2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Robust Differentiable Collision Detection for General Objects

Jiayi Chen[1,2], Wei Zhao[3,4], Liangwang Ruan[1,2], Baoquan Chen[1], He Wang[1,2†]

***Abstract*— Collision detection is a core component of robotics applications such as simulation, control, and planning. Traditional algorithms like GJK+EPA compute *witness points*—the closest or deepest-penetration pairs between two objects—but are inherently non-differentiable, preventing gradient flow and limiting gradient-based optimization in contact-rich tasks such as grasping and manipulation. Recent work introduced efficient first-order randomized smoothing to make witness points differentiable; however, their direction-based formulation is restricted to convex objects and lacks robustness for complex geometries. In this work, we propose a robust and efficient differentiable collision detection framework that supports both convex and concave objects across diverse scales and configurations. Our method introduces distance-based first-order randomized smoothing, adaptive sampling, and equivalent gradient transport for robust and informative gradient computation. Experiments on complex meshes from DexGraspNet and Objaverse show significant improvements over existing baselines. Finally, we demonstrate a direct application of our method for dexterous grasp synthesis to refine the grasp quality. The code is available at https://github.com/JYChen18/DiffCollision.**

I. Introduction

Collision detection is a long-standing and fundamental problem in robotics and computer graphics, with applications ranging from simulation and control to planning and learning. A key outcome of collision detection is the *witness points* [1]—the closest points when objects are separated or the deepest penetration points when they overlap. These points specify where and how objects interact: they directly indicate contact positions and allow computation of contact distance and direction, making them a cornerstone of contact modeling [2], [3].

Despite the critical role of these witness points, traditional algorithms such as GJK [4], [5], EPA [6], and MPR [7] are non-differentiable, preventing gradients from propagating through them. This lack of differentiability is one possible reason limiting the potential of gradient-based optimization in contact-rich tasks such as grasping and manipulation. As a result, robotics has often relied on gradient-free approaches such as reinforcement learning [8], [9] and sampling-based planners [10], in contrast to the success of gradient-based methods in deep learning.

Recent work has begun improving the differentiability of contact modeling [11], [12], [13], but mainly focuses on improving the contact solvers and smoothing the *magnitude* of contact forces with respect to inter-object distance, known as *force-at-a-distance*. In contrast, the differentiability of witness points themselves has received little attention. This omission is critical: without gradients through witness points, it is difficult to enforce desired contact positions or adjust force directions on complex shapes such as concave regions (Figure 2).

To bridge this gap, Montaut et al. [14] introduced efficient first-order randomized smoothing approaches to make witness point differentiable. However, their direction-based method is limited to convex objects and has only been extensively tested on simple meshes (as few as 12 vertices), falling short of the demands of real robotic applications.

In this work, we propose a general approach to calculate the derivative of witness points that applies to both convex

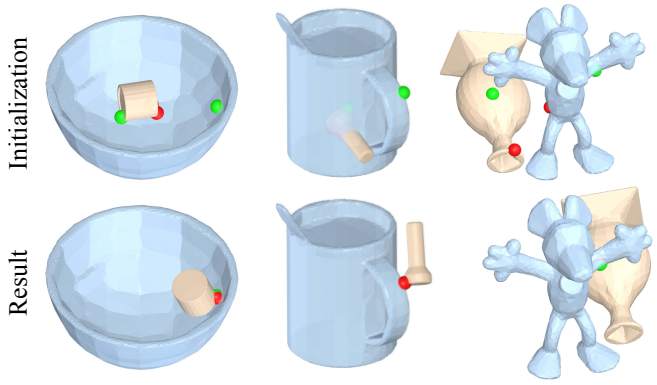

Fig. 1: **Task illustration.** We solve for the relative pose that aligns target points (green) with witness points (red). This task validates the computed derivatives of the witness points *with respect to* the object poses.

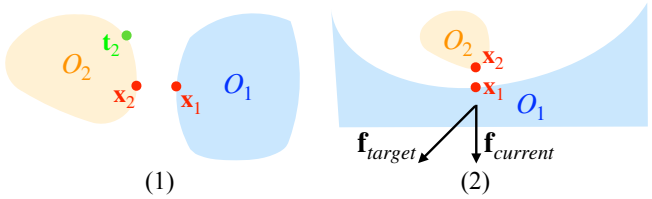

Fig. 2: **Motivation.** When optimizing the pose T_2 of object O_2, treating the witness points $\mathbf{x}_1, \mathbf{x}_2$ as fixed on objects O_1 and O_2 without considering their derivatives cannot handle *at least* two cases: (1) enforcing a specified point $\mathbf{t}_2$ to be a witness point (i.e., as the task in Fig. 1); (2) enforcing object O_2 to produce a target contact force $\mathbf{f}_{\text{target}}$ on a concave object O_1, where ignoring $\partial \mathbf{x}_1 / \partial T_2$ actually assumes that $\mathbf{x}_1$ is fixed. In this situation, to generate a leftward contact force, one would move $\mathbf{x}_2$ rightward, which is incorrect. By contrast, our method accounts for $\partial \mathbf{x}_1 / \partial T_2$, recognizing that when $\mathbf{x}_2$ moves rightward, $\mathbf{x}_1$ also shifts rightward even more. Thus, the correct update is to move O_2 leftward.

[1]Peking University. [2]Galbot. [3]Tsinghua University. [4]The Hong Kong Polytechnic University. †Corresponding author: hewang@pku.edu.cn.

and concave objects. As shown in Figure 1, our method is robust to object scale, handles both penetrating and non-penetrating configurations, and scales to complex shapes such as those in DexGraspNet [15] and Objaverse [16].

Our method belongs to the family of first-order randomized smoothing but introduces three key innovations. First, instead of relying on direction–based smoothing derived from GJK optimality conditions [14], we propose *distance–based smoothing*, which generalizes naturally beyond convex shapes. Second, rather than approximating local geometry from neighboring vertices—which is mesh-sensitive and not parallel-friendly—we propose an *adaptive sampling* strategy, yielding greater robustness. Third, we introduce *equivalent gradient transport*, which improves gradient quality when only one object's pose is optimized, a scenario frequently encountered in grasping and manipulation.

Experiments on both convex and concave objects demonstrate significant improvements over existing baselines. On complex meshes from DexGraspNet and Objaverse, our method achieves a median error below 0.1 mm across 400 random object pairs with 1024 tasks each. At mm-level accuracy, we outperform baselines by more than 40%. Our approach is also memory- and time-efficient, and easily parallelizable on GPUs. Finally, we demonstrate a direct application to dexterous grasp refinement, achieving improved grasp quality.

