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

Amortized NeuralSDF-Mesh Collision Detection for Robotic Contact Simulation

Jinhee Yun[1], Jeongmin Lee[1,2], Sunkyung Park[1], and Dongjun Lee[1]

***Abstract*— Collision detection is a fundamental problem in robotics, but handling collisions between non-convex objects remains challenging. A common approach for representing non-convex geometry is a signed distance function (SDF). Voxel-based SDF (VoxelSDF) enables fast distance queries but suffers from discretization artifacts and high memory costs. Neural implicit SDF (NeuralSDF) provides a continuous and memory-efficient representation with generalization, yet their slow query speed has limited their use in collision detection. To overcome these limitations, this paper proposes a novel amortized NeuralSDF–mesh collision detection framework. NeuralSDF–mesh collisions are formulated as a constrained optimization problem at the triangle level, and the Karush–Kuhn–Tucker conditions are derived to enable the amortization. A learning-based amortized optimization directly predicts collisions in a single forward pass, eliminating iterative optimization procedures. The amortized model adopts an auto-decoder architecture, extending the advantages of NeuralSDF in memory efficiency and category-level generalization to collision detection. Experiments demonstrate substantial speedups over baseline methods while maintaining comparable contact quality and reduced memory usage. The proposed approach also exhibits category-level generalization to unseen objects and can be applied to various robotic simulation scenarios.**

I. Introduction

Collision detection is a fundamental problem for physics-based simulation, manipulation, and planning in robotics. Handling collisions between non-convex objects is particularly challenging, as both the accuracy and efficiency of collision detection depend on shape representation. Collision between geometric primitives such as boxes, spheres, and capsules can be efficiently handled with analytic methods [1], however, these primitive representations cannot model complex non-convex shapes. Collision between convex meshes can be processed using the Gilbert-Johnson-Keerthi method [2] and the Expanding Polytope Algorithm [3], which are well established for convex shapes. However, extending these methods to non-convex shapes requires convex decomposition, often compromising geometric fidelity while increasing computational complexity.

A signed distance function (SDF) is a representation that encodes the shortest distance to an object's surface. Its implicit formulation can represent non-convex shapes, leading to extensive use in robotics. SDFs can be implemented in several forms, including analytic expressions, voxel-based discretizations, or neural network parameterizations. Analytic SDF [4] provides exact formulas for simple shapes. Voxel-based SDF (VoxelSDF) [5] discretizes objects into volumetric grids storing precomputed signed distances, enabling fast queries via lookup tables. However, it suffers from discretization artifacts, resolution dependence, and high memory cost due to per-object storage, which limits scalability in complex or large-scale scenarios. More recently, neural implicit SDF (NeuralSDF) [6], [7] has been introduced, representing signed distance fields using neural networks. Compared to VoxelSDF, NeuralSDF provides a continuous representation with higher geometric fidelity and improved memory efficiency. Moreover, it supports category-level generalization [6], [8], allowing adaptation to unseen objects within the same category through shape code optimization without additional training.

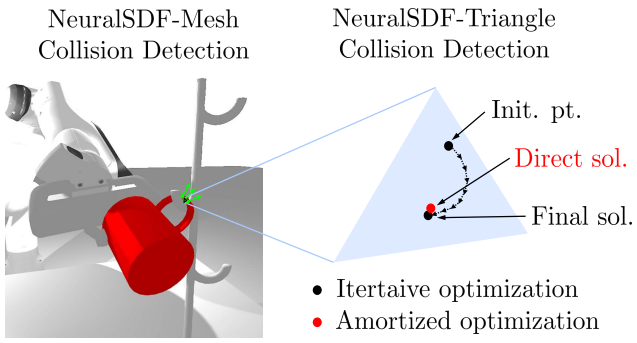

Fig. 1: Illustration of proposed amortized NeuralSDF-mesh collision detection framework. Multiple collisions between NeuralSDF and mesh are decomposed into NeuralSDF-triangle collision problems. In contrast to typical iterative optimizations, our method directly identifies a contact.

However, applying NeuralSDF in downstream robotics applications exposes a critical limitation, most notably in collision detection of robotic contact simulations. Each distance query of NeuralSDF requires a forward pass of neural network, which is significantly slower than analytic computation available in analytic SDF or the direct table lookups available in VoxelSDF. Since collision detection algorithms using SDFs are typically solved with iterative optimization, this inefficiency is further exacerbated as each optimization demands tens of NeuralSDF queries [9]. Widely used physics simulators such as MuJoCo [10] and Isaac

This research was supported by Korea Planning & Evaluation Institute of Industrial Technology (KEIT) grant funded by Ministry of Trade, Industry and Resources (MOTIR) (RS-2024-00419641, RS2025-25455303).

[1]The authors are with the Department of Mechanical Engineering, Institute of Advanced Machines and Design (IAMD) and Institute of Engineering Research (IOER), Seoul National University, Seoul, Republic of Korea.

[2]The author is with Holiday Robotics, Seoul, Republic of Korea.

Corresponding author: Dongjun Lee. E-mail: yjhs0932@snu.ac.kr, jeongmin.lee@holiday-robotics.com, sunk1136@snu.ac.kr, djlee@snu.ac.kr.

Sim with PhysX [11] also rely on iterative optimization to solve SDF–SDF and SDF–mesh collisions, respectively. However, MuJoCo primarily supports analytic SDFs, while Isaac Sim supports only VoxelSDF representations for SDF-based collision detection. Prior work [12] explored real-time collision handling between general SDFs, but did not support NeuralSDF, explicitly noting it as a limitation. Other works [13], [14] solve SDF-mesh collision problems using three iterative numerical algorithms, but they also consider only VoxelSDF representations. To the best of our knowledge, no prior work has addressed the inefficiency of NeuralSDF queries for SDF-based collision detection. As a result, NeuralSDF remains less practical for time-sensitive robotics applications.

