AutoHood3D: A Multi-Modal Benchmark for Automotive Hood Design and Fluid–Structure Interaction

Vansh Sharma[*]
University of Michigan
Ann Arbor, MI, USA

Harish Jai Ganesh
University of Michigan
Ann Arbor, MI, USA

Maryam Akram
Ford Research and Innovation Center
Dearborn, MI, USA

Wanjiao Liu
Ford Research and Innovation Center
Dearborn, MI, USA

Venkat Raman
University of Michigan
Ann Arbor, MI, USA

Abstract

This study presents a new high-fidelity multi-modal dataset containing 16000+ geometric variants of automotive hoods useful for machine learning (ML) applications such as engineering component design and process optimization, and multiphysics system surrogates. The dataset is centered on a practical multiphysics problem—hood deformation from fluid entrapment and inertial loading during rotary-dip painting. Each hood is numerically modeled with a coupled Large-Eddy Simulation (LES)-Finite Element Analysis (FEA), using 1.2M cells in total to ensure spatial and temporal accuracy. The dataset provides time-resolved physical fields, along with STL meshes and structured natural language prompts for text-to-geometry synthesis. Existing datasets are either confined to 2D cases, exhibit limited geometric variations, or lack the multi-modal annotations and data structures—shortcomings we address with AutoHood3D. We validate our numerical methodology, establish quantitative baselines across five neural architectures, and demonstrate systematic surrogate errors in displacement and force predictions. These findings motivate the design of novel approaches and multiphysics loss functions that enforce fluid–solid coupling during model training. By providing fully reproducible workflows, AutoHood3D enables physics-aware ML development, accelerates generative-design iteration, and facilitates the creation of new FSI benchmarks. Dataset and code URLs in D.1.

1 Introduction

Classical numerical solvers [1, 2] provide accurate predictions for practical systems such as propulsion systems [3, 4, 5] and turbine performance [6, 7, 8]; however, they are computationally expensive and often intractable for iterative design processes [9]. The latest advances in computational fluid dynamics (CFD) now incorporate ML techniques to develop turbulence closure models [10, 11], modify wall-modeled LES [12], and for real-time flow control strategies [13, 14]. Traditional ML approaches have improved both predictive accuracy [15, 16] and time-to-solution across a variety of CFD applications [17, 18]. Despite these advances, iterative design workflows rely predominantly on design-of-experiments methodologies integrated with CFD solvers to drive large-scale optimization studies.

[*]Corresponding Author: vanshs@umich.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

In recent years, several comprehensive aerodynamic datasets [19, 20] have been published, covering everything from 2D airfoil profiles [21] to full 3D vehicle exteriors [22] and complete car geometries [23]. These large-scale collections serve to bridge the gap between data-driven surrogate models and high-fidelity CFD solvers. However, most publicly available repositories are RANS based, confined to 2D configurations, or lack both a sufficient sample size and a transparent generation workflow to support in-depth design exploration. Moreover, real-world applications frequently involve additional physics, such as multiphase flows or coupled fluid-structure interactions, that these datasets do not capture. To date, no resource offers a reproducible, end-to-end pipeline alongside a large-scale, multiphysics 3D dataset.

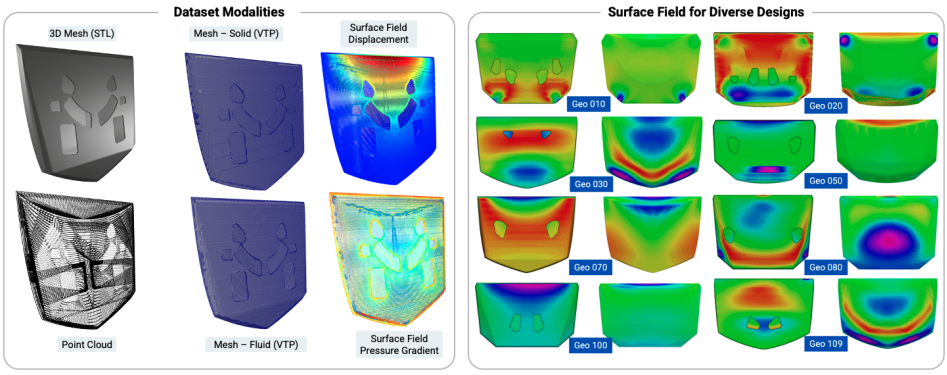

Figure 1: Dataset modalities (left): base 3D STL hood shell, surface point-cloud sampling, raw fluid and solid meshes, and sample surface fields (normalized displacement and pressure-gradient) extracted on the hood surface. Example variants (right): selected hood geometries with interpolated deformation fields mapped onto the STL, demonstrating how different cutout topologies yield distinct deflection patterns on both the front and rear faces.