In summary, our contributions are:

- An open-source, robust, and efficient differentiable collision detection framework for both convex and concave objects, built upon distance-based smoothing with adaptive sampling and equivalent gradient transport;
- Competitive empirical performance on large-scale object datasets, demonstrating the robustness of our method to object shapes, scales and configurations;
- A preliminary application to dexterous grasp refinement.

II. Related Work

A. Discrete Collision Detection for Meshes

A major class of collision detection algorithms leverages the *Minkowski sum* to reduce collision detection to an origin-in-polytope test. These methods are highly efficient and widely used in rigid-body simulators such as MuJoCo [17] and PyBullet [18]. Representative algorithms include the Gilbert-Johnson-Keerthi (GJK) algorithm [4], [5], which iteratively searches for the closest point to the origin in the Minkowski difference; the Expanding Polytope Algorithm (EPA) [6], which refines the GJK result to compute penetration depth and witness points; and the Minkowski Portal Refinement (MPR) algorithm [7], which combines elements of both for practical efficiency. These methods assume convex meshes, and concave objects typically require convex decomposition [19], [20].

Another line of work relies on exact triangle intersection tests, which check for intersections between mesh triangles. While computationally more expensive, these methods are naturally parallelizable on GPUs. A common variant is the Separating Axis Test (SAT) [21], again limited to convex shapes. More general methods explicitly check vertex–face and edge–edge pairs, allowing them to handle concave or deformable objects. However, they cannot identify deepest penetration points and typically require penetration-free configurations at each timestep [11], making them less common in robotics applications.

All of the above methods are non-differentiable, except for vertex–face and edge–edge tests, which provide highly limited and local gradients. In contrast, our method computes derivatives of witness points that are compatible with any forward collision detection algorithm, enabling gradient flow through witness points.

B. Differentiable Collision Detection

Interest in differentiable collision detection has grown only recently, largely motivated by the progress of differentiable simulation [22], [23], [24]. Tracy et al. [25] formulate collision detection as a convex optimization problem of scaling the two objects, which naturally yields derivatives. However, their method is restricted to convex primitives and scales poorly to complex meshes compared to GJK. Montaut et al. [14] introduce random smoothing techniques to approximate derivatives. The zeroth-order method, akin to finite differences, requires multiple collision queries per backward pass and is thus inefficient. The first-order method leverages the optimality conditions of the GJK algorithm, offering efficiency but restricted to strictly convex shapes and not robust enough for complex meshes.

Inspired by [14]'s first-order method, we propose a general approach for both convex and concave objects with improved robustness, while preserving computational efficiency.

III. Preliminaries on the $\mathrm{SE}(3)$ Lie Group

a) Lie algebra: The tangent space of $\mathrm{SE}(3)$ at the identity element is denoted as Lie algebra $\mathfrak{se}(3)$:

$$\mathfrak{se}(3) = \left\{ \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} \;\middle|\; \omega, v \in \mathbb{R}^3 \right\} \tag{1}$$

where $[\omega]_\times$ is the skew-symmetric matrix of ω. The standard exponential map

$$\exp : \mathfrak{se}(3) \to \mathrm{SE}(3) \tag{2}$$

is a local diffeomorphism around the identity and serves as a natural retraction, mapping elements of the tangent space back onto the group $\mathrm{SE}(3)$.

b) Adjoint operator: For $T \in \mathrm{SE}(3)$, the adjoint operator describes how a rigid-body transformation conjugates Lie algebra elements:

$$\mathrm{Ad}_T : \mathfrak{se}(3) \to \mathfrak{se}(3), \quad \mathrm{Ad}_T(\xi) \;=\; T\,\xi\,T^{-1}. \tag{3}$$

The adjoint operator satisfies the fundamental identity

$$T \exp(\xi) T^{-1} = \exp\big((\mathrm{Ad}_T \xi)\big), \tag{4}$$

which is widely used in robotics to transport velocities, Jacobians, and differentials between coordinate frames.

For further details on $\mathrm{SE}(3)$, its Lie algebra, and the adjoint operator, we refer the reader to [26].

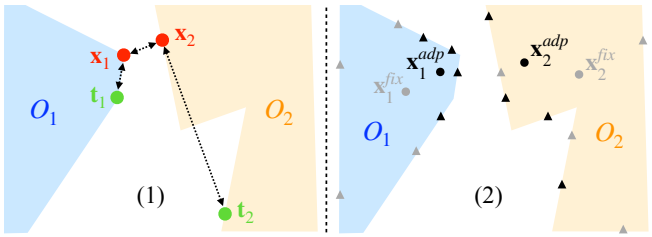

Fig. 3: **(1) Task formulation.** The witness points $\mathbf{x}_1, \mathbf{x}_2$ calculated by collision detection are expected to match specified target points $\mathbf{t}_1, \mathbf{t}_2$ via the losses shown by dotted lines. **(2) Smoothing witness points.** Adaptive sampling yields better surface samples (triangles) than fixed sampling, improving the approximation of $\mathbf{x}_1, \mathbf{x}_2$.

IV. Methods

A. Task Formulation

Let O_i be objects with poses $T_i \in SE(3)$ for $i \in \{1, 2\}$. We denote world-frame points as $\mathbf{p}_i = T_i \mathbf{p}_{i,o}$, where $\mathbf{p}_{i,o}$ is the representation in the local frame of O_i. Next, we define:

- **Witness points** $\mathbf{x}_i$: The closest/penetrating points between O_1 and O_2 computed via GJK/EPA. $i \in \{1, 2\}$.
- **Target points** $\mathbf{t}_i$: The target contact points specified per task and fixed in the local frame of O_i. $i \in \{1, 2\}$.
- **Boundary samples** $\{\mathbf{v}_{i,j}\}_{j=1}^N$: N points randomly sampled to approximate the object surface geometry.

The objective is to optimize the poses T_1, T_2 such that the witness points approach the targets:

$$\min_{T_1, T_2} \mathcal{L} = \|\mathbf{x}_1 - \mathbf{x}_2\|^2 + \|\mathbf{x}_1 - \mathbf{t}_1\|^2 + \|\mathbf{x}_2 - \mathbf{t}_2\|^2 \quad (5)$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ depend on T_1 and T_2. Our proposed algorithm calculates $\nabla_{T_i} \mathcal{L}$ through $\mathbf{x}_i$, as shown in Algorithm 1.

