Tetrahedron-Tetrahedron Intersection and Volume Computation Using Neural Networks

Erendiro Pedro[1], Hermann Meißenhelter[2] and Gabriel Zachmann[2]

[1]Institute of Engineering, Polytechnic of Porto, Portugal
[2]University of Bremen, Germany

Abstract

Narrow-phase collision detection is a critical bottleneck in physics-based simulation, traditionally relying on exact but sequential CPU-bound algorithms that struggle to exploit massive GPU parallelism. In this work, we reframe the tetrahedron-tetrahedron intersection as a learned inference problem. We leverage principles of hierarchical permutation invariance—derived from DeepSets and PointNet—to construct a neural architecture that jointly predicts intersection status and volumetric overlap directly from vertex coordinates. We generate a large dataset of 12 million pairs covering five distinct tetrahedron pair configurations, ensuring the model learns robust geometric decision boundaries. Our most accurate model achieves 99.3% mean classification accuracy across five contact configurations and predicts overlap volume with MAE $\approx 2.88 \times 10^{-3}$ *(for positive volumes) while remaining entirely GPU-resident. On consumer-grade GPU, our pipeline outperforms the exact CGAL library by 93× in detection speed and over 16,821× when volume computation is included. This "volume-for-free" paradigm offers a transformative trade-off for real-time applications where approximate is acceptable but ultra-fast geometric reasoning is paramount.*

CCS Concepts

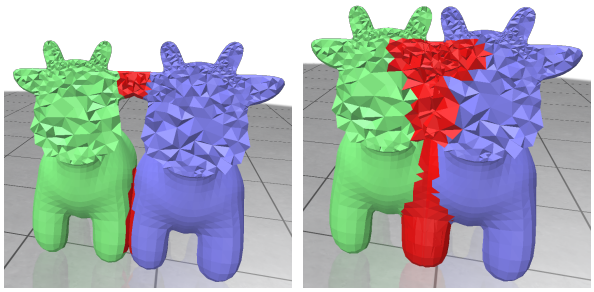
• ***Computing methodologies*** → ***Collision detection; Neural networks;***

1. Introduction and Related Work

Narrow-phase collision detection remains a critical bottleneck in physics simulation, particularly for tetrahedral meshes in finite element methods, where millions of intersection checks occur per frame. Traditional exact algorithms (GJK [GJK88], EPA [vdB01], SAT [Eri05,GPR02]) rely on sequential branching logic that creates severe thread divergence on modern GPUs, underutilizing available parallelism. Fisher and Lin [FL01] characterize *intersection volume* as the "most complicated yet accurate method" to define intersection extent.

To address this scalability gap, we reframe tetrahedron-tetrahedron intersection as a learned inference problem: given vertex coordinates $\mathcal{T}_1, \mathcal{T}_2 \subset \mathbb{R}^3$, we learn a mapping $f : \mathbb{R}^{24} \to \{0,1\} \times [0, V_{\max}]$ to predict the intersection status $y \in \{0,1\}$ and the volumetric overlap $v \in [0, V_{\max}]$. The existence of exact geometric algorithms establishes a deterministic ground-truth labeling function, yet approximating such predicates via neural inference has received little direct attention in the literature.

Closest to our setting, Önal and Zafeirakopoulos [ÖZ19] pioneered the application of machine learning to computationally hard geometric problems, specifically e stimating p olytope volumes. Their best model achieved an accuracy of 96.31% and an $R^2 = 0.998$ for volume regression using fixed-size encodings. While tetrahedron intersection has received limited attention,



(a) Shallow intersection (b) Deep intersection

Figure 1: ***Mesh intersection detection using TetrahedronPairNet****: Two tetrahedralized mesh objects at different penetration depths. Our method analyzes relative tetrahedron pair configurations and predicts their intersection status: red indicates intersecting tetrahedra, while green/blue indicate non-intersecting regions. (a) Shallow intersection with limited penetration. (b) Deeper intersection with a larger interpenetration volume.*

DOI: 10.2312/egs.20261016

point cloud processing offers relevant architectural insights. PointNet [QSMG17] pioneered permutation-invariant learning through shared MLPs and symmetric aggregation (max pooling), processing sets directly from coordinates. In parallel Zaheer et al [ZKR*18], proved that any continuous permutation-invariant function can be decomposed as $\rho(\sum_{x\in X}\phi(x))$, where ϕ maps elements into a latent space and ρ transforms the aggregated set representation.