We present the AutoHood3D dataset (shown in Figure 1) to overcome the aforementioned gaps. In particular, the dataset poses a unique challenge for current data-driven methods due to its multiphysics modality. The multiphysics challenge at the core of this study is the deformation of automotive hoods during the rotary dip paint process [24, 25]. The automotive hood consists of two principal elements: an exterior panel designed for aerodynamic flow management and aesthetic styling, and an interior reinforcement, typically a hollow honeycomb lattice that accommodates fluid lines and provides thermal insulation while resisting static and dynamic deflection. Aerodynamic efficiency remains a key design factor, but pedestrian safety regulations [26, 27] and utility requirements impose orthogonal constraints on the overall hood design. However, during the rotary dip paint process, rapid rotation of the carrier arm entrains residual coating fluid within the engineered cutouts and recesses, subjecting the shell to transient body forces that, combined with inertial loading, produce localized deflections [24]. Because these manufacturing-induced loads are rarely incorporated into early-stage design analyses, unexpected deformations occur in thin-walled, geometrically complex regions. This behavior is analogous to industrial occurrences, such as, distortion of composite panels from immersion coatings, and finds a conceptual parallel in adaptive warping wing designs [28].

The dataset comprises over 16,000 unique 3D hood geometries with time-resolved FSI solution snapshots, provided in multiple modalities (see Figure 1). All geometry generation scripts, FSI simulation workflows, and analysis tools are released under an open-source license and readily available to the developer community. We expect these modular workflows to accelerate the creation of new large-scale 3D FSI benchmarks, enabling temporal autoregressive modeling and physics-informed [29] PDE learning applications. In an overarching way, our contributions can be summarized as follows:

- Introduce the first open-source dataset of more than 16,000 3D automotive hood geometries with engineered cutouts, providing extensive design diversity for fluid dynamics and structural studies.
- Provide a fully reproducible, end-to-end workflow, from STL generation to FSI co-simulation, designed for scalable synthesis of design variants of new components, and extensible to diverse engineering contexts.

- Deliver a truly multimodal benchmark, comprising high-fidelity STL meshes, raw CFD solutions with time-resolved flow and surface fields, FEA displacement and stress outputs, surface point clouds, and tokenized natural language annotations, accompanied by performance benchmarks for five neural architectures.

2 Related Work

In scientific-ML (SciML) domain, the integration of AI into CFD to accelerate high-fidelity simulations has become a prominent research thrust. Large-scale repositories, such as PDEBench [30], PINNacle [31], Airfrans [21], and BubbleML [32], provide comprehensive datasets for PDE learning, airfoil optimization, and multiphase flow modeling. Beyond application-specific datasets, cross-domain databases such as BLASTNet [19] and ThePDEWell [20] aggregate varied PDE system results and spatio-temporal solution fields, enabling systematic benchmarking and transferability studies across scientific disciplines. In the industrial domain, large-scale datasets such as DrivAerNet++ [22] and WindsorML [23] have emerged, presenting realistic scenarios that enable engineers to predict surface pressure distributions, volumetric flow characteristics, and the resulting forces and moments on novel geometries under varied operating conditions. Despite their individual strengths, existing datasets exhibit notable gaps: some provide extensive geometric and solution data but lack generative workflows for creating new variants [31, 20, 19]; others deliver end-to-end pipelines, yet remain confined to 2D cases with only limited extensions to three dimensions [21, 32]; and a few support full 3D flow simulations but offer limited design diversity due to computational costs [23]. Consequently, no public dataset unifies broad design variations, reproducible generation processes, and a comprehensive 3D FSI benchmark with multiple modalities such as point cloud and STL.

In parallel, recent work [33] has shown that point clouds serve as a versatile representation for physical world and scene reconstruction [34] across applications such as virtual reality, object recognition, autonomous navigation [35], and robotic manipulation. For example, PointLLM [36] and related models [37] use joint text–point-cloud embeddings [38] to generate 3D geometries from natural language prompts [39]. Despite these advances [40, 41], the development of text-grounded CAD point cloud models is hindered by the absence of large-scale, diverse 3D engineering datasets with detailed natural language annotations. Furthermore, point cloud data are inherently sparse and typically lack semantic context, limiting their utility for comprehensive scene understanding [42, 33]. To bridge these domains, we augment our FSI dataset with paired natural language annotations and point cloud outputs, creating a text-to-geometry corpus for supervised fine-tuning of LLMs in CAD-driven generative-AI workflows.