B. Distance–Based Softmax Smoothing

Without loss of generality, we focus on $\mathbf{x}_1$, as $\mathbf{x}_1$ and $\mathbf{x}_2$ are symmetric and the results for $\mathbf{x}_2$ follow analogously.

We begin by reformulating $\mathbf{x}_1$ as

$$\mathbf{x}_1 = \underset{\mathbf{v} \in T_1 \partial O_1}{\operatorname{argmax}} \left(- \|\mathbf{v} - \mathbf{x}_2\|^2 \right), \quad (6)$$

where ∂O_1 denotes the boundary of object O_1 and $T_1 \partial O_1$ denotes the object boundary in the world frame. This formulation is exact when the two objects do not overlap. In penetration cases, it becomes an approximation, but we find empirically that our method still performs well—likely because the penetration is penalized by the first term in the loss function (Eq. 5). Since penetration is typically undesirable in robotic tasks, such a term is commonly included in gradient-based methods, making this formulation a reasonable approximation.

To obtain a differentiable surrogate, we sample N points $\{\mathbf{v}_{1,j}\}_{j=1}^N \in T_1 \partial O_1$ and replace the non-smooth argmax with a softmax relaxation. Specifically, we define the score value $u_j \in \mathbb{R}$ and softmax weights $w_j \in \mathbb{R}$

$$u_j = -\|\mathbf{v}_{1,j} - \mathbf{x}_2\|^2, \quad w_j = \frac{\exp(u_j/\tau)}{\sum_{k=1}^N \exp(u_k/\tau)} \quad (7)$$

Algorithm 1 Robust Differentiable Collision Detection

Input: Objects O_1, O_2;Poses T_1, T_2;Gradients $\nabla_{\mathbf{x}_1}\mathcal{L}, \nabla_{\mathbf{x}_2}\mathcal{L}$
Output: Witness points $\mathbf{x}_1, \mathbf{x}_2$; Gradients $\nabla_{T_1}\mathcal{L}, \nabla_{T_2}\mathcal{L}$

function FORWARD(T_1, T_2)
  ▷ *Compute witness points using standard solver*
  $\mathbf{x}_1, \mathbf{x}_2 \leftarrow$ GJK(O_1, O_2, T_1, T_2)
  SAVEFORBACKWARD($O_1, O_2, T_1, T_2, \mathbf{x}_1, \mathbf{x}_2$)
  return $\mathbf{x}_1, \mathbf{x}_2$

function BACKWARD($\nabla_{\mathbf{x}_1}\mathcal{L}, \nabla_{\mathbf{x}_2}\mathcal{L}$)
  ▷ *Fixed or Adaptive Sampling (Sec. IV-C)*
  $\mathbf{V}_1, \mathbf{V}_2 \leftarrow$ SampleSurface($O_1, O_2, T_1, T_2, \mathbf{x}_1, \mathbf{x}_2$)
  ▷ *Distance-based Randomized Smoothing (Sec. IV-B)*
  $\mathbf{x}_1^\star, \mathbf{x}_2^\star \leftarrow$ WeightedSum($\mathbf{V}_1, \mathbf{V}_2, \mathbf{x}_1, \mathbf{x}_2$) ▷ Eq. 8
  ▷ *Derivative Calculation (Sec. IV-D)*
  $\frac{\partial \mathbf{x}_1}{\partial T_1}, \frac{\partial \mathbf{x}_2}{\partial T_2} \leftarrow$ ComputeSelfJacob($\mathbf{x}_1^\star, \mathbf{x}_2^\star$) ▷ Eq. 11
  $\frac{\partial \mathbf{x}_1}{\partial T_2}, \frac{\partial \mathbf{x}_2}{\partial T_1} \leftarrow$ ComputeCrossJacob($\mathbf{x}_1^\star, \mathbf{x}_2^\star$) ▷ Eq. 12
  $\nabla_{T_1}\mathcal{L} \leftarrow \nabla_{\mathbf{x}_1}\mathcal{L} \cdot \frac{\partial \mathbf{x}_1}{\partial T_1} + \nabla_{\mathbf{x}_2}\mathcal{L} \cdot \frac{\partial \mathbf{x}_2}{\partial T_1}$ ▷ Chain rule
  $\nabla_{T_2}\mathcal{L} \leftarrow \nabla_{\mathbf{x}_1}\mathcal{L} \cdot \frac{\partial \mathbf{x}_1}{\partial T_2} + \nabla_{\mathbf{x}_2}\mathcal{L} \cdot \frac{\partial \mathbf{x}_2}{\partial T_2}$
  ▷ *Equivalent Gradient Transport (Sec. IV-E)*
  $\nabla_{T_1}\mathcal{L}, \nabla_{T_2}\mathcal{L} \leftarrow$ EGT($T_1, T_2, \nabla_{T_1}\mathcal{L}, \nabla_{T_2}\mathcal{L}$)
  return $\nabla_{T_1}\mathcal{L}, \nabla_{T_2}\mathcal{L}$

where τ is the temperature parameter. In practice, we find that setting $\tau = \text{std}(\mathbf{u})$ (i.e., the standard deviation of u_i) works well, as it adapts automatically to object scale and avoids manual tuning. The witness point is then approximated by the weighted combination of sampled points:

$$\mathbf{x}_1 \approx \mathbf{x}_1^\star = \sum_{j=1}^N w_j \mathbf{v}_{1,j} = \mathbf{V}_1 \mathbf{w}_1^T, \quad (8)$$

where $\mathbf{w}_1 = [w_1, ..., w_N] \in \mathbb{R}^N$ and $\mathbf{V}_1 = [\mathbf{v}_{1,1}, ..., \mathbf{v}_{1,N}] \in \mathbb{R}^{3 \times N}$. If $\mathbf{x}_{1,o} \in \{\mathbf{v}_{1,j,o}\}$, then $\|\mathbf{x}_1 - \mathbf{x}_1^\star\| \to 0$ as $\tau \to 0$.

C. Sampling Strategies

The sampled points $\mathbf{v}_{1,j,o}$ provide a discrete approximation of the object surface and are thus critical to the accuracy of the witness point approximation. To balance quality and efficiency, we first generate a large candidate set (hundreds of points) as a preprocessing step, and then select a small subset (e.g., 16 points) in each iteration as the local approximation.

