Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

TE-SDF: Tetra-Encoded Signed Distance Field for Memory-Efficient and Accurate Collision Detection

Harim Ji
School of Mechanical Engineering
Seoul National University
Email: jiharim0911@snu.ac.kr

Yongseok Lee
Department of Robotics and Mechatronics
DGIST
Email: yslee@dgist.ac.kr

Dongjun Lee†
School of Mechanical Engineering
Seoul National University
Email: djlee@snu.ac.kr

***Abstract*—A signed distance field (SDF) is a widely used geometric representation for robust collision detection between complex geometries, which is crucial for contact-rich simulations. While numerous works have studied SDF representations and SDF-based collision detection, achieving both memory efficiency and high accuracy while maintaining scalability remains a challenge. In this paper, we propose a novel SDF representation, Tetra-Encoded SDF (TE-SDF), which combines the adaptive spatial discretization of a tetrahedral mesh with exact-distance evaluation localized to each tetrahedron by encoding a compact set of candidate surface faces per tetrahedron. We demonstrate the effectiveness of TE-SDF in contact-rich simulation by implementing a fully GPU-accelerated collision detector based on TE-SDF and integrating it into a GPU-accelerated simulation framework. Our results show that TE-SDF enables memory-efficient, accurate, and scalable collision detection, expanding the domain of robotic simulation scenarios that can be handled in practice.**

I. Introduction

A signed distance field (SDF) implicitly represents the geometry by encoding the distance to the closest surface point throughout a volumetric domain. This property has motivated extensive research and applications of SDFs for efficient and robust collision detection in contact-rich simulations across various fields, such as graphics [29, 17, 22], animations [30], games [6], and robotics [34, 24, 10, 11]. The ability of SDF-based collision detections to robustly handle complex geometries enables the simulation of challenging robotic tasks, such as complex assembly [44, 24, 34].

A common approach to representing SDFs is a voxel-based method that stores precomputed distance values on a dense voxel grid. Evaluation at an arbitrary query point is performed using trilinear interpolation of the values at the eight voxel corners surrounding it. This can be viewed as combining the voxel-grid spatial discretization of the domain with a local linear function, resulting in a piecewise linear approximation of the underlying SDF. This representation is simple and computationally efficient, making it a standard choice in prior works [34, 1, 44, 17]. However, the accuracy of voxel-based representations is constrained by the grid resolution, resulting in a prohibitively large memory footprint for geometries with tight tolerance requirements.

†Corresponding author.

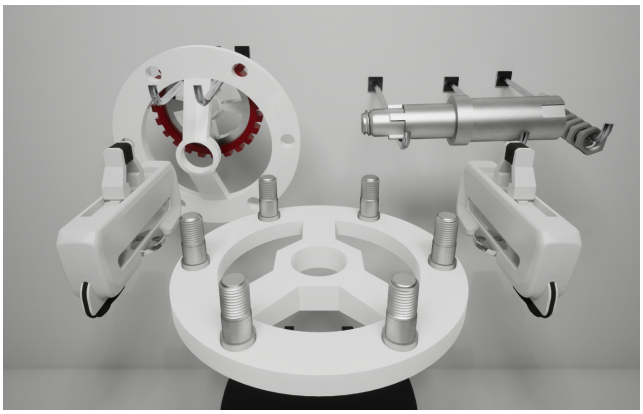


Fig. 1: Overview of 'Robo-assembly' scenario consisting of three peg-in-hole and seven bolt–nut assembly tasks. We use the same meshes for both visualization and collision detection of the assembled meshes.

To mitigate this cost, various studies propose memory-efficient variants by using alternative spatial discretizations, such as octrees [13], or by employing different local functions, such as polynomial [22] or neural network [4, 2, 37, 45]. While these approaches can reduce memory cost, evaluating the SDF values requires complex procedures, such as traversing trees and inferring neural networks. This overhead limits the usage of these representations for Face–SDF collision detection, which requires sequential SDF evaluations coming from the per-face optimization scheme [29]. Consequently, these representations are restricted to non-iterative methods, such as Node–SDF evaluations [22, 26], which may miss contacts between sharp features and edges. Moreover, these methods still approximate the underlying SDF.

In this paper, we propose a novel SDF representation, *Tetra-Encoded SDF* (TE-SDF), which we abbreviate as *TES*. TES is motivated by the following observations regarding tetrahedral meshes (tetramesh) and SDFs:

- **Tetramesh:** Tetrameshes inherently provide adaptive spatial discretizations, allocating more tetrahedra in geometrically detailed regions and fewer tetrahedra in simple regions [40].
- **SDF:** Given an arbitrary query point, if a small set of candidate surface triangles (faces) that may be closest

to that point is known, then evaluating the *exact* SDF at that point reduces to a few point–triangle distance computations.

Based on these observations, TES implicitly represents the exact SDF by encoding a compact set of candidate faces for each tetrahedron. This enables an SDF query at a point within a given tetrahedron to be evaluated using only a few point–triangle distance computations. By combining the adaptive spatial discretization of a tetrahedral mesh with exact-distance evaluations localized to each tetrahedron, TES provides a memory-efficient and accurate SDF representation. Also, due to its simple, localized SDF evaluations, TES enables efficient GPU-accelerated collision detection for both Node–SDF and Face–SDF methods. We implement and integrate this new collision detection framework with a GPU-accelerated simulation [19] and evaluate it in challenging contact-rich scenarios involving multiple complex meshes and tight geometric tolerances (Fig. 1; Sec. VII-B and Sec. VII-C). Across these scenarios, TES meets the accuracy requirements while consuming only a few tens of megabytes (MB) of memory and maintains collision detection time costs on the order of a few milliseconds. To the best of our knowledge, this combination of memory efficiency, accuracy, and scalability for collision detection of multiple complex geometries has not been reported. Moreover, TES can maintain accurate SDF evaluation in the neighborhood of the surface, even under geometric deformation, without additional runtime overhead, making it promising for soft-body simulations as well. An implementation of TES will be made available as a library.[1][2]

In contrast to prior work that also stores faces for localized exact-distance evaluation [18, 26], TES fundamentally differs in its spatial discretization strategy and its practical suitability for contact-rich simulation. Specifically, Huang et al. [18] relies on a voxel-grid discretization, which inherits the memory inefficiency of voxel-based SDF representations and has not been demonstrated as a collision detection method. In contrast, Liu and Kim [26] employ a rectangular swept sphere (RSS) as a spatial discretization. However, this approach requires a tree traversal for each SDF query, which can be computationally expensive and is primarily suited to Node–SDF collision detection. Moreover, the method [26] is not designed to handle a large number of SDF objects simultaneously, limiting its scalability in scenes with many interacting geometries.

The rest of the paper is organized as follows. An overview of previous works and preliminaries is explained in each Sec. II and Sec. III. Structural definitions and methodology for constructing TES are given in each Sec. IV and Sec. V. Implementation details of the collision detection with TES are given in Sec. VI. Evaluations of TES and collision detection performance are shown in Sec. VII. Finally, discussions and conclusions are made in each Sec. VIII and Sec. IX.