3 Dataset Workflow

The dataset generation workflow is organized into three broad phases. In the first phase we process the inner–hood designs from [43] to reconstruct complete hood shell and segment cutouts as ordered point clouds. In the second phase focused on design-variation synthesis, we generate new hood variants by embedding the previously extracted cutouts chosen under user-defined spacing and plane-separation constraints. Finally, in the third phase, consisting of the FSI simulation pipeline, we automate mesh generation and solver initialization to perform fluid–structure analyses at scale. Detailed methodologies are as follows.

3.1 Base Geometry Generation

We adopt 100 inner-only hood geometries from [43] as our base set. To obtain a complete dual-shell model, each inner-hood design is processed with a convex-hull algorithm [44] to produce an outer envelope surface. The resulting hull is then translated and rotated so that the inner shell is centered at the global origin and aligned approximately perpendicular to the incoming flow direction. Finally, a uniform thickness of 10 mm is extruded from the hull to form the full outer and inner hood shell. To generate a diverse set of engineering cutouts on these base shells, we draw on a collection of approximately 250 inner-hood geometries from [43], each of which contains about four to eight cutouts. Each feature of the cutout is parameterized by smooth Bézier curves, thus avoiding sharp corners that can induce stress concentrations. To isolate these cutouts curves, we project the 3D inner hood surface onto a reference plane and employ the SAM-2 segmentation model [45] to delineate

individual cutout zones (see Fig. 2). From each segmented region, the curve boundary is extracted to produce an ordered point cloud, which is then stored in a design database.

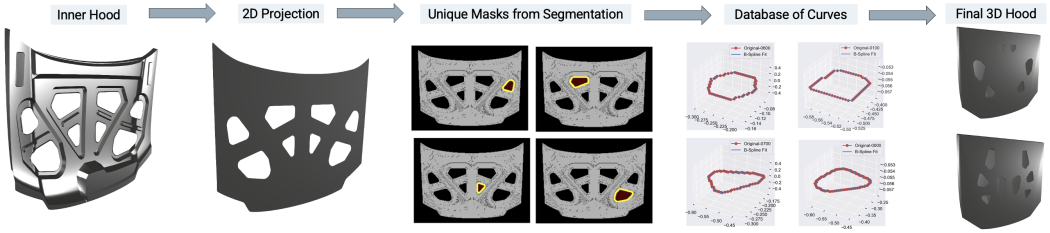

Figure 2: Workflow for generating multiple hood geometries. Starting from the base inner-hood CAD (left), the surface is projected onto a 2D plane and segmented into individual cutout masks. Each mask boundary is extracted as an ordered point-cloud curve. These curves are then re-embedded into the full 3D hood shell to produce the final geometries with engineered openings (right).

The design database comprises approximately 1,750 unique cutout profiles, each characterized by its perimeter and enclosed area. We apply the K-Means [46] clustering algorithm in the two-dimensional feature space (perimeter, area), which yields seven distinct groups (see Fig. 13). When generating new hood variants, we sample between one and four curves from the unique clusters, enforcing bilateral symmetry about the vehicle (or hood) centerline. Two user-defined parameters govern the layout: (1) the minimum distance between adjacent curves on a given cutting plane, and (2) the axial separation between mirrored cutout planes. For each design iteration, we uniformly sample these parameters from prescribed ranges to ensure both engineering relevance and sufficient diversity in the resulting hood geometries. Additional information provided in Appendices D.3 and D.4

3.2 Solver Setup

All simulations were carried out using the open-source OpenFOAM v2312 framework [47]. For the fluid phase, we developed UM_pimpleFoam, a CPU-based LES solver that extends OpenFOAM's transient, incompressible pimpleFoam solver with the Spalart–Allmaras Delayed Detached-Eddy-Simulation (DDES) turbulence closure [48]. The primary modification in UM_pimpleFoam is the inclusion of an additional transient forcing term in the momentum-conservation equation, enabling explicit time-dependent body-force effects within the Navier–Stokes system. For the structural phase, we used UM_solidDisplacementFoam, built on top of OpenFOAM's solidDisplacementFoam utility. This solver assumes a small-strain linear-elastic constitutive model, Hooke's law, to compute the hood stress and displacement fields. Since the predicted deformations are minor and exert negligible spatio-temporal perturbations on the flow field, we adopt a one-way coupling approach: the fluid solver drives the structural response without reverse feedback. Multiple studies have shown that one-way coupling can capture the correct system response [49, 50, 51, 52]. Critically, the fluid and solid solvers execute concurrently, exchanging pressure field at each coupling interval. Additional details on computational setup and solver test cases are provided in Appendix C.