Formally, let $P_{\text{static}} = P_{\text{ver}} \cup P_{\text{sur}}$ be the fixed set of object vertices and random surface samples. At each iteration, we augment this with the current witness point $\mathbf{x}_{1,o}$ to form the full candidate set $P = P_{\text{static}} \cup \{\mathbf{x}_{1,o}\}$. We then select the active local subset by filtering P based on the Euclidean distance to $\mathbf{x}_{1,o}$:

$$\{\mathbf{v}_{1,j,o}\} = \{\mathbf{v}_{1,j,o} \in P : \|\mathbf{v}_{1,j,o} - \mathbf{x}_{1,o}\| \le \alpha\}. \quad (9)$$

We consider two strategies for choosing α:

- **Fixed Sampling**: α is a constant hyperparameter used throughout the optimization.
- **Adaptive Sampling**: α is dynamically updated at each iteration as $\alpha = \max\left(\|\mathbf{t}_1 - \mathbf{x}_1\|, \varepsilon\right)$, where $\mathbf{t}_1$ is the

target contact point and ε is a small positive constant for numerical stability. Adaptive sampling is particularly helpful for concave objects with complex geometries.

D. Derivative Calculation

Applying the chain rule on Eq. 8:

$$\frac{\partial \mathbf{x}_1}{\partial T_1} \approx \frac{\partial \mathbf{x}_1^\star}{\partial T_1} = \frac{\partial \mathbf{x}_1^\star}{\partial \mathbf{V}_1}\frac{\partial \mathbf{V}_1}{\partial T_1} + \frac{\partial \mathbf{x}_1^\star}{\partial \mathbf{w}_1}\frac{\partial \mathbf{w}_1}{\partial T_1}. \tag{10}$$

When differentiating with respect to T_1, we treat $\mathbf{x}_2$ as fixed. Under this assumption, the expression becomes

$$\frac{\partial \mathbf{x}_1}{\partial T_1} \approx \frac{\partial \mathbf{x}_1^\star}{\partial \mathbf{V}_1}\frac{\partial \mathbf{V}_1}{\partial T_1} + \frac{\partial \mathbf{x}_1^\star}{\partial \mathbf{w}_1}\frac{\partial \mathbf{w}_1}{\partial \mathbf{V}_1}\frac{\partial \mathbf{V}_1}{\partial T_1}. \tag{11}$$

This approximation is exact when $\mathbf{x}_2$ is a vertex of T_2O_2, since small perturbations of T_1 leave its position unchanged. More generally, we find empirically that the same assumption also yields a reasonable approximation when $\mathbf{x}_2$ lies on the surface of O_2. An analogous derivation gives $\partial \mathbf{x}_2/\partial T_2$.

Next, we consider the cross derivative of $\mathbf{x}_1$ with respect to T_2. Since $\partial \mathbf{V}_1/\partial T_2 = 0$, we obtain

$$\frac{\partial \mathbf{x}_1}{\partial T_2} \approx \frac{\partial \mathbf{x}_1^\star}{\partial T_2} = \frac{\partial \mathbf{x}_1^\star}{\partial \mathbf{w}_1}\frac{\partial \mathbf{w}_1}{\partial \mathbf{x}_2}\frac{\partial \mathbf{x}_2}{\partial T_2}. \tag{12}$$

Because an approximation of $\partial \mathbf{x}_2/\partial T_2$ has already been derived as Eq. 11, it can be directly substituted into the above expression to evaluate $\partial \mathbf{x}_1/\partial T_2$, thereby avoiding reusing the assumption that $\mathbf{x}_{2,o}$ is fixed. By symmetry, the cross derivative $\partial \mathbf{x}_2/\partial T_1$ can be obtained in the same manner.

E. Equivalent Gradient Transport

In practical robotic tasks such as grasping, it is common to optimize only the robot's pose while keeping the object fixed. Suppose T_1 is fixed during optimization. In this case, directly updating T_2 alone can be inefficient, because the task's inherent symmetry is broken and T_2 may need larger adjustments than T_1 to reach the target points $\mathbf{t}_1$ if the initialization is far from the target. To tackle it, we propose a strategy to transfer the update that would nominally apply to T_1 to T_2 in an equivalent manner, as shown in Figure 4.

Given the gradient in the ambient space $G_i = \partial L/\partial T_i \in \mathbb{R}^{4\times 4}$, the corresponding Lie algebra element is obtained as

$$\xi_i = \mathrm{Proj}_{\mathfrak{se}(3)}\big(T_i^{-1}G_i\big), \quad i = 1, 2, \tag{13}$$

where $\mathrm{Proj}_{\mathfrak{se}(3)}(\cdot)$ denotes the projection of a 4×4 matrix onto the linear space $\mathfrak{se}(3)$. The following lemma states how to obtain the *equivalent gradient* transported from T_1 to T_2.

Lemma 1. For any $T_1, T_2 \in \mathrm{SE}(3)$, $\xi_1 \in \mathfrak{se}(3)$, and $\lambda \in \mathbb{R}$, updating T_1 with $-\lambda\xi_1$ is equivalent, in terms of the relative pose, to updating T_2 with $-\lambda\tilde{\xi}_2$, where

$$\tilde{\xi}_2 = -\mathrm{Ad}_{T_2^{-1}T_1}\xi_1. \tag{14}$$

Formally, the equivalent relative pose is given by

$$\left(T_1\exp(-\lambda\xi_1)\right)^{-1}T_2 = T_1^{-1}\left(T_2\exp(-\lambda\tilde{\xi}_2)\right). \tag{15}$$

Proof. Since $(\exp(-\lambda\xi_1))^{-1} = \exp(\lambda\xi_1)$, we have

$$\big(T_1\exp(-\lambda\xi_1)\big)^{-1}T_2 = \exp(\lambda\xi_1)T_1^{-1}T_2. \tag{16}$$

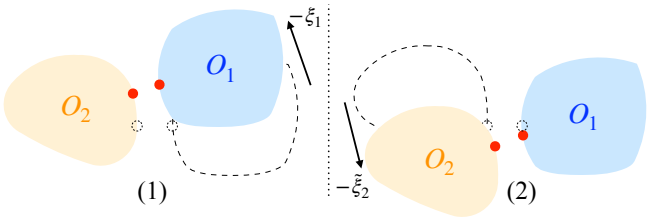

Fig. 4: **Equivalent gradient transport (EG).** Updating T_1 with gradient ξ_1 produces the same relative pose as updating T_2 with our proposed *equivalent gradient* $\tilde{\xi}_2$. The object and witness points before the update are shown as dotted lines.