[1]https://github.com/HarimJi/TES-sdk

[2]https://github.com/HarimJi/TES-Encoder

II. Related works

A. Representation of SDF

Evaluation of SDFs with a **voxel-based** representation can be done efficiently with several lookups and a single trilinear interpolation. However, this method incurs an $O(n^3)$ memory cost, where n is the grid resolution, resulting in intensive memory usage for handling large, detailed geometries. In this context, memory-efficient representations of SDFs have been extensively studied. **Hierarchical** data structures can assign smaller voxels to geometrically detailed regions while placing larger voxels in simple regions [13, 32, 33], thereby greatly reducing the memory cost compared to using dense grids. Koschier et al. [22] extends this idea by expressing the SDF within the hierarchically placed voxel as polynomials instead of linear functions. Liu and Kim [26] utilize a bounding volume hierarchy (BVH) with rectangular swept spheres as its bounding primitive for spatial discretization and use locally stored faces for exact SDF evaluations. Huang et al. [18] also store faces but use a dense voxel grid for spatial discretization. Another branch of research on compact representations of SDF uses a **neural network** (NN). Park et al. [37] uses an auto-decoder architecture to train NN-SDF for general shapes encoded as latent vectors. Chabra et al. [4] use voxel-based discretization of the entire geometry into local shapes and use [37] to represent local SDFs. Lee et al. [24] and Bertiche et al. [2] show that NN-SDF can be used for collision detection in simulations

B. SDF-based collision detection

A straightforward way to use SDFs for collision detection is to perform **Node–SDF** evaluations, in which the SDF of one mesh is evaluated at every node of the other mesh. This method has been successfully implemented for contact-rich cloth simulation [14], rigid-body simulations [17, 44], and soft-body simulations [30, 31]. However, testing only the nodes can miss contacts in cases of edge–edge contact or when large faces are involved. To overcome this issue, Macklin et al. [29] introduce a **Face–SDF** collision detection by solving a per-face optimization problem for the SDF with the Frank-Wolfe method. This method can robustly handle challenging collision-detection scenarios involving complex meshes, such as bolt-nut assemblies [34]. Liu et al. [27] show that optimization-based methods can also be applied to **SDF–SDF** collision detection with the projected gradient method and multiple initial guesses. Mujoco [11] also supports SDF-based collision detection with a similar approach, while it can handle universal SDF representations. While the SDF-based method is generally more robust to artifacts arising from discrete-time collision detection than surface-based methods, several works have also explored **continuous collision detection** (CCD) using SDFs. Bender et al. [1] formulate the node–SDF CCD as a 1-dimensional root-finding problem. Pelletier-Guénette et al. [38] extend the optimization problem formulated in [29] to the temporal domain, introducing a temporally augmented Frank-Wolfe method combined with Golden-section line search to

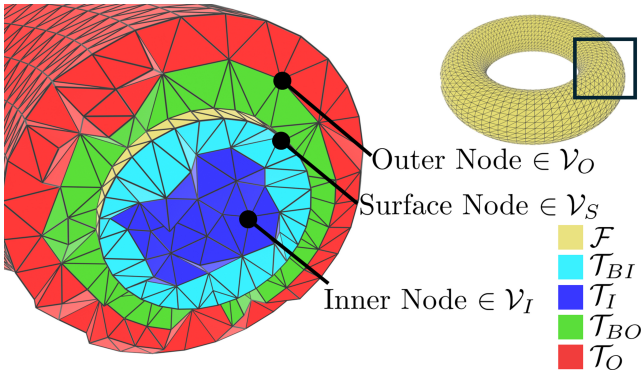


Fig. 2: Tetramesh generated from a trimesh of a torus.

compute the time of impact (TOI).

III. Preliminary

A. Tetrahedral mesh

Given a triangulated surface mesh (trimesh) of a three-dimensional solid, a tetrahedral mesh (tetramesh) discretizes a volumetric domain into tetrahedral elements, which include the interior and maybe the exterior region around the surface. In this paper, we define a tetramesh G of a trimesh as a tuple

$$G := (\mathcal{V}, \mathcal{F}, \mathcal{T}) \tag{1}$$

where $\mathcal{V} = \{v_i \in \mathbb{R}^3\}_{i=1}^{N_V}$, $\mathcal{F} = \{f_i \subset \mathbb{R}^3\}_{i=1}^{N_F}$, and $\mathcal{T} = \{t_i \subset \mathbb{R}^3\}_{i=1}^{N_T}$ is a set of each nodes (vertices), triangular faces, and tetrahedra. For notational convenience, we abuse notation and identify a set of faces $F \in \mathcal{F}$ with its spatial region $\bigcup_{f\in F} f$ when no ambiguity arises. We adopt the same abuse of notation for a set of tetrahedra $T \in \mathcal{T}$.

As shown in Fig. 2, the node set $\mathcal{V}$ and tetrahedron set $\mathcal{T}$ can be classified as follows,

$$\mathcal{V} = \mathcal{V}_S \cup \mathcal{V}_I \cup \mathcal{V}_O, \quad \mathcal{T} = \mathcal{T}_{BI} \cup \mathcal{T}_I \cup \mathcal{T}_{BO} \cup \mathcal{T}_O$$

where $\mathcal{V}_S$, $\mathcal{V}_I$, $\mathcal{V}_O$ each denote the set of *surface*, *inner*, and *outer* vertices while, $\mathcal{T}_{BI}$, $\mathcal{T}_I$, $\mathcal{T}_{BO}$, $\mathcal{T}_O$ each denote the set of *boundary-inner*, *inner*, *boundary-outer*, and *outer tetrahedra*. We call a tetrahedron an inner tetrahedron if all of its vertices are inner vertices, and a boundary-inner tetrahedron if it contains at least one surface vertex. The definitions of outer tetrahedron and boundary-outer tetrahedron are analogous.

B. Signed Distance Field

In this paper, we define the Signed Distance Field (SDF) of a tetramesh $G = (\mathcal{V}, \mathcal{F}, \mathcal{T})$, $\phi_G : \mathcal{T} \to \mathbb{R}$ as follows:

$$\phi_G(x) = \begin{cases} -d(x, \mathcal{F}) & \text{if } x \in \mathcal{T}_{BI} \cup \mathcal{T}_I \\ d(x, \mathcal{F}) & \text{if } x \in \mathcal{T}_{BO} \cup \mathcal{T}_O \end{cases}$$

where, the distance function, $d(\cdot, \mathcal{X}) : \mathbb{R}^3 \to \mathbb{R}$ for some set $\mathcal{X} \in \mathbb{R}^3$ is defined as, $d(x, \mathcal{X}) = \inf_{y\in\mathcal{X}} \|x - y\|$. Notice that this definition is a restriction of the traditional SDF to the domain where it is tetrahedralized. But this doesn't confine the domain to the interior of the surface as long as $\mathcal{T}_{BO} \cup \mathcal{T}_O$ is not empty. In practice, the exterior region can be selectively tetrahedralized in areas where pre-contact constraint activation is important, such as tight assembly interfaces (e.g., bolt–nut engagement) or interactions involving fast-moving thin objects (e.g., falling spoons). An ablation study comparing the robustness of the collision detection with and without the exterior tetrahedral region is provided in Appendix F.

C. GPU-Memory Hierarchy

In this section, we briefly review the GPU memory hierarchy, which is crucial for high-performance GPU acceleration. We focus on three types of memory relevant to this work: register, global memory, and local memory. Registers are on-chip memories that provide local, low-latency, and small storage for a thread. Global memories, on the other hand, are off-chip memory accessible to all threads. They provide large storage but with higher latency than registers. Local memories are private to a thread, similar to registers, but are off-chip, having similar latency to global memories. These are used automatically when memories exceeding the register capacity must remain in a thread, a phenomenon referred to as *register spilling*. When implementing acceleration on GPUs, kernels (functions executed on the GPU) should be designed to avoid excessive register pressure per thread (i.e., the amount of register memory used per thread) to avoid register spilling [8].

IV. Tetra-Encoded Signed Distance Field

The majority of the computation for evaluating the SDF value, $\phi_G(x)$, comes from the distance calculation, $d(x, \mathcal{F})$. However, such a calculation reduces to a trivial point-triangle distance evaluation once the closest face to the query point x is known. We extend this idea to the tetrahedron level: suppose the query point x is known to be inside a tetrahedron $t \in \mathcal{T}$ and the set $E_t \subset \mathcal{F}$ satisfying the following condition is given:

$$f \in E_t \iff \exists y \in \text{int}(t) \quad s.t.\ d(y, f) = d(y, \mathcal{F}) \tag{2}$$

That is, a face f is included in E_t if and only if there exists a point inside the tetrahedron t for which f is the closest face among all faces in $\mathcal{F}$. Then we have,

$$d(x, \mathcal{F}) = d(x, E_t)$$

which can be computed with $n(E_t)$ point-triangle distance evaluations. We refer to this set E_t as the set of *encoded faces* of the tetrahedron t, which can be interpreted as the set of faces that can *possibly be the closest* to points within t.

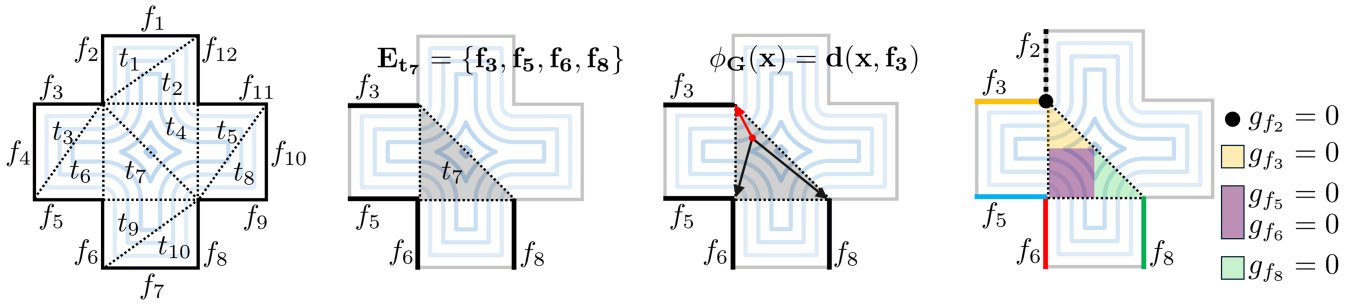
With this context, we define the Tetra-Encoded SDF (TES) of the tetramesh $G = (\mathcal{V}, \mathcal{F}, \mathcal{T})$, as the extended tuple of G:

$$TES(G) := (\mathcal{V}, \mathcal{F}, \mathcal{T}, \mathcal{E}) \tag{3}$$

where $\mathcal{E}$ is an indexed set of encoded face sets defined as follows:

$$\mathcal{E} := \{E_t \subset \mathcal{F}\}_{t\in\mathcal{T}}$$

With this definition, the evaluation of SDF with TES, given the query point x that is known to be inside $t \in \mathcal{T}$, reduces



(a) (b) (c) (d)

Fig. 3: (a) A tetramesh of a "cross" illustrated in 2D for clarity. In this projection, tetrahedra reduce to triangles and faces reduce to line segments. The mesh consist of 12 faces $(f_1, f_2, \ldots, f_{12})$ and 10 tetrahedra $(t_1, t_2, \ldots, t_{10})$. For clarity, the SDF is visualized only in the interior region. (b) Encoded faces associated with tetrahedron t_7. (c) SDF evaluation at the query point (red) using 4 point–triangle distance computations over the encoded faces. Note that f_5 and f_6 have the same closest point to the query point. (d) Regions where the gap function associated with each encoded face is zero. Note that $g_{f_2} = 0$ only at the boundary of t, thus f_2 is not included in E_{t_7}.

to a few point–triangle evaluations as follows:

$$\phi_G(x) = \begin{cases} -\,d\,(x, E_t)\,, & t \in \mathcal{T}_{BI} \cup \mathcal{T}_I \\ \quad d(x, E_t)\,, & t \in \mathcal{T}_{BO} \cup \mathcal{T}_O \end{cases} \tag{4}$$

The overall concept of TES is illustrated in Fig. 3.

V. Encoding

In this section, we describe the methodology for obtaining the sets of encoded face sets, $\mathcal{E}$, to construct the TES collider for a given tetramesh G before the simulation. A complete algorithm for TES encoding is summarized in Algorithm 1.

Given a tetrahedron t, to decide whether the face f is in $E_t \in \mathcal{E}$, we introduce the gap function $g_f : \mathbb{R}^3 \to \mathbb{R}$ by

$$g_f(x) := d(x, f) - d(x, \mathcal{F})$$

Membership of f in E_t can be characterized as the following feasibility problem derived from (2):

$$\text{find } x \in \text{int}(t) \quad \text{s.t. } g_f(x) = 0 \tag{5}$$

The face f belongs to E_t if and only if the feasibility problem admits a solution, as shown in Fig. 3d.

A. Lipschitz pruning

Instead of checking every $f \in \mathcal{F}$ by solving the feasibility problem (5), we prune the majority of the faces to have $\mathcal{F}' \subset \mathcal{F}$ by using the following lemma:

Lemma 1. *Given $\mathcal{B}(x_0, R) \supset \text{int}\,(t)$, then we have the following:*

$$f \notin E_t \Leftarrow g_f(x_0) \geq 2R \tag{6}$$

where $\mathcal{B}(x_0, R) \subset \mathbb{R}^3$ denotes an open ball centered at x_0 with radius R .

Thus, by setting x_0 as the center of the tetrahedron and R to tightly include $\text{int}\,(t)$, we can prune out the majority of the faces by a simple predicate (6) before solving the feasibility problem. Once the candidate face set $\mathcal{F}'$ is obtained, we can further reduce the computation from the fact that for any $x \in t$, we have, $d(x, \mathcal{F}) = d(x, \mathcal{F}')$. That is, the evaluation of $d(x, \mathcal{F})$ can be done in $n(\mathcal{F}')$ point-triangle distance evaluations. Detailed proof of Lemma 1 is provided in Appendix A.

B. GPU-accelerated brute force method

To solve the feasibility problem (5), we adopt a brute-force sampling strategy. Sampling points set, $t^{\alpha,Q}$, is chosen as a barycentric grid of resolution $Q \in \mathbb{N}$ over a homogeneously shrunken tetrahedron obtained by scaling t by a factor $\alpha \in (0, 1)$ to approximate the interior domain $\text{int}(t)$ by a finite set. For practical encoding timings, we accelerate the process by assigning a GPU thread to each sample point $x \in t^{\alpha,Q}$. Each thread evaluates the gap function $g_f(x)$ at the assigned sample point for every candidate face, $f \in \mathcal{F}'$, and inserts f into E_t whenever $g_f(x) \leq \epsilon$ for a certain threshold ϵ, as described in Algorithm 1. Since missing a valid face in the encoded face set E_t can lead to an uncontrolled loss of SDF accuracy, our primary objective is to avoid false negatives. Brute-force evaluation provides a conservative and controllable test for membership in E_t. We describe details for sampling points $t^{\alpha,Q}$, and a formal proof that there exists a sampling density above which no valid faces are missed, provided in Appendix B.

While a natural alternative is to formulate the problem as an optimization over t by minimizing the gap function $g_f(x)$ and checking whether its minimum is below a threshold ϵ, this approach is often unreliable in practice due to the non-smoothness and non-convexity of the SDF, making the result sensitive to initialization and lacking convergence guarantees [38].

C. Distance-ordered truncation

We argue that exact SDF evaluation only in a neighborhood of the object surface is sufficient for accurate and stable

physics simulation, similarly stated in [10, 2], based on the following observations:

- In contact-rich simulations, collision events and constraint activations happen predominantly in the vicinity of the object surface.
- When deep penetration occurs, the exact SDF is less critical; it is sufficient that the approximate SDF provides meaningful contact features that reduce penetration.

Within this context, we introduce a distance-ordered truncation method of the encoded face set to $E'_t \subset E_t$ which regulates the cardinality of the encoded set, $n(E'_t) \leq E_{\max}$, while preserving accuracy near the surface. This relaxes the diversity of the sizes of the encoded face sets within $\mathcal{E}$, which can degrade SDF evaluation performance on the GPU during simulation (one process that performs a few point-triangle distance evaluations may have to wait for another that performs hundreds of such evaluations). The truncation is done as follows: for every face $f \in E_t$, we calculate d_f, which is the minimum distance encoded by f in the tetrahedron region t:

$$d_f = \min_{x \in t,\ g_f(x)=0} d(x, f) \tag{7}$$

We then order the encoded faces in E_t by d_f in ascending order and include the first $E_{\max}$ faces in E'_t. The rationale behind this method is to prioritize the encoded faces that contribute to the smaller distance value, thus preserving accuracy near the surface. In practice, we calculate the approximation of d_f under the discretized tetrahedron described in Sec. V-B:

$$d_f \approx \min_{x \in t^{\alpha,Q},\ g_f(x) \leq \epsilon} d(x, f) \tag{8}$$

In this way, the ordering can be done along with encoding, without additional post-processing.

Algorithm 1 TES Encoding with Distance-Ordered Truncation

Require: Tetramesh $G = (\mathcal{V}, \mathcal{F}, \mathcal{T})$, shrink factor $\alpha \in (0, 1)$, grid resolution Q, threshold ϵ, maximum cardinality of encoded faces $E_{\max}$
Ensure: Encoded face sets $\mathcal{E} = \{E_t\}_{t \in \mathcal{T}}$

```
for all tetrahedra t ∈ T do
                                        ▷ — Lipschitz pruning —
    Initialize F' = ∅
    Set B(x_0, R) ⊃ int(t)
    for all faces f ∈ F in parallel do
        if g_f(x_0) < 2R then
            F' ← F' ∪ {f}
        end if
    end for
                                        ▷ — Brute-force feasibility test —
    Initialize E_t ← ∅
    Initialize distance priority map {d_f ← ∞}_{f∈F'}
    for all sample points x ∈ t^{α,Q} in parallel do
        Compute d_trimesh ← d(x, F')
        for all faces f ∈ F' do
            if d(x, f) − d_trimesh ≤ ε then
                E_t ← E_t ∪ {f}
                d_f ← min{d_f, d(x, f)}
            end if
        end for
    end for
                                        ▷ — Distance-ordered truncation —
    Sort faces in E_t by d_f in ascending order
    Truncate E_t to retain the first E_max faces
end for
    return E = {E_t}_{t∈T}
```

D. Encoding for deformation

The TES constructed in the previous section can be extended to support deforming trimeshes while preserving accurate SDF evaluation in a neighborhood of the surface. We consider deformations that alter node positions while maintaining the connectivity of the underlying mesh. For every boundary tetrahedra, $t \in \mathcal{T}_{BI} \cup \mathcal{T}_{BO}$, we augment the encoded face sets E_t, to include all faces that are adjacent to the surface vertices of t with corresponding distance priority as $d_f = 0$ before distance-ordered truncation. This ensures that faces that may become locally closest under deformation are captured in the encoded set.

VI. GPU-accelerated collision detection

In this section, we describe the implementation of a GPU-accelerated collision detection with TES. To ensure generality and scalability, we design the collision detection to handle multiple meshes within a single execution pass: even when multiple mesh pairs are potentially in contact, all collision queries are resolved within a unified GPU kernel rather than launching a separate kernel per pair. This design enables a consistent, minimal number of kernel launches per simulation time step, thereby reducing kernel launch overhead. We implement two mutually exclusive collision detection modes: Node–TES and Face–TES.

A. Node–TES

The Node–TES generates contact features by evaluating SDF values and gradients at the mesh nodes of each TES collider with respect to all other TES colliders in the scene, similar to [2, 14, 17, 22, 26, 44]. We first conduct a broad-phase to identify every overlapping node–tetrahedron pair, which can be efficiently implemented on GPUs using the method of Karras [20]. In the narrow-phase, we assign one GPU thread to each node–tetrahedron pair, (v, t), which is passed from the broad-phase, where v is a node of one TES collider and t is a tetrahedron of another TES collider. The thread then evaluates the SDF value at v and the gradient using (4) by sequentially conducting point–triangle distance evaluations with faces encoded in E_t. A detailed algorithm is provided in Appendix C.

B. Face–TES

The Face–TES generates contact features for each overlapping face-tetrahedron pair by evaluating the minimum of the localized signed distances over the mesh faces, following a

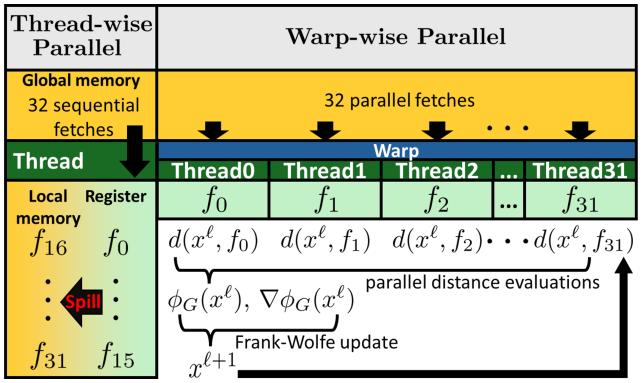


Fig. 4: Comparison between thread-wise and warp-wise parallel Face–TES collision detection scheme with $n(E_t) = 32$. Thread-wise parallelization spills half of the fetched faces from global to local memory, resulting in performance loss. Warp-wise parallelization has no such spill, as it contains all necessary data in registers. This scheme also enables parallelized face set fetching and SDF evaluation.

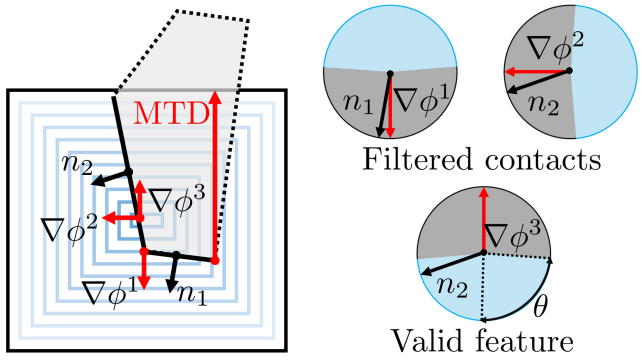


Fig. 5: Contact features generated by Face–TES method. Here, two faces, each having normal as n_1 and n_2, are tested against the SDF of a rectangle. Contact normals $\nabla\phi^1$ and $\nabla\phi^2$ are filtered out from the cosine-filter, whereas $\nabla\phi^3$ is considered as a valid contact normal.

similar approach in [29]. We first perform a broad-phase to identify every potentially overlapping face-tetrahedron pair, followed by a middle phase that computes the intersection between the face and the tetrahedron t, which becomes a convex polygon P of up to 6 vertices. The narrow phase evaluates contact features by solving the following optimization problem with an iterative Frank-Wolfe method:

$$\min_{x\in P}\phi_G(x) = \min_{x\in P}\begin{cases} -d\,(x, E_t)\,, & \text{if } t \in \mathcal{T}_{BI}\cup\mathcal{T}_I \\ \;\;\,d(x, E_t)\,, & \text{if } t \in \mathcal{T}_{BO}\cup\mathcal{T}_O \end{cases} \tag{9}$$

which can be derived from (4) and the fact that $P \subset t$. A naive approach to accelerate this narrow-phase on a GPU is to assign one thread per optimization problem. However, this approach often leads to poor performance due to excessive register pressure on the GPU thread. Specifically, consider the case where $n(E_t) = 32$, as illustrated in Fig. 4. At each iteration, the thread should fetch geometric data for the 32 faces from global memory to evaluate the SDF value and gradient using (4). To avoid redundant memory transactions, the thread should store these faces in its local register for later iterations. However, this leads to excessive register pressure, causing register spilling and degrading the performance as described in Sec. III-C. To avoid these issues, we assign a warp (a GPU architecture-defined execution unit consisting of 32 threads executed in lockstep [9]) rather than a thread to each optimization problem within the narrow-phase kernel. Specifically, as illustrated in Fig. 4, each thread within a warp is assigned one face $f \in E_t$ that is small enough to fit in the local register without spilling. Then the signed distance value and the gradient required for each Frank-Wolfe iteration can be computed cooperatively by the 32 threads: each thread evaluates the point–triangle distance in parallel and combines the results using the warp shuffle operation. Note that although we assume 32 encoded faces for simplicity, we can mask out excess threads if we have fewer faces or make extra fetches per thread if we have more. For example, if $n(E_t) \le 96$, each thread will fetch and store at most 3 faces. By distributing data loading, storing, and computation across threads, warp-level parallelism significantly reduces per-thread register pressure and enables efficient parallel evaluation. A detailed algorithm for the Face–TES method is provided in Appendix C.