Inter-solver communication preCICE library [53] provides the essential adapter to couple multiphysics simulations by mapping boundary value exchanges at each coupling interval [54]. We adopt a nearest-projection scheme to map the fluid-solver pressure field onto the structural-solver mesh (shown in Fig. 3) and to transfer deformation if bidirectional coupling were to be employed in future. Because both meshes are generated from an identical STL source, the geometric discrepancy is negligible, ensuring that the nearest-projection operator introduces minimal interpolation error.

Computational Mesh Mesh generation was performed with OpenFOAM's SnappyHexMesh utility, incorporating targeted surface boundary refinements to capture near-wall flow features [55, 23]. The final LES mesh comprises approximately 750,000 polyhedral cells ($\Delta = 0.003125$ m) in total, of which 170,000 to 180,000 cells are concentrated in the boundary layer region surrounding the surface of the hood to ensure accurate capture of pressure distribution phenomena. For structural analysis, a surface mesh of approximately 400,000 cells ($\Delta = 0.00220$ m) was extracted from the same STL geometry, providing sufficient spatial resolution to resolve the dominant deflection mode [56].

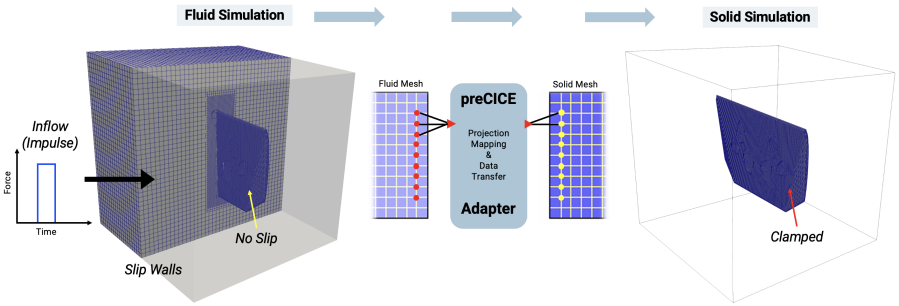

Figure 3: One-way coupled FSI co-simulation overview. An impulsive inflow acceleration is applied to the fluid domain, which uses free-slip exterior walls and a no-slip hood surface. The computed pressure field on the fluid mesh is transferred via preCICE's nearest-projection adapter to the solid mesh, where the hood shell is fully clamped for structural deformation analysis.

Boundary Conditions The computational domain is tightly wrapped around the hood to concentrate the resolution on the fluid loads acting on its surface. At the downstream boundary, a zero-gradient (extrapolation) outflow condition is imposed to suppress spurious reflections back into the domain. The hood geometry is placed normal to the incoming flow, with its engineered cutouts facing upstream. We apply a time-dependent acceleration impulse at the inlet, that is, a uniform force history imposed simultaneously on all fluid cells, while the flow is initialized in a quiescent state. For all simulations, a constant acceleration pulse of magnitude $\alpha = 2.7$ m/s^2 is applied over a fixed duration of $t = 0.07$ s at 30° to the flow direction (sampling time 0.01s, total of 8 solutions). The structural domain is defined as the complement of the fluid mesh (with only the hood shell), and is fully clamped at its mounting interfaces with material properties set as: density (ρ_s) = 2700 kg/m^3, Poison ratio (ν) = 0.33 and Young's modulus (E) = 68.9e9 N/m. Figure 3 details the boundary conditions and domain configuration used in the co-simulation. In the fluid domain, all faces have slip or "inviscid" conditions and has no fluid boundary layer growth; the hood is treated as a no-slip wall to capture viscous effects at the solid–fluid interface.

HPC Setup Simulations were executed on our internal HPC cluster, comprising 16 compute nodes with 8 NVIDIA H100 SXM 80GB GPUs each and a total of 1536 CPU cores (Intel Xeon Platinum 8468 2.1G). Each simulation case was parallelized over 12 cores with 72 GB of memory requirement. The full simulation dataset occupies approximately 1.2 TB of storage and consists of 37,000 output files. In aggregate, the co-simulation campaign consumed roughly 5000 CPU-core hours.

3.3 Dataset for Generative AI

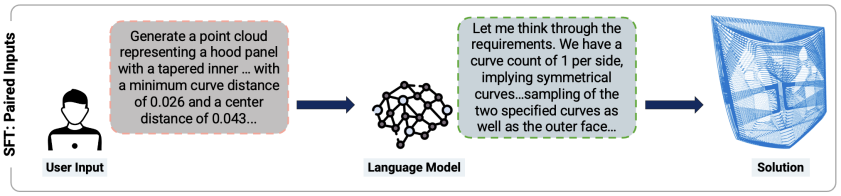

Figure 4: Supervised fine-tuning (SFT) example data point: an LLM is trained to map structured text prompts, specifying cutout parameters such as curve count and geometric spacing, to the corresponding 3D point cloud hood geometry.