By the adjoint identity Eq. 4,

$$\exp(\lambda\xi_1)T_1^{-1}T_2 = T_1^{-1}T_2\exp\big(\lambda\,\mathrm{Ad}_{T_2^{-1}T_1}\xi_1\big). \tag{17}$$

Substituting $\tilde{\xi}_2 = -\mathrm{Ad}_{T_2^{-1}T_1}\xi_1$ yields Eq. 15. □

It is worth noting that the equivalent gradient transport technique is not limited to cases where T_1 and T_2 are decision variables. It can be implemented as a plug-in module and applied when T_1 and T_2 act as intermediate transformations within a larger computation graph, as long as we project the obtained gradient $\xi_2 + \tilde{\xi}_2$ from $\mathfrak{se}(3)$ back to the ambient space using the inverse of Eq. 13. For example, in the application shown in Section V-E, T_2 is calculated by forward kinematics and this module can still be used.

F. Comparison with First-order Method in [14]

Our method is inspired by [14] and therefore shares some components, such as softmax smoothing and point sampling. However, both are redesigned in our work for robustness.

For softmax smoothing, their score value is defined as

$$u_j = \langle \mathbf{v}_{1,j}, \mathbf{y}\rangle, \quad \mathbf{y} = \begin{cases} \mathbf{x}_2 - \mathbf{x}_1, & \text{if no penetration} \\ \mathbf{x}_1 - \mathbf{x}_2, & \text{otherwise.} \end{cases} \tag{18}$$

This *direction-based smoothing* leverages the optimality conditions of the GJK/EPA algorithm but comes with several limitations. Most importantly, it only works reliably when both objects are strictly convex. Using their score vector, points $\mathbf{v}_{1,j}$ lying near a plane orthogonal to $\mathbf{y}$ become indistinguishable, and concave regions cannot be handled at all because GJK algorithm is not applicable. In addition, numerical issues arise when $\mathbf{x}_1$ is extremely close to $\mathbf{x}_2$, making $\mathbf{y}$ nearly zero and the direction ill-defined (see Appendix) . In contrast, our *distance-based smoothing* extends naturally to general objects and avoids the numerical instability caused by vanishing directions.

For the sampling strategy, [14] considers vertices within a fixed-level neighborhood (e.g., 3- or 5-ring) around the witness point, which is sensitive to mesh quality. In contrast, we address this issue by filtering candidates pre-sampled on object surfaces by their distance to the witness point.

Overall, our scheme (**Distance + Adaptive + EG**) substantially enhances robustness on general objects compared to the previous method [14] (**Direction + Neighbor**).

Method	DexGraspNet (Convex)			DexGraspNet (Concave)			Objaverse (Convex)			Objaverse (Concave)		
	D5↓	D9↓	Acc(%)↑	D5↓	D9↓	Acc(%)↑	D5↓	D9↓	Acc(%)↑	D5↓	D9↓	Acc(%)↑
Analytical	1.0e-4	2.0e-3	4.0	3.3e-5	1.2e-3	15.2	5.1e-5	1.4e-3	8.9	3.3e-5	8.9e-4	13.7
Finite Difference	2.7e-6	1.5e-4	40.6	3.9e-6	1.8e-4	35.9	2.5e-6	1.7e-4	41.3	8.0e-6	2.6e-4	27.9
RS-0 [14]	1.5e-6	1.3e-4	45.3	2.6e-6	1.2e-4	38.0	1.5e-6	1.8e-4	46.0	8.6e-6	2.9e-4	27.4
RS-1-Dir [14]	2.2e-6	4.0e-4	45.6	1.7e-5	1.7e-3	28.3	3.8e-7	3.2e-4	55.6	2.8e-5	1.8e-3	22.1
Ours	**4.3e-9**	**5.6e-7**	**91.1**	**6.5e-9**	**9.0e-6**	**80.3**	**5.4e-9**	**1.2e-6**	**89.4**	**1.0e-7**	**4.5e-5**	**61.8**

TABLE I: **Quantitative comparison.** Our method significantly outperforms all baselines across datasets and geometries.

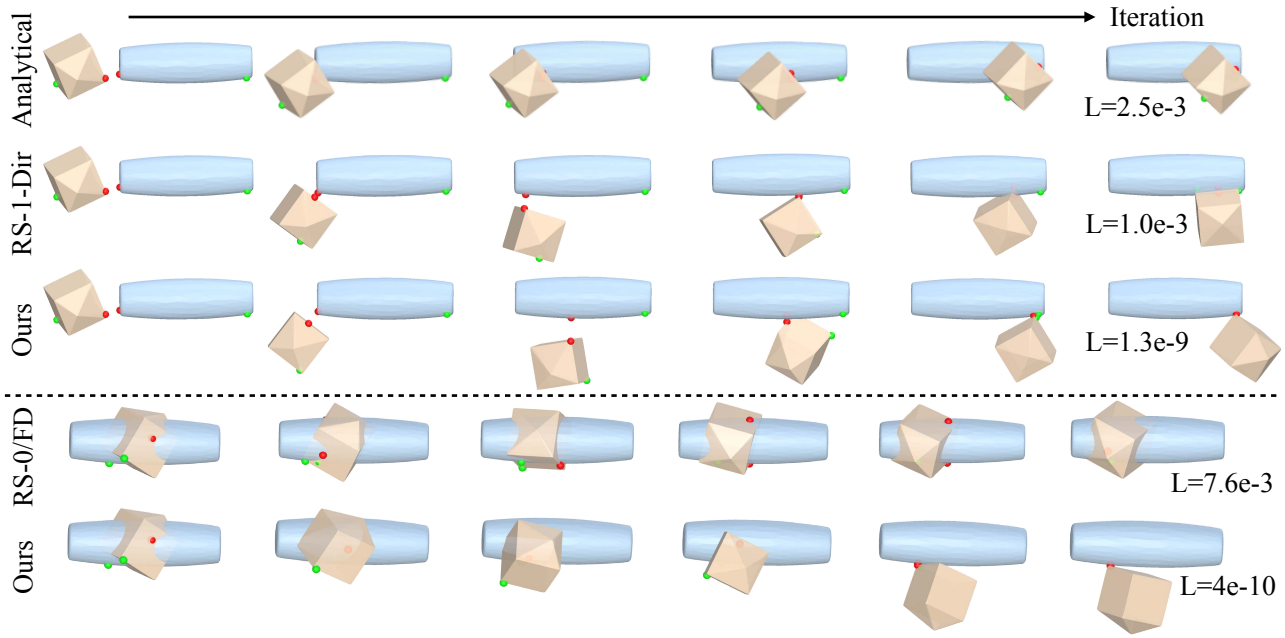

Fig. 5: **Qualitative comparison on convex objects.** Different baselines exhibit distinct failure patterns: *Analytical* often gets stuck due to zero derivatives at vertices (cols. 2, 5); *RS-1-Dir* struggles to disambiguate vertices lying near a plane (col. 6); *RS-0* and *Finite Difference (FD)* often fail to resolve initial penetrations. In contrast, *Ours* performs well in both scenarios.