C. Cosine filter

Collision detection with TES provides multiple local contact features rather than a single global feature (e.g., [16, 35]). This strategy of generating multiple local features by solving small local problems circumvents the challenging task of finding the global feature in complex, non-convex geometries, and is generally adopted across various collision detection methods [25, 5, 29, 23]. However, this locality can generate spurious contact features even when the local problems are correctly solved. For example, suppose we generate local contact features between a pair of overlapping geometries as shown in Fig. 5, using the Face–TES method. Here, a spurious contact feature with contact normal $\nabla\phi^1$ pointing to the opposite side of the minimal translational distance (MTD) [16] is generated, which is obtained by minimizing the SDF on the face with its normal being n_1. Note that this phenomenon is not specific to TES but rather general across methods for finding local contact features. To prevent this issue, we add a 'Cosine filter' at the end of the Face–TES collision detection, which filters out the contact feature if the following condition is true:

$$\nabla\phi \cdot n > -\cos\theta \tag{10}$$

where $\nabla\phi$ is the gradient of the SDF used as contact normal, and n is the oriented normal of the face being tested. This process can be added to the returning process of the Face–TES method without requiring any additional kernel launches.

VII. Experiments

In this section, we evaluate the TES encoding and the collision detection, each described in Sec. V and Sec. VI.

TABLE I: Encoding statistics where "full" indicates no distance-ordered truncation, and "32" indicates truncation with $E_{\max} = 32$. Details on memory, maximum error, exactness-band, and error-band are given in Appendix D.

Name	$N_{\mathcal{V}}$	$N_{\mathcal{F}}$	$N_{\mathcal{T}}$	Q / ϵ	#enc. face	memory (MB)	max error (%)	exactness-band	1% error-band
					avg / max	full / 32	full / 32	full / 32	full / 32
M16 bolt	13695	23418	63164	128 / 0	10.7 / 437	3.580 / 3.260	0 / 35.7	1.0 / 0.121	1.0 / 0.373
M16 Nut	3147	5968	9142	128 / 0	8.4 / 59	0.436 / 0.433	0 / 52.6	1.0 / 0.406	1.0 / 0.489
Hole	4559	8448	22405	128 / 0	7.36 / 30	0.975 / 0.975	0 / 0	1.0 / 1.0	1.0 / 1.0
Shaft	1881	3334	6111	256 / 1E-7	29.4 / 280	0.782 / 0.598	0 / 28.4	1.0 / 0.105	1.0 / 0.114
Spoon	2588	3138	12301	128 / 1E-9	15.8 / 35	0.941 / 0.941	0 / 0.73	1.0 / 0.158	1.0 / 1.0
Gripper	3353	5176	11429	64 / 0	16.1 / 33	0.888 / 0.888	0 / 7E-3	1.0 / 0.97	1.0 / 1.0
Deformed gripper	"	"	"	-	"	"	8.9 / 8.9	0.151 / 0.151	0.303 / 0.303

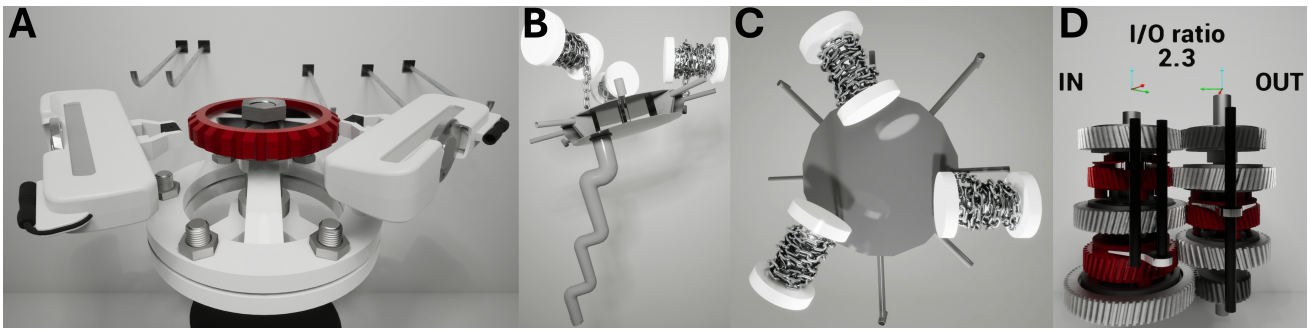


Fig. 6: Illustrations of simulation scenarios described in Sec. VII-B. **A**: Robo-assembly scenario after completion. **B**, **C**: SAM suspended with chain spools. **D**: Engine transmission scenario.

Every experiment is done with an Intel Core i9-13900K CPU, and NVIDIA RTX 5090 GPU PC. We also recommend that the readers see the supplemental video for more details.

A. TES encoding results

We construct TES for various meshes using Algorithm 1 with the scaling parameter set as $\alpha = 0.999$. The remaining parameters (Q and ϵ) are chosen to achieve an exact SDF reconstruction. These meshes will be used in the simulation scenarios described in Sec. VII-B, Sec. VII-C, and Sec. VII-D. The TES domain is chosen to include the object's outer region wherever it is geometrically or functionally critical, such as along the bolt thread or around the surface of a thin spoon. The domain is then tetrahedralized using **Gmsh** [15]. Detailed figures of the meshes are provided in Appendix D. We compare TES against the ground-truth SDF on a uniform voxel grid of sample points, with voxel size set to the bounding-box diagonal length divided by 2^{10}. Statistics are shown in Table I.

1) Accuracy: TES reproduces the ground-truth SDF to within numerical precision by appropriately selecting the encoding parameters. Moreover, even when distance-ordered truncation is applied with $E_{\max} = 32$, TES preserves SDF accuracy in a neighborhood of the surface, as indicated by a non-zero exactness band. This indicates that truncation preserves SDF accuracy in the near-surface region, which is most relevant for collision detection during simulation.

2) Memory Efficiency: TES exhibits low memory consumption across all tested assets (e.g., a highly detailed `M16 bolt` mesh requires only a few megabytes of memory). We further compare the memory consumption of TES with voxel-based SDF representations in Sec. VII-C, showing the memory-efficiency improvements under tight-tolerance requirements.

3) Deformability: We evaluate TES under deforming geometry using the `Gripper` mesh. Specifically, we apply an arbitrary deformation to obtain a `Deformed gripper` mesh, and evaluate the SDF using TES, which is encoded from the undeformed mesh. As shown in Table I and Fig. 11 in Appendix D, TES maintains exactness near the surface under deformation and distance-ordered truncation, demonstrating its suitability for extension to soft-body simulation.

B. GPU-accelerated simulation

We test the collision detection performance in challenging scenarios, as shown in Fig. 6, by integrating with a GPU-accelerated simulation (e.g., [19]). We use the Face–TES method with fixed iteration, $K = 16$, cosine-filter threshold, $\cos\theta = 0.3$, and use distance-ordered truncated TES-colliders with $E_{\max} = 32$. The time step is set to 5ms for every scene. The statistics and performance for each scenario are summarized in Table II. The results show that our framework robustly and accurately handles collision detection in challenging scenarios while maintaining practical performance.

1) Robo-assembly: A pair of robot grippers, teleoperated by a VR user, conducts a long-horizon assembly task. Collision detection should be fast and accurate enough for interactive and physically realistic assembly.

2) Chained suspended aerial manipulator: A suspended aerial manipulator (e.g., SAM [39]) is pulled upward via three chain spools, each containing 100 chain links. Contact becomes extremely intense where the chains are being spooled.

TABLE II: Collision detection statistics for the scenarios shown in Fig. 6 using the Face–TES method. BP, MP, and NP correspond to the broad, middle, and narrow phases, respectively, and "#NP queries" denotes the number of narrow-phase optimization problems (9) generated.

Name	memory (MB)	time cost (ms)	#NP queries
		BP / MP / NP	avg / max
Robo-assembly	5.80	0.81 / 0.10 / 0.10	9094 / 16711
Chained SAM	52.91	2.1 / 0.20 / 0.49	140097 / 185370
Engine transmission	8.05	0.71 / 0.09 / 0.08	2073 / 4513
Bolt–nut	3.69	1.41 / 0.19 / 0.14	20746 / 26185
Peg-in-hole	2.59	0.70 / 0.15 / 0.21	38424 / 55600
Pile of shafts	88.24	1.71 / 0.09 / 0.05	8053 / 150678

3) Engine transmission: The red gears control the transition between five transmission ratios. Power flows from the input to the output shaft via contact features.

4) Pile of shafts: 128 number of `Shafts` used in the 'Robo-assembly' scenario is scattered arbitrarily as shown in Fig. 8. The initial state has many deeply penetrating object pairs. TE-SDF robustly resolves these penetrations within a few frames. Detailed discussions are provided in Sec. VII-D.

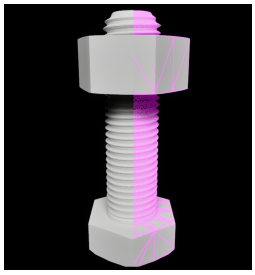
C. Comparison with voxel-based SDF

We compare the memory footprint of TES against a voxel-based SDF representation in a frictionless M16 bolt–nut assembly (dimensions according to ISO 4032) and a complex-shaped frictionless peg-in-hole. We vary the clearance to evaluate collision accuracy under increasingly strict geometric tolerances. As a baseline, we use a dense voxel-grid SDF collider in Isaac Sim for a straightforward, controlled comparison of memory costs across different resolutions. To reduce discrepancies arising from differences in the underlying dynamics solvers [19, 28], we use various timesteps ($1\,\mathrm{ms} \sim 5\,\mathrm{ms}$) and set the maximum number of solver iterations.

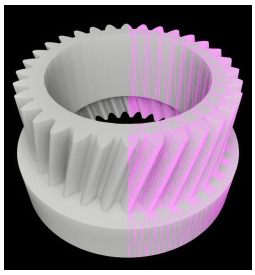
As shown in Table III, TES exhibits physically reliable behavior (e.g., the nut and peg fall under gravity) under all tested tolerances for both methods (Node–TES and Face–TES) while consuming only a few megabytes (MB) of memory. In contrast, the voxel-based SDF requires substantially more memory to represent the geometry for tight-tolerance collision detection. These results indicate that TES provides a memory-efficient alternative for scenarios involving multiple complex and detailed geometries. Further analysis of geometrical error for voxel-based SDF is discussed in Appendix. G.

D. Analysis of Distance-Ordered Truncation

We further study how the distance-ordered truncation parameter $E_{\max}$ affects the accuracy and runtime cost of collision detection using the 'Pile of shafts' scenario shown in Fig. 8. Specifically, we test a range of truncation settings, $E_{\max} = 32, 64, \ldots, 256, 288$ ($>$ full), and analyze their effects on the geometric accuracy of TE-SDF, the robustness under the existence of deeply penetrating objects, and the time cost for Face–TES narrow-phase.



(a) Bolt-nut



(b) Peg-in-hole

Fig. 7: Scenarios used for comparisons between TES and voxel-based SDF.

TABLE III: Comparison results with voxel-based SDF shown in Fig. 7. $Q_\star$ is the quality settings of voxel-based SDF provided in Isaac Sim. S and F denote each success and failure in the simulation without artifacts. Details for memory cost evaluation for voxel-based SDF are provided in Appendix E.

Nut tolerance: 100 / 50 / 25 μm						
Time step (ms)	Voxel-SDF (Q_{nut} / Q_{bolt})				TES	
	256 / 256	512 / 512	512 / 1024	1024 / 1024	Node	Face
1	F / F / F	S / F / F	S / S / F	S / S / F	S / S / S	S / S / S
3	F / F / F	S / F / F	S / S / F	S / S / F	S / S / S	S / S / S
5	F / F / F	S / F / F	S / S / F	S / S / F	S / S / S	S / S / S
Memory (Mb)	15.82	126.26	502.08	1007.52	3.69	

Peg tolerance: 150 / 100 / 50 μm						
Time step (ms)	Voxel-SDF (Q_{peg} / Q_{hole})				TES	
	256 / 512	512 / 256	512 / 512	1024 / 1024	Node	Face
1	F / F / F	S / S / F	S / S / F	S / S / S	S / S / S	S / S / S
3	S / F / F	S / S / F	S / S / F	S / S / S	S / S / S	S / S / S
5	S / F / F	S / S / F	S / S / F	S / S / S	S / S / S	S / S / S
Memory (Mb)	70.20	45.54	102.41	815.03	2.59	

As $E_{\max}$ increases, TE-SDF approaches the exact SDF as more encoded faces are retained for distance evaluation; in our setting, $E_{\max} = 256$ already recovers the exact SDF as shown in Table IV. Nevertheless, the minimal setting $E_{\max} = 32$ shows robust resolution of deeply penetrating objects, as shown in Fig. 8. This shows that although the distance-ordered truncation may provide an inaccurate SDF in deep regions, it still provides valid contact features to resolve penetration. We further benchmark the Face–TES narrow-phase time cost at both the 'Start frame' and the 'End frame' of the scenario, which contain 150,678 and 8,208 narrow-phase queries, respectively. As shown in Fig. 9, larger values of $E_{\max}$ increase the narrow-phase cost due to additional face fetches, point–face distance evaluations, register pressure, and use of shared memories [7]. However, the time cost gap between the smallest and largest $E_{\max}$ settings narrows as the scenario progresses, because the number of narrow-phase queries decreases and fewer SDF evaluations are performed in deep-interior regions as the penetrations between the objects are being resolved.

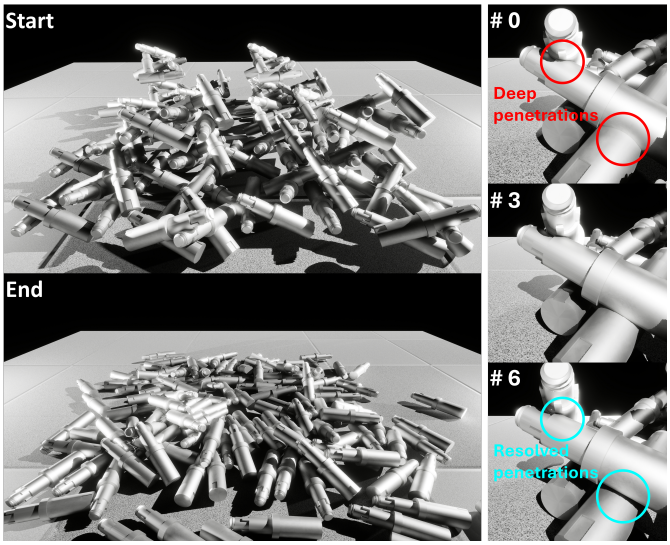


Fig. 8: (Right) Overview of the 'Start frame' and the 'End frame' of the 'Pile of shafts' scenario with $E_{\max} = 32$. (Left) Close-up view of deep penetrations at the initial state. These penetrations are resolved within a few frames (frame #0 ~ frame #6).

TABLE IV: Statistics of memory cost and geometric accuracy of the `Shaft` mesh with varying $E_{\max}$ settings. Notice that statistics for $E_{\max} = 32,\ 288$ ($>$ full) are already given in Table I.

$E_{\max}$	64	96	128	160	192	224	256
memory (MB)	0.709	0.750	0.770	0.778	0.781	0.782	0.782
max error (%)	22.74	18.28	16.57	14.07	3.38	0.02	0
exactness-band	0.215	0.307	0.399	0.505	0.545	0.601	1.0
1% error-band	0.282	0.395	0.536	0.555	0.555	1.0	1.0

VIII. Discussion

While TES enables effective collision detection in challenging scenarios, there is some promising future work. First, TES admits multiple variants that exploit the localized surface information per tetrahedron in different ways. For example, instead of storing encoded faces, one could define the local SDF via encoded nodes, similar to Implicit Moving-Least Squares [12]. In particular, using a smooth local approximation (surrogate SDFs) may allow coarser tetrahedralization in geometrically smooth regions, thereby reducing both memory and computational cost during the broadphase. Second, the current Face–TES collision detection generates contact features for all overlapping face–tetrahedron pairs, potentially resulting in more contact constraints than Face–SDF methods. Although this can be effectively handled with GPU-accelerated dynamics solvers (e.g., [41, 19]), this can be excessive when coupled with second-order solvers (e.g., Newton-type methods [43, 3]) or sequential update methods (e.g., projected Gauss–Seidel [28]). A possible solution is to integrate TES with various contact clustering methods, such as K-means clustering [36] or data-driven methods [21]. Third, spatial discretization schemes with other primitives, such as the hexahedron [42], can be used in combination with a localized distance evaluation method per primitive along with warp-wise parallelism. Finally, integrating TES into soft-body simulation is a promising direction.

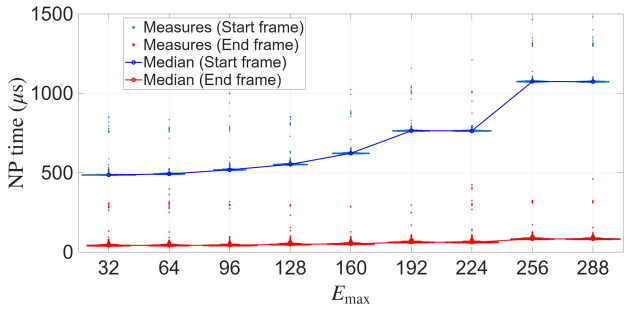


Fig. 9: Time cost benchmark for Face-TES narrow-phase with 'Pile of shafts' scenario under varying truncation settings. We conduct 3k measurements per truncation setting and report the average.

IX. Conclusion

In this paper, we propose TES, a memory-efficient and accurate SDF representation based on tetrahedralized spatial discretization and localized exact distance evaluation per tetrahedron. We implement both GPU-accelerated construction of TES and a collision detection framework based on this novel representation. We demonstrate the effectiveness of this framework on challenging collision-detection scenarios. We further show that this representation can handle runtime geometry deformation with minimal additional cost, demonstrating that TES is also suitable for deformable bodies.

Acknowledgment

Harim Ji and Dongjun Lee 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. Yongseok Lee is with the Department of Robotics and Mechatronics, Daegu Gyeongbuk Institute of Science & Technology (DGIST), Daegu, Republic of Korea. This work was supported by the Korea Planning & Evaluation Institute of Industrial Technology (KEIT) grant funded by the Ministry of Trade, Industry and Resources (MOTIR) (No. RS-2024-00441872, No. RS-2024-00419641, No. RS-2025-25455303).

References

[1] J Bender, C Duriez, F Jaillet, and G Zachmann. Continuous collision detection between points and signed distance fields. In *Workshop on virtual reality interaction and physical simulation*, volume 8, 2014. URL https://viterbi-web.usc.edu/~jbarbic/ccd/XuBarbicVRIPHYS2014.pdf.

[2] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Neural implicit surfaces for efficient and accurate collisions in physically based simulations, 2021. URL https://arxiv.org/abs/2110.01614. arXiv:2110.01614.

[3] Alejandro M. Castro, Frank N. Permenter, and Xuchen Han. An unconstrained convex formulation of compliant contact. *Trans. Rob.*, 39(2):1301–1320, April 2023. ISSN 1552-3098. doi: 10.1109/TRO.2022.3209077. URL https://doi.org/10.1109/TRO.2022.3209077.

[4] Rohan Chabra, Jan E. Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 608–625, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58526-6.

[5] Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. Vertex block descent. 43(4), July 2024. ISSN 0730-0301. doi: 10.1145/3658179. URL https://doi.org/10.1145/3658179.

[6] Epic Games Corporation. Unreal engine documentation. https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-7-documentation, 2026. Accessed: 2026-01-07.

[7] Nvidia Corporation. Using shared memory in cuda c/c++. https://developer.nvidia.com/blog/using-shared-memory-cuda-cc/, 2013. Accessed: 2026-05-06.

[8] Nvidia Corporation. Cuda c++ programming guide. https://docs.nvidia.com/cuda/cuda-programming-guide/02-basics/writing-cuda-kernels.html, 2026. Accessed: 2026-01-10.

[9] Nvidia Corporation. Cuda c++ programming guide. https://docs.nvidia.com/cuda/cuda-programming-guide/03-advanced/advanced-kernel-programming.html#advanced-kernels-hardware-implementation-simt-architecture, 2026. Accessed: 2026-01-10.

[10] NVIDIA Corporation. Colliders — create and destroy a static collider. https://docs.omniverse.nvidia.com/kit/docs/omni_physics/latest/dev_guide/rigid_bodies_articulations/collision.html#createsdfcollider, 2026. URL https://docs.omniverse.nvidia.com/kit/docs/omni_physics/latest/dev_guide/rigid_bodies_articulations/collision.html#createsdfcollider. Accessed: 2026-01-08.

[11] Google DeepMind. Computation — mujoco documentation. https://mujoco.readthedocs.io/en/stable/computation/index.html, 2026. URL https://mujoco.readthedocs.io/en/stable/computation/index.html. Accessed: 2026-01-08.

[12] Y. Du, Y. Li, S. Coros, and B. Thomaszewski. Robust and artefact-free deformable contact with smooth surface representations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '24, page 1–13, Goslar, DEU, 2024. Eurographics Association. doi: 10.1111/cgf.15187. URL https://doi.org/10.1111/cgf.15187.

[13] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 249–254, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1581132085. doi: 10.1145/344779.344899. URL https://doi.org/10.1145/344779.344899.

[14] Arnulph Fuhrmann, Gerrit Sobotka, and Clemens Groß. Distance fields for rapid collision detection in physically based modeling. In *Proceedings of GraphiCon*, volume 2003, pages 58–65. Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences . . . , 2003.

[15] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. doi: https://doi.org/10.1002/nme.2579. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579.

[16] 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*, 4(2):193–203, 1988. doi: 10.1109/56.2083. URL https://ieeexplore.ieee.org/abstract/document/2083.