We constructed a text-based paired corpus of 2587 hood geometries by leveraging the Gemma3 model (gemma3-27b-it)[57] in its vision-language configuration. For each sample, we authored a concise text prompt encoding the engineered cutout parameters: minimum intercurve spacing, center plane offset, and total curve count along with a snapshot of the geometry and passed it through Gemma3's multimodal encoder. The model then generated the corresponding 3D point cloud description of the hood shell and a chain-of-thought procedure, which we captured as the supervised output (see Fig. 4). This dataset supports training and supervised fine-tuning (SFT) of LLMs to map natural

language design specifications onto 3D point cloud representations, thus establishing a foundation for generative AI CAD workflows ranging from zero shot shape synthesis and parametric editing to language-driven mesh refinement and automated design variation generation. Additional details in SI.

3.4 Dataset Structure

The dataset encompasses automotive hood designs across multiple modalities, ensuring diverse representations and enabling seamless integration into a variety of downstream ML and generative AI applications.

- 3D Base Geometries: High-fidelity STL meshes of complete hood shells, suitable for CAD workflows, CFD mesh generation, design optimization, and generative modeling.
- CFD Results: Time-resolved surface-interpolated velocity and pressure fields at eight temporal snapshots, ideal for data-driven fluid dynamics studies and autoregressive sequence models.
- FEA Results: Displacement and stress fields on the hood under CFD-derived loads, allowing structural performance analysis and surrogate model training.
- Point Clouds: Surface point cloud representations of the hood with variable point densities, for use in reconstruction algorithms.
- Engineered STL Variations: A diverse set of STL hood shells with integrated cutouts, curated for generative-design, topology-optimization, and virtual-reality applications.
- Text Descriptions: Tokenized natural language annotations for each STL file, facilitating transformer-based mapping between textual metadata and geometric models.

The dataset is organized into two levels, a sample subset of 4500 (approximately) and an extended sample superset of 12000+ hoods, structured in a curriculum learning sequence [58]. Beyond these datasets, an additional set consisting of 10000+ hood geometries with randomized cutout configurations is included. Within each tier, samples progress from designs with the fewest cutout curves to those with the most, enabling staged training from simpler to more complex geometric variations. Additional details in Appendix D.

Dataset Access The dataset is released under CC BY-NC 4.0 and is freely available without requiring an account (see D.1); full download instructions and metadata (Croissant format) appear in SI and GitHub. Due to the large size of the data, pre-split subsets for training, validation, in-distribution (id) testing, and out-of-distribution (ood) testing are also provided.

4 ML Benchmarking

Accurate prediction of fluid-induced deformations is essential in the automotive hood design process. Because rapid evaluation of numerous design variants, each subject to multiple performance constraints, is required, low-latency surrogate models are indispensable. In this section, building on previous benchmarks [22, 21, 23], we present ML approaches for predicting the normalized deformation response across different hood geometries. Using PyTorch [59], we evaluate five architectures—MLP, PointNet [60], GraphSAGE [61], Graph U-Net [62], and PointGNNConv [63]—each composed of an encoder, core model, and decoder. An MLP-based encoder first maps each hood vertex (or point-cloud sample) and its auxiliary attributes (e.g., surface pressure) into a latent feature vector. The core model then aggregates these embeddings, via point- or graph-based operations, to capture local and global geometric context. Finally, an MLP-based decoder regresses the normalized deformation at each node. For this study, we used the 4500 hood geometries subset, partitioned into training (75%), validation (15%), id-testing (10%), and ood-testing (single isolated geometry) segments. Model inputs consist of point-cloud coordinates, surface normals, and mesh-connectivity data, while outputs comprise the velocity components U, pressure p, and displacement D. All models were trained for 750 [2] epochs using the mean squared error (MSE) loss function. Comprehensive hyperparameter configurations and architectural details are provided in Appendix A.

[2] Graph U-Net results are reported at 320 epochs due to its longer per-epoch runtime. However, the training loss curve aligns closely with those of the leading models trained for 750 epochs.

Benchmarking Metric Model performance is evaluated using the Mean Squared Error (MSE), defined as the average of the squared differences between predicted and true deformations. Due to its quadratic error term, the MSE is particularly sensitive to outliers in prediction errors.

4.1 Benchmarking Results

4.1.1 In-Distribution Tests

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP	**0.25**	0.27	**0.44**	**0.37**	2.80	0.51	0.76
PointNet	0.31	**0.26**	0.48	0.55	**0.29**	**0.40**	**0.43**
GraphSAGE	4.89	1.31	2.04	4.22	14.88	3.98	6.86
Graph U-Net [2]	1.91	0.86	1.18	3.09	2.89	0.92	0.97
PointGNNConv	4.50	1.42	2.69	7.70	11.71	8.92	15.41

Table 1: ID Test: Mean squared error on the different normalized fields for all the models.