V. Experiments

A. Setup and Evaluation Metrics

Our experimental setup differs from [14] in two key aspects. First, we evaluate on objects that are closer to real robotic applications, rather than rough convex shapes. Second, we use plain gradient descent without line search, to better highlight the quality of derivatives.

Objects. We randomly select assets from the 10k-object dataset used in Dexonomy [27], where 5k are from DexGraspNet [15] and 5k from Objaverse [16]. Each object is scaled so its bounding box diagonal lies in $[0.01, 0.2]$m.

Experimental Setup. We evaluate two settings: (1) using the convex hull of each object (denoted as **convex**), and (2) applying CoACD [20] for convex decomposition (denoted as **concave**). For each setting, we fix 100 random object pairs shared across all methods. From each pair, 1,024 target point pairs $(\mathbf{t}_{1,o}, \mathbf{t}_{2,o})$ are sampled from mesh vertices or surfaces, resulting in 100k tasks per method. In each task, Object 1 is fixed while Object 2 is randomly initialized around it.

Optimizer. We run gradient descent for 2k iterations. To avoid numerical instability, pose derivatives are normalized, since they can be extremely large at the beginning and tiny near convergence. The rotation step size s_r decays from $10 \rightarrow 1 \rightarrow 0.1$ at 200 and 1800 iterations, while the translation step size $s_t = s_r/100$.

Implementation. We use Coal [28] for narrow-phase collision detection, parallelized with OpenMP [29]. For concave objects, we add a broad-phase check using bounding spheres. Our proposed derivatives for witness points are implemented in PyTorch [30], supporting batched processing of multiple target points on a GPU.

Metrics. (1) **D5**: median loss at the final iteration, (2) **D9**: ninth decile of final loss, (3) **Acc (%)**: fraction of tasks with final loss $<$1e-6. For reference, since the loss sums three squared distances, a per-point displacement of 1 mm (1e-6 after squaring), yields a total loss of about 3e-6.

B. Comparison with Baselines

We compare against four baselines:

a) Analytical: computes vertex–face derivatives at witness points, representing a special case of brute-force vertex–face and edge–edge check. Since brute-force cannot find witness points during penetration, we still use GJK as the forward function. In practice, witness points rarely lie on edge–edge pairs, possibly due to forward-backward mismatch and numerical issues, so only vertex–face derivatives are considered.

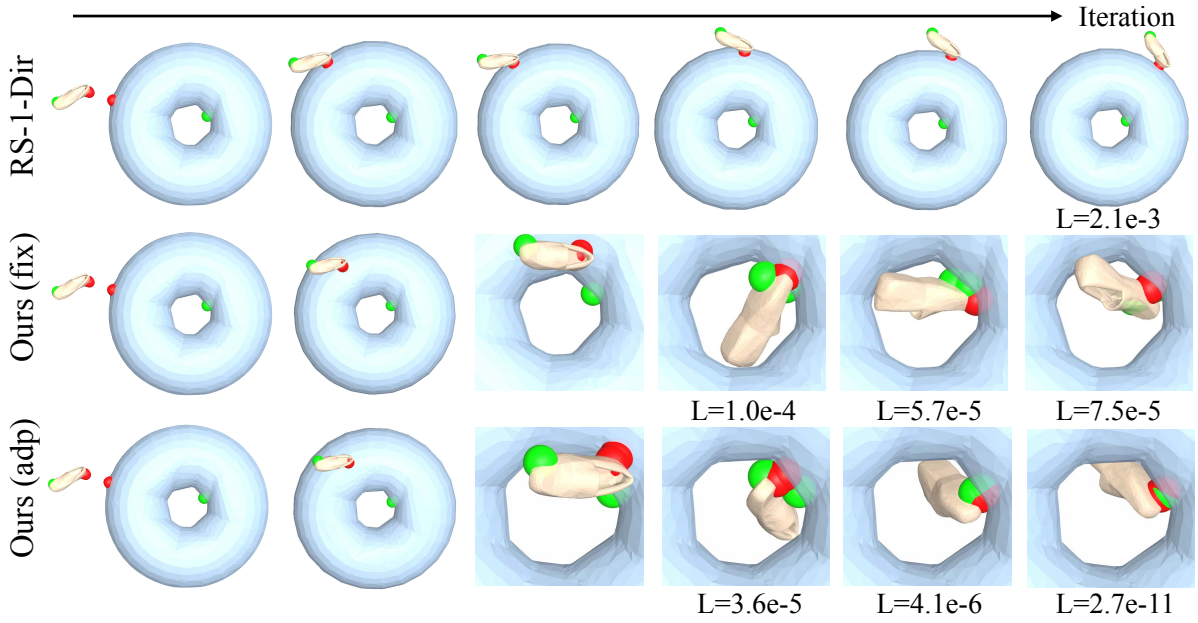

Fig. 6: **Qualitative comparison on concave objects.** *RS-1-Dir* fails dramatically when the target points lie on concave regions, regardless of the sampling strategy. Our proposed *RS-1-Dist* with adaptive sampling can converges well.

b) Finite Difference: perturbs each input dimension and computes central differences.

c) RS-0: 0th-order random smoothing [14], which can be viewed as an enhanced finite difference using Gaussian samples as perturbations. Due to its computational cost, we run RS-0 and Finite Difference with only 128 target points (1/8 of the other settings) to avoid excessive runtime.

d) RS-1-Dir: 1st-order direction-based random smoothing with neighbor-based sampling [14]. To make it parallel-friendly, neighboring vertices within the fifth level set are randomly subsampled to maintain a fixed number per batch.

As shown in Table I, our method clearly outperforms all baselines. We achieve a median error below 0.1 mm across all tasks and outperform the baselines by more than 40% in terms of mm-level accuracy.

Among baselines, **Analytical** performs the worst and often gets stuck on vertices, as shown in Figure 5. This occurs because witness points located on vertices have zero derivatives—small pose changes do not alter the witness. Even when the derivatives are nonzero, they depend on a single face, making the results highly sensitive to mesh quality.