In this paper, we address this issue by introducing the amortized NeuralSDF–mesh collision detection framework. Our method directly predicts collisions between NeuralSDFs and meshes in a single forward pass, eliminating iterative numerical procedures. Specifically, we formulate collisions between NeuralSDFs and triangles as a constrained optimization problem and derive the optimality conditions from the Karush–Kuhn–Tucker (KKT) conditions [15]. Leveraging this formulation, we design a learning-based amortized model with a loss that enforces the derived conditions, enabling successful unsupervised learning. We adopt an auto–decoder architecture to extend the category-level generalization property of NeuralSDF to collision detection. The network handles the collisions of unseen objects within the same category without additional training. Additionally, we structurally incorporate permutation equivariance through shared MLPs [16], [17] for efficient learning. The main contributions of this paper are:

- We present a novel amortized NeuralSDF–mesh collision detection framework with learning-based approach.
- We derive optimality conditions of NeuralSDF-mesh collision and leverage them to train amortized model, achieving substantial speed improvements.
- We extend memory efficiency and category-level generalization of NeuralSDF to collision detection.

The rest of the paper is organized as follows. Sec. II formulates NeuralSDF-mesh collision detection problem and derives the KKT optimality conditions. Sec. III presents our learning-based amortized collision detection framework, including network architecture, training strategy with loss functions and implementation details. Sec. IV reports experimental results, evaluating collision detection performance and effectiveness of amortized model in terms of generalization and structure.

II. NeuralSDF-Mesh Collision Detection

A. Problem Formulation

We consider the collision detection problem between an object represented as a NeuralSDF and another represented as a triangle mesh. NeuralSDF ($f_{\text{SDF}} : \mathbb{R}^3 \to \mathbb{R}$) is a continuous function which maps a 3D point to its signed distance to the object surface, and the triangle mesh is defined as a set of triangular faces. As illustrated in Fig. 1, multiple collisions can occur between NeuralSDF and mesh, and we decompose these collisions into elementary NeuralSDF–triangle collision problems, where each triangular face of mesh object is considered independently. For each triangular face, the objective is to identify a point that minimizes the signed distance of NeuralSDF inside the triangle, which corresponds to the closest point between the NeuralSDF object and the triangular face. This can be formulated as the following constrained optimization problem:

$$\begin{aligned} &\min_{\alpha,\beta,\gamma} f_{\text{SDF}}\left(\alpha\mathbf{v}_1 + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3\right) \\ &\text{s.t.} \quad \alpha + \beta + \gamma = 1, \quad \alpha, \beta, \gamma \geq 0 \end{aligned} \tag{1}$$

where $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$ are the vertex coordinates of the triangle and $\alpha, \beta, \gamma \in \mathbb{R}$ are the barycentric coordinates of the closest point to be optimized. The constraints ensure that the optimized solution lies within the triangle.

Our formulation adopts the same optimization structure as previous studies [13], [14], which have formulated the SDF–mesh collision detection problem. However, we observe that their iterative numerical optimization exacerbates the query inefficiency of NeuralSDF, thereby motivating the need for amortized optimization [18]. As illustrated on the right side of Fig. 1, the amortized approach directly approximates the solution, replacing typical iterative procedures and reducing computational cost.

B. KKT Optimality Conditions

From the problem of NeuralSDF-triangle collisions (1), we derive the optimality conditions using the Karush-Kuhn-Tucker (KKT) conditions [15] of the optimization problem. The corresponding Lagrangian function is defined as:

$$\begin{aligned} &\mathcal{L}(\alpha, \beta, \gamma, \lambda, \mu_1, \mu_2, \mu_3) \\ &= f_{\text{SDF}}(\mathbf{p}) + \lambda(\alpha + \beta + \gamma - 1) - \mu_1\alpha - \mu_2\beta - \mu_3\gamma \end{aligned} \tag{2}$$

where $\lambda \in \mathbb{R}$ is the Lagrangian multiplier for the equality constraint, $\mu_1, \mu_2, \mu_3 \in \mathbb{R}$ are the Lagrangian multipliers for the inequality constraints, and $\mathbf{p} \in \mathbb{R}^3$ denotes the closest point defined as $\mathbf{p} = \alpha\mathbf{v}_1 + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3$ with barycentric coordinates α, β, γ. The KKT conditions, including stationarity, primal-dual feasibility, and complementary slackness, can be expressed as follows:

$$\begin{aligned} \mathbf{v}_i^T \nabla f_{\text{SDF}}(\mathbf{p}) + \lambda - \mu_i = 0, \quad i \in \{1, 2, 3\} \\ 0 \leq \alpha \perp \mu_1 \geq 0 \\ 0 \leq \beta \perp \mu_2 \geq 0 \\ 0 \leq \gamma \perp \mu_3 \geq 0. \end{aligned} \tag{3}$$

Considering all possible cases of α, β and γ within the constraints, the following inequality equation is satisfied:

$$(\mathbf{p} - \mathbf{v}_i)^T \nabla f_{\text{SDF}}(\mathbf{p}) \geq 0, \quad i \in \{1, 2, 3\}. \tag{4}$$

The equality in (4) holds for a vertex $\mathbf{v}_i$ when the corresponding inequality constraint is inactive. Since at least one