Table 1 and Fig. 5 demonstrate that simple architectures, namely MLP and PointNet, outperform more complex graph-based methods on ID test data. As shown in the table, the MLP achieves the lowest MSE on U_x and U_z, and on p, while PointNet shows the best overall balance of errors across all fields. The visualizations of the field map in Fig. 5 confirm these quantitative findings. The MLP and PointNet reconstructions capture the primary patterns in $\|U\|$ and p with minimal smoothing, and reproduce the high-deflection zones around cutout edges in $\|D\|$. In contrast, graph-based models, such as GraphSAGE tend to underpredict peak values and introduce spurious oscillations in regions of sharp geometric change. Graph networks are based on a fixed neighborhood radius or neighbor count, which forces a trade-off between over-smoothing fine cutout details and depriving nodes of broader context. In particular, the dramatic degradation in D_x performance is possibly due to displacement being a secondary field driven by the integrated pressure load, so any upstream errors in velocity or pressure predictions amplify downstream effects. See Appendix B for further analysis of MLP performance compared to graph models (homophily and ablation tests).

4.1.2 Out-of-Distribution Test

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP	**1.49**	1.89	2.97	4.96	181.02	5.87	19.81
PointNet	3.48	2.89	6.87	15.45	**73.79**	17.83	25.55
GraphSAGE	3.24	1.81	2.65	6.99	84.34	6.07	21.71
Graph U-Net [2]	2.04	1.29	1.45	**4.05**	126.62	7.69	16.13
PointGNNConv	2.84	**1.51**	**1.20**	5.79	130.08	**4.78**	**13.67**

Table 2: OOD Test: Mean squared error on the different normalized fields for all the models.

The OOD test set reveals a different trend for model performance: graph-based architectures outperform point- and MLP-based models when faced with novel geometries (Table 2). While MLP achieves the lowest error on U_x and PointNet on D_x, both exhibit substantial degradation in p and D predictions outside the training distribution. In contrast, Graph U-Net attains the best OOD MSE for pressure 4.05×10^{-2} and maintains competitive errors across all velocity components, demonstrating a robust generalization of global flow characteristics. Similarly, PointGNNConv achieves the lowest error on transverse components D_y and D_z, indicating its efficacy in capturing localized deformation induced by complex cutout topologies. The field-map visualizations (Figure 6) confirm these quantitative results: Graph U-Net and PointGNNConv reproduce sharp features in pressure and the concentrated deflection zones around cutouts with high fidelity, whereas simpler models produce overly smooth or smeared predictions.

Runtime Analysis Table 3 summarizes the inference times per sample and the parameter counts for each architecture, highlighting their relative efficiency profiles. MLP and PointNet achieve sub-10

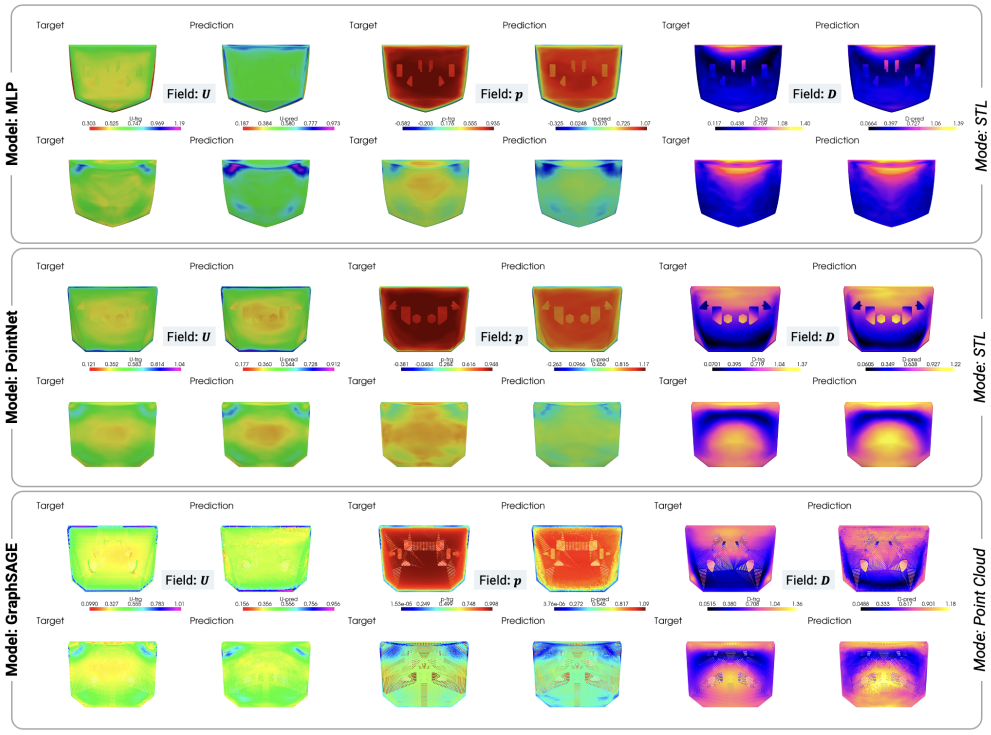

Figure 5: In-distribution test predictions for the three lowest-error models. Each block shows results for a specific mode—MLP (STL), PointNet (STL), and GraphSAGE (Point Cloud)—with target (left) and predicted (right) fields plotted for $\|U\|$, p, and $\|D\|$ on the front (top row of each block) and back (bottom row of each block) surfaces of the hood geometry.

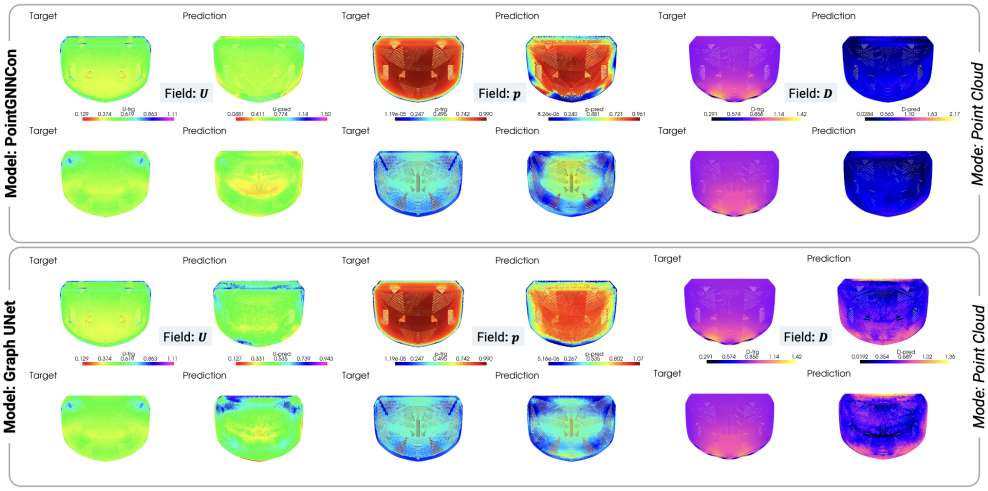

Figure 6: OOD test predictions for the top two graph-based models using point-cloud modality. For PointGNNConv (top block) and Graph U-Net [2] (bottom block), target (left) and predicted (right) fields are shown for $\|U\|$, p, and $\|D\|$ on the front (upper row of each block) and back (lower row of each block) surfaces of a representative OOD hood geometry.

ms inference with strong in-distribution accuracy (see Table 1), ideal for high-throughput prototyping. Graph U-Net and PointGNNConv, while costing 200–250 ms per inference, provide higher OOD fidelity (see Table 2) using advanced message passing. GraphSAGE offers a compromise with 11 ms latency and moderate generalization.

Model	Parameter Count	Time per Epoch (s)	Time per Inference (ms)
MLP	319623	96 ± 4	9.66
PointNet	154463	120 ± 3	11.14
GraphSAGE	93703	147 ± 3	9.74
Graph U-Net	3550535	1520 ± 35	248.86
PointGNNConv	168598	366 ± 4	34.34

Table 3: Comparison of model sizes and runtimes. Inference time averaged for 200 samples.

5 Conclusion

We have presented AutoHood3D, the first open source, large-scale benchmark for 3D automotive hood design and one-way fluid-structure interaction. Comprising more than 16000 parametric variants and an excess of 108000 time-resolved LES–FEA snapshots on a refined cell mesh, the dataset provides sufficient geometric diversity and high-fidelity multiphysics solutions. Our fully documented pipeline, which spans convex shell reconstruction, cutout segmentation, numerical meshing, and preCICE-mediated co-simulation, ensures complete reproducibility and extensibility to other solvers and new physical scenarios.