RS-0 outperforms **Finite Difference** due to more diverse perturbations providing better derivative estimates. However, they often fail to resolve initial penetrations, as shown in Figure 5. Moreover, both methods are extremely time-consuming (Section V-D) and thus limiting practical use. We also find that these two methods are very sensitive to the optimization step size (See Appendix) .

RS-1-Dir performs better than RS-0 and FD on convex objects but worse on concave ones, consistent with our analysis that direction-based randomized smoothing is inherently limited to convex geometries. Moreover, we identify another major limitation of direction-based smoothing—its strong sensitivity to the contact margin (see Appendix) .

Acc@1e-6 (%) ↑		DexGraspNet		Objaverse	
		Convex	Concave	Convex	Concave
Dir	Neighbor	53.7	37.9	57.9	26.1
	Fixed	63.2	52.5	64.0	37.8
	Adaptive	62.2	55.8	61.4	41.9
Dist	Neighbor	85.6	70.2	88.5	53.1
	Fixed	89.9	75.1	91.0	55.8
	Adaptive	**91.2**	**80.6**	**89.5**	**62.4**
Dir	T_1, T_2	71.2	63.3	68.9	46.1
	T_2(w/o EG)	29.8	26.6	33.4	21.3
	T_2(w/ EG)	62.2	55.8	61.4	41.9
Dist	T_1, T_2	90.9	**80.6**	**89.5**	**62.6**
	T_2(w/o EG)	70.8	64.4	73.3	50.4
	T_2(w/ EG)	**91.2**	**80.6**	**89.5**	62.4

TABLE II: **Ablation study.** (1) *Smoothing:* our distance-based formulation consistently outperforms the direction-based baseline [14]. (2) *Sampling:* adaptive α improves accuracy, especially for concave objects. (3) *Equivalent gradient (EG):* when only T_2 is optimized, EG recovers performance comparable to jointly optimizing T_1 and T_2.

The results in Table I are reported under its best-performing configuration, i.e., with a contact margin of $\beta = 10^{-3}$.

C. Ablation Study

In this section, we evaluate three design choices:

- **Smoothing strategy**: direction-based smoothing [14] vs. distance-based smoothing (ours)
- **Sampling**: neighbor-based [14], fixed α, adaptive α
- **Equivalent gradient (EG)**: optimizing both T_1, T_2 vs. only T_2 with/without EG

The default setting is Distance + Adaptive + T_2(w/ EG). Since direction-based smoothing is highly sensitive to the contact margin, we report all results in this section as the average performance under two contact margin settings, $\beta = 0$ and $\beta = 10^{-3}$ (see Appendix for details) .

Time (µs)	DexGraspNet		Objaverse	
	Convex	Concave	Convex	Concave
Forward	8.3	23.4	8.7	29.2
Backward (RS-0 family)	58.6	488.3	59.3	1269.5
Backward (RS-1 family)	17.7	19.6	18.9	21.4

TABLE III: **Runtime analysis.** RS-1 family (Ours) is much more efficient than RS-0, especially for concave objects.

	Sampling			Derivative		EG
	Neighbor	Fixed	Adaptive	Dir	Dist	
Time (µs)	13.5	9.8	10.4	5.7	4.5	1.1

TABLE IV: **Runtime breakdown of backward components.** Neighbor-based sampling runs in C++ with OpenMP on CPU; all other components run in PyTorch on GPU.

Table II shows three key findings. First, our distance-based smoothing consistently outperforms the direction-based baseline across all settings, even when the baseline is enhanced with our adaptive sampling and equivalent gradient transport. As shown in Figure 6, direction-based smoothing often fails dramatically on challenging concave objects. A possible reason the baseline does not completely fail on concave objects is that some randomly sampled target points happen to lie on locally convex regions.

Second, neighbor-based sampling performs poorly, likely due to its sensitivity to mesh quality, making it unsuitable for in-the-wild objects. Both fixed and adaptive α perform well on convex shapes, while adaptive sampling is especially beneficial for concave ones.

Finally, equivalent gradient transport is essential when only T_2 is optimized. Without EG, performance degrades due to the mismatch between the current and target contact point on object 1, since moving an object itself typically requires smaller pose changes than moving another object. EG compensates for this effect and restores performance to the level of jointly optimizing T_1 and T_2.

D. Runtime Analysis

We evaluate five methods grouped into two families, reporting representative runtimes in Table III since intra-family variances are negligible. The **RS-0 family** (Finite Difference, RS-0) is computationally expensive as each backward pass requires numerous forward collision detections. This becomes prohibitive for complex concave objects, such as those in the Objaverse dataset. Conversely, the **RS-1 family** (Analytical, RS-1-Dir, and **Ours**) maintains consistent efficiency regardless of mesh complexity. All benchmarks were performed using an Nvidia 3090 GPU and Intel Xeon Platinum 8255C CPU. Table IV shows a detailed runtime breakdown of each backward component. Sampling is time-consuming as it computes distances from the witness point to the entire pre-sampled candidate set, with complexity linear in the number of candidates. However, this operation is easily parallelizable on modern GPUs, mitigating the overhead.

E. Preliminary Results on Dexterous Grasp Synthesis

To further demonstrate the applicability of our method to downstream robotic tasks, we conduct a preliminary study on dexterous grasp synthesis. A grasp generated by Dexonomy [27] is used as initialization, and target contact point pairs are manually annotated on the object and fingertips. The optimizable variables include the hand 6D root pose and joint angles, with the loss term defined in Eq. 5. Without any modification, our method allows gradients to propagate from the loss through the fingertip poses—computed via differentiable forward kinematics—back to the root pose and joint angles in an end-to-end manner.

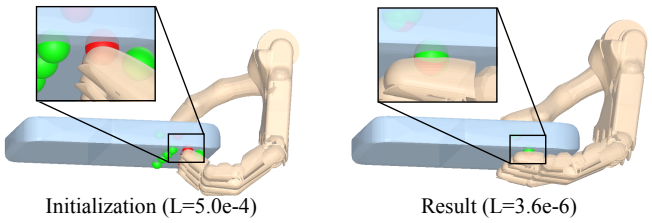

Fig. 7: **Preliminary grasp refinement with our method.** Starting from a grasp generated by Dexonomy [27] and manually annotated target contact points, our method back-propagates gradients from Eq. 5 to the hand root pose and joint angles, successfully bringing the targets into contact.

