Instant Self-Intersection Repair for 3D Meshes

WONJONG JANG, POSTECH, South Korea
YUCHEOL JUNG, POSTECH, South Korea
GYEONGMIN LEE, POSTECH, South Korea
SEUNGYONG LEE, POSTECH, South Korea

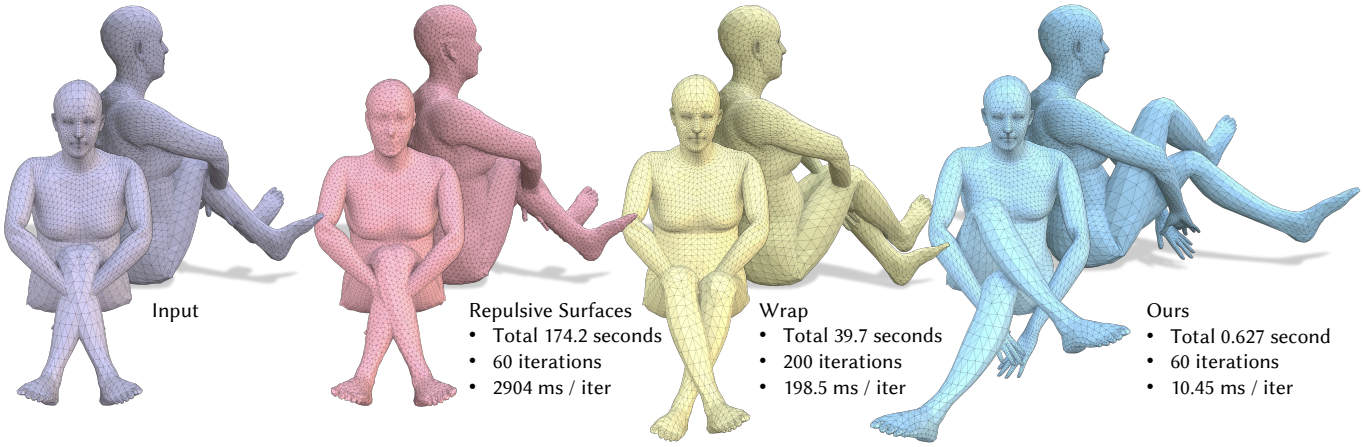


Fig. 1. Our instant self-intersection repair method resolves complex self-intersections in static 3D surface meshes that existing approaches [Wrap 2024; Yu et al. 2021a] fail to handle. Our novel algorithm achieves both robust intersection repair and remarkable computational efficiency, enabling practical applications in various 3D modeling scenarios.

Self-intersection repair in static 3D surface meshes presents unique challenges due to the absence of temporal motion and penetration depth information—two critical elements typically leveraged in physics-based approaches. We introduce a novel framework that transforms local contact handling into a global repair strategy through a combination of local signed tangent-point energies and their gradient diffusion. At the heart of our method is a key insight: rather than computing expensive global repulsive potentials, we can effectively approximate long-range interactions by diffusing energy gradients from local contacts throughout the mesh surface. In turn, resolving complex self-intersections reduces to simply propagating local repulsive energies through standard diffusion mechanics and iteratively solving tractable local optimizations. We further accelerate convergence through our momentum-based optimizer, which adaptively regulates momentum based on gradient statistics to prevent overshooting while maintaining rapid intersection repair. The resulting algorithm handles a variety of challenging scenarios, from shallow contacts to deep penetrations, while providing computational efficiency suitable for interactive applications.

CCS Concepts: • **Computing methodologies** → **Shape modeling**; • **Mathematics of computing** → *Continuous optimization.*

Authors' Contact Information: Wonjong Jang, POSTECH, South Korea, wonjong@postech.ac.kr; Yucheol Jung, POSTECH, South Korea, ycjung@postech.ac.kr; Gyeongmin Lee, POSTECH, South Korea, kyung45250@postech.ac.kr; Seungyong Lee, POSTECH, South Korea, leesy@postech.ac.kr.

ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3731427

Additional Key Words and Phrases: Self-intersection, shape optimization, static 3D surface meshes

ACM Reference Format:
Wonjong Jang, Yucheol Jung, Gyeongmin Lee, and Seungyong Lee. 2025. Instant Self-Intersection Repair for 3D Meshes. *ACM Trans. Graph.* 44, 4 (August 2025), 14 pages. https://doi.org/10.1145/3731427

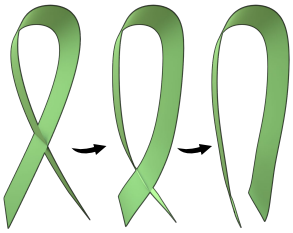
1 Introduction

Self-intersections in geometric modeling arise when different parts of a surface pass through each other, creating physically invalid configurations that need to be repaired. These geometric anomalies occur in various scenarios—from dynamic cases where surfaces intersect during motion, to static cases where intersections exist in a single mesh configuration. Especially, static cases commonly arise in applications like deformation or rigging transfer, where source mesh poses are mapped to differently shaped target characters. Our goal is to repair self-intersections of static surface meshes by computing minimal geometric deformation that eliminates invalid surface configurations while faithfully preserving the original shape characteristics.

While self-intersection detection and resolution have been well studied in physics simulations, these solutions take full advantage of temporal information and penetration depth estimation, both of which are unavailable in static surface mesh scenarios. Continuous collision detection approaches [Ferguson et al. 2021; Lan et al. 2022a,b; Li et al. 2020, 2022] assume continuous object motion between previous collision-free frame and current frame, allowing them to detect and resolve collisions by comparing previous and

current states. Discrete collision detection approaches [Ding and Schroeder 2019; Macklin et al. 2016; Müller et al. 2007] assume volumetric or tetrahedral meshes where penetration depths can be effectively estimated [Chen et al. 2023; Je et al. 2012].

However, static 3D surface meshes present several fundamental challenges: there is no temporal information about how and when intersections occurred; the absence of a clear interior volume makes it ambiguous to determine inside/outside regions; and in complex penetrations, finding closest surface points and deciding resolution directions becomes highly challenging. The last issue is particularly problematic since resolving local triangle collisions does not necessarily lead to a complete global self-intersection resolution—the algorithm must infer an appropriate global deformation direction with only local geometric information available.



In this paper, we present a novel method for instant self-intersection repair in static 3D surface meshes that combines efficient local energy computation with gradient diffusion. Our approach is built on the proven effectiveness of tangent-point energy [Sassen et al. 2024; Yu et al. 2021a,b] for self-intersection prevention, but introduces a critical improvement: rather than computing computationally expensive global potentials, we focus computational effort on local potentials at intersection regions and approximate global potentials through smooth propagation of local potentials over the surface domain. As a part of this strategy, we revise the original unsigned tangent-point energy formulation into a signed version to better direct the repulsive forces. This local-to-global strategy significantly reduces computational overhead and enables self-intersection repair for static 3D surfaces without well-defined interior volume, as demonstrated in the inset figure.

To accelerate convergence, we exploit the observation that vertices tend to move coherently during self-intersection repair. Based on this insight, we develop an adaptive momentum-based optimization scheme that employs repulsive coherence while preventing overshooting artifacts. Our novel optimizer incorporates momentum to achieve rapid convergence, but automatically moderates its effect when self-intersections have been repaired, ensuring rapid and stable repair even in cases of deep penetrations.

The integration of these components—*local signed tangent-point energy*, *gradient diffusion*, and *adaptive momentum optimization*—enables our method to handle a wide range of self-intersecting surface meshes from shallow surface contacts to complex deep penetrations. The robustness and speed of our method make it particularly valuable for interactive applications like deformation transfer, character animation, and real-time mesh editing, where self-intersections can frequently occur.

The main contributions of our work include:

- A novel local signed tangent-point energy formulation, combined with a gradient diffusion scheme for efficient intersection repair.
- An adaptive momentum-based optimization method that accelerates convergence while preventing overshooting artifacts.
- An extensive benchmark dataset of surface meshes with challenging self-intersections, allowing rigorous evaluation of self-intersection repair methods.
- A comprehensive validation across various scenarios, including user-controlled repair and applications in deformation transfer, inter-object intersection repair, and key pose correction.

2 Related Work

2.1 Self-Intersection Handling

Physics Simulation. Self-intersection resolution in physics simulation employs either Continuous Collision Detection (CCD) or Discrete Collision Detection (DCD). CCD methods, such as the incremental potential contact approach [Li et al. 2020] and its extensions [Fang et al. 2021; Ferguson et al. 2021; Lan et al. 2022a,b; Li et al. 2022], detect the first moment of contact between elements starting from a known intersection-free state. More relevant to our work are DCD methods, which detect and resolve collisions after they occur. DCD methods typically resolve intersections through various computational strategies, including minimizing penetration volume [Allard et al. 2010; Wang et al. 2012], applying constraints [Bouaziz et al. 2023; Macklin et al. 2016; Müller et al. 2007; Verschoor and Jalba 2019], and using penalty forces [Belytschko and Neal 1991; Ding and Schroeder 2019; Drumwright 2007; Huněk 1993]. However, these methods are primarily designed for volumetric or tetrahedral meshes and depend on robust approximations of penetration depths.

For self-intersection repair on static surface meshes, an intuitive approach would be to first perform volumetric discretization and then apply a DCD approach to the resulting representation. However, conventional tetrahedralization methods such as TetGen [Hang 2015] assume self-intersection-free input surface meshes. More recent approaches like TetWild [Hu et al. 2018] and fTetWild [Hu et al. 2020] allow inputs with self-intersections, but glue disparate self-intersecting parts through their ϵ-envelopes. Although Sacht et al. [2013] and Overby et al. [2021] utilize conformal mean curvature flow (cMCF) to eliminate self-intersections before tetrahedralization, cMCF generally does not guarantee convergence to self-intersection-free surfaces. Li and Barbič [2018] proposed an alternative method using cell complexes and immersion graphs to decompose meshes into self-intersection-free components before tetrahedralization. These approaches incur not only considerable computational overhead due to slow tetrahedralization processes but also the additional burden of physics simulations to resolve self-intersections.

Tangent-Point Energy. Repulsive Curves [Yu et al. 2021b] leverage *tangent-point energies* [Buck and Orloff 1995; Strzelecki and von der Mosel 2013, 2018] to handle space curves by providing infinite barriers to self-intersection through global pairwise point interactions. Repulsive Surfaces [Yu et al. 2021a] adapt these potentials to surface geometries through discretization schemes and hierarchical solvers, enabling robust self-intersection avoidance in surface meshes. Repulsive Shells [Sassen et al. 2024] introduce shape spaces with collision-aware metrics based on tangent-point energy, ensuring that continuous shape interpolation remains intersection-free by default. In contrast to these approaches that use tangent-point energies as global repulsive potentials, our method computes

local tangent-point energies specifically on intersecting triangles and approximates global energies through surface-based gradient diffusion for robust and efficient self-intersection repair.

2.2 Diffusion over 3D Meshes

The mesh Laplacian operator [Dziuk 1988; Pinkall and Polthier 1993] provides a mathematical foundation for modeling diffusion processes across discrete surfaces [Meyer et al. 2003]. This operator facilitates various geometry processing tasks through energy diffusion, with the heat method [Crane et al. 2017] being a notable example that computes geodesic distances by simulating heat flow. Extending the diffusion concept, recent works leverage gradient diffusion using the mesh Laplacian matrix as a low-pass filter on gradients for enforcing smoothness constraints on diverse geometric tasks: Nicolet et al. [2021] achieve smooth inverse rendering solutions with fewer gradient descent steps, Jung et al. [2023] regularize non-rigid mesh registration, and Wang et al. [2024] constrain rigidness in multi-view facial reconstruction. Similarly, our method utilizes gradient diffusion to smoothly propagate sparse gradients across the mesh surface.

2.3 Optimization with Momentum

Momentum-based optimizations [Nesterov 1983; Polyak 1964] accelerate convergence by accumulating past gradients, and modern optimizers like Adam [Kingma and Ba 2015] further extend this approach by incorporating adaptive per-parameter learning rates. However, while these methods improve convergence rates, they can produce overshooting artifacts [Ogata 1995] that yield undesirable shapes in geometric optimization. To address this limitation, we propose an adaptive optimization method that accelerates convergence via aggressive updates during self-intersection repair and prevents overshooting by eliminating momentum after resolving self-intersections.

3 Overview

Our framework repairs self-intersections through iterations of four main steps:

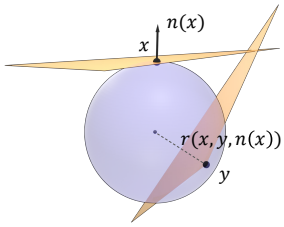
i. Identify intersecting triangle pairs with CUDA-optimized self-intersection detection algorithm [Choutas 2019; Karras 2012].
ii. Compute local signed tangent-point energies for each detected intersection pair (Section 4).
iii. Propagate the derivatives of local repulsive energies across the mesh surface (Section 5.1).
iv. Accelerate optimization through momentum while preventing overshooting artifacts (Section 6.2).

The key innovation of our approach is an efficient local-to-global strategy: rather than computing expensive global repulsive energies, we concentrate computation on actual intersection regions and leverage diffusion to smoothly propagate repulsive forces. Combined with momentum-based optimization, this approach enables both efficient and robust handling of complex self-intersections.

4 Local Signed Tangent-Point Energy

We introduce an efficient energy formulation for repairing self-intersections including complex and deep penetrations on mesh surfaces. Building on the effective discrete tangent-point energy (TPE) for surfaces [Yu et al. 2021a], we introduce two key improvements: a localized formulation that operates directly on a self-intersecting triangle pair (Section 4.1) and a signed formulation (Section 4.2) that enhances the original unsigned approach for consistent repulsion forces.

4.1 Local Discrete Tangent-Point Energy



Given two points x, y and a normal vector $n(x)$ at x, the tangent-point energy is defined as the reciprocal of the radius $r(x, y, n(x))$ of the minimal sphere that is tangent to x and intersects y. For a discrete triangle mesh, this energy can be efficiently computed using midpoint quadrature over triangle pairs [Yu et al. 2021a]. Given a mesh with triangle set $\mathcal{T}$, the discrete tangent-point energy $\hat{\Phi}$ considers all triangle pairs:

$$\hat{\Phi}(x) := \sum_{t_1 \in \mathcal{T}} \sum_{\substack{t_2 \in \mathcal{T} \\ t_1 \neq t_2}} a_{t_1} a_{t_2} K(c_{t_1}, c_{t_2}, n_{t_1}), \tag{1}$$

where a_{t_1}, a_{t_2} are triangle areas, c_{t_1}, c_{t_2} are centroids, and n_{t_1} denotes the unit normal of triangle t_1. The discrete kernel K is defined as:

$$K(x, y, n) := \frac{|\langle n, x - y \rangle|^\alpha}{|x - y|^{2\alpha}}. \tag{2}$$

The exponent α determines the repulsion strength.

This global formulation, while adequate for minor intersections [Yu et al. 2021a, Fig. 26], struggles with deep penetrations due to deceptive high-energy pairs. These pairs typically occur in high-curvature regions where locally adjacent vertices generate significant energy values despite not being actual intersections. While these pairs contribute high energy to Eq. 1, their gradients often counteract the resolution of true self-intersections, making the optimization process inefficient.

We address this problem by introducing a local formulation that concentrates only on intersecting triangle pairs $(t_1, t_2) \in C$:

$$\hat{\Phi}(x) = \sum_{(t_1, t_2) \in C} a_{t_1} a_{t_2} (K(c_{t_1}, c_{t_2}, n_{t_1}) + K(c_{t_2}, c_{t_1}, n_{t_2})).$$

This localized formulation reduces the computational complexity from $O(n^2)$ pairs, or $O(n \log n)$ pairs when using a bounding volume hierarchy, to $O(k)$ where $k \ll n$, resulting in substantial computational gains.

Although this triangle-to-triangle approach prevents centroid intersections, it cannot guarantee that vertices do not penetrate triangles. To address this, we propose a vertex-to-triangle formulation:

$$\hat{\Phi}(x) = \sum_{(t_1, t_2) \in C} \Big(\sum_{v_1 \in t_1} a_{t_2} K(c_{t_2}, v_1, n_{t_2}) + \sum_{v_2 \in t_2} a_{t_1} K(c_{t_1}, v_2, n_{t_1}) \Big) \tag{3}$$

This formulation enables vertices to receive direct repulsive forces, rather than averaged forces computed using triangle centroids.

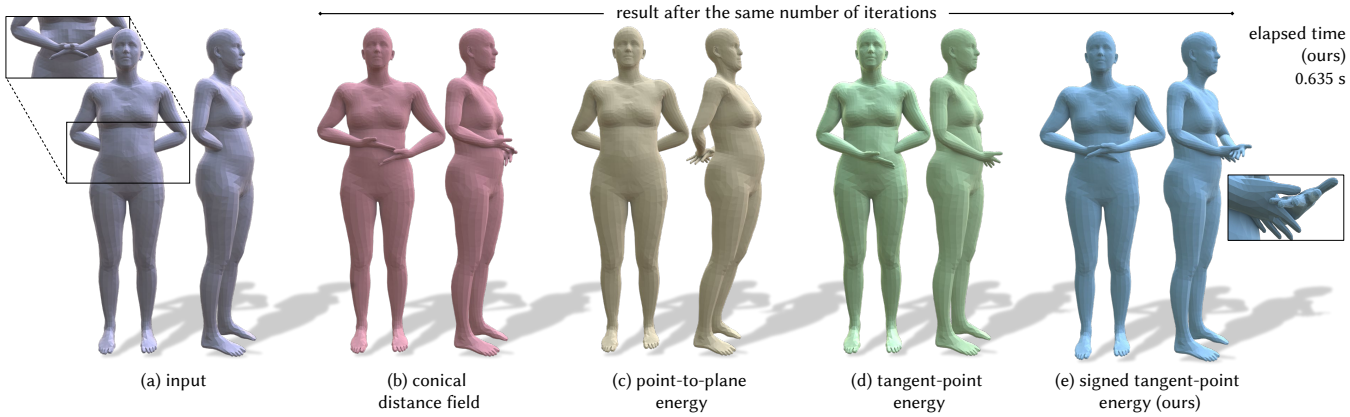


Fig. 2. Effects of repulsive energies. (a) Input mesh with deep self-penetrations, (b-d) alternative energy formulations exhibit slow convergences, with (c) simply moving along surface normals producing unnatural deformations, (e) our signed tangent-point energy achieves the fastest convergence.

4.2 Signed Formulation

The tangent-point kernel (Eq. 2) in its original form is an *unsigned* energy function that achieves its minimum value of zero when the vector $x - y$ is tangential to the surface. When applied globally to all triangle pairs, this energy cannot reach its minimum except in the case of a planar mesh; otherwise, it distributes surface curvatures uniformly. In our local formulation where only intersecting triangle pairs are considered, this unsigned energy can simply reach its minimum by settling vertices on the tangent plane, rather than properly resolving self-intersections through repulsion (Fig. 3).

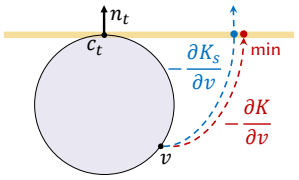


Fig. 3. When a penetrating vertex v receives repulsive energy from a triangle centroid c_t with normal vector n_t, the derivative of the unsigned kernel K leads vertex v to settle on c_t's tangent plane, while the derivative of the signed kernel K_s guides v to escape outward from the surface.

To prevent vertices from settling on the tangent plane and maintain repulsive forces throughout the intersection repair process, we introduce a *signed* reformulation of the original energy:

$$K_s(c_t, v, n_t) = \frac{\langle n_t, c_t - v\rangle^\alpha}{|c_t - v|^{2\alpha}}, \tag{4}$$

where α is chosen to be a positive odd number to preserve the sign of the numerator.

This signed formulation ensures consistent gradient directions regardless of whether vertices lie inside or outside the surface, with the energy decreasing monotonically as vertices move away from the paired triangle centroid along the surface normal direction. As illustrated in Fig. 2, this modification produces significantly faster convergence compared to both the unsigned formulation and conventional energies, such as conical distance field [Tzionas et al. 2016] and point-to-plane energy [Rusinkiewicz and Levoy 2001].

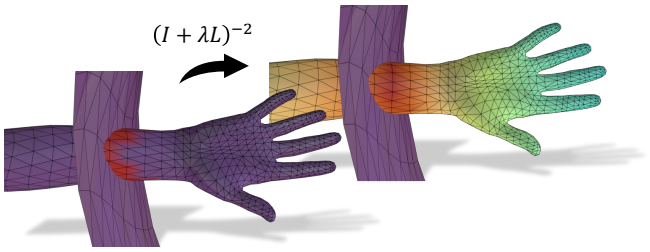


Fig. 4. Visualization of gradient diffusion. We apply two iterations of diffusion with a large time step ($\lambda \geq 99$) to propagate the repulsive energy gradients from local contact triangles to deeply penetrating vertices.

4.3 TPE for Avoidance vs. Repair of Self-Intersections

The tangent-point energy functional acts as a key component for self-intersection avoidance in previous works [Sassen et al. 2024; Yu et al. 2021a,b]. For avoiding self-intersections, it is important to create a barrier with infinite energy by setting the exponent $\alpha > 4$. The barrier enables maintaining intersection-free states during mesh optimization for an initial configuration with no self-intersections.

However, for our self-intersection repair task where the input mesh already contains intersections, such an infinite energy barrier can be problematic as it may trap vertices in self-intersecting regions. Thus, for self-intersection repair, we need to set the exponent $\alpha < 4$ to create a finite energy barrier that discourages self-intersections and simultaneously allows vertices to escape from intersecting regions. Based on this observation, our experiments employ $\alpha = 3$, which enforces strong repulsion while preserving the sign of Eq. 4.

5 Gradient Diffusion and Regularization

We introduce gradient diffusion and shape-preserving regularization to resolve self-intersections while maintaining geometric features of input meshes. The diffusion mechanism propagates local repulsive forces to broader mesh regions (Section 5.1), and differential coordinate-based regularization preserves geometric features (Section 5.2).

5.1 Repulsive Gradient Diffusion

Unlike global repulsive potentials (Eq. 1) that generate dense gradient fields on the entire mesh, our local repulsive energy produces sparse gradients localized to self-intersecting triangles. To propagate these repulsive forces to vertices of non-intersecting triangles, we perform gradient diffusion through a diffusion matrix $(\mathbf{I}+\lambda\mathbf{L})^{-1}$, combining the identity matrix $\mathbf{I}$ and the discrete Laplacian matrix $\mathbf{L}$. We apply this diffusion matrix twice to achieve a sufficient global distribution of local gradients, and the final diffused gradients are obtained by:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta(\mathbf{I}+\lambda\mathbf{L})^{-2}\frac{\partial\hat{\Phi}}{\partial\mathbf{x}},$$

where η is the step size for gradient descent and λ controls the diffusion radius, with a larger value resulting in broader gradient propagation. In all our experiments, we set $\lambda = 99$, except for Fig. 13 where $\lambda = 999$. Fig. 4 visualizes the effect of gradient propagation.

Although this approach shares similarities with *Sobolev preconditioning* [Karatson and Loczi 2005; Neuberger 1985; Osher et al. 2022; Park et al. 2021], its fundamental purpose differs. Sobolev preconditioning typically addresses an optimization problem with global minimum by improving the condition number of the system matrix to achieve a better convergence rate. In our case, we utilize the diffusion matrix to propagate sparse gradients smoothly over the mesh surface, obtaining a dense gradient approximation that guides our dynamic optimization process as the objective function evolves with changing self-intersections.

In terms of computational efficiency, the advantage of sparse linear system $(\mathbf{I}+\lambda\mathbf{L})$ used for gradient diffusion is that it remains constant throughout the optimization process. This enables us to pre-compute a Cholesky decomposition [Naumov 2011] on the system matrix, allowing for near-linear time solution of the sparse system at each iteration. As demonstrated in Sections 7.2 and 7.3, this approach to resolving self-intersections via iterative local optimization significantly reduces computational cost and produces plausible shapes suitable for practical applications.

5.2 Shape Preserving Regularization

Another key objective in our self-intersection repair is to preserve the original geometric features of the input mesh. To achieve the objective, we employ differential coordinates [Lipman et al. 2004; Sorkine et al. 2004], computed using the Laplace-Beltrami operator $\mathbf{L}$ [Dziuk 1988; Pinkall and Polthier 1993]. These coordinates approximate the mean curvature at each vertex, characterizing local geometric variations, and preserving them leads to maintaining the relative geometric relationships between vertices. Our regularization term preserving the initial differential coordinates $\boldsymbol{\delta}$ is defined as:

$$E(\mathbf{x}) = \|\mathbf{L}\mathbf{x} - \boldsymbol{\delta}\|^2. \tag{5}$$

Minimizing this energy term ensures that vertices are repositioned with minimal distortions from their original differential relationships during intersection repair.

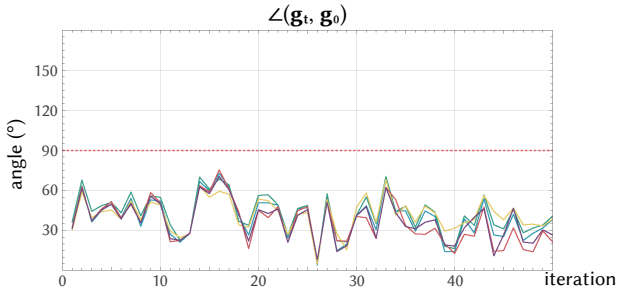


Fig. 5. Gradient consistency analysis. We measure angles between the current and initial gradients, $\mathbf{g}_t$ and $\mathbf{g}_0$, for five vertices of the same repaired body part during gradient descent. The gradients maintain consistent repulsion directions throughout the repair process.

5.3 Final Self-Intersection Repair Energy

By combining Eqs. 3 and 5, our final objective function for self-intersection repair is formulated as:

$$\Phi_{\text{total}}(\mathbf{x}) = \hat{\Phi}(\mathbf{x}) + \beta E(\mathbf{x}), \tag{6}$$

where β controls the influence of regularization and is set to $\beta = 10^6$ in our experiments. Finally, we diffuse the derivatives of this objective function and update vertex positions using:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta(\mathbf{I}+\lambda\mathbf{L})^{-2}\frac{\partial\Phi_{total}}{\partial\mathbf{x}}.$$

In the optimization process, our repulsive energy $\hat{\Phi}(\mathbf{x})$ varies per iteration as it depends on the intersecting triangle pairs detected at each step. During optimization, the role of $\hat{\Phi}(\mathbf{x})$ is not to reach a minimum but to provide consistent repulsion directions toward a self-intersection-free configuration. When self-intersections are repaired, the repulsive energy $\hat{\Phi}(\mathbf{x})$ is substantially attenuated as intersecting triangle pairs disappear, allowing the shape-preserving regularization to become dominant. In subsequent optimization iterations, this regularization attempts to restore the original shape and may reintroduce self-intersections, resulting in oscillatory behavior. Since the final state at termination may not represent the optimal solution due to these oscillations, we select the configuration with the minimum number of self-intersecting triangles observed during the optimization process as our final solution.

6 Momentum-Based Acceleration

Although gradient descent can effectively minimize self-intersection repair energy (Eq. 6), the optimization process often requires numerous iterations, particularly for self-intersecting meshes with deep penetrations. In this section, we present an acceleration method based on momentum control for our optimization. Our method builds on the popular Adam optimizer [Kingma and Ba 2015] but introduces crucial modifications to address overshooting artifacts of momentum-based acceleration.

6.1 Gradient Consistency

The effectiveness of momentum-based acceleration in self-intersection repair stems from the consistent gradient directions during

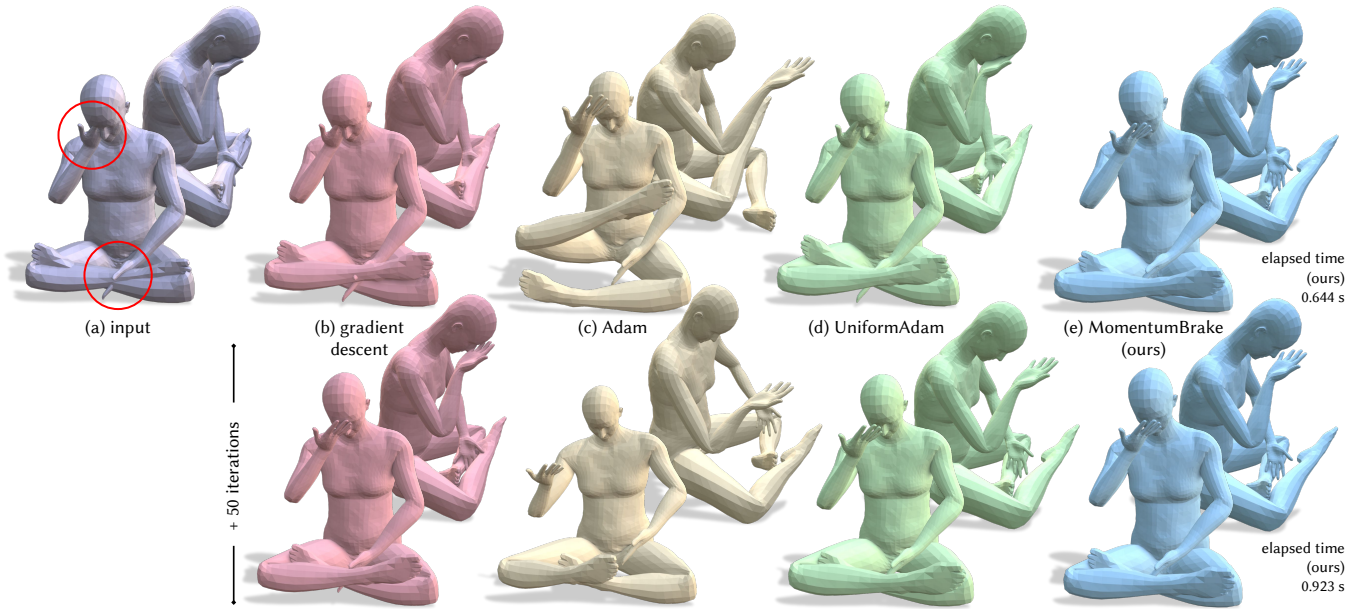


Fig. 6. Comparison of optimization methods. (a) Original mesh with self-intersections. (b) Standard gradient descent shows slow convergence compared to momentum-based optimizers. (c) Adaptive learning rate mechanism in Adam results in inconsistent and sensitive geometry updates. (d) UniformAdam exhibits two key issues: reduced convergence speed due to momentum normalization (top) and overshooting artifacts caused by residual momentum after intersection repair (bottom). (e) Our approach achieves rapid and stable convergence by adaptively controlling momentum during intersection repair with automatic momentum reinitialization.

optimization, illustrated in Fig. 5. This directional consistency indicates that momentum-based optimization can substantially accelerate convergence while preserving solution quality.

For momentum-based optimization, we employ the Adam optimizer, which utilizes both the first and second moments—exponential moving averages of gradients and their squares, respectively. The optimizer applies an adaptive step size for each variable based on the second moment. However, as shown in Fig. 6 (c), this adaptive normalization can disturb smooth geometry updates, as vertices with low second moments become disproportionately sensitive to geometry updates. Nicolet et al. [2021] address this issue with UniformAdam, which uniformly normalizes the first moments using the infinity norm of the second moments. While this uniform normalization resolves sensitive and inconsistent geometry updates, it unnecessarily reduces gradient magnitudes and impedes optimization progress (Fig. 6 (d)). We found that accelerated convergence can be achieved by simply removing this redundant normalization, as demonstrated in Fig. 6 (e).

6.2 MomentumBrake

While momentum-based optimization accelerates convergence, it can introduce *overshooting* artifacts [Ogata 1995]. Unlike in deep neural network training where overshooting can help escape local minima [Sutskever et al. 2013], such behavior in mesh optimization can lead to unnatural shape deformation and hinder convergence to desirable solution. We address this limitation by introducing MomentumBrake, an optimizer that adaptively regulates momentum based on gradient statistics.

MomentumBrake computes an approximate standard deviation $\boldsymbol{\sigma}$ using the optimizer's first moments $\mathbf{m}_{t-1}$ and second moments $\mathbf{v}_{t-1}$ at time $t-1$:

$$\boldsymbol{\sigma}_t = \sqrt{\mathbf{v}_{t-1} - \mathbf{m}_{t-1}^2}.$$

Using this standard deviation, MomentumBrake establishes bounds of $\pm\alpha\boldsymbol{\sigma}$ to determine when momentum should be applied. The momentum term is preserved when the current gradient falls within these bounds but is reset when gradients exceed them. Formally, we update the momentum term $\mathbf{m}_t$ at time t using:

$$\mathbf{m}_t = \begin{cases} \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t & \text{if } \left|\mathbf{g}_t - \mathbf{m}_{t-1}\right| \le \alpha \left|\boldsymbol{\sigma}_t\right| \\ \mathbf{g}_t & \text{otherwise} \end{cases} \tag{7}$$

where $\mathbf{g}_t$ is the current gradient, β_1 is the momentum coefficient, and α is the magnitude of bounds. This adaptive mechanism prevents residual momentum from causing overshooting while retaining the benefits of momentum-based acceleration. The complete procedure is detailed in Algorithm 1.

To ensure stability in early optimization, we apply bias correction to the estimates of mean $\boldsymbol{\mu}_t$ and standard deviation $\boldsymbol{\sigma}_t$ using moving average coefficients β_1 and β_2. This bias correction is crucial in early iterations to compensate for the initialization of the first and second moments as zero. Furthermore, for the first 10 iterations, we deliberately employ standard momentum updates, as early estimates of mean $\boldsymbol{\mu}_t$ and standard deviation $\boldsymbol{\sigma}_t$ could be unreliable due to an insufficiently developed gradient trajectory. Applying gradient bounds during these initial stages would result in repeated momentum resets, undermining the benefits of momentum-based

Algorithm 1 MomentumBrake

Require: initial vertices $\mathbf{x}_0$, a step size $\eta > 0$, coefficients used for moving averages $\beta_1, \beta_2 \in [0, 1)$, standard deviation multiplier $\alpha > 0$

initialize $\mathbf{m}_0 \leftarrow \mathbf{0}$, $\mathbf{v}_0 \leftarrow \mathbf{0}$
for $t = 1$ **to** T **do**
  $\mathbf{g}_t \leftarrow -(\mathbf{I} + \lambda \mathbf{L})^{-2} \frac{\partial \Phi_{total}}{\partial \mathbf{x}}$
  $\boldsymbol{\mu}_t \leftarrow \mathbf{m}_{t-1} / (1 - \beta_1^t)$ ▷ Bias corrected mean
  $\boldsymbol{\sigma}_t \leftarrow \sqrt{\mathbf{v}_{t-1} / (1 - \beta_2^t) - \boldsymbol{\mu}_t^2}$ ▷ Bias corrected std.
  $\mathbf{mask}_t \leftarrow \mathbb{1}[\boldsymbol{\mu}_t - \alpha \boldsymbol{\sigma}_t \leq \mathbf{g}_t \leq \boldsymbol{\mu}_t + \alpha \boldsymbol{\sigma}_t]$
  if $t \leq 10$ **then**
    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t$
  else
    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{mask}_t \odot \mathbf{m}_{t-1} + (1 - \beta_1 \mathbf{mask}_t) \odot \mathbf{g}_t$ ▷ Eq. 7
  end if
  $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2$
  $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t / (1 - \beta_1^t)$ ▷ Bias correction
  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \eta \hat{\mathbf{m}}_t$
end for
return $\mathbf{x}_t$

acceleration and causing the optimizer to behave similarly to standard gradient descent.

This adaptive mechanism on momentum update establishes a bound on momentum update magnitude, inhibiting sudden momentum fluctuations. Our optimizer enforces that the momentum change is bounded in both momentum preserving and reset cases:

$$|\mathbf{m}_t - \mathbf{m}_{t-1}| = \begin{cases} (1 - \beta_1) \left| \mathbf{g}_t - \mathbf{m}_{t-1} \right| \leq (1 - \beta_1) \alpha \left| \boldsymbol{\sigma}_t \right| & \triangleright \text{ preserve} \\ |\mathbf{m}_t - \mathbf{m}_{t-1}| > \alpha \left| \boldsymbol{\sigma}_t \right| & \triangleright \text{ reset} \end{cases}$$

In the momentum-preserving case, this derivation demonstrates an *implicit* Lipschitz-type bound, where momentum updates are proportionally constrained relative to gradient statistics $\boldsymbol{\sigma}_t$ rather than explicit input difference $\left| \mathbf{g}_t - \mathbf{g}_{t-1} \right|$. Such a bound ensures stable momentum updates during intersection repair as long as gradient consistency is maintained. Conversely, when gradients deviate significantly from recent patterns, momentum is reset, enabling the system to quickly suppress overshooting artifacts.

7 Experiments

We conducted comprehensive experiments to evaluate our self-intersection repair framework. First, we compare our method against recent methods with functionality to resolve self-intersections in static 3D surface meshes. We then analyze computational efficiency by examining the detailed performance of our framework's components. We also show the flexibility of our approach through user-controlled repair and demonstrate its effectiveness for various practical applications, including deformation transfer, inter-object intersection repair, and key pose correction. Finally, we showcase the results of our method on a variety of examples (Fig. 17).

7.1 Implementation Details

We perform a pre-processing step that normalizes the input mesh to unit scale by dividing all vertex coordinates by their spatial range, as the tangent-point energy is not scale-invariant. After repairing self-intersections using our method, we restore the mesh to its original scale in a post-processing step.

We implemented our method in PyTorch with CUDA acceleration. Our implementation utilizes a CUDA-optimized self-intersection detection algorithm [Choutas 2019; Karras 2012] and sparse matrix solver [Naumov 2011; Nicolet et al. 2021] for computational efficiency. All experiments and performance evaluations were conducted on a desktop workstation equipped with an AMD Ryzen 7950X3D processor, an NVIDIA GeForce RTX 4090 GPU, and 64GB RAM.

7.2 Comparisons

7.2.1 Benchmark. To extensively evaluate our method, we propose a benchmark dataset for the self-intersection repair task. Our benchmark consists of 60 diverse surface meshes containing self-intersections, categorized into three groups: full-body human meshes, animal meshes, and miscellaneous meshes. For the human models, we generated 30 challenging self-intersection scenarios by manually rigging an SMPL model [Loper et al. 2015] using yoga poses as references. The animal meshes comprise 20 models derived from four artist-created animals [Radik Bilalov 2025]—deer, bear, wolf, and hare—each with five distinct poses manually modified to induce self-intersections. Finally, we created 10 miscellaneous objects from scratch, including six molecular knot shapes [Schaufelberger 2020, Fig. 1] along with pretzel, Möbius strip, Celtic knot, and disconnected Klein bottle shapes.

7.2.2 Baselines. We evaluate our method against two established approaches for self-intersection repair on static surface meshes: Repulsive Surfaces [Yu et al. 2021a] and the off-the-shelf software Wrap [Wrap 2024]. For Repulsive Surfaces, we used the official implementation to resolve self-intersections. For Wrap, we employed its *Fix Intersections* tool, which resolves self-intersections using an as-rigid-as-possible approach. This tool requires calibration of five parameters: three weights for controlling collision, smoothness, and initial vertex preservation, and two distance thresholds for resolution and maximum collision. Obtaining a satisfactory result requires extensive parameter tuning tailored to each input geometry, which is a considerably time-consuming process due to the tool's slow optimization performance. To ensure fair comparison, we manually tuned all five parameters for each input.

7.2.3 Experimental Results. We evaluate our method against the baselines on our proposed benchmark. In Table 1, we first compare the number of self-intersecting triangles remaining after repair. Since each mesh contains a different number of triangles, we also present the normalized number of self-intersecting triangles, computed as the ratio to the total triangle count. While comparative methods still leave numerous self-intersections, our method successfully repairs self-intersections, except for folded geometries that are geodesically adjacent but produce conflicting repulsion directions, as discussed in the limitations section (Sec. 7.6). To demonstrate our method's effectiveness in completely eliminating geodesically

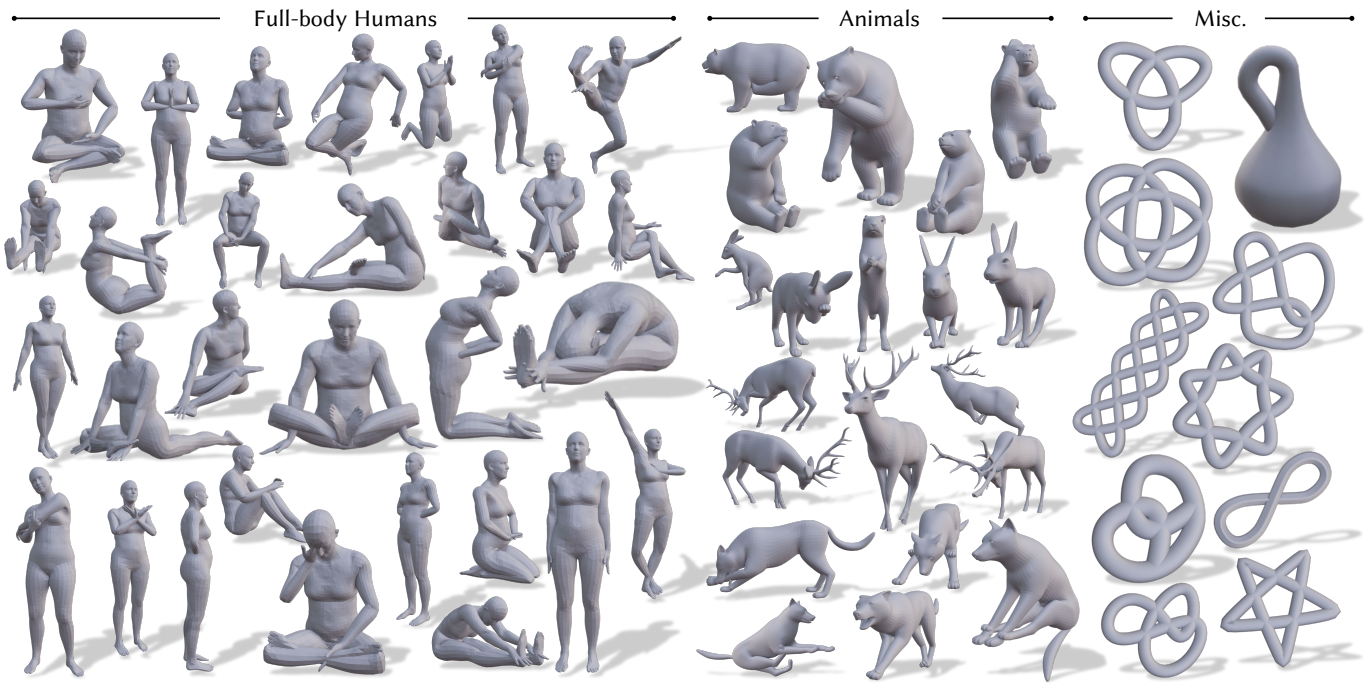


Fig. 7. Our self-intersection benchmark dataset. It contains diverse surface meshes with self-intersections categorized into three groups: full-body humans, animals, and miscellaneous objects including knots.

distant self-intersections, we additionally report the maximum geodesic distance between self-intersecting triangle pairs in the resulting meshes. When measuring geodesic distances, we first normalize all meshes to unit scale. Our method shows small maximum geodesic distances, indicating that self-intersections between disparate parts are successfully repaired.

In Table 2, we also compare the computational efficiency of our method with the baselines. We measured the time required for each method after performing optimization with a fixed number of 60 iterations. Other methods require substantially longer execution time to resolve self-intersections, whereas our method demonstrates superior performance, completing repairs within one second.

As illustrated in Fig. 16, we tested the limits of geometric complexity that can be handled by different methods using examples ranging from shallow penetrations (rows 1–2) to deep penetrations and intertwined structures (rows 3–6). Other methods encounter significant limitations in resolving deep penetrations and complex entangled structures. In addition, Repulsive Surfaces [Yu et al. 2021a] produces denser meshes due to dynamic remeshing, which is crucial for minimizing its tangent-point energy. In contrast, our method effectively handles these challenging cases without any remeshing operations.

7.3 Analysis

7.3.1 Performance. We analyze computational performance and GPU resource utilization of our method. Fig. 8 presents a detailed breakdown of execution times for different components in our pipeline and peak GPU memory usage during the repair phase.

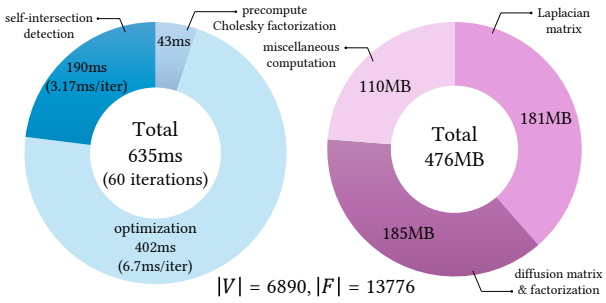


Fig. 8. Analysis of execution time and memory consumption. Our method demonstrates high efficiency, requiring only 635 ms for 60 iterations with a peak GPU memory usage of 476 MB. This computational efficiency makes our system viable for practical deployments in various use cases.

When tested on SMPL models [Loper et al. 2015] with complex self-intersections, our method takes an average repair time of 635 ms per model with peak GPU memory usage of 476 MB. When processing multiple data in succession, our framework takes average 484 ms with a warmed-up GPU.

Excluding self-intersection detection, our optimization framework contains three important steps: tangent-point energy calculation, gradient diffusion, and momentum updates. The tangent-point energy calculation is computationally inexpensive as it processes only a few intersecting triangle pairs. The gradient diffusion is performed via fast linear system solving using matrix pre-factorization. Momentum updates involve simple arithmetic operations. These

Table 1. Quantitative evaluation of self-intersection repairs using our benchmark dataset. We report both the number of remaining self-intersections (NSI) after repair and the normalized count (NC) relative to the total triangle count to avoid mesh density bias. We also measure the maximum geodesic distance (MGD) between self-intersecting triangles, where a lower value indicates successful resolution of self-intersections between geodesically distant parts.

	Repulsive Surfaces [2021a]			Wrap [2024]			Ours		
	Human	Animal	Misc.	Human	Animal	Misc.	Human	Animal	Misc.
NSI	481.57	227.6	571.3	418.43	480.65	265.9	243.27	64.75	32.2
NC	0.0223	0.0078	0.1082	0.0301	0.0230	0.0510	0.0176	0.0030	0.0042
MGD	0.6175	0.4853	3.6303	0.3720	0.9253	4.4773	0.1191	0.0450	0.0165

Table 2. Comparison with other methods on wall-clock performance. We measured the average execution time using our benchmark dataset.

	Repulsive Surfaces	Wrap	Ours
Execution time (sec)	82.440	43.247	0.635

efficient steps enable our optimization to achieve remarkable performance at 6.7 ms per iteration.

7.3.2 Parameter Tuning. Our method incorporates two controllable parameters: the diffusion radius λ and the regularization weight β. Setting the diffusion radius λ too small causes our method to become excessively sensitive to local updates, while an overly large λ produces undesirable stiff geometry updates. Similarly, an insufficient regularization weight β fails to preserve the original input shape, whereas an excessive β inhibits necessary deformation. Achieving desirable results requires tuning of these two parameters. As demonstrated in Fig. 9, our method produces valid outputs within the acceptable parameter ranges: $5 \times 10^5 \leq \beta \leq 2 \times 10^6$ and $9 \leq \lambda \leq 140$.

In contrast to Wrap [Wrap 2024], which necessitates precise tuning of five parameters whenever input mesh characteristics change, our approach demonstrates robust generalization across different inputs once the parameters are properly set. We used fixed parameters for all experiments, except in Fig. 13, where we intentionally increased the diffusion radius λ to achieve stronger geometric stiffness.

7.3.3 Robustness of Repulsive Energies. While simple self-intersections with shallow penetrations can be easily resolved, robust repulsive energies are essential for handling more complex cases in diverse surface meshes. To evaluate the robustness of different repulsive energies, we perform a stress test on a challenging example featuring complex self-intersections. As demonstrated in Fig. 10, conventional energies become stuck or diverge even with sufficient iterations. In contrast, our proposed signed tangent-point energy exhibits superior robustness in this challenging case. This improvement is derived from two key properties: the inherent ability of the tangent-point energy to discourage self-intersections and its consistent repulsive forces induced by the signed formulation.

7.3.4 Convergence Rate of Optimizers. We evaluate our MomentumBrake optimizer against gradient descent and UniformAdam [Nicolet et al. 2021]. We exclude the standard Adam optimizer [Kingma and Ba 2015] from this comparison due to its inherent limitations in geometry updates caused by adaptive normalization (Fig. 6). To

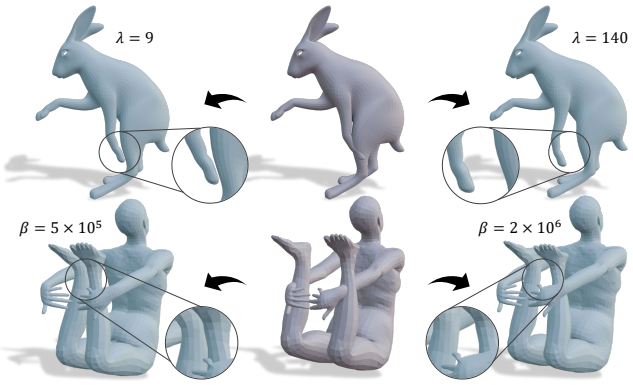


Fig. 9. Our method supports a broad range of parameter values, consistently producing valid results across the spectrum. Within the range, each parameter introduces distinct geometric variations: the diffusion radius λ modulates geometric stiffness, and the regularization weight β controls the degree of shape preservation.

Table 3. Convergence rates of optimizers on our benchmark. We report the average number of iterations required to reach a self-intersection-free configuration.

	Gradient Descent	UniformAdam	MomentumeBrake (Ours)
Human	115.43	82.97	47.63
Animal	64.35	53.65	34.75
Misc.	24.7	19.4	12.1
Total	83.28	62.60	37.42

quantitatively assess the effectiveness of the optimizer, we measure the number of iterations required to repair self-intersections on our dataset (Table 3). UniformAdam generally achieves faster convergence than gradient descent, but its redundant normalization hinders convergence by reducing the step size. Our MomentumBrake optimizer achieves the fastest convergence by leveraging momentum and removing redundant normalization, while moderating overshooting artifacts through momentum reset.

7.4 Repulsion Direction Control

While surface normals of self-intersecting triangles primarily determine the repulsive direction in tangent-point energy, simply following these geometric constraints may not always produce results

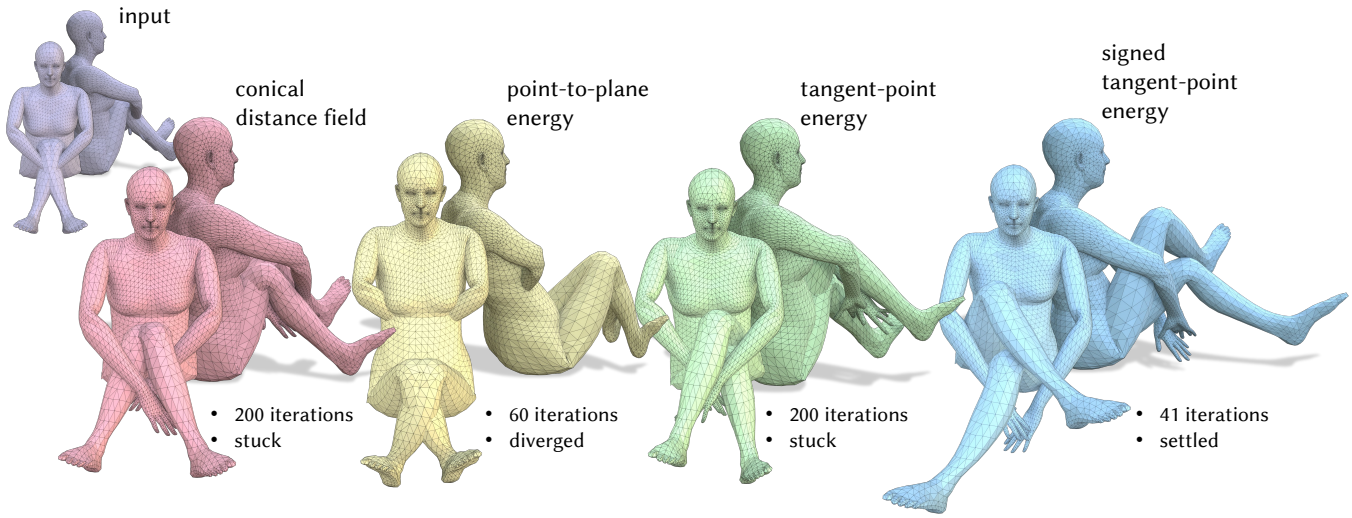


Fig. 10. Robustness of repulsive energies. Given an input surface mesh with complex self-intersections, conical distance field and unsigned tangent-point energy fail to resolve the intersections, becoming stuck in repetitive solutions. Point-to-plane energy diverges toward collapsed geometry due to continuous backward updates on hands. Our signed tangent-point energy successfully repairs this challenging case.

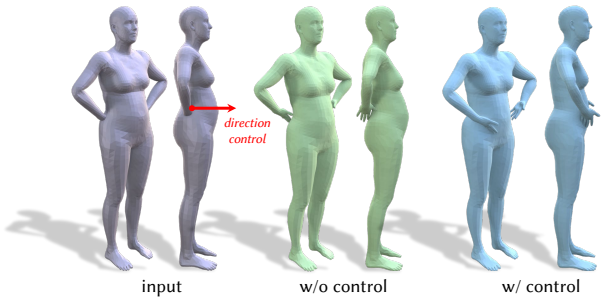


Fig. 11. Repulsive direction control. While our method finds a geometrically close solution (middle), it may not always coincide with the user intention. Our method offers direct control for the repulsion direction (right).

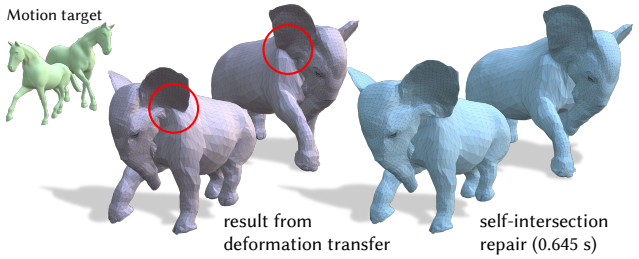


Fig. 12. Self-intersection repair in deformation transfer. When transferring motion via deformation transfer, self-intersection-free results are not generally guaranteed. By combining our method with deformation transfer, we achieve immediate generation of physically valid meshes with the intended motion.

that match user intentions. To allow a user to guide the deformation process, we introduce an intuitive control mechanism. Given a preferred repair direction for a user-specified surface region, the surface normals of triangles that intersects with the surface region are replaced with the user-specified repair direction in tangent-point energy calculation. This approach effectively enables user control in our self-intersection repair process (Fig. 11).

7.5 Applications

7.5.1 Self-Intersection-Free Deformation Transfer. Deformation transfer [Sumner and Popović 2004] is a valuable tool for mesh animation reuse by mapping animation sequences defined on a source mesh to the target mesh with different topology. This technique can significantly enhance artists' productivity by allowing them to repurpose existing animations for different character models. While this seminal technique has demonstrated its practical value over decades, it still faces a persistent limitation: the generated animations often contain self-intersections due to shape discrepancy between source and target meshes. By combining our self-intersection repair framework with deformation transfer, we can instantly obtain plausible transfer results without self-intersections (Fig. 12).

7.5.2 Inter-Objects Intersection Repair. Our method can be easily extended to handle intersections between two different objects beyond self-intersection repair. Our process for inter-object intersection repair follows the same principles as self-intersection repair, with one key distinction: we employ separate diffusion matrices and compute distinct gradient diffusion of local repulsive energies for each object. This approach robustly resolves inter-object intersections while preserving the geometric characteristics of each object (Fig. 13).

7.5.3 Key Pose Repair for Motion Sequence without Self-Intersection. A common approach in motion animation production is to generate key poses and interpolate between them. However, when key

Fig. 13. *Draw the sword from the stone.* Our method can repair inter-object penetrations using only local repulsive energies from intersecting triangles.

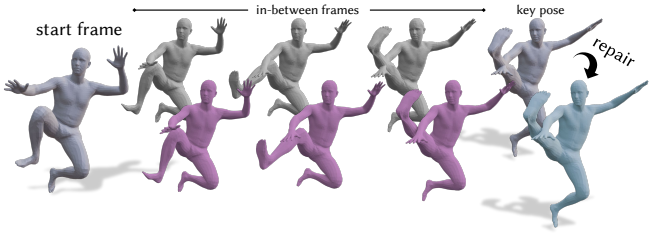
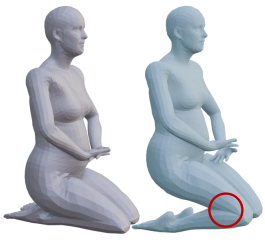


Fig. 14. Key pose repair for self-intersection-free motion sequence. Key pose meshes containing self-intersections may lead to implausible shapes when used for generating in-between motion frames. Our automatic method for repairing self-intersections can notably reduce the need for manual correction of key poses.

poses contain self-intersections, the resulting animation sequence would exhibit physically invalid motions. Our framework can be integrated into the artist's workflow by automatically repairing self-intersecting poses (Fig. 14), eliminating the need for manual adjustments to character poses. This automation can reduce artist workload, ensuring physically plausible animation sequences.

7.6 Limitations

Our iterative local optimization and momentum-based updates assume locally consistent gradients of repulsive energy, as mentioned in Section 6.1. An input violating this assumption, e.g., a triangle mesh with randomly connected vertices on a unit circle that contains self-intersections, may induce inconsistent repulsion, which remains a limitation of our method. In addition, our method assumes the existence of a topologically self-intersection-free solution. If this condition is not met, any existing method, including ours, will inevitably fail (Fig. 15).



While our method effectively repairs self-intersections by combining local repulsive energies and gradient diffusion, it exhibits varying effectiveness depending on the geodesic proximity of self-intersecting regions. For geodesically distant self-intersecting parts, our gradient diffusion successfully approximates global repulsion forces, leading to effective repair. However, when self-intersecting regions are geodesically adjacent, the diffused gradients can interfere with each other, causing attenuation of repulsive forces. This interference may result in slower convergence or incomplete resolution of local self-intersections.

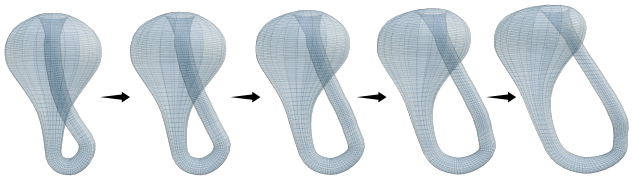


Fig. 15. Limitation. Like all existing approaches, our method fails on surfaces such as the Klein bottle, which do not admit a topologically self-intersection-free embedding.

8 Conclusion

We presented a novel framework for self-intersection repair in static 3D surface meshes by reformulating global intersection handling as an efficient local optimization problem. Our method combines local signed tangent-point energy with gradient diffusion, and incorporates the MomentumBrake optimizer that statistically regulates momentum to prevent overshooting while achieving rapid convergence. Extensive evaluations demonstrate that our approach enables fast and robust repair of complex self-intersections, providing a practical solution for geometry processing pipelines.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was supported by NRF grants (RS-2023-00280400, RS-2024-00451947) and IITP grants (RS-2022-II220290, RS-2024-00437866, RS-2021-II212068) funded by the Korean government (MSIT).

References

Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G Kry. 2010. Volume contact constraints at arbitrary resolution. In *ACM SIGGRAPH 2010 papers*. 1–10.

Ted Belytschko and Mark O Neal. 1991. Contact-impact by the pinball algorithm with penalty and Lagrangian methods. *Internat. J. Numer. Methods Engrg.* 31, 3 (1991), 547–572.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2023. Projective dynamics: Fusing constraint projections for fast simulation. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 787–797.

Gregory Buck and Jeremey Orloff. 1995. A simple energy function for knots. *Topology and its Applications* 61, 3 (1995), 205–214.

He Chen, Elie Diaz, and Cem Yuksel. 2023. Shortest Path to Boundary for Self-Intersecting Meshes. *ACM Trans. Graph.* 42, 4 (2023).

Vassilis Choutas. 2019. torch-mesh-isect. https://github.com/vchoutas/torch-mesh-isect.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2017. The heat method for distance computation. *Commun. ACM* 60, 11 (2017), 90–99.

Ounan Ding and Craig Schroeder. 2019. Penalty force for coupling materials with Coulomb friction. *IEEE transactions on visualization and computer graphics* 26, 7 (2019), 2443–2455.

Evan Drumwright. 2007. A fast and stable penalty method for rigid body simulation. *IEEE transactions on visualization and computer graphics* 14, 1 (2007), 231–240.

Gerhard Dziuk. 1988. *Finite elements for the Beltrami operator on arbitrary surfaces*. Springer.

Yu Fang, Minchen Li, Chenfanfu Jiang, and Danny M Kaufman. 2021. Guaranteed globally injective 3D deformation processing. *ACM Trans. Graph.* 40, 4 (2021).

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M Kaufman, and Daniele Panozzo. 2021. Intersection-free rigid body dynamics. *ACM Trans. Graph.* 40, 4 (2021).

Si Hang. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw* 41, 2 (2015), 11.
Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast tetrahedral meshing in the wild. *ACM Trans. Graph.* 39, 4 (2020), 117–1.
Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60.
I Huněk. 1993. On a penalty formulation for contact-impact problems. *Computers & structures* 48, 2 (1993), 193–203.
Changsoo Je, Min Tang, Youngeun Lee, Minkyoung Lee, and Young J Kim. 2012. PolyDepth: Real-time penetration depth computation using iterative contact-space projection. *ACM Trans. Graph.* 31, 1 (2012), 1–14.
Yucheol Jung, Hyomin Kim, Gyeongha Hwang, Seung-Hwan Baek, and Seungyong Lee. 2023. Mesh density adaptation for template-based shape reconstruction. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–10.
J Karatson and L Loczi. 2005. Sobolev gradient preconditioning for the electrostatic potential equation. *Computers & Mathematics with Applications* 50, 7 (2005), 1093–1104.
Tero Karras. 2012. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*. 33–37.
Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980
Lei Lan, Danny M Kaufman, Minchen Li, Chenfanfu Jiang, and Yin Yang. 2022a. Affine body dynamics: Fast, stable & intersection-free simulation of stiff materials. *arXiv preprint arXiv:2201.10022* (2022).
Lei Lan, Guanqun Ma, Yin Yang, Changxi Zheng, Minchen Li, and Chenfanfu Jiang. 2022b. Penetration-free projective dynamics on the GPU. *ACM Trans. Graph.* 41, 4 (2022).
Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy R Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4 (2020).
Xuan Li, Minchen Li, and Chenfanfu Jiang. 2022. Energetically consistent inelasticity for optimization time integration. *ACM Trans. Graph.* 41, 4 (2022), 1–16.
Yijing Li and Jernej Barbič. 2018. Immersion of self-intersecting solids and surfaces. *ACM Trans. Graph.* 37, 4 (2018), 1–14.
Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossi, and Hans-Peter Seidel. 2004. Differential coordinates for interactive mesh editing. In *Proceedings Shape Modeling Applications, 2004.* IEEE, 181–190.
Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16.
Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*. 49–54.
Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 35–57.
Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *J. Vis.* 18, 2 (2007), 109–118.
Maxim Naumov. 2011. Parallel solution of sparse triangular linear systems in the preconditioned iterative methods on the GPU. *NVIDIA Corp., Westford, MA, USA, Tech. Rep. NVR-2011* 1 (2011).
Yurii Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Dokl. Akad. Nauk. SSSR*, Vol. 269. 543.
JW Neuberger. 1985. Steepest descent and differential equations. *Journal of the Mathematical Society of Japan* 37, 2 (1985), 187–195.
Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large steps in inverse rendering of geometry. *ACM Trans. Graph.* 40, 6 (2021), 1–13.
Katsuhiko Ogata. 1995. *Discrete-time control systems*. Prentice-Hall, Inc.
Stanley Osher, Bao Wang, Penghang Yin, Xiyang Luo, Farzin Barekat, Minh Pham, and Alex Lin. 2022. Laplacian smoothing gradient descent. *Research in the Mathematical Sciences* 9, 3 (2022), 55.
Matthew Overby, Danny Kaufman, and Rahul Narain. 2021. Globally Injective Geometry Optimization with Non-Injective Steps. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 111–123.
Jea-Hyun Park, Abner J Salgado, and Steven M Wise. 2021. Preconditioned accelerated gradient descent methods for locally Lipschitz smooth objectives with applications to the solution of nonlinear PDEs. *Journal of Scientific Computing* 89, 1 (2021), 17.
Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics* 2, 1 (1993), 15–36.
Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics* 4, 5 (1964), 1–17.
Radik Bilalov. 2025. *RedDeer3D*. https://radbill.artstation.com/
Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 145–152.
Leonardo Sacht, Alec Jacobson, Daniele Panozzo, Christian Schüller, and Olga Sorkine-Hornung. 2013. Consistent volumetric discretizations inside self-intersecting surfaces. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 147–156.
Josua Sassen, Henrik Schumacher, Martin Rumpf, and Keenan Crane. 2024. Repulsive Shells. *ACM Trans. Graph.* 43, 4 (2024).
Fredrik Schaufelberger. 2020. Open questions in functional molecular topology. *Communications Chemistry* 3, 1 (2020), 182.
Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 175–184.
Paweł Strzelecki and Heiko von der Mosel. 2013. Tangent-Point Repulsive Potentials for a Class of Non-smooth m-dimensional Sets in n. Part I: Smoothing and Self-avoidance Effects. *Journal of Geometric Analysis* 23, 3 (2013), 1085–1139.
Paweł Strzelecki and Heiko von der Mosel. 2018. *Geometric curvature energies: facts, trends, and open problems*. Universitätsbibliothek der RWTH Aachen.
Robert W Sumner and Jovan Popović. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph.* 23, 3 (2004).
Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*. PMLR, 1139–1147.
Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. 2016. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision* 118 (2016), 172–193.
Mickeal Verschoor and Andrei C Jalba. 2019. Efficient and accurate collision response for elastically deformable models. *ACM Trans. Graph.* 38, 2 (2019), 1–20.
Bin Wang, François Faure, and Dinesh K Pai. 2012. Adaptive image-based intersection volume. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–9.
Yating Wang, Ran Yi, Ke Fan, Jinkun Hao, Jiangbo Lu, and Lizhuang Ma. 2024. Learning Topology Uniformed Face Mesh by Volume Rendering for Multi-view Reconstruction. *arXiv preprint arXiv:2404.05606* (2024).
Wrap. 2024. *Wrap*. https://faceform.com/
Chris Yu, Caleb Brakensiek, Henrik Schumacher, and Keenan Crane. 2021a. Repulsive surfaces. *ACM Trans. Graph.* 40, 6 (2021).
Chris Yu, Henrik Schumacher, and Keenan Crane. 2021b. Repulsive curves. *ACM Trans. Graph.* 40, 2 (2021).

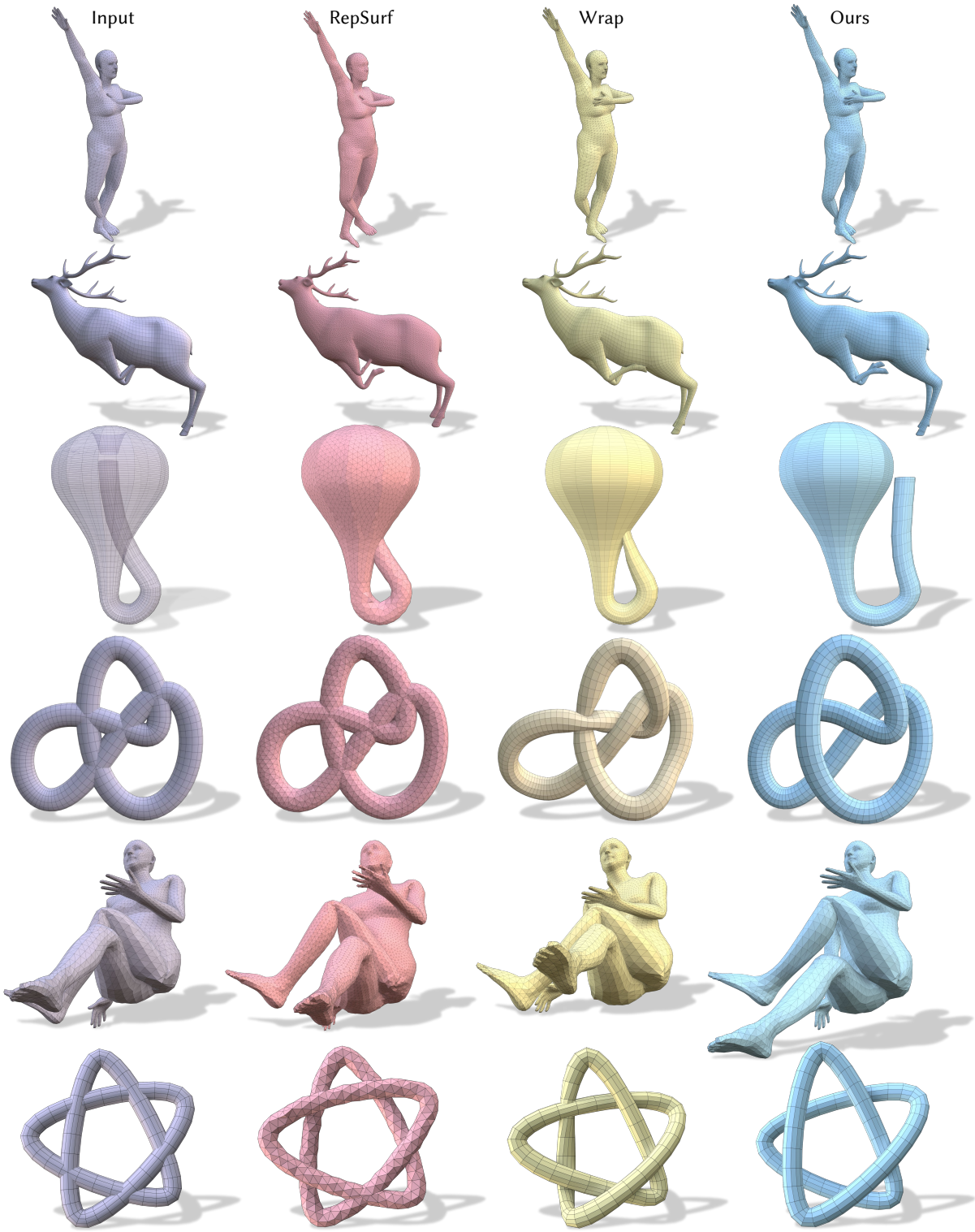


Fig. 16. Comparison of our method with Repulsive Surfaces (RepSurf) [Yu et al. 2021a] and Wrap [Wrap 2024] in handling various self-intersections. While Repulsive Surfaces and Wrap can partially resolve shallow penetrations with slow convergences, our method successfully resolves self-intersections across all penetration depths and achieves significantly faster computational performance.

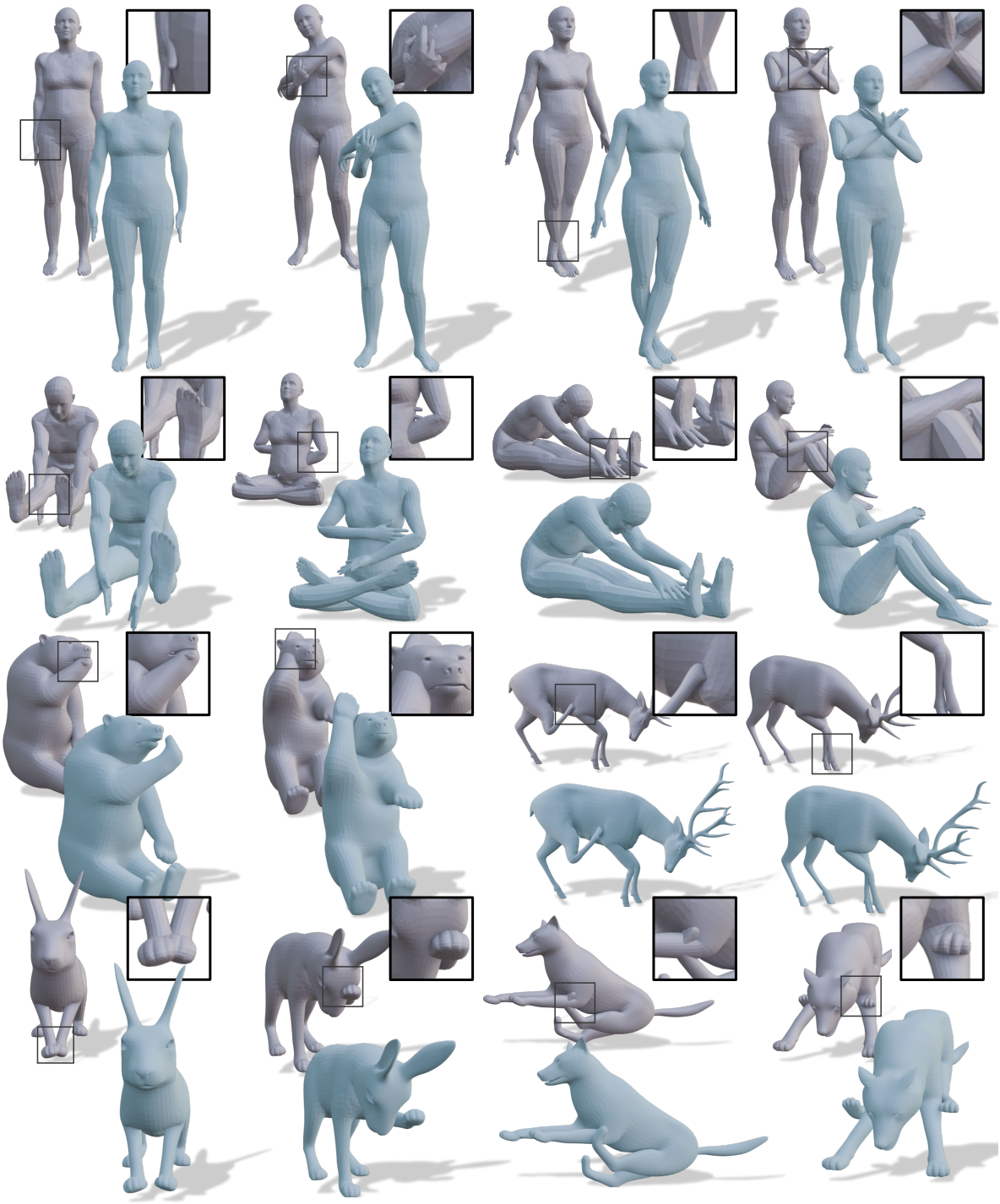


Fig. 17. Self-intersection repair on diverse 3D surface meshes with self-intersections. All cases are resolved in under one second.