To the best of our knowledge, this work presents the first end-to-end framework for joint tetrahedron intersection and overlap volume prediction via GPU-accelerated inference. By pairing standard deep learning architectures with rigorous synthetic data, we surpass classical exact geometric predicates in raw throughput via three advantages: (i) SIMD-Optimized Execution using branch-free tensor operations to eliminate warp divergence; (ii) Zero-Overhead Volume Estimation computed as a latent byproduct of detection; and (iii) Hard Deterministic Latency providing a predictable computational floor regardless of geometry or scene density.

We summarize our contributions below:

- A neural architecture for joint tetrahedron intersection classification and intersection volume regression.
- Five data generation strategies corresponding to distinct tetrahedron pair configurations: no–intersection, vertex–face, edge–face, face–face, and volumetric interpenetration.
- Evaluation showing 99.3% mean accuracy over the 5 strategies, with speedups of $> 93\times$ for intersection detection and $> 16{,}821\times$ for volume computation compared to exact sequential CPU predicates.

2. Dataset Generation

Each tetrahedron is i.i.d generated uniformly within the unit cube $[0,1]^3$, deriving ground truth labels via exact geometric predicates from CGAL [The25]. Binary intersection status is determined using `CGAL::do_intersect`. For volume, we employ `CGAL::Nef_polyhedron_3` to perform robust boolean intersection operations. The resulting Nef structure is then converted into a closed polyhedron, and the exact volume is computed via `CGAL::Polygon_mesh_processing::volume`.

Figure 2: ***Data Generation Strategies****: visualizations of randomly generated .obj tetrahedron–tetrahedron pairs corresponding to five topological configurations. (1) non-intersecting pairs (N-I), (2, 3, and 4) show contact configurations (V-F, E-F, F-F) sampled via constraints, and (8) volumetric interpenetration (Vol.).*

2.1. Strategies for Generating Training Data

These strategies, illustrated in Figure 2, are detailed further in the Supplementary Material:

1. **No Intersection:** Pairs $\mathcal{T}_1, \mathcal{T}_2 \sim \mathcal{U}([0,1]^3)$ are generated via rejection sampling, using `CGAL::do_intersect` to filter candidates and guarantee strictly disjoint volumes.
2. **Vertex–Face:** $\mathcal{T}_2$ is anchored to a point p on a face F of $\mathcal{T}_1$. To prevent penetration, the three non-contact vertices are spherically sampled strictly within the outward half-space defined by F's normal. An angular buffer ($\epsilon = 10^{-16}$) prevents precision-based intersections near the boundary.
3. **Edge–Face:** Two vertices of $\mathcal{T}_2$ are projected onto the supporting plane of a face of $\mathcal{T}_1$, aligning an edge. To maintain disjointness, the remaining two vertices are spherically sampled strictly within the outward hemisphere, utilizing the $\epsilon = 10^{-16}$ angular buffer for numerical stability.
4. **Face–Face:** Three vertices of $\mathcal{T}_2$ are projected onto the supporting plane of a face of $\mathcal{T}_1$. The fourth vertex is spherically sampled relative to their centroid strictly within the outward hemisphere (buffered by $\epsilon = 10^{-16}$). Finally, `CGAL::do_intersect` ensures the coplanar triangle actually overlaps the target face.
5. **Volumetric Intersection:** Candidates $\mathcal{T}_1, \mathcal{T}_2 \sim \mathcal{U}([0,1]^3)$ are repeatedly generated via rejection sampling until `CGAL::do_intersect` explicitly confirms an intersection.

2.2. Distribution and Volume Constraints

To avoid trivial zero-volume heuristics caused by the disjoint-dominant nature of naive sampling, we enforce the constraints summarized in Table 1.

Table 1: ***Dataset Constraints****: Stratification and normalization strategies used to balance the data distribution and stabilize training.*

Constraint	Implementation & Rationale
Class Stratification	**50% Disjoint** ($y=0$) and **50% Intersecting** ($y=1$), split as V–F (5%), E–F (7%), F–F (20%), and Volumetric (18%) to accurately capture decision boundaries.
Volume Norm.	Intersections are log-uniformly sampled $v \in [10^{-7}, 20^{-2}]$. A global scale $\lambda = 1000$ shifts targets to $\approx [10^{-4}, 20]$ to prevent vanishing gradients.

2.3. Geometric Transformations for Input Normalization

During both training and inference, we reduce input dimensionality by applying canonicalizing geometric transformations (Table 2; details in Supp. Material).

3. TetrahedronPairNet

Figure 1 demonstrates the method's capability to detect intersections at varying penetration depths between two tetrahedralized mesh objects. TetrahedronPairNet (Figure 3) implements a symmetric architecture grounded in PointNet [QSMG17] and

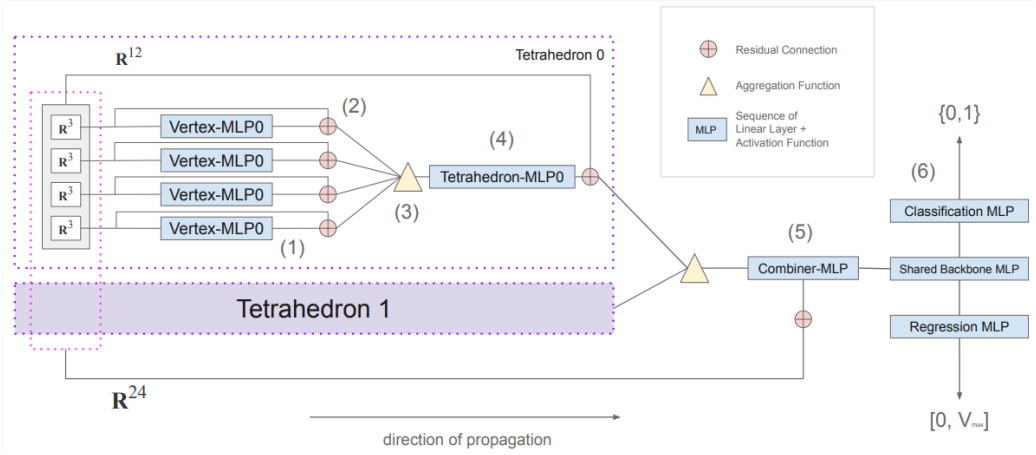


Figure 3: ***Schematic of TetrahedronPairNet.*** *A permutation-invariant, DeepSets-based architecture. (1–3)* ***Vertex Level****: Each 3D vertex is processed by a shared Vertex-MLP with a local residual skip, then aggregated. (4)* ***Tetrahedron Level****: Features are refined symmetrically by a shared Tetrahedron-MLP with a local 12D residual from each tetrahedron. (5) Both tetrahedron embeddings are pooled into a Combiner-MLP, which integrates a global 24D residual from the raw input. (6) A Shared Backbone MLP feeds dual heads: classification (intersection status) and regression (overlap volume).*

Table 2: ***Input Normalization Strategies****: Geometric transformations used to project tetrahedron pairs into a canonical frame.*

Strategy	Mechanism & Properties
Principal Axis Alignment (PCA)	**Mechanism**: Applies a rigid rotation derived from the PCA of $\mathcal{T}_1$ to both tetrahedra, aligning $\mathcal{T}_1$'s principal axes with the global basis. **Properties**: Standardizes orientation while strictly preserving distances, scale, and volume.
Unitary Tetrahedron Alignment (Unit)	**Mechanism**: Applies an affine transformation that maps $\mathcal{T}_1$ to the unit simplex $\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\}$ and expresses $\mathcal{T}_2$ in this new frame. **Properties**: Preserves boolean intersection status; intersection volume is scaled linearly by the determinant of the affine map.