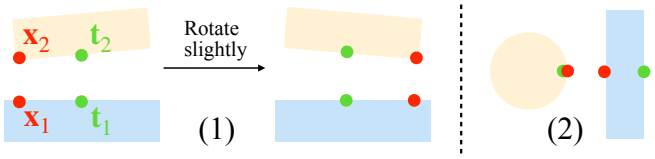

Fig. 8: **Failure cases.** (1) Discontinuity of point contact model for face-face contact. (2) A local minimum of the loss term that traps the optimization.

As shown in Figure 7, our method successfully refines the grasp to make all annotated target points in contact. The current formulation relies on manually annotated target points, which limits scalability. However, since our method is not restricted to a specific loss (particularly in the fixed sampling setting), it could be combined with differentiable grasp quality metrics [31], [32] to eliminate the need for manual annotation. We leave this integration for future work.

VI. Limitations and Future Works

Our method still faces several limitations. First, when both target points lie on planar faces, convergence to low loss is difficult because face–face contact is inherently discontinuous in the point–contact model (Figure 8, Left). Hydroelastic contact models [33], [34] used in Drake [35] provide a smoother alternative, though they currently do not extend to non-contact scenarios and incur higher computational cost. Second, optimization may become trapped in local minima when the target point lies on the opposite side of the object (Figure 8, Right). Finally, while our experiments demonstrate robustness on a toy problem, further validation is needed to assess applicability in gradient-based pipelines for tasks like dexterous grasping and manipulation.

VII. Conclusions

In this work, we presented a general approach for differentiating witness points from collision detection, extending beyond convex shapes to arbitrary meshes. Experiments on standard object datasets showed robustness across object scale, configuration, and mesh complexity. We further provided preliminary results on applying our method to a downstream task, dexterous grasp refinement. Future work will focus on integrating our method into planning and control frameworks to support downstream contact-rich tasks in robotic grasping and manipulation.

Appendix

We evaluate the following parameter variations to further validate our method's robustness against hyperparameters:

- **Contact Margin (β):** We modify Eq. 5 to

$$\mathcal{L} = \|\mathbf{x}_1 - \mathbf{x}_2 + \beta\mathbf{n}\|^2 + \|\mathbf{x}_1 - \mathbf{t}_1\|^2 + \|\mathbf{x}_2 - \mathbf{t}_2\|^2, \quad (19)$$

  where $\beta \in [0, 10^{-2}]$ is the contact margin and $\mathbf{n}$ is the contact normal. The direction-based baseline fails when β is small, , whereas our method remains highly stable.
- **Optimization Step Size (s_r, s_t):** Our method demonstrates significantly less sensitivity to step size variations compared to the baselines.

Due to space constraints, detailed experimental setups and results are provided in the appendix of the arXiv version.

References

[1] C. Ericson, *Real-time collision detection*. Crc Press, 2004.

[2] S. Andrews, K. Erleben, and Z. Ferguson, "Contact and friction simulation for computer graphics," in *ACM SIGGRAPH 2022 Courses*, ser. SIGGRAPH '22. New York, NY, USA: Association for Computing Machinery, 2022.

[3] Q. Le Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, "Contact models in robotics: a comparative analysis," *IEEE Transactions on Robotics*, 2024.

[4] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 2002.

[5] L. Montaut, Q. Le Lidec, V. Petrik, J. Sivic, and J. Carpentier, "Gjk++: Leveraging acceleration methods for faster collision detection," *IEEE Transactions on Robotics*, vol. 40, pp. 2564–2581, 2024.

[6] G. Van Den Bergen, "Proximity queries and penetration depth computation on 3d game objects," in *Game developers conference*, vol. 170, 2001, p. 209.

[7] G. Snethen, "Xenocollide: Complex collision made simple," in *Game programming Gems 7*. Course Technology, 2008, pp. 165–178.

[8] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[9] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, vol. 8, no. 84, p. eadc9244, 2023.

[10] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo," *arXiv preprint arXiv:2212.00541*, 2022.

[11] M. Li, Z. Ferguson, T. Schneider, T. R. 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, p. 49, 2020.

[12] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on robotics*, vol. 39, no. 6, pp. 4691–4711, 2023.

[13] W. Jin, "Complementarity-free multi-contact modeling and optimization for dexterous manipulation," *arXiv preprint arXiv:2408.07855*, 2024.

[14] L. Montaut, Q. L. Lidec, A. Bambade, V. Petrik, J. Sivic, and J. Carpentier, "Differentiable collision detection: a randomized smoothing approach," *arXiv preprint arXiv:2209.09012*, 2022.

[15] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang, "Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation," *arXiv preprint arXiv:2210.02697*, 2022.

[16] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 13 142–13 153.

[17] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[19] K. Mamou and F. Ghorbel, "A simple and efficient approach for 3d mesh approximate convex decomposition," in *2009 16th IEEE international conference on image processing (ICIP)*. IEEE, 2009, pp. 3501–3504.

[20] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[21] S. Gottschalk, "Separating axis theorem," 1996.

[22] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[23] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax–a differentiable physics engine for large scale rigid body simulation," *arXiv preprint arXiv:2106.13281*, 2021.

[24] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," https://github.com/nvidia/warp, March 2022, NVIDIA GPU Technology Conference (GTC).

[25] K. Tracy, T. A. Howell, and Z. Manchester, "Differentiable collision detection for a set of convex primitives," *arXiv preprint arXiv:2207.00669*, 2022.

[26] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[27] J. Chen, Y. Ke, L. Peng, and H. Wang, "Dexonomy: Synthesizing all dexterous grasp types in a grasp taxonomy," *arXiv preprint arXiv:2504.18829*, 2025.

[28] J. Pan, S. Chitta, D. Manocha, J. Mirabel, J. Carpentier, and L. Montaut, "Coal - An extension of the Flexible Collision Library," Feb. 2025. [Online]. Available: https://github.com/coal-library/coal

[29] L. Dagum and R. Menon, "Openmp: an industry standard api for shared-memory programming," *IEEE computational science and engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[31] J. Chen, Y. Chen, J. Zhang, and H. Wang, "Task-oriented dexterous hand pose synthesis using differentiable grasp wrench boundary estimator," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 5281–5288.

[32] J. Chen, Y. Ke, and H. Wang, "Bodex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 01–08.

[33] R. Elandt, E. Drumwright, M. Sherman, and A. Ruina, "A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 8238–8245.

[34] J. Masterjohn, D. Guoy, J. Shepherd, and A. Castro, "Velocity level approximation of pressure field contact patches," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 593–11 600, 2022.

[35] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu