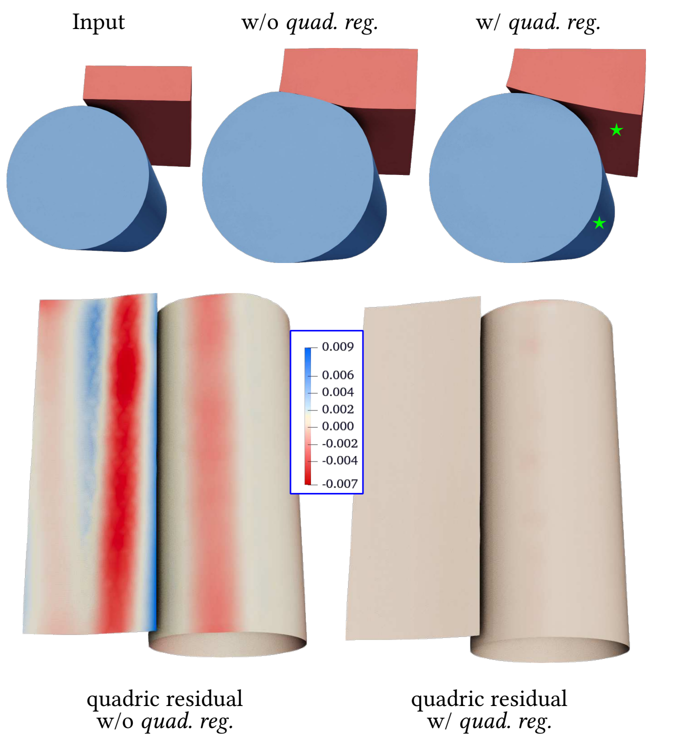


Fig. 14: **Quadric fitting as additional shape regularization.** Quadric regularization is applied to the two surface patches marked with ⋆. The regularized patches are maintained to be quadric surfaces during the offsetting.

Quadric regularization energy. The energy term for avoiding trivial solutions:

$$E_{\text{qreg}} = \sum_{P_i} \left(\sqrt{\|A_i\|_F^2 + 2\,\|B_i\|^2} - 1 \right)^2. \tag{16}$$

We set the weights $w_{\text{qpos}} = 1e10\overline{|f|}/|V|$ and $w_{\text{qreg}} = 1e9\overline{|f|}|P_i|/|V|$ for each patch P_i in Figure 14, where $|V|$ is the number of all vertices and $|P_i|$ is the number of vertices in patch P_i.

2) *Padding layer generation:* Our method keeps the connectivity of the input surface. Therefore, it is possible to generate the padding layer for the input surface using a swept-volume-like method as shown in Figure 15. We offset a quadrangle mesh for a few steps, then connect the offset vertices to form the padding layer. Note that the tangential relaxation method is not applied in this case, as it may perturb the locations of the vertices, especially near the singularities, which may lead to inverted elements in the padding layer. Of course, to apply our idea for high-quality boundary-layer generation, more detailed control (*e.g.*, thickness and distortion) of the volumetric mesh between the top and bottom meshes should be provided.

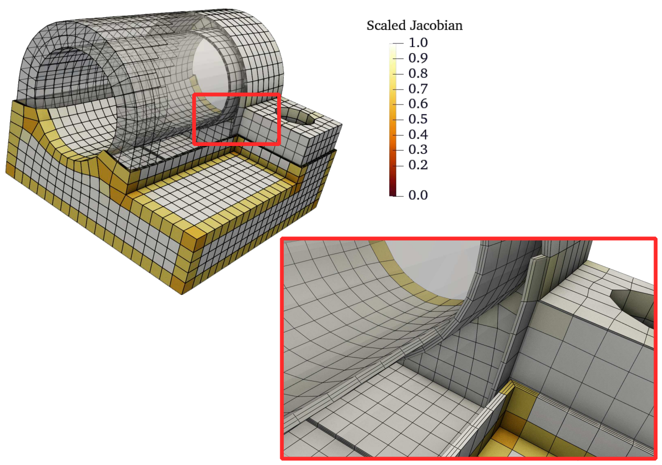


Fig. 15: **Padding layer generation.** Our method can maintain a one-to-one mapping of the vertices from the input surface to the offset surface. This allows us to offset a quadrangle mesh to obtain a hexahedral padding layer.

VII. Conclusion

We propose to offset a mesh surface by a hybrid representation for 3D objects, which uses the boundary surface mesh as the primary structure, supplemented by an associated smooth field. The field is defined by convolving the generalized winding number field of the surface mesh with a Gaussian kernel, and is differentiable *w.r.t.* both spatial locations and mesh vertices. Utilizing this differentiability, we develop a versatile offsetting technique that accommodates various shape regularizations. More importantly, this hybrid representation can help control intersection issues without explicit intersection detection and handling, and it shares similarities with the 2D counterpart [10].

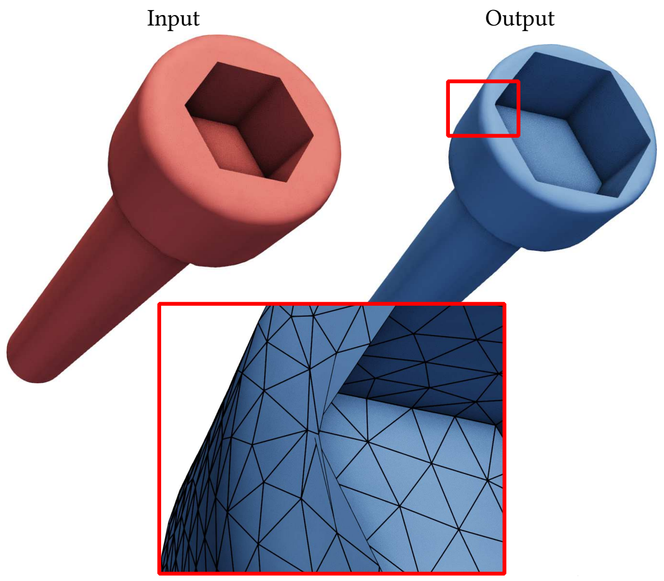


Fig. 16: **Limitations due to connectivity.** As we only use a simple tangential relaxation, triangles may flip due to the connectivity limitation. One may use a more advanced remeshing algorithm to avoid this issue.

Although our method and its current implementation show good behavior in our experiments, there are still limitations and can be further improved.

1) Intersections: Our method may produce intersections due to the following reasons:

- As explained before, our method only penalizes intersections implicitly through an energy term involving the smoothed winding number. This penalty may not be strong enough compared with heavily weighted shape regularizations. Besides, the energy is discretized by a simple quadrature, which may miss intersections.
- In regions with small local feature sizes (*e.g.*, sharp corners and thin plates), the smoothed winding number with a large Gaussian kernel approximates the SDF poorly and may even be unreliable for inside/outside classification (see [10]).
- The tangential relaxation may introduce flipped triangles (see Figure 16) due to the connectivity limitation.

Therefore, while deep intersections are generally avoided, shallow intersections may still occur. To alleviate the issue, one can adjust the weights, the quadrature scheme, and the radius of the Gaussian kernel, along with mesh adaptivity.

To further reduce intersections and even guarantee an intersection-free outcome, one can use fTetWild [48] as a post-processing step, as suggested by PCO [28], if connectivity does not need to be preserved. Since the possible intersections are usually shallow, the recent non-interior-point collision-resolving method [44] may also be used as a post-processing step when the input mesh is intersection-free and can be tetrahedralized. Interior-point methods such as IPC [34] or [35] may also be used along with our backward scheme, which is an interesting direction for future work.

2) Non-CAD models: Our method is designed for triangulated CAD models, which are composed of smooth patches with sharp features. In Figure 17, we show some results of applying our method to general models. For models with high-frequency details, it is hard to preserve the dihedral angles during offsetting; thus, the optimization gets stuck at small offsets due to the current shape regularization. Besides, the tangential relaxation easily introduces overlapping triangles on such models. To apply our method to such models, better regularization energy and element quality handling is required.

3) Extensions to other explicit shape representation: To apply the idea in this paper to arbitrary polygon meshes, point clouds, or even B-spline surfaces, the key is to compute the smoothed generalized winding number reliably and efficiently. Using the quadrature of the regularized Poisson kernel should be possible if the error is acceptable. The remaining task is to compute the derivatives *w.r.t.* the parameterization of the shape, *i.e.*, the vertex locations for polygon meshes, the point locations for point clouds, and the control points for B-spline surfaces.

Acknowledgments

We would like to thank all the anonymous reviewers for their valuable comments and suggestions. This work was supported

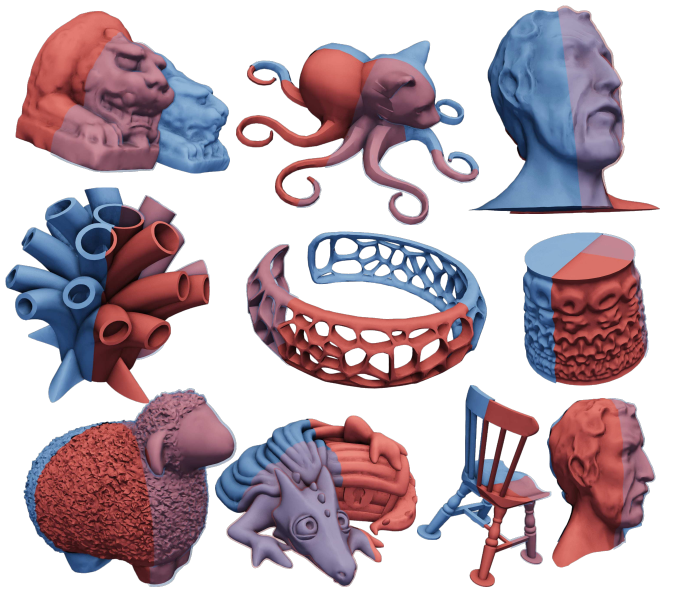


Fig. 17: **Non-CAD models.** Our method is potentially applicable to non-CAD models. Here we show some preliminary results by offsetting models from [48] dataset.

in part by NSFC (No. 62472382, No. U24B20154), "Pioneer" and "Leading Goose" R&D Program of Zhejiang Province (No. 2025C01086). We also thank for the generous support from a few friendly companies.

References

[1] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision detection for deformable objects," *Computer Graphics Forum*, vol. 24, no. 1, pp. 61–81, 2005. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2005.00829.x

[2] N. M. Patrikalakis and P. V. Prakash, "Free-form plate modeling using offset surfaces," *Journal of Offshore Mechanics and Arctic Engineering*, vol. 110, no. 3, pp. 287–294, 08 1988. [Online]. Available: https://doi.org/10.1115/1.3257064

[3] C. C.L. Wang and Y. Chen, "Thickening freeform surfaces for solid fabrication," *Rapid Prototyping Journal*, vol. 19, no. 6, pp. 395–406, 09 2013. [Online]. Available: https://doi.org/10.1108/RPJ-02-2012-0013

[4] X. Qu and B. Stucker, "A 3d surface offset method for stl-format models," *Rapid Prototyping Journal*, vol. 9, no. 3, pp. 133–141, Jan 2003. [Online]. Available: https://doi.org/10.1108/13552540310477436

[5] S.-J. Kim, D.-Y. Lee, and M.-Y. Yang, "Offset triangular mesh using the multiple normal vectors of a vertex," *Computer-Aided Design and Applications*, vol. 1, no. 1-4, pp. 285–291, 2004. [Online]. Available: https://doi.org/10.1080/16864360.2004.10738269

[6] C.-M. Chuang and H.-T. Yau, "A new approach to z-level contour machining of triangulated surface models using fillet endmills," *Computer-Aided Design*, vol. 37, no. 10, pp. 1039–1051, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448504002143

[7] M. Malosio, N. Pedrocchi, and L. M. Tosatti, "Algorithm to offset and smooth tessellated surfaces," *Computer-Aided Design and Applications*, vol. 6, no. 3, pp. 351–363, 2009. [Online]. Available: https://www.tandfonline.com/doi/abs/10.3722/cadaps.2009.351-363

[8] I. L. Yi, Y.-S. Lee, and H. Shin, "Mitered offset of a mesh using qem and vertex split," in *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, ser. SPM '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 315–320. [Online]. Available: https://doi.org/10.1145/1364901.1364945

[9] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, p. 163–169, Aug. 1987. [Online]. Available: https://doi.org/10.1145/37402.37422

[10] H. Sun, J. Wang, H. Bao, and J. Huang, "Gauwn: Gaussian-smoothed winding number and its derivatives," in *SIGGRAPH Asia 2024 Conference Papers*, ser. SA '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3680528.3687569

[11] J. T. Beale, W. Ying, and J. R. Wilson, "A simple method for computing singular or nearly singular integrals on closed surfaces," *Communications in Computational Physics*, vol. 20, no. 3, p. 733–753, 2016.

[12] H. Chen, B. Miller, and I. Gkioulekas, "3d reconstruction with fast dipole sums," *ACM Trans. Graph.*, vol. 43, no. 6, Nov. 2024. [Online]. Available: https://doi.org/10.1145/3687914

[13] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, ser. SGP '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 175–184. [Online]. Available: https://doi.org/10.1145/1057432.1057456

[14] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum, "Subspace gradient domain mesh deformation," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 1126–1134. [Online]. Available: https://doi.org/10.1145/1179352.1142003

[15] Z. Levi and C. Gotsman, "Smooth rotation enhanced as-rigid-as-possible mesh animation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 2, pp. 264–277, 2015.

[16] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, p. 317–324. [Online]. Available: https://doi.org/10.1145/311535.311576

[17] W. von Funck, H. Theisel, and H.-P. Seidel, "Vector field based shape deformations," *ACM Trans. Graph.*, vol. 25, no. 3, p. 1118–1125, Jul. 2006. [Online]. Available: https://doi.org/10.1145/1141911.1142002

[18] K. Crane, U. Pinkall, and P. Schröder, "Robust fairing via conformal curvature flow," *ACM Trans. Graph.*, vol. 32, 2013.