DeepSets [ZKR*18] to satisfy the requirement of permutation invariance for tetrahedral vertex sets, skip connections are integrated to ensure robust gradient propagation and enable high-capacity feature extraction [ZTHR22, MQY*22]. To minimize computational overhead and maximize raw throughput, we deliberately omit normalization layers, though these remain optional for specific convergence needs.

4. Experimental Results

We established a CGAL baseline using exact predicates and evaluate a `TetrahedronPairNet` with the parameters presented in Table 3. Performance was measured on an Intel Core i5-1035G1 for sequential CPU execution and an NVIDIA GeForce MX330 (1.95 GB) for GPU inference.

Table 4 presents the performance profile.

Training Configuration		Model Architecture	
Training Samples	12,000,000	**Precision**	float32
Optimizer	AdamW	**Total Parameters**	73,370
Learning Rate	10^{-3}	**Model Size**	633 KB
Batch Size	2048	**Activation Function**	ReLU
Epochs	50	**Aggregation Function**	Max-Pooling
Loss Function	0.5 BCE + 0.5 RMSLE	**Vertex MLP**	[24, 24]
Class Balance	0.5 positive/negative	**Tetrahedron MLP**	[48, 48]
Input Normalization	PCA + Unit	**Combiner MLP**	[96, 96]
Data Distribution	[50%, 5%, 7%, 20%, 18%]	**Shared Backbone**	[1024]
Volume Scaling	10^3	**Classification Head**	[128, 64, 32, 1]
		Regression Head	[128, 128, 1]

Table 3: *TetrahedronPairNet architecture and training hyperparameters. The model uses a shared feature backbone for simultaneous intersection classification and volume regression. MLP blocks consist of sequential linear layers followed by the specified activation; dimensions denote hidden unit counts; list length denotes the number of layers.*

5. Limitations

While promising, our approach assumes a fixed data distribution; thus, scene-specific distributional shifts may reduce accuracy and require fine-tuning. Additionally, three contact configurations are currently missing from the training data: vertex–vertex, vertex–edge, and edge–edge. However, we detail possible generation strategies for these cases in the supplementary material. Finally, compared to our classification results, the volume regression accuracy remains less reliable, especially for deep penetrations. Therefore, for accuracy-sensitive simulations, we position TetrahedronPairNet as a fast, GPU-resident pre-pass to filter pairs and provide low-cost volume estimates. Exact predicates can then be selectively applied to predicted intersections, low-confidence outputs, or large penetrations.

Test Subset / Metric	Detect Acc.	Vol. MAE	Notes
Degenerate & Non-Intersecting Cases			
No Intersection	98.02%	1.67×10^{-7}	Baseline negative class
Vertex–Face	99.07%	4.27×10^{-9}	Correctly predicts ≈ 0 volume
Edge–Face	99.97%	4.72×10^{-9}	Correctly predicts ≈ 0 volume
Face–Face	99.92%	1.65×10^{-9}	Correctly predicts ≈ 0 volume
Volumetric Intersection Analysis			
Volumetric (Overall)	**99.07%**	$\mathbf{2.88 \times 10^{-3}}$	**Volumetric collision case**
Small ($v < 0.0067$)	—	1.47×10^{-3}	88% Bin Acc. (34% of samples)
Medium ($0.0067 \leq v < 0.0133$)	—	2.80×10^{-3}	60% Bin Acc. (33% of samples)
Large ($0.0133 \leq v < 0.0200$)	—	4.44×10^{-3}	41% Bin Acc. (33% of samples)
Global Averages	**99.37%**	—	Aggregated across 500k samples
Computational Performance (Average of 30 runs)			
Method	**Throughput**	**Latency (Pre. + Inf.)**	**Speedup (Detect / Vol)**
CGAL (Detection, CPU)	6,352 samples/s	157μs	Baseline
CGAL (+ Volume, CPU)	35 samples/s	28.31 ms	Baseline
TetrahedronPairNet (CPU, BS=2048)	271,313 samples/s	3.69μs (1.80 + 1.89)	43× / 7,751×
TetrahedronPairNet (GPU, BS=2048)	588,744 samples/s	1.70μs (1.11 + 0.59)	93× / 16,821×

Table 4: *Performance benchmarks (500k samples; 100k/class). High accuracy at contact thresholds demonstrates robust identification of topological transitions. Regression error scales with intersection magnitude; predictive accuracy is highest for small overlaps—critical for collision onset—but degrades at high penetration. BS: Batch Size; FP32: Float32; MAE: Mean Absolute Error on a unit cube scale ($V_{cube} = 1.0$).*

6. Conclusion

TetrahedronPairNet achieves 99.3% mean accuracy and a volume estimation MAE of $\approx 2.88 \times 10^{-3}$, delivering a **93**× speedup for intersection detection and a **16,821**× speedup for volume estimation over exact CGAL predicates. By reformulating intersection as learned inference, we trade absolute exactness for massive parallel throughput, enabling high-speed narrow-phase filtering and providing differentiable collision signals for optimization. While our baseline relies on minimal inductive biases, integrating richer geometric priors is expected to further improve the accuracy–throughput frontier. Ultimately, this work demonstrates that the perceived sequentiality of geometric kernels is often an artifact of traditional design, opening the door to massively parallel, learned geometric algorithms.

All code and artifacts are open source and available in the supplementary material.

References

[Eri05] ERICSON C.: *Real-Time Collision Detection*. Morgan Kaufmann, 2005. 1

[FL01] FISHER S., LIN M.: Fast penetration depth estimation for elastic bodies using deformed distance fields. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)* (2001), vol. 1, pp. 330–336 vol.1. 1

[GJK88] GILBERT E. G., JOHNSON D. W., KEERTHI S. S.: A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation 4*, 2 (1988), 193–203. 1

[GPR02] GANOVELLI F., PONCHIO F., ROCCHINI C.: Fast tetrahedron-tetrahedron overlap algorithm. *Journal of Graphics Tools 7*, 2 (2002), 17–25. doi:10.1080/10867651.2002.10487557. 1

[MQY*22] MA X., QIN C., YOU H., RAN H., FU Y.: Rethinking network design and local geometry in point cloud: A simple residual mlp framework, 2022. arXiv:2202.07123. 3

[ÖZ19] ÖNAL U., ZAFEIRAKOPOULOS Z.: A machine learning framework for volume prediction. In *Analysis of Experimental Algorithms* (Cham, 2019), Kotsireas I., Pardalos P., Parsopoulos K. E., Souravlias D., Tsokas A., (Eds.), Springer International Publishing, pp. 408–423. 1

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2

[The25] THE CGAL PROJECT: *CGAL User and Reference Manual*, 6.1 ed. CGAL Editorial Board, 2025. 2

[vdB01] VAN DEN BERGEN G.: Proximity queries and penetration depth computation on 3d game objects. In *Proceedings of Game Developers Conference* (2001). 1

[ZKR*18] ZAHEER M., KOTTUR S., RAVANBAKHSH S., POCZOS B., SALAKHUTDINOV R., SMOLA A.: Deep sets, 2018. 2, 3

[ZTHR22] ZHANG L., TOZZO V., HIGGINS J., RANGANATH R.: Set norm and equivariant skip connections: Putting the deep in deep sets. In *Proceedings of the 39th International Conference on Machine Learning* (17–23 Jul 2022), vol. 162 of *Proceedings of Machine Learning Research*, PMLR, pp. 26559–26574. 3