[17] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, July 2003. ISSN 0730-0301. doi: 10.1145/882262.882358. URL https://doi.org/10.1145/882262.882358.

[18] Jian Huang, Yan Li, Roger Crawfis, Shao Chiung Lu, and Shuh Yuan Liou. A complete distance field representation. In *Proceedings of the Conference on Visualization '01*, VIS '01, page 247–254, USA, 2001. IEEE Computer Society. ISBN 078037200X.

[19] Harim Ji, Hyunsu Kim, Jeongmin Lee, Somang Lee, Seoki An, Jinuk Heo, Youngseon Lee, Yongseok Lee, and Dongjun Lee. Gpu-accelerated subsystem-based admm for large-scale interactive simulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13868–13874, 2025. doi: 10.1109/ICRA55743.2025.11128665. URL https://ieeexplore.ieee.org/document/11128665.

[20] Tero Karras. 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*, EGGH-HPG'12, page 33–37, Goslar, DEU, 2012. Eurographics Association. ISBN 9783905674415. URL https://diglib.eg.org/bitstream/handle/10.2312/EGGH.HPG12.033-037/033-037.pdf?sequence=1.

[21] Myungsin Kim, Jaemin Yoon, Dongwon Son, and Dongjun Lee. Data-driven contact clustering for robot simulation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8278–8284, 2019. doi: 10.1109/ICRA.2019.8793938. URL https://ieeexplore.ieee.org/document/8793938.

[22] Dan Koschier, Crispin Deul, Magnus Brand, and Jan Bender. An hp-adaptive discretization algorithm for signed distance field generation. *IEEE Transactions on Visualization and Computer Graphics*, 23(10):2208–2221, 2017. doi: 10.1109/TVCG.2017.2730202. URL https://ieeexplore.ieee.org/abstract/document/7987773.

[23] Lei Lan, Ran Luo, Marco Fratarcangeli, Weiwei Xu, Huamin Wang, Xiaohu Guo, Junfeng Yao, and Yin Yang. Medial elastics: Efficient and collision-ready deformation via medial axis transform. *ACM Trans. Graph.*, 39(3), April 2020. ISSN 0730-0301. doi: 10.1145/3384515. URL https://doi.org/10.1145/3384515.

[24] Minji Lee, Jeongmin Lee, Jaemin Yoon, and Dongjun Lee. Real-time physically-accurate simulation of robotic snap connection process. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5173–5180, 2021. doi: 10.1109/IROS51168.2021.9636246. URL https://ieeexplore.ieee.org/document/9636246.

[25] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. Incremental potential contact: intersection- and inversion-free, large-deformation dynamics. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. doi: 10.

1145/3386569.3392425. URL https://doi.org/10.1145/3386569.3392425.

[26] Fuchang Liu and Young J. Kim. Exact and adaptive signed distance fieldscomputation for rigid and deformablemodels on gpus. 20(5):714–725, May 2014. ISSN 1077-2626. doi: 10.1109/TVCG.2013.268. URL https://doi.org/10.1109/TVCG.2013.268.

[27] Pengfei Liu, Yuqing Zhang, He Wang, Milo K. Yip, Elvis S. Liu, and Xiaogang Jin. Real-time collision detection between general sdfs. *Computer Aided Geometric Design*, 111: 102305, 2024. ISSN 0167-8396. doi: https://doi.org/10.1016/j.cagd.2024.102305. URL https://www.sciencedirect.com/science/article/pii/S0167839624000396.

[28] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. MIG '16, page 49–54, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345927. doi: 10.1145/2994258.2994272. URL https://doi.org/10.1145/2994258.2994272.

[29] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Zach Corse. Local optimization for robust signed distance field collision. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(1), May 2020. doi: 10.1145/3384538. URL https://doi.org/10.1145/3384538.

[30] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. Efficient elasticity for character skinning with contact and collisions. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450309431. doi: 10.1145/1964921.1964932. URL https://doi.org/10.1145/1964921.1964932.

[31] Nathan Mitchell, Mridul Aanjaneya, Rajsekhar Setaluri, and Eftychios Sifakis. Non-manifold level sets: a multivalued implicit surface representation with applications to self-collision processing. *ACM Trans. Graph.*, 34(6), November 2015. ISSN 0730-0301. doi: 10.1145/2816795.2818100. URL https://doi.org/10.1145/2816795.2818100.

[32] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.*, 32(3), July 2013. ISSN 0730-0301. doi: 10.1145/2487228.2487235. URL https://doi.org/10.1145/2487228.2487235.

[33] Ken Museth. Nanovdb: A gpu-friendly and portable vdb data structure for real-time rendering and simulation. In *ACM SIGGRAPH 2021 Talks*, pages 1–2. 2021.

[34] Yashraj Narang, Kier Storey, Iretiayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, Ankur Handa, and Dieter Fox. Factory: Fast contact for robotic assembly. In *Robotics: Science and Systems*, 2022.

[35] Chong Jin Ong and E.G. Gilbert. Growth distances: new measures for object separation and penetration. *IEEE Transactions on Robotics and Automation*, 12(6):888–903, 1996. doi: 10.1109/70.544772. URL https://ieeexplore.ieee.org/abstract/document/544772.

[36] M.A. Otaduy and M.C. Lin. A modular haptic rendering algorithm for stable and transparent 6-dof manipulation. *IEEE Transactions on Robotics*, 22(4):751–762, 2006. doi: 10.1109/TRO.2006.876897. URL https://ieeexplore.ieee.org/document/1668258.

[37] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. doi: 10.1109/CVPR.2019.00025. URL https://ieeexplore.ieee.org/document/8954065.

[38] Joël Pelletier-Guénette, Alexandre Mercier-Aubin, and Sheldon Andrews. Real-time triangle-sdf continuous collision detection. *Proc. ACM Comput. Graph. Interact. Tech.*, 8(4), August 2025. doi: 10.1145/3747862. URL https://doi.org/10.1145/3747862.

[39] Yuri S. Sarkisov, Min Jun Kim, Davide Bicego, Dzmitry Tsetserukou, Christian Ott, Antonio Franchi, and Konstantin Kondak. Development of sam: cable-suspended aerial manipulator. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5323–5329, 2019. doi: 10.1109/ICRA.2019.8793592. URL https://ieeexplore.ieee.org/document/8793592.

[40] Hang Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2), February 2015. ISSN 0098-3500. doi: 10.1145/2629697. URL https://doi.org/10.1145/2629697.

[41] Alessandro Tasora, Dan Negrut, and Mihai Anitescu. *GPU-Based Parallel Computing for the Simulation of Complex Multibody Systems with Unilateral and Bilateral Constraints: An Overview*, pages 283–307. Springer Netherlands, Dordrecht, 2011. ISBN 978-90-481-9971-6. doi: 10.1007/978-90-481-9971-6_14. URL https://doi.org/10.1007/978-90-481-9971-6_14.

[42] Timothy J. Tautges. The generation of hexahedral meshes for assembly geometry: survey and progress. *International Journal for Numerical Methods in Engineering*, 50(12):2617–2642, 2001. doi: https://doi.org/10.1002/nme.139. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.139.

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109. URL https://ieeexplore.ieee.org/document/6386109.

[44] Hongyi Xu, Yili Zhao, and Jernej Barbič. Implicit multibody penalty-baseddistributed contact. *IEEE Transactions on Visualization and Computer Graphics*, 20(9):1266–1279, 2014. doi: 10.1109/TVCG.2014.2312013. URL https://ieeexplore.ieee.org/document/6767148.

[45] Ehsan Zobeidi and Nikolay Atanasov. A deep signed directional distance function for shape representation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2689–2695, 2024. doi: 10.1109/IROS58592.2024.10801713. URL https://ieeexplore.ieee.org/abstract/document/10801713.