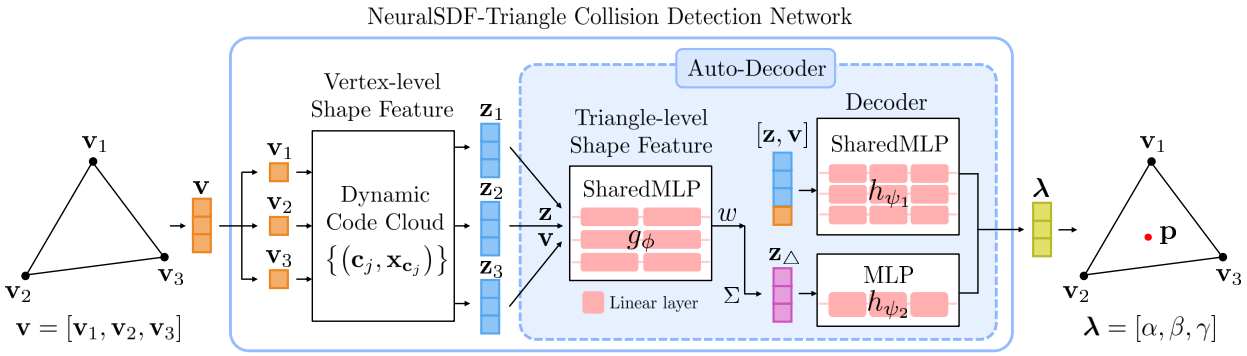

Fig. 2: Network architecture of the proposed amortized NeuralSDF-Traingle collision detection framework. Based on an auto-decoder architecture, the network predicts the contact point $\mathbf{p}$ as barycentric coordinates $\boldsymbol{\lambda} = [\alpha, \beta, \gamma]$ on the given triangle vertices $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$, directly addressing NeuralSDF–triangle collisions.

of the three inequality constraints must be inactive, we derive the optimality condition as:

$$\begin{aligned} (\nabla f_{\text{SDF}}(\mathbf{p}))^T(\mathbf{p} - \mathbf{v}_{\min}) = 0, \\ \text{where } \mathbf{v}_{\min} = \underset{\mathbf{v}\in\{\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3\}}{\arg\min} \nabla f_{\text{SDF}}(\mathbf{p})^T\mathbf{v} \end{aligned} \tag{5}$$

where the $\mathbf{p}$ is the optimal solution, serving as the contact point between the NeuralSDF and the triangle. The $\mathbf{v}_{\min}$ denotes the vertex with the minimum projection onto the gradient of SDF at $\mathbf{p}$, corresponding to the farthest vertex in the negative gradient direction. The derived optimality condition characterizes the solution of the optimization problem of NeuralSDF-triangle collisions and provides a way to adopt an amortized framework.

III. Amortized Collision Detection Framework

We propose a learning-based framework that amortizes the solution of the constrained optimization problem (1) in NeuralSDF–triangle collision detection.

A. Network Architecture

Most NeuralSDF models [6], [8] adopts an auto-decoder architecture that takes a point coordinate and a corresponding latent shape feature vector as inputs to enable category-level generalization. We extend this architecture to a collision detection network that takes the vertex coordinates of the triangle $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$ and corresponding shape feature vectors as inputs and produces the barycentric coordinates $\alpha, \beta, \gamma \in \mathbb{R}$ of the contact point $\mathbf{p} \in \mathbb{R}^3$ as outputs. The network architecture is illustrated in Fig. 2.

For the shape feature vectors, we consider both vertex-level and triangle-level representations, denoted as $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \in \mathbb{R}^{32}$ and $\mathbf{z}_\triangle \in \mathbb{R}^{32}$. We adopt the dynamic code cloud introduced in [8], which adaptively captures fine geometric details, although it can generally be replaced by other design choices. The vertex-level shape feature vectors are defined for each triangle vertex and computed by distance-based interpolation [8] of shape codes as:

$$\mathbf{z}_i(\mathbf{v}_i) = \frac{\sum_{j=1}^{N_{\text{code}}} \|\mathbf{v}_i - \mathbf{x}_{\mathbf{c}_j}\|_2^{-3}\mathbf{c}_j}{\sum_{j=1}^{N_{\text{code}}} \|\mathbf{v}_i - \mathbf{x}_{\mathbf{c}_j}\|_2^{-3}} \tag{6}$$

where $\mathbf{c}_j \in \mathbb{R}^{32}$ and $\mathbf{x}_{\mathbf{c}_j} \in \mathbb{R}^3$ are a shape code and its position of the dynamic code cloud $\{(\mathbf{c}_j, \mathbf{x}_{\mathbf{c}_j})\}_{j=1}^{N_{\text{code}}}$ with $N_{\text{code}} = 1376$.

The triangle-level shape feature vector is designed to capture shape variations within the triangle. We define it as a weighted sum of the vertex-level shape feature vectors $\mathbf{z}_i$ with learnable weights w_i as:

$$\begin{aligned} \mathbf{z}_\triangle(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) &= \sum_{i=1}^{3} w_i \mathbf{z}_i(\mathbf{v}_i), \\ w_i &= g_\phi\big([\|\mathbf{v}_i - \bar{\mathbf{v}}\|_2/p,\ \{\cos(\mathbf{z}_j - \mathbf{z}_i)\}_{j\neq i}]\big) \end{aligned} \tag{7}$$

where $\bar{\mathbf{v}}$ and p denote the centroid and perimeter of the triangle. The network g_ϕ predicts weights w_i using as input the normalized centroid distance of each vertex $\mathbf{v}_i$ and cosine similarities between vertex-level shape feature vectors $\mathbf{z}_i$, enabling the weights to learn local geometric and shape variations. The network $g_\phi : \mathbb{R}^3 \to \mathbb{R}$ is implemented as a shared multilayer perceptron (shared MLP) [16], [17], meaning that the same parameterized network is applied to each input. The MLP is designed with (3–16–1) dimensions with a ReLU activation, followed by a softmax function to normalize the weights. The shared MLP structure ensures permutation equivariance with respect to input ordering. Consequently, the weighted sum of each weights and vertex-level shape feature vectors in (7) is permutation invariant, meaning that reordering the vertices $\mathbf{v}_i$ does not change the resulting triangle-level shape feature vector $\mathbf{z}_\triangle$. This invariance improves both training stability and efficiency.

The decoder predicts the contact point within the triangle that minimizes the signed distance, using both the vertex-level shape feature vectors $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$ and the triangle-level shape feature vector $\mathbf{z}_\triangle$. Two decoder functions, h_{ψ_1} and h_{ψ_2}, are designed to interpret the two types of embeddings accordingly and are defined as:

$$\begin{aligned} \mathbf{y}_1 &= [h_{\psi_1}([\mathbf{z}_1; \mathbf{v}_1]), h_{\psi_1}([\mathbf{z}_2; \mathbf{v}_2]), h_{\psi_1}([\mathbf{z}_3; \mathbf{v}_3])] \\ \mathbf{y}_2 &= h_{\psi_2}(\mathbf{z}_\triangle)\mathbf{1}_3 \\ \boldsymbol{\lambda} &= [\alpha, \beta, \gamma] = \sigma(\mathbf{y}_1 + \mathbf{y}_2) \end{aligned} \tag{8}$$

where $\mathbf{y}_1, \mathbf{y}_2, \boldsymbol{\lambda} \in \mathbb{R}^3$, $[\mathbf{z}_i; \mathbf{v}_i]$ denotes the concatenation of vertex-level shape feature vector $\mathbf{z}_i$ and vertex coordinate $\mathbf{v}_i$, and σ is the softmax function. The network $h_{\psi_1} : \mathbb{R}^{35} \to \mathbb{R}$ is implemented as a shared MLP with (35-32-32-1) dimensions, applied independently to each input to ensure permutation equivariance. The network $h_{\psi_2} : \mathbb{R}^{32} \to \mathbb{R}$ is implemented as an MLP with (32-32-32-1) dimensions. The output $\boldsymbol{\lambda} = [\alpha, \beta, \gamma]$ represents the barycentric coordinates of the contact point within the triangle, satisfying $\alpha+\beta+\gamma = 1$ through the softmax function. Since the vertex-level shape feature vectors $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$ are permutation equivariant and the triangle-level shape feature vector $\mathbf{z}_\triangle$ is permutation invariant with respect to the triangle vertices, the decoder yields permutation equivariant output $\boldsymbol{\lambda}$. As a result, the predicted contact point $\mathbf{p}$, defined as $\mathbf{p} = \alpha\mathbf{v}_1 + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3$, is invariant to vertex ordering, ensuring robustness of the network.

The auto-decoder architecture supports category-level generalization, building upon the design of NeuralSDF. It can handle unseen objects within the same category by optimizing only shape codes from the dynamic code cloud. Since the network learns the relations among shape features and among triangle vertices, the trained auto-decoder can be directly applied to unseen objects without additional training.

B. Objective Function

We incorporate the optimality condition in (5) into our amortized framework by formulating it as a loss function:

$$\mathcal{L}_{\text{op}} = \left(\nabla f_{\text{SDF}}(\mathbf{p})\right)^T (\mathbf{p} - \mathbf{v}_{\text{min}}) \tag{9}$$

where $\mathbf{p}$ denotes the predicted contact point, computed from the triangle vertices $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ and the network outputs $\boldsymbol{\lambda} = [\alpha, \beta, \gamma]$ in (8). This optimality loss $\mathcal{L}_{\text{op}}$ enforces the predicted point to satisfy the KKT optimality condition of constrained optimization problem without additional supervision, thereby enabling unsupervised learning. The minimum signed distance loss $\mathcal{L}_{\text{min}}$ is formulated to encourage the predicted point to attain the minimum signed distance as

$$\mathcal{L}_{\text{min}} = \max(f_{\text{SDF}}(\mathbf{p}),\ 0). \tag{10}$$

The network is jointly trained with two loss functions in an unsupervised manner, and the total loss is defined as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{op}} + \lambda_{\text{min}}\mathcal{L}_{\text{min}}. \tag{11}$$

where $\lambda_{\text{min}} = 1$.

C. Training Details

The training dataset consists of sets of triangles surrounding the object that may collide with it. The dataset is collected for each object. To apply the same training procedure across various objects, each object is normalized into the unit cube $[-0.5, 0.5]^3$ and triangles are sampled around the object under the following conditions:

- The triangle edge lengths lie within $[0.03, 0.08]$.
- The triangle area lies within $[0.0002, 0.002]$.
- The signed distance of the triangle centroid lies within $[0, 0.1]$.
- The minimum signed distance from the triangle to the object is positive.

All numerical values are defined with respect to the normalized object scale and can therefore be interpreted as percentages of the object size.

For training stability, the triangle size is restricted and the ranges are chosen based on the default edge lengths commonly used in mesh generation software such as MeshLab [19], and Blender [20]. The signed distances used in the conditions are computed using each NeuralSDF model and the point which attains the minimum signed distance is obtained by solving an optimization problem using the Frank-Wolfe algorithm with 100 iterations. Under these conditions, each dataset is constructed using 100k triangles.

The network is implemented in PyTorch and trained using the Adam optimizer with a learning rate of 1×10^{-4}, exponential decay rates of 0.9 and 0.999, and a batch size of 5000. Training is performed for 3000 epochs, taking approximately 3.0 hours on an Intel Core i7-12700F CPU and an NVIDIA RTX 4060 GPU.

IV. Evaluation

We evaluate the proposed amortized NeuralSDF-mesh collision detection framework in terms of detection performance and the effectiveness of the amortized model.

A. Collision Detection Performance

We compare our amortized NeuralSDF-mesh collision detection framework with existing SDF-mesh collision detection approaches. As baselines, we consider the method that solve the SDF-mesh contact problem via iterative numerical optimization using the Frank-Wolfe (FW) algorithm, which is widely used in IsaacSim and prior research [11], [13], [14]. We implement two FW-based methods with different SDF representations, NeuralSDF and VoxelSDF, denoted as NeuralSDF+FW and VoxelSDF+FW. Our proposed method is denoted as NeuralSDF+Ours.

The collision detection performance evaluation is conducted at three levels: triangle-level, object-level, and simulation-level. We focus our analysis on the advantages of NeuralSDF over VoxelSDF representations and the benefits of amortized optimization compared to iterative optimization.

As contact objects, we select mugs and bottles, commonly used in manipulation tasks, and gears and bolts, commonly used in assembly tasks. While mugs and bottles represent objects with typical geometric complexity, gears and bolts introduce highly complex geometries that create contact-rich scenarios, enabling a robustness evaluation of the proposed method.

1) Triangle-level Evaluation: We first evaluate the performance on a triangle dataset of 10k samples generated using the same sampling strategy as the training dataset. For each triangle, we assess whether the detected contact point $\mathbf{p}$ satisfies the optimality of the NeuralSDF–triangle collision problem, using an optimality metric defined as:

$$\text{Optimality} = \left(\nabla f_{\text{SDF}}(\mathbf{p})\right)^T (\mathbf{p} - \mathbf{v}_{\text{min}})\,/p \tag{12}$$

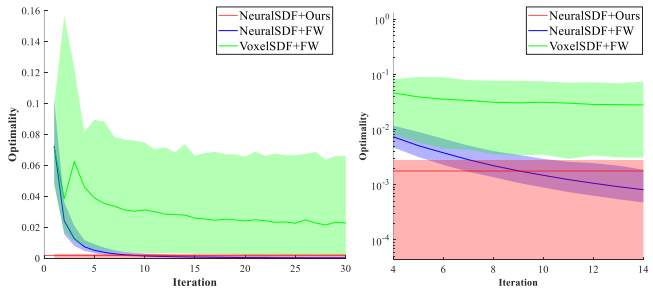

Fig. 3: Optimality comparison of NeuralSDF+Ours, NeuralSDF+FW, and VoxelSDF+FW at the triangle level. Results are shown in linear scale (left) and logarithmic scale (right).

Object	Optimality	Time [ms]		
		NeuralSDF + Ours	NeuralSDF + FW (Iter)	VoxelSDF + FW (Iter)
Mug	7.843e−04	0.038	64.5 (11.2)	∞ (∞)
Bottle	1.762e−03	0.038	51.9 (9.05)	∞ (∞)
Gear	4.641e−03	0.038	29.8 (5.16)	∞ (∞)
Bolt	4.252e−03	0.038	34.2 (5.95)	∞ (∞)

TABLE I: Computation time comparison to achieve the same level of optimality for NeuralSDF+Ours, NeuralSDF+FW, and VoxelSDF+FW.

where $\mathbf{v}_{\text{min}}$ and p are defined in (5).

The comparison results are shown in Fig. 3. The y-axis shows the optimality metric defined per triangle. The median is plotted as a line, with the first and third quartiles shown as a shaded region. NeuralSDF and amortized model are trained per object, and VoxelSDF is computed at a resolution of 1024. The results correspond to the bottle object in Table I.

Fig. 3 shows that NeuralSDF+Ours achieves a low optimality value in a single forward pass, while NeuralSDF+FW converges gradually and reaches a similar level around iteration 9. In contrast, VoxleSDF+FW does not converge to same level of optimality. Similar trends are observed for the other objects, and the numerical results are summarized in Table I.

To compare computational efficiency, we measure the time required to achieve the same level of optimality for three methods. We report the optimality value, computation time, and the number of iterations required for the FW-based method. NeuralSDF+Ours consistently achieves convergence in approximately 0.038 ms per triangle, while NeuralSDF+FW requires 5-11 iterations and 30-65 ms of computation time. VoxelSDF+FW fails to converge to the same optimality. Since NeuralSDF represents object geometry more accurately than VoxelSDF, NeuralSDF-based method can detect solutions with low optimality values. Furthermore, the amortized approach does not require iterative procedures, providing advantages in terms of computational efficiency.

2) Object-level Evaluation: We evaluate pairwise object collisions in terms of contact quality, computation time, and memory usage within a single simulation step. The collision detection procedure in a single step contact simulation consists of broad-phase and narrow-phase collision detection. In the broad-phase, Axis-Aligned Bounding Box overlap test is used to identify candidate triangles that may potentially collide. In the narrow-phase, three methods are applied to detect contacts on the candidate triangles, followed by contact feature extraction. In the contact feature extraction, SDF values and its gradients are queried, as contact features are defined as:

$$\begin{aligned} \mathbf{p} &= \alpha \mathbf{v}_1 + \beta \mathbf{v}_2 + \gamma \mathbf{v}_3 \\ d &= f_{\text{SDF}}(\mathbf{p}) \\ \mathbf{n} &= \nabla f_{\text{SDF}}(\mathbf{p}). \end{aligned} \tag{13}$$

A point is classified as a valid contact if $d \leq \varepsilon$, where ε denoted the contact margin.

The evaluation results are in Fig. 4 and Table II. Following four scenarios are evaluated: a mug hanging on a rack (Mug–Rack), a bottle placed on a holder (Bottle–Holder), two spur gears in mesh (Gear–Gear), and a bolt–nut assembly (Bolt–Nut). NeuralSDF and amortized model are trained per object, and VoxelSDF is computed with a truncation distance of 0.1 for normalized objects and a resolution of 1024, which corresponds to the maximum resolution supported by IsaacSim. The iteration limits for NeuralSDF+FW are set to the value required to achieve same level of optimality as reported in Table I, while the limit for VoxelSDF+FW is set to 50. The contact margin is set to $\varepsilon = 1 \times 10^{-3}$. The computation time and memory usage are measured for the entire collision detection procedures in CPU, and the number of candidate triangles and total mesh triangles are also reported.

In Fig. 4, NeuralSDF+Ours and NeuralSDF+FW produce consistent contact points, whereas VoxelSDF+FW often fails to capture contacts, either missing them or producing physically inconsistent contact normals. In Table II, NeuralSDF+Ours and NeuralSDF+FW achieve reliable collision detection, producing a similar number of contacts. The small differences may arise from approximation errors in the learned NeuralSDF for complex geometries and the filtering of contacts using a small contact margin. Furthermore, even when both methods detects an optimal solution, the location of contact points may be slightly different. While NeuralSDF+Ours achieves comparable collision detection quality, it is 2–12 times faster than NeuralSDF+FW. In contrast, VoxelSDF is faster but fails to capture collisions due to a small contact margin. The total memory usage of NeuralSDF+Ours comes from 60.7 MB for NeuralSDF model and 0.1 MB for our amortized collision detection model. This demonstrates that our framework is highly memory-efficient. In contrast, VoxelSDF+FW requires memory on the order of GB and particularly fails to produce plausible contacts for thin and complex-shaped objects.

3) Simulation-level Evaluation: We demonstrate the applicability of our method in robotic contact simulation level. Extending the single-step contact scenarios presented in Sec. IV-A.2, we implement full simulation experiments for four scenarios (Mug-Rack, Bottle-Holder, Gear-Gear, and

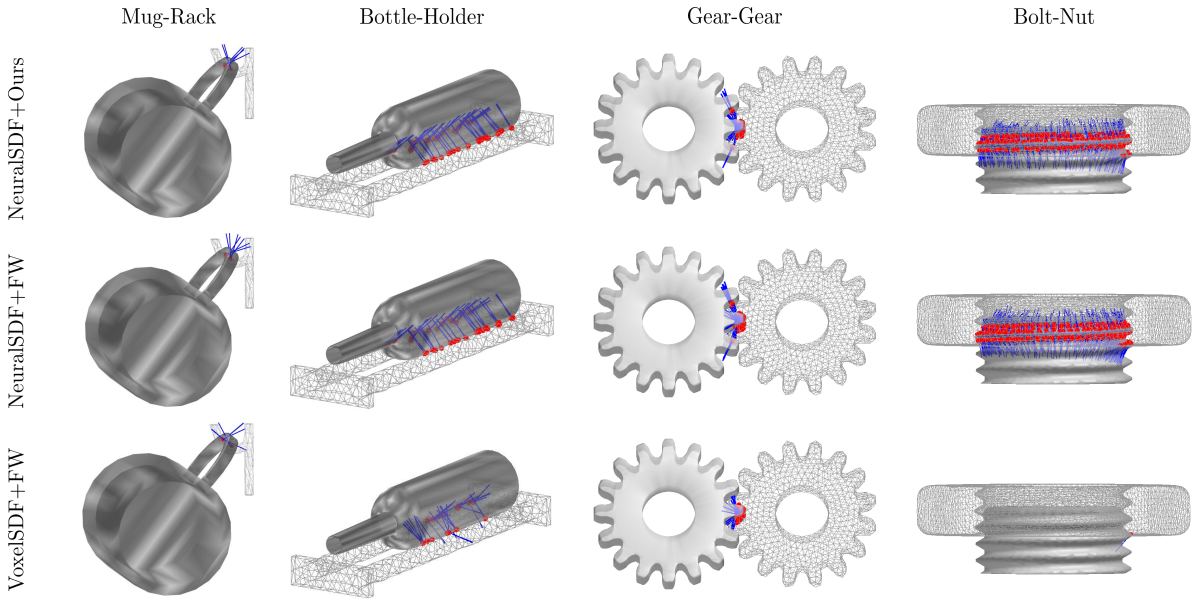

Fig. 4: Contact feature visualization in pairwise object collisions for NeuralSDF+Ours, NeuralSDF+FW, and VoxelSDF+FW.

Scenario	Method	Contacts	Time [s] ↓	Memory [MB] ↓	Iteration Limit	Resolution	Triangles (Cand. / Total)
Mug–Rack	NeuralSDF + Ours	6	0.150	60.8	-	-	18 / 228
	NeuralSDF + FW	9	1.778	60.7	12	-	
	VoxelSDF + FW	9	0.013	18417	50	1024	
Bottle–Holder	NeuralSDF + Ours	115	0.926	60.8	-	-	132 / 924
	NeuralSDF + FW	116	7.619	60.7	10	-	
	VoxelSDF + FW	49	0.073	3715	50	1024	
Gear–Gear	NeuralSDF + Ours	66	4.940	60.8	-	-	93 / 2800
	NeuralSDF + FW	70	8.785	60.7	6	-	
	VoxelSDF + FW	39	4.327	9033	50	1024	
Bolt–Nut	NeuralSDF + Ours	1010	14.245	60.8	-	-	1714 / 9548
	NeuralSDF + FW	1111	61.656	60.7	6	-	
	VoxelSDF + FW	1	3.764	15248	50	1024	

TABLE II: Collision detection performance comparison for NeuralSDF+Ours, NeuralSDF+FW, and VoxelSDF+Fw. The number of detected contact points and triangles, computation time and memory usage are reported. Iteration limits are set for the FW-based algorithms, and resolution is specified for the VoxelSDF representation.

Bolt-Nut) as shown in Fig. 5. The Franka Emika Panda robot with a parallel gripper is used in the first two scenarios. For simulation, we employ the cascaded Newton-based augmented Lagrangian (CANAL) [21] method as the contact solver, which can manage multi-contact accurately and robustly. All simulations are implemented in C++, with the Eigen and OpenGL libraries. Video results are provided in the accompanying video.

In the Mug-Rack scenario, the gripper grasps a mug and hangs it onto the rack. Contact are detected stably on NeuralSDF-based methods, while VoxelSDF-based method detects contacts inconsistently and sometimes results in penetration. In the Bottle-Holder scenario, NeuralSDF-based methods enables the gripper to stably grasp a bottle and place it on the holder without penetration. In contrast, with VoxelSDF+FW, the gripper fails to generate proper contacts during grasping, making it impossible to grasp and lift the bottle. Even when the bottle is forcibly dropped onto the holder, no contacts are detected and the bottle penetrates and falls through the holder. This failure is consistent with Table II, where VoxelSDF+FW produces no valid contact points with holders. The slender geometry of bottles is difficult to capture at voxel resolution, which explains the limited performance.

In the Gear-Gear scenario, one gear is forced to rotate and the meshed gear rotates accordingly. With NeuralSDF-based methods, the two gears rotate smoothly with stable collision detection. Hoewever, With VoxelSDF+FW, although the gears rotate, intermittent sticking behavior is observed due to inconsistent contact. In the Bolt-Nut scenario, a nut is threaded onto a bolt and rotated. With NeuralSDF-based methods, numerous contact points are detected in each simulation step, enabling stable rotation of the nut. In contrast, with VoxelSDF+FW, it fails to detect any contacts,

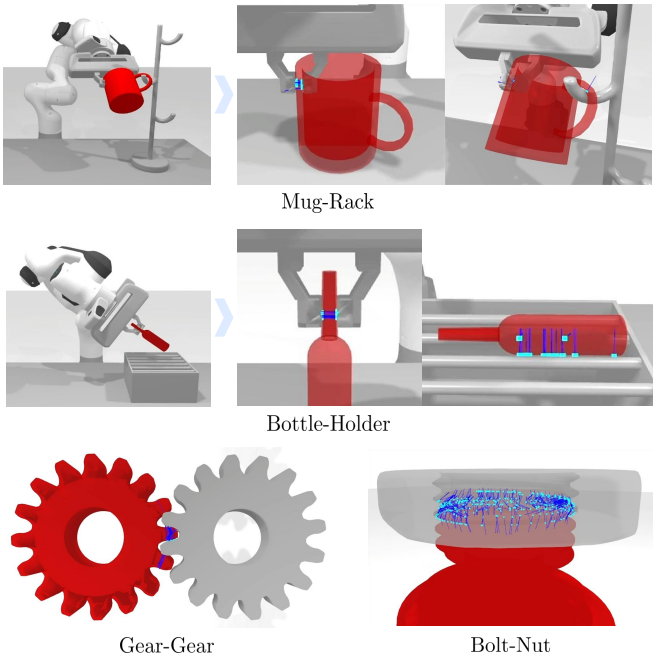

Fig. 5: Snapshots of robotic contact simulations. Mug–Rack (top), Bottle–Holder (middle), Gear–Gear (bottom left), and Bolt–Nut (bottom right).

consistent with Table II.

Across all scenarios, NeuralSDF enables reliable contact simulation due to its powerful geometric representation, and NeuralSDF+Ours is significantly faster and more computationally efficient than NeuralSDF+FW.

B. Category-level Generalization

We evaluate the category-level generalization ability of our amortized model on unseen objects. Following the property of auto-decoder architecture, unseen objects from the same category can be handled by optimizing only shape codes from the dynamic code cloud, while keeping the network parameters fixed. To validate this, we train NeuralSDF and our collision detection network on ten mugs and ten bottles from the ShapeNet dataset [22], and then test them on another ten unseen mugs and ten unseen bottles from the same category. We restrict our experiments to mugs and bottles, as ShapeNet provides mesh datasets for these categories.

Fig. 6 visualizes the contact points in single-step contact scenarios of Mug–Rack and Bottle–Holder for unseen objects. Our method consistently predicts plausible contact points and contact normals. Table III summarizes the optimality metric (12) and normalized mean error (NME) of the prediced contact point for seen and unseen objects. NME is defined as the Euclidean error normalized by the triangle edge lengths, where ground truth is obtained by FW with 500 iterations. The results show that performance on unseen objects is comparable to that on seen objects, confirming the effective category-level generalization of our framework. Fig. 7 shows the application of simulation level, and the plausible contacts are well detected both seen and unseen

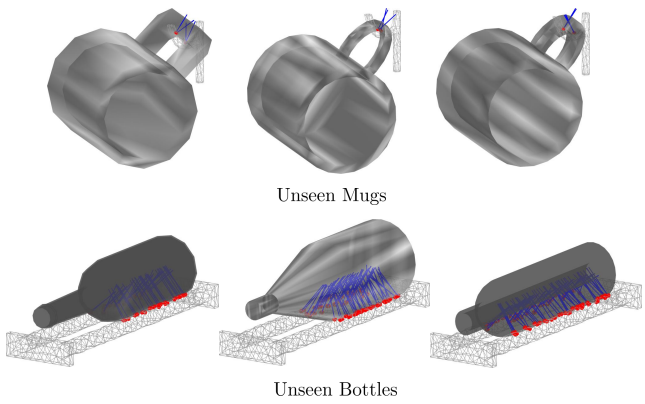

Fig. 6: Contact feature visualization on unseen mug and bottle objects.

	Optimality		NME	
	Seen	Unseen	Seen	Unseen
Mug	8.8576e−04	9.7915e−04	0.0058	0.0089
Bottle	1.9080e−03	4.4341e−03	0.0152	0.0176

TABLE III: Category-level generalization performance, comparing the optimality and normalized mean error (NME) for seen and unseen objects.

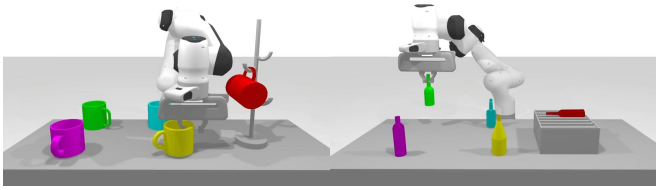

Fig. 7: Snapshots of robotic contact simulations for both seen and unseen objects.

objects.

Although our evaluation is conducted only for two objects, category-level generalization is an inherent property of auto-decoder architecture. Therefore, if the NeuralSDF model is well trained at the category level for arbitrary objects, the amortized model is expected to achieve similar collision detection performance.

C. Ablation Study

An ablation study is conducted to validate each network component, by removing the optimality loss $\mathcal{L}_{\text{op}}$, the minimum signed distance loss $\mathcal{L}_{\text{min}}$, the vertex-level shape feature vectors $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$, and the triangle-level shape feature vector $\mathbf{z}_\triangle$. Performance is measured using the optimality metric in (12) and the normalized mean error (NME) of the predicted point. Table IV shows that removing the optimality loss $\mathcal{L}_{\text{op}}$ significantly degrades performance, yielding the largest differences in both optimality and NME. While the network can be trained using only the minimum signed distance loss, the performance is less competitive. This result confirms that $\mathcal{L}_{\text{op}}$ is indispensable for predicting optimal solutions and

Model Variant	Optimality ↓	NME ↓
Full (Ours)	7.8433e−04	0.0102
w/o $\mathcal{L}_{\text{op}}$	9.5106e−02	0.3049
w/o $\mathcal{L}_{\text{min}}$	4.1231e−03	0.0455
w/o $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$	4.2045e−02	0.1819
w/o $\mathbf{z}_\triangle$	4.4112e−03	0.0460

TABLE IV: Ablation study. The optimality and the normalized mean error (NME) of the predicted contact point are compared.

enables unsupervised learning, as it is directly derived from the optimality conditions.

V. Conclusion

In this work, we presented a novel amortized NeuralSDF–mesh collision detection framework with a learning-based approach. We formulate the NeuralSDF-triangle collision problem as a constrained optimization and derive the corresponding KKT optimality conditions. We leverage the conditions to train our amortized model that predicts the contact point directly, replacing typical iterative procedures. The proposed amortized model, implemented with an auto-decoder architecture, not only improves query speed but also achieves memory efficiency and category-level generalization. Its effectiveness is validated through extensive evaluations.

Since the proposed method is learning-based, contact prediction may degrade for triangles whose sizes differ from the conditions of the training dataset. However, the triangle size adopted in the dataset corresponds to a commonly used values and is set to be sufficiently small in practice. Therefore, for meshes with triangles sizes outside the range, remeshing the mesh to satisfy the dataset conditions enables the proposed method to be applied a broader range of contact scenarios.

The proposed method provides fast and plausible solutions, making it useful in contact-rich scenarios where a large number of contacts must be processed efficiently. However, for tasks requiring high-precision contacts, our learning-based approach may not always be sufficient, and a trade-off between accuracy and computation cost may need to be considered. In addition, the proposed framework relies on NeuralSDF for the loss functions and the definition of contact features, and therefore its performance inherently depends on the quality of the NeuralSDF representation. In such cases, the proposed method can be effectively used as a warm-starting option for NeuralSDF+FW, reducing the computational cost of early iterations, while allowing additional FW refinement to ensure accuracy.

Moreover, our work addresses the SDF-mesh collision problem accurately and efficiently, and can be applied not only to the collision detection of contact simulation but also to other applications that require collision computation. It can be extended to manipulation and planning problems in robotics [23], where collision avoidance is critical. Furthermore, since both NeuralSDF and amortized model are based on neural networks, the entire pipeline provides differentiable contact features, making the method suitable for gradient-based optimization methods in robotic problems.

References

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

[2] 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.

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

[4] I. Quilez, "Inigo quilez, distance functions," *URL: https://iquilezles.org/articles/distfunctions*, vol. 2, 2010.

[5] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[6] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[7] L. Schirmer, T. Novello, V. da Silva, G. Schardong, D. Perazzo, H. Lopes, N. Goncalves, and L. Velho, "Geometric implicit neural representations for signed distance functions," *Computers & Graphics*, vol. 125, p. 104085, 2024.

[8] T. Li, X. Wen, Y.-S. Liu, H. Su, and Z. Han, "Learning deep implicit functions for 3d shapes with dynamic code clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 840–12 850.

[9] G. DeepMind, "MuJoCo XML Reference : sdf_iterations," https://mujoco.readthedocs.io/en/3.1.2/XMLreference.html.

[10] 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.

[11] N. Developer, "Physx sdk," https://developer.nvidia.com/physx-sdk.

[12] P. Liu, Y. Zhang, H. Wang, M. K. Yip, E. S. Liu, and X. Jin, "Real-time collision detection between general sdfs," *Computer Aided Geometric Design*, vol. 111, p. 102305, 2024.

[13] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and Z. Corse, "Local optimization for robust signed distance field collision," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 3, no. 1, pp. 1–17, 2020.

[14] J. Pelletier-Guénette, A. Mercier-Aubin, and S. Andrews, "Real-time triangle-sdf continuous collision detection," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 8, no. 4, pp. 1–22, 2025.

[15] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Traces and emergence of nonlinear programming*. Springer, 2013, pp. 247–258.

[16] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[18] B. Amos *et al.*, "Tutorial on amortized optimization," *Foundations and Trends® in Machine Learning*, vol. 16, no. 5, pp. 592–732, 2023.

[19] A. Muntoni and P. Cignoni, "PyMeshLab," Jan. 2021.

[20] B. Foundataion, "Blender python api," https://docs.blender.org/api/current/index.html.

[21] J. Lee, M. Lee, S. Park, J. Yun, and D. Lee, "Variations of augmented lagrangian for robotic multi-contact simulation," *IEEE Transactions on Robotics*, 2025.

[22] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[23] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2022.