[19] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*. CRC Press, 2010. [Online]. Available: https://books.google.com.pe/books?id=AuXqBgAAQBAJ

[20] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton–jacobi formulations," *J. Comput. Phys.; (United States)*, vol. 79, no. 1, 11 1988. [Online]. Available: https://www.osti.gov/biblio/6694968

[21] P. Mullen, A. McKenzie, Y. Tong, and M. Desbrun, "A variational approach to eulerian geometry processing," *ACM Trans. Graph.*, vol. 26, no. 3, p. 66–es, Jul. 2007. [Online]. Available: https://doi.org/10.1145/1276377.1276459

[22] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Physics-inspired topology changes for thin fluid features," in *ACM SIGGRAPH 2010 Papers*, ser. SIGGRAPH '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1833349.1778787

[23] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Deforming meshes that split and merge," in *ACM SIGGRAPH 2009 Papers*, ser. SIGGRAPH '09. New York, NY, USA: Association for Computing Machinery, 2009. [Online]. Available: https://doi.org/10.1145/1576246.1531382

[24] F. Da, C. Batty, and E. Grinspun, "Multimaterial mesh-based surface tracking," *ACM Trans. Graph.*, vol. 33, no. 4, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2601097.2601146

[25] P. Heiss-Synak, A. Kalinov, M. Strugaru, A. Etemadi, H. Yang, and C. Wojtan, "Multi-material mesh-based surface tracking with implicit topology changes," *ACM Trans. Graph.*, vol. 43, no. 4, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3658223

[26] S. Schaefer and J. Warren, "Dual marching cubes: Primal contouring of dual grids," *Computer Graphics Forum*, vol. 24, no. 2, pp. 195–201, 2005. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2005.00843.x

[27] D. Zint, N. Maruani, M. Rouxel-Labb, and P. Alliez, "Feature-preserving offset mesh generation from topology-adapted octrees," *Computer Graphics Forum*, vol. 42, no. 5, p. e14906, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14906

[28] L. Wang, X. Wang, P. Wang, S. Chen, S. Xin, J. Guo, W. Wang, and C. Tu, "Pco: Precision-controllable offset surfaces with sharp features," *ACM Trans. Graph.*, vol. 43, no. 6, Nov. 2024. [Online]. Available: https://doi.org/10.1145/3687920

[29] Blender Online Community, *Blender - a 3D modelling and rendering package (Version 4.4.0)*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2025. [Online]. Available: http://www.blender.org

[30] D. Zint, Z. Chen, Y. Zhu, D. Zorin, T. Schneider, and D. Panozzo, "Topological offsets," *ACM Trans. Graph.*, vol. 44, no. 4, Jul. 2025. [Online]. Available: https://doi.org/10.1145/3731157

[31] P. Palfrader and M. H. and, "Computing mitered offset curves based on straight skeletons," *Computer-Aided Design and Applications*, vol. 12, no. 4, pp. 414–424, 2015. [Online]. Available: https://doi.org/10.1080/16864360.2014.997637

[32] S. Huber, "The topology of skeletons and offsets," in *Proceedings of the 34th European Workshop on Computational Geometry (EuroCG '18)*, Mar. 2018.

[33] W. Jung, H. Shin, and B. K. Choi, "Self-intersection removal in triangular mesh offsetting," *Computer-Aided Design and Applications*, vol. 1, no. 1-4, pp. 477–484, 2004. [Online]. Available: https://doi.org/10.1080/16864360.2004.10738290

[34] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: intersection-and inversion-free, large-deformation dynamics," *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392425

[35] Y. Fang, M. Li, C. Jiang, and D. M. Kaufman, "Guaranteed globally injective 3d deformation processing," *ACM Trans. Graph.*, vol. 40, no. 4, Jul. 2021. [Online]. Available: https://doi.org/10.1145/3450626.3459757

[36] Z. Jiang, T. Schneider, D. Zorin, and D. Panozzo, "Bijective projection in a shell," *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3414685.3417769

[37] R. V. Garimella and M. S. Shephard, "Boundary layer mesh generation for viscous flow simulations," *International Journal for Numerical Methods in Engineering*, vol. 49, no. 1-2, pp. 193–218, 2000. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-0207%2820000910/20%2949%3A1/2%3C193%3A%3AAID-NME929%3E3.0.CO%3B2-R

[38] B. Payne and A. Toga, "Distance field manipulation of surface models," *IEEE Computer Graphics and Applications*, vol. 12, no. 1, pp. 65–71, 1992.

[39] A. Biswas and V. Shapiro, "Approximate distance fields with non-vanishing gradients," *Graphical Models*, vol. 66, no. 3, pp. 133–159, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1524070304000049

[40] V. Rvachev, T. Sheiko, V. Shapiro, and I. Tsukanov, "Transfinite interpolation over implicitly defined sets," *Computer Aided Geometric Design*, vol. 18, no. 3, pp. 195–220, 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167839601000152

[41] J. Manson and S. Schaefer, "Analytic rasterization of curves with polynomial filters," *Computer Graphics Forum*, vol. 32, no. 2pt4, pp. 499–507, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12070

[42] R. Kolluri, "Provably good moving least squares," *ACM Trans. Algorithms*, vol. 4, no. 2, May 2008. [Online]. Available: https://doi.org/10.1145/1361192.1361195

[43] Y. Du, Y. Li, S. Coros, and B. Thomaszewski, "Robust and artefact-free deformable contact with smooth surface representations," *Computer Graphics Forum*, vol. 43, no. 8, p. e15187, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.15187

[44] H. Chen, E. Diaz, and C. Yuksel, "Shortest path to boundary for self-intersecting meshes," *ACM Trans. Graph.*, vol. 42, no. 4, Jul. 2023. [Online]. Available: https://doi.org/10.1145/3592136

[45] A. Jacobson, L. Kavan, and O. Sorkine-Hornung, "Robust inside-outside segmentation using generalized winding numbers," *ACM Trans. Graph.*, vol. 32, no. 4, Jul. 2013. [Online]. Available: https://doi.org/10.1145/2461912.2461916

[46] G. Barill, N. G. Dickson, R. Schmidt, D. I. W. Levin, and A. Jacobson, "Fast winding numbers for soups and clouds," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201337

[47] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo, "Tetrahedral meshing in the wild," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 60:1–60:14, Jul. 2018. [Online]. Available: http://doi.acm.org/10.1145/3197517.3201353

[48] Y. Hu, T. Schneider, B. Wang, D. Zorin, and D. Panozzo, "Fast tetrahedral meshing in the wild," *ACM Trans. Graph.*, vol. 39, no. 4, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392385

[49] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit (4th ed.)*. Kitware, 2006.

[50] H. Si, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw.*, vol. 41, no. 2, Feb. 2015. [Online]. Available: https://doi.org/10.1145/2629697

[51] L. Sacht and A. Jacobson, "Cascading upper bounds for triangle soup pompeiu-hausdorff distance," *Computer Graphics Forum*, vol. 43, no. 5, p. e15129, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.15129

[52] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[53] H. Cao, G. Xu, R. Gu, J. Xu, X. Zhang, T. Rabczuk, Y. Luo, and X. Gao, "Robust and feature-preserving offset meshing," 2024. [Online]. Available: https://arxiv.org/abs/2412.15564

[54] H. Cao, G. Xu, R. Gu, J. Xu, X. Zhang, and T. Rabczuk, "A parallel feature-preserving mesh variable offsetting method with dynamic programming," 2023. [Online]. Available: https://arxiv.org/abs/2310.08997

Haoran Sun received the B.Eng. degree from College of Computer Science and Technology, Zhejiang University in 2021, and is currently pursuing the Ph.D. degree at State Key Lab of CAD and CG, Zhejiang University. His research interests include computer graphics and geometry processing.

Shuang Wu received the B.Sc. degree from School of Mathematics, Southeast University in 2024, and is currently pursuing the Ph.D. degree at State Key Lab of CAD and CG, Zhejiang University. His research interests include computer graphics and geometry processing.

Hujun Bao received the B.Sc. and Ph.D. degrees in applied mathematics from Zhejiang University in 1987 and 1993, respectively. He is currently a Cheung Kong professor with the State Key Lab of CAD and CG, Zhejiang University. His research interests include geometry computing, vision computing, real-time rendering, and virtual reality.

Jin Huang received the Ph.D. degree from Computer Science Department, Zhejiang University, in 2007. He is currently a professor with the State Key Lab of CAD and CG, Zhejiang University, China. His research interests include geometry processing and physically based simulation. He was a reviewer of ACM SIGGRAPH, Eurographics, Pacific Graphics, IEEE TVCG, *etc*. He was the recipient of the Excellent Doctoral Dissertation Award of China Computer Federation for his PhD.

Supplementary Material Variational Mesh Offsetting by Smoothed Winding Number

Haoran Sun, Shuang Wu, Hujun Bao*, and Jin Huang

In this supplementary material, we provide some details and additional results of the method.

Appendix A
Implementation details

In this section, we provide the details of the implementation of our method.

A.1 Numerical methods for the smoothed winding number and its gradients

A.1.1 Gradients

The field value computation in our paper is the same as [1], but [1] does not give the formula for the derivative computation. Here, we extend the method of gradients computation from [2] to the 3D case instead of computing the exact gradients of this field.

For the triangle face f with vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$, assuming $W_\sigma^f(\mathbf{q})$ to be the contribution of f to the overall smoothed winding number field, we compute its gradients *w.r.t.* the vertex locations as

$$\frac{\partial}{\partial \mathbf{v}_i} W_\sigma^f(\mathbf{q}; \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) = \int_f G_\sigma(\mathbf{q}-\mathbf{x}) \frac{\partial d(\mathbf{x})}{\partial \mathbf{v}_i} \, \mathrm{d}\mathbf{x}, \tag{A1}$$

where $d(\mathbf{x})$ is the signed distance from $\mathbf{x}$ to the plane that f lies on, its gradients being

$$\frac{\partial d(\mathbf{x})}{\partial \mathbf{v}_i} = \frac{\mathbf{n}\mathbf{n}^\top}{2|f|}(\mathbf{x}-\mathbf{v}_j) \times (\mathbf{v}_k - \mathbf{v}_j), \tag{A2}$$

where $|f|$ stands for the area of the triangle f, $\mathbf{n}$ is the unit normal of f, and i, j, k is the cyclic permutation of $1, 2, 3$.

Note that the symbol W_σ^f by itself does not have an exact meaning. Only its gradients as a whole are defined and can be accumulated over all the faces to obtain the gradients of the smoothed winding number field.

When the surface is simple and closed (without self-intersections), the gradients above equal the exact gradients of the smoothed winding number field. When the surface has open boundaries, this formulation only captures the local jump of the winding number along the normal direction of the surface and ignores the gradient influences introduced by the smoothness of the generalized winding number. This behavior is more desirable in our applications.

A.1.2 Numerical method

The integral can be computed by numerical quadrature efficiently. Following [3], we sample 6 quadrature points $\{\mathbf{x}_k^f\}$ with weights $\{w_k^f\}$ on each face f. Then, the smoothed winding number field in Equation (10) in the main paper can be approximated as

$$W_\sigma(\mathbf{q}) \approx \sum_{f\in F} \sum_{k=1}^{6} w_k^f \mathbf{n}_f \cdot X(\mathbf{q}, \mathbf{x}_k^f), \tag{A3}$$

and the gradients Equation (A1) are computed by

$$\frac{\partial}{\partial \mathbf{v}_i} W_\sigma(\mathbf{q}) \approx \sum_{f\in\mathcal{N}(\mathbf{v}_i)} \sum_{k=1}^{6} w_k^f G_\sigma(\mathbf{q}-\mathbf{x}_k^f) \frac{\partial d(\mathbf{x}_k^f)}{\partial \mathbf{v}_i}, \tag{A4}$$

where $\mathcal{N}(\mathbf{v}_i)$ is the one-ring neighborhood of vertex $\mathbf{v}_i$. The same as [2], the gradients *w.r.t.* spatial location are simply:

$$\frac{\partial}{\partial \mathbf{q}} W_\sigma(\mathbf{q}) = -\sum_i \frac{\partial}{\partial \mathbf{v}_i} W_\sigma(\mathbf{q}). \tag{A5}$$

In contrast to [2], we use a more common BVH construction like [4] for acceleration instead of using a special BVH construction to meet their closed-boundary requirements. More details can be found in Section A.3.

A.2 Nonlinear optimization solver

For the optimization problem (Equation (6) in the main paper), we use the Levenberg-Marquardt algorithm, which is a common damped Gauss-Newton algorithm. The damping factor λ is updated by the following rule:

$$\lambda \leftarrow \begin{cases} \lambda/2 & \text{if } E^{(k+1)} < E^{(k)}, \\ \lambda \times 3 & \text{if } E^{(k+1)} \geq E^{(k)}. \end{cases} \tag{A6}$$

A step is accepted if $E^{(k+1)} \leq 1.05E^{(k)}$, *i.e.*, a slight increase of the energy is allowed to increase the convergence speed.

To reduce the computational cost of matrix factorization and allocation, in each offset step, we fix the sparsity pattern of the Jacobian matrix. Although the derivative at sample

The authors are with State Key Lab of CAD and CG, Zhejiang University, Hangzhou, 310027, China. E-mail: {hrsun, shuangwu2002}@zju.edu.cn, {bao, hj}@cad.zju.edu.cn.
Corresponding author: Hujun Bao.

point $\mathbf{s}$ is non-zero for all the vertices because of Gaussian convolution, it decays fast. So as an approximation, we only compute the derivative of the smoothed winding number for each sample point $\mathbf{s}$ *w.r.t.* at most 8 nearest vertices within 10σ selected at the beginning of the step, thus the sparsity pattern of the Jacobian matrix is fixed in each offset step.

A.3 BVH acceleration

For the input triangle mesh, the exact smoothed winding number field can be evaluated by summing over all faces

$$W_\sigma(\mathbf{q}) = \sum_{f\in F} \int_f \mathbf{n}_f \cdot X(\mathbf{q},\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{A7}$$

leading to a complexity of $O(|F|)$. Therefore, we adopt the similar BVH-based approximation as [4] when evaluating the smoothed winding number field and its gradients.

Each BVH node is associated with a cluster of triangles enclosed by its bounding box, the set of triangles denoted as B. The cluster is represented by the area-weighted barycenters of the triangles:

$$\mathbf{p} = \sum_{f\in B} |f|\mathbf{c}_f \Big/ \sum_{f\in B} |f|. \tag{A8}$$

When the query point $\mathbf{q}$ is far away from the BVH node, we use the first order Taylor expansion at the cluster center $\mathbf{p}$:

$$\begin{aligned}\mathbf{n}\cdot X(\mathbf{q},\mathbf{x}) \approx & \mathbf{n}\cdot X(\mathbf{q},\mathbf{p}) \\ & + (\mathbf{n}\otimes(\mathbf{x}-\mathbf{p}))\cdot \nabla_{\mathbf{x}} X(\mathbf{q},\mathbf{p}).\end{aligned} \tag{A9}$$

The exact form of $\nabla_{\mathbf{x}} X(\mathbf{q},\mathbf{p})$ is

$$\nabla_{\mathbf{x}} X(\mathbf{q},\mathbf{p}) = \frac{1}{4\pi r^3}(I-3\hat{\mathbf{r}}\otimes\hat{\mathbf{r}})\Psi(r) + G_\sigma(\mathbf{q}-\mathbf{p})\hat{\mathbf{r}}\otimes\hat{\mathbf{r}}, \tag{A10}$$

where $\hat{\mathbf{r}} = (\mathbf{q}-\mathbf{p})/r$, and $\Psi(r) = 2(\Phi_\sigma(r) - r g_\sigma(r))$. However, since $\Psi(r) \approx 1$ and $G_\sigma(\mathbf{q}-\mathbf{p}) \approx 0$ when r is large, we use a simpler and more efficient form as

$$\nabla X(\mathbf{q},\mathbf{p}) \approx \frac{I - 3\hat{\mathbf{r}}\otimes\hat{\mathbf{r}}}{4\pi r^3}, \tag{A11}$$

i.e., the same form of the GWN [4].

When the query point $\mathbf{q}$ is far enough away from the node, *i.e.*, r is larger than β ($\beta = 1.75$ in our implementation) times the diagonal length of its bounding box, we use the far-field approximation

$$\begin{aligned} W_\sigma^B(\mathbf{q}) = & \left(\sum_{f\in B} |f|\mathbf{n}_f\right)\cdot X(\mathbf{q},\mathbf{p}) \\ & + \left(\sum_{f\in B} |f|\mathbf{n}_f \otimes (\mathbf{c}_f - \mathbf{p})\right)\cdot \frac{I-3\hat{\mathbf{r}}\otimes\hat{\mathbf{r}}}{4\pi r^3}\end{aligned} \tag{A12}$$

for values. Otherwise, we adopt quadrature rules to evaluate the integrals within each triangle.

A.4 Setting of $\{o_i\}$

Simply setting o_i to a constant for all s_i in Equation (12) in the main paper is not good enough, as the minimizer of

$$\left(\left[W_\sigma^{(k+1)}(\mathbf{s}_i) - o_i\right] - W_\sigma^{(k)}(\mathbf{s}_i)\right)^2 \tag{A13}$$

may be a value $W_\sigma^{(k+1)}(\mathbf{s}_i) > 1$ or $W_\sigma^{(k+1)}(\mathbf{s}_i) < 0$, which we will not accept as it indicates an embedding with issues. Therefore, we set o_i by the following simple heuristic:

$$o_i = \operatorname{clamp}\left(W_\sigma^{(k)}(\mathbf{s}_i) + \hat{o}, 0.05, 0.95\right) - W_\sigma^{(k)}(\mathbf{s}_i), \tag{A14}$$

where $\hat{o}$ is the global expected offset, and we use $\hat{o} = 0.2$ by default.

A.5 Tangential relaxation

In our implementation, the following simple tangential relaxation from [5] is used.

For a corner vertex, *i.e.*, a vertex connected by more than two feature edges (we use the CAD patch boundaries as feature edges; they can also be defined by a dihedral-angle threshold), we fix such vertices during the tangential relaxation.

For a vertex $\mathbf{v}$ on sharp edges, *i.e.*, a vertex connected by two feature edges, we first find a smoothed location $\hat{\mathbf{v}}$, then project it to the sharp direction:

$$\mathbf{v} \leftarrow \mathbf{v} + \left\langle \hat{\mathbf{v}} - \mathbf{v}, \frac{\mathbf{v}_+ - \mathbf{v}_-}{\|\mathbf{v}_+ - \mathbf{v}_-\|}\right\rangle, \tag{A15}$$

where $\hat{\mathbf{v}} = \frac{1}{|e_-|+|e_+|}(|e_-|\mathbf{m}_- + |e_+|\mathbf{m}_+)$ and $\mathbf{m}_-, \mathbf{m}_+$ are the midpoints of the next and previous sharp edges, and $|e_-|, |e_+|$ are the lengths of the next and previous sharp edges.

For a non-feature vertex $\mathbf{v}$, we first find a smoothed location $\hat{\mathbf{v}}$, then project it to the tangent plane:

$$\mathbf{v} \leftarrow \hat{\mathbf{v}} - \langle \hat{\mathbf{v}} - \mathbf{v}, \mathbf{n}\rangle, \tag{A16}$$

where $\hat{\mathbf{v}} = \frac{1}{\sum |f_i|}\sum |f_i|\mathbf{b_i}$, $\mathbf{b_i}$ is the barycenter of the i-th triangle incident to $\mathbf{v}$ and $\mathbf{n}$ is the vertex normal.

Appendix B
Proofs of Theorem 1 in the main paper

The bivariate vector field $X(\mathbf{q},\mathbf{x})$ is defined as

$$X(\mathbf{q},\mathbf{x}) = \int_{\mathbb{R}^3\setminus\{\mathbf{x}\}} G_\sigma(\mathbf{q}-\mathbf{y})\nabla_{\mathbf{x}}\mathcal{G}(\mathbf{y},\mathbf{x})\,\mathrm{d}\mathbf{y}. \tag{A17}$$

We need to prove that the above equals to Equation (A25).

Lemma 1. *$X(\mathbf{q},\mathbf{x})$ satisfies*

$$\nabla_{\mathbf{q}}\cdot X(\mathbf{q},\mathbf{x}) + G_\sigma(\mathbf{q}-\mathbf{x}) = 0. \tag{A18}$$

Proof. This lemma has appeared in [1, Equation (12), (47) - (49)]. It can also be proved in more detail as follows.

To avoid the singularity of the Green's function, we write $X(\mathbf{q},\mathbf{x})$ as

$$\begin{aligned} X(\mathbf{q},\mathbf{x}) &= \int_{\mathbb{R}^3\setminus\{\mathbf{x}\}} G_\sigma(\mathbf{q}-\mathbf{y})\nabla_{\mathbf{x}}\mathcal{G}(\mathbf{y},\mathbf{x})\,\mathrm{d}\mathbf{y} \\ &= \lim_{\substack{r\to 0\\ R\to+\infty}} \int_{\Omega(\mathbf{x},r,R)} G_\sigma(\mathbf{y}-\mathbf{q})\nabla_{\mathbf{x}}\mathcal{G}(\mathbf{y},\mathbf{x})\,\mathrm{d}\mathbf{y} \\ &=: \lim_{\substack{r\to 0\\ R\to+\infty}} \tilde{X}(\mathbf{q},\mathbf{x}), \end{aligned} \tag{A19}$$

where $\Omega = \Omega(\mathbf{x}, r, R) = \{\mathbf{y} \in \mathbb{R}^3 \mid r \le \|\mathbf{y}-\mathbf{x}\| \le R\}$. By taking the divergence *w.r.t.* $\mathbf{q}$ and using the symmetry of both G_σ and $\mathcal{G}$, we have

$$\begin{aligned} \nabla_{\mathbf{q}}\cdot\tilde{X}(\mathbf{q},\mathbf{x}) &= \nabla_{\mathbf{q}}\cdot\int_\Omega G_\sigma(\mathbf{y}-\mathbf{q})\nabla_{\mathbf{x}}\mathcal{G}(\mathbf{y},\mathbf{x})\,\mathrm{d}\mathbf{y} \\ &= \int_\Omega \langle\nabla_{\mathbf{q}}G_\sigma(\mathbf{y}-\mathbf{q}),\nabla_{\mathbf{x}}\mathcal{G}(\mathbf{y},\mathbf{x})\rangle\,\mathrm{d}\mathbf{y} \\ &= \int_\Omega \langle\nabla G_\sigma(\mathbf{y}-\mathbf{q}),\nabla_{\mathbf{y}}\mathcal{G}(\mathbf{y},\mathbf{x})\rangle\,\mathrm{d}\mathbf{y}. \end{aligned} \tag{A20}$$

Since $\mathcal{G}(\mathbf{y},\mathbf{x})$ is harmonic in Ω, by applying Green's first identity, we have

$$\begin{aligned} \nabla_{\mathbf{q}}\cdot\tilde{X}(\mathbf{q},\mathbf{x}) = &-\int_\Omega G_\sigma(\mathbf{y}-\mathbf{q})\Delta\mathcal{G}(\mathbf{y},\mathbf{x})\,\mathrm{d}\mathbf{y} \\ &+\oint_{\Gamma_R} G_\sigma(\mathbf{y}-\mathbf{q})\nabla_{\mathbf{y}}\mathcal{G}(\mathbf{y},\mathbf{x})\cdot\mathbf{n}\,\mathrm{d}\mathbf{y} \\ &+\oint_{\Gamma_r} G_\sigma(\mathbf{y}-\mathbf{q})\nabla_{\mathbf{y}}\mathcal{G}(\mathbf{y},\mathbf{x})\cdot\mathbf{n}\,\mathrm{d}\mathbf{y} \\ =:&\,0 + I_R + I_r, \end{aligned} \tag{A21}$$

where Γ_R and Γ_r are the boundary spheres of Ω at R and r respectively.

For I_R, we have

$$\begin{aligned} |I_R| &= \left|\oint_{\Gamma_R} G_\sigma(\mathbf{y}-\mathbf{q})\frac{(\mathbf{y}-\mathbf{x})}{4\pi\|\mathbf{y}-\mathbf{x}\|^3}\cdot\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{y}-\mathbf{x}\|}\,\mathrm{d}\mathbf{y}\right| \\ &= \frac{1}{4\pi R^2}\oint_{\Gamma_R} G_\sigma(\mathbf{y}-\mathbf{q})\,\mathrm{d}\mathbf{y} \\ &= \frac{1}{4\pi R^2}\oint_{\Gamma_R} C\exp\left(-\frac{\|\mathbf{y}-\mathbf{x}+\mathbf{x}-\mathbf{q}\|^2}{2\sigma^2}\right)\mathrm{d}\mathbf{y} \\ &\le \frac{1}{4\pi R^2}\oint_{\Gamma_R} C\exp\left(-\frac{(R-\|\mathbf{x}-\mathbf{q}\|)^2}{2\sigma^2}\right)\mathrm{d}\mathbf{y} \\ &= C\exp\left(-\frac{(R-\|\mathbf{x}-\mathbf{q}\|)^2}{2\sigma^2}\right), \end{aligned} \tag{A22}$$

so $I_R \to 0$ as $R\to+\infty$.

For I_r, we have

$$\begin{aligned} I_r &= \oint_{\Gamma_r} G_\sigma(\mathbf{y}-\mathbf{q})\frac{(\mathbf{y}-\mathbf{x})}{4\pi\|\mathbf{y}-\mathbf{x}\|^3}\cdot\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{y}-\mathbf{x}\|}\,\mathrm{d}\mathbf{y} \\ &= -\frac{1}{4\pi r^2}\oint_{\Gamma_r} G_\sigma(\mathbf{y}-\mathbf{q})\,\mathrm{d}\mathbf{y}, \end{aligned} \tag{A23}$$

which leads to

$$\lim_{r\to 0} I_r = -G_\sigma(\mathbf{x}-\mathbf{q}) = -G_\sigma(\mathbf{q}-\mathbf{x}), \tag{A24}$$

thus Equation (A18) holds.

□

Proof of Theorem 1 in the main paper. Recall Equation (11) in the main paper, *i.e.*, the regularized Poisson kernel of [6],

$$\frac{\mathbf{q}-\mathbf{x}}{2\pi r^3}\big(rg_\sigma(r)-\Phi_\sigma(r)\big), \tag{A25}$$

Note that the singularity of Equation (A25) at $\mathbf{q}=\mathbf{x}$ is removable.

We first show that $\tilde{X}(\mathbf{q},\mathbf{x})$ is curl-free:

$$\begin{aligned} \nabla_{\mathbf{q}}\times\tilde{X}(\mathbf{q},\mathbf{x}) &= \int_\Omega -\nabla G_\sigma(\mathbf{y}-\mathbf{q})\times\nabla_{\mathbf{x}}\mathcal{G}(\mathbf{y},\mathbf{x})\,\mathrm{d}\mathbf{y} \\ &= \int_\Omega \frac{G_\sigma(\mathbf{y}-\mathbf{q})}{4\pi\sigma^2\|\mathbf{y}-\mathbf{x}\|^3}(\mathbf{y}-\mathbf{q})\times(\mathbf{x}-\mathbf{y})\,\mathrm{d}\mathbf{y} \\ &= (\mathbf{q}-\mathbf{x})\times\int_{\Omega'}\frac{G_\sigma(\mathbf{y}+\mathbf{x}-\mathbf{q})}{4\pi\sigma^2\|\mathbf{y}\|^3}\mathbf{y}\,\mathrm{d}\mathbf{y}, \end{aligned} \tag{A26}$$

where Ω' is a spherically symmetric region obtained by translating Ω by $-\mathbf{x}$. The vector-valued integral is parallel to $(\mathbf{q}-\mathbf{x})$ by symmetry, thus $\tilde{X}(\mathbf{q},\mathbf{x})$ and $X(\mathbf{q},\mathbf{x})$ are both curl-free, and Equation (A25) is also curl-free because it is a radial vector field.

Since $X(\mathbf{q},\mathbf{x})$ vanishes at infinity, according to the Helmholtz theorem, the curl-free solution to Equation (A18) is unique, so it is sufficient to verify that Equation (A25) satisfies Equation (A18) by direct computation. Observing the term $\frac{\mathbf{q}-\mathbf{x}}{2\pi r^3}$ is divergence-free for all $\mathbf{q}\ne\mathbf{x}$, we have

$$\begin{aligned} \nabla_{\mathbf{q}}\cdot\text{(A25)} &= \frac{\mathbf{q}-\mathbf{x}}{2\pi r^3}\cdot\frac{\mathbf{q}-\mathbf{x}}{r}rg'_\sigma(r) \\ &= \frac{1}{2\pi r}\left(-g_\sigma(r)\frac{r}{\sigma^2}\right) \\ &= -G_\sigma(\mathbf{q}-\mathbf{x}),\quad \mathbf{q}\ne\mathbf{x}, \end{aligned} \tag{A27}$$

and by continuity, the above also holds for $\mathbf{q}=\mathbf{x}$.

□

Appendix C
Additional results

In this section, we provide additional results, including comparison with PFP (Figure 1), timing (Table 1) and comparison of different smoothed winding number computation methods (Section C.2).

C.1 Extensions - Topology changes

C.2 Comparison of smoothed winding number computation

While the formulation based on the extended divergence theorem in [2] can also be applied to 3D, an efficient numerical method for the 3D case is not yet found. One reason is that their formulation prevents the easy usage of numerical quadrature methods. To show this, we use numerical quadrature to replace the original quasi-analytical one in [2] which uses Gaussian kernel integral intensively, and compare it with our method using the RWN [1] formulation. As shown in Figure 2, this method is much faster in 3D.

To make this point clearer, we also backport the RWN formulation to 2D (see Section C.2) and compare it with [2] and its quadrature version. From the results shown in Figure 2, we can see that the error of the smoothed winding number computed by our method decreases quickly with an increasing number of quadrature points, while

#	ABC Index	Direction	$\lvert V\rvert$	$\lvert F\rvert$	Number of Evaluation			Time / s			
					Value	Gradient	Factorize	Value	Gradient	Factorize	Total
A	0006	Inward	3629	7258	4 137 060	16 257 920	270	20.59	0.47	30.50	70.06
B	0093	Inward	3221	6454	2 994 656	11 720 464	217	11.31	0.34	11.86	36.40
C	0168	Inward	5021	10 046	5 183 736	20 333 104	243	32.85	0.58	36.05	92.50
D	0168	Outward	5021	10 046	4 058 584	15 832 496	187	26.23	0.44	29.76	74.10
E	0144	Inward	3660	7324	3 544 816	13 878 690	227	21.79	0.39	24.78	62.81
F	0206	Outward	9237	18 470	7 424 940	28 960 960	186	57.87	0.81	59.03	151.71
G	0093	Outward	3221	6454	2 620 324	10 223 136	188	13.28	0.30	14.69	39.46
H	2762	Outward	8746	17 500	7 735 000	30 240 000	206	68.19	0.88	70.19	176.13
I	0144	Outward	3660	7324	2 841 712	11 065 040	179	20.23	0.32	20.80	53.95
J	1525	Outward	8392	16 972	9 470 376	37 198 037	264	97.64	1.04	96.60	239.17
K	2762	Inward	8746	17 500	9 870 000	38 780 000	267	86.06	1.09	89.39	224.49
L	0137	Inward	8721	17 502	8 961 024	35 144 016	241	64.64	1.00	56.90	165.94
M	0017	Inward	3803	7350	3 601 500	14 112 000	230	13.12	0.41	24.88	56.74
N	0137	Outward	8721	17 502	8 645 988	33 883 872	232	89.88	1.00	67.09	199.24
O	5066	Outward	6931	13 862	7 291 412	28 611 168	248	47.10	0.80	47.69	127.96
P	0017	Outward	3803	7350	3 322 200	12 994 800	211	16.02	0.38	31.55	66.50
Q	6818	Outward	14 356	28 708	14 124 336	55 349 024	231	130.86	1.56	99.74	303.47
R	5066	Inward	6931	13 862	6 681 484	26 171 456	226	42.09	0.74	39.52	111.39
S	0181	Inward	27 891	55 790	27 671 840	108 455 313	233	321.80	3.08	261.82	724.55
T	0181	Outward	27 891	55 790	27 225 520	106 670 480	229	319.59	3.07	243.73	701.48

TABLE 1: The timing of the results shown in Figure 3 in the main paper. "Value": computing the value of the smoothed winding number. "Gradient": computing the gradient of the smoothed winding number *w.r.t.* a vertex. "Factorize": factorizing the sparse matrix in the Levenberg-Marquardt method.

the extended-divergence-theorem-based method [2] needs a much larger number of quadrature points to achieve the same accuracy. We also observe that the original numerical method of [2] is still the best choice for closed 2D polygons. However, the RWN formulation still has the advantage of handling open-boundary cases.

2D backport of RWN formulation

The Green's function in 2D is

$$\mathcal{G}(\mathbf{x},\mathbf{y}) = \frac{1}{2\pi}\log\|\mathbf{x}-\mathbf{y}\|, \tag{A28}$$

and the winding number of a curve Γ is defined as

$$\begin{aligned} w\left(\mathbf{q}\right) &= \int_\Gamma \nabla_\mathbf{x}\mathcal{G}(\mathbf{q},\mathbf{x})\cdot\mathbf{n}\,\mathrm{d}\mathbf{x} \\ &= \int_\Gamma \frac{\mathbf{x}-\mathbf{q}}{2\pi\|\mathbf{x}-\mathbf{q}\|^2}\cdot\mathbf{n}\,\mathrm{d}\mathbf{x} \end{aligned} \tag{A29}$$

The smoothed winding number in the 2D case yields

$$\begin{aligned} W_\sigma\left(\mathbf{q}\right) &= (w * G_\sigma)(\mathbf{q}) \\ &= \int_\Gamma \mathbf{n}\cdot X(\mathbf{q},\mathbf{x})\,\mathrm{d}\mathbf{x}, \end{aligned} \tag{A30}$$

where

$$X(\mathbf{q},\mathbf{x}) = \nabla_\mathbf{x}\mathcal{G}(\mathbf{q},\mathbf{x})\left(1-\exp\left(-\frac{\|\mathbf{x}-\mathbf{q}\|^2}{2\sigma^2}\right)\right). \tag{A31}$$

For a polygonal curve Γ, we have

$$\begin{aligned} W_\sigma\left(\mathbf{q}\right) &= \sum_{\gamma\in\Gamma}\int_\gamma \mathbf{n}\cdot X(\mathbf{q},\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &\approx \sum_{\gamma\in\Gamma}\sum_k w_k^\gamma\mathbf{n}\cdot X(\mathbf{q},\mathbf{x}_k^\gamma), \end{aligned} \tag{A32}$$

where $(\mathbf{x}_k^\gamma, w_k^\gamma)$ denote the Gauss–Legendre quadrature points and weights on the line segment γ.

REFERENCES

[1] H. Chen, B. Miller, and I. Gkioulekas, "3d reconstruction with fast dipole sums," *ACM Trans. Graph.*, vol. 43, no. 6, Nov. 2024. [Online]. Available: https://doi.org/10.1145/3687914

[2] H. Sun, J. Wang, H. Bao, and J. Huang, "Gauwn: Gaussian-smoothed winding number and its derivatives," in *SIGGRAPH Asia 2024 Conference Papers*, ser. SA '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3680528.3687569

[3] D. A. Dunavant, "High degree efficient symmetrical gaussian quadrature rules for the triangle," *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 1129–1148, 1985. [Online]. Available: https://api.semanticscholar.org/CorpusID:120117894

[4] G. Barill, N. G. Dickson, R. Schmidt, D. I. W. Levin, and A. Jacobson, "Fast winding numbers for soups and clouds," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201337

[5] D. Sieger and M. Botsch, "The Polygon Mesh Processing Library," Aug. 2023. [Online]. Available: https://github.com/pmp-library/pmp-library

[6] J. T. Beale, W. Ying, and J. R. Wilson, "A simple method for computing singular or nearly singular integrals on closed surfaces," *Communications in Computational Physics*, vol. 20, no. 3, p. 733–753, 2016.

[7] H. Cao, G. Xu, R. Gu, J. Xu, X. Zhang, T. Rabczuk, Y. Luo, and X. Gao, "Robust and feature-preserving offset meshing," 2024. [Online]. Available: https://arxiv.org/abs/2412.15564

[8] H. Cao, G. Xu, R. Gu, J. Xu, X. Zhang, and T. Rabczuk, "A parallel feature-preserving mesh variable offsetting method with dynamic programming," 2023. [Online]. Available: https://arxiv.org/abs/2310.08997

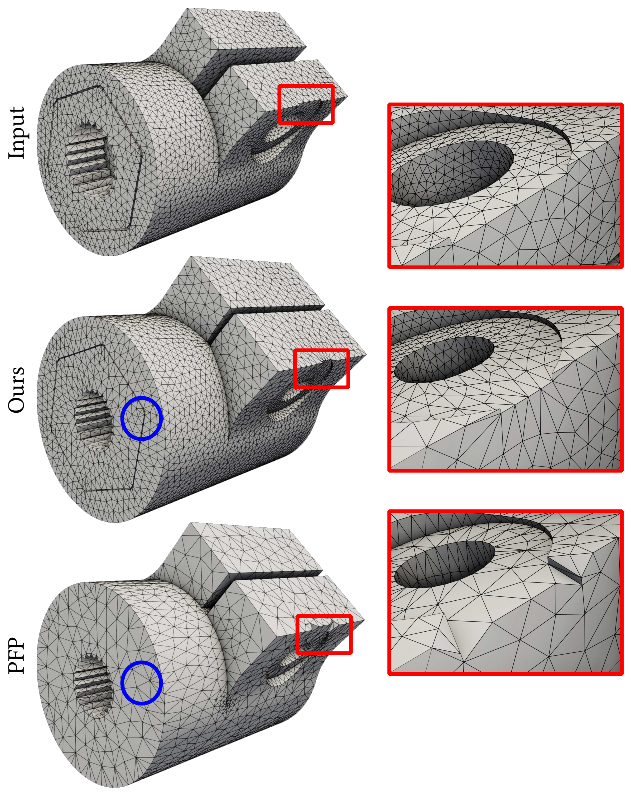


Fig. 1: **Comparison with PFP.** We are aware of the preprint [7], which can be seen as a direct derivation of the previous preprint PFP [8], with the similar local quadratic optimization and dynamic programming for vertex offsetting positions but an improved mesh extraction after that in the pipelines. However, as there is only an official implementation of PFP, we compare our method with it. PFP can generate similar results to ours with the initial features preserved. Unlike Blender in Figure 11 in the main paper, PFP can handle the closely placed surface parts (blue) by a different strategy to ours that adapts the topology. It can also be observed that their local optimization may generate incorrect offsetting positions in some cases (red).

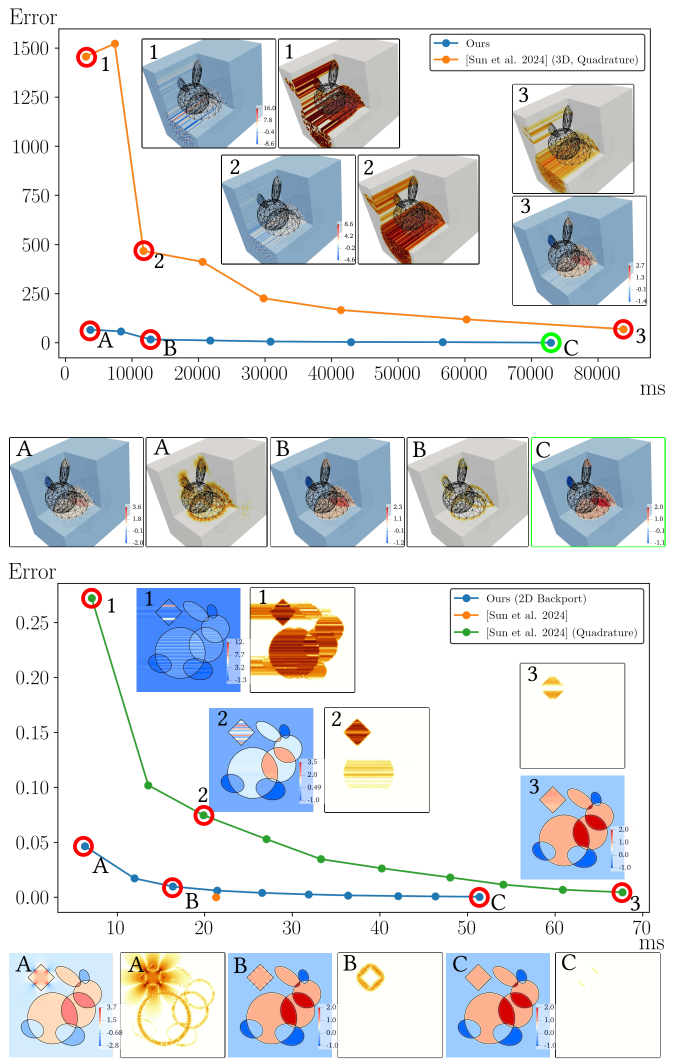


Fig. 2: **Comparison of the smoothed winding number computation with [2].** We compute the smoothed winding number at the vertices of $100 \times 100 \times 100$ (3D) and 100×100 (2D) grids. Involved methods are: ours (computation using the regularized winding number), [2] (computation using the extended divergence theorem, quasi-analytical) and [2] (computation using the extended divergence theorem, quadrature). For quadrature, there are $1, 4, 7, 13, 19, 27, 37, 48$ quadrature points per triangle and $1, 3, 5, 7, 9, 11, 13, 15, 17, 19$ quadrature points per segment. Field and error distribution are shown. Error is the L2 norm against green framed as the ground-truth in 3D, and the orange point in 2D. BVH is not used for all the methods.