To support data-driven and generative AI workflows, AutoHood3D is released in multiple modalities (STL, raw CFD/FEA meshes and fields, point clouds) and includes structured natural language prompts for supervised fine-tuning of LLMs in text-to-geometry synthesis. We further establish performance baselines across five neural architectures: from MLP and PointNet (inference times of 7–11 ms, in-distribution MSE $\sim 10^{-3}$) to Graph U-Net and PointGNNConv (34–249 ms, OOD MSE as low as 1.2×10^{-2}). These results quantify the speed-accuracy trade-offs between model classes, providing clear guidance for selecting architectures that align with specific performance and generalization requirements.

We anticipate that AutoHood3D will accelerate research in surrogate modeling, aerodynamic optimization, manufacturing process design, and physics-informed machine learning. By lowering the barrier to large-scale 3D FSI data generation and offering a modular open source workflow, our work lays the groundwork for future datasets to advance generative AI and data-driven engineering.

6 Limitations and Future Scope

Despite its contributions, AutoHood3D has several opportunities for improvements. First, cutouts are currently defined via 2D projections; extending them to fully 3D contours would capture more intricate geometric features. Second, our ML models are restricted to point and graph-based architectures; investigating mesh-agnostic neural operator approaches [64] may improve predictive accuracy. Third, clustering cutouts by area and perimeter may overlook higher-order shape descriptors - incorporating curvature or modal deformation modes could enhance design-space exploration. Finally, the use of MSE loss does not explicitly enforce the physical conservation of momentum or force across the fluid–structure interface; developing multiphysics-aware training objectives that penalize these residuals could improve fidelity.

In the future, we will integrate non-linear structural solvers and introduce additional geometric parameters (e.g., hood mass distribution and material heterogeneity) to support advanced design-optimization studies. We also plan to diversify the loading conditions, varying impulse magnitudes and orientations, to fully cover the operational envelope of manufacturing and aerodynamic scenarios.

Acknowledgments and Disclosure of Funding

This work was supported at the University of Michigan by Ford Motor Company under the grant "Manufacturability-constrained closure design using physics-informed artificial intelligence". The authors thank Elliot Kimmel for discussions on FSI and Sudeep Katakol and Dr.Shivam Barwey for reviewing the initial manuscript and providing insightful feedback.

References

[1] Hrvoje Jasak. Openfoam: Open source cfd in research and industry. *International journal of naval architecture and ocean engineering*, 1(2):89–94, 2009.

[2] Shivank Sharma, Ral Bielawski, Oliver Gibson, Shuzhi Zhang, Vansh Sharma, Andreas H Rauch, Jagmohan Singh, Sebastian Abisleiman, Michael Ullman, Shivam Barwey, et al. An amrex-based compressible reacting flow solver for high-speed reacting flows relevant to hypersonic propulsion. *arXiv preprint arXiv:2412.00900*, 2024.

[3] Jelle Houtman and Sebastian Timme. Global stability analysis of elastic aircraft in edge-of-the-envelope flow. *Journal of Fluid Mechanics*, 967, 7 2023.

[4] Sabet Seraj, Anil Yildirim, Joshua L. Anibal, and Joaquim R.R.A. Martins. Dissipation and time step scaling strategies for low and high Mach number flows. *Journal of Computational Physics*, 491:112358, 7 2023.

[5] Sebastian Abisleiman, Vansh Sharma, Ral Bielawski, and Venkat Raman. Structure of three-dimensional conical oblique detonation waves. *Combustion and Flame*, 274:113971, 2025.

[6] Pier Carlo Nassini, Daniele Pampaloni, Roberto Meloni, and Antonio Andreini. Lean blow-out prediction in an industrial gas turbine combustor through a LES-based CFD analysis. *Combustion and Flame*, 229:111391, 3 2021.

[7] Maciej Chmielewski, Paweł Niszczota, and Marian Gieras. Combustion efficiency of fuel-water emulsion in a small gas turbine. *Energy*, 211:118961, 9 2020.

[8] Shuzhi Zhang, Vansh Sharma, Venkat Raman, Tristan T Shahin, Alexander J Hodge, Rohan M Gejji, Robert P Lucht, and Carson D Slabaugh. Three-dimensional analysis of hydrogen fuel effects in multi-tube combustor. *Proceedings of the Combustion Institute*, 41:105790, 2025.

[9] Mouadh Yagoubi, David Danan, Milad Leyli-abadi, Jean-Patrick Brunet, Jocelyn Ahmed Mazari, Florent Bonnet, maroua gmati, Asma Farjallah, Paola Cinnella, Patrick Gallinari, and Marc Schoenauer. Neurips 2024 ml4cfd competition: Harnessing machine learning for computational fluid dynamics in airfoil design, 2024.

[10] Anudhyan Boral, Zhong Yi Wan, Leonardo Zepeda-Nunez, James Lottes, Qing Wang, Yi-Fan Chen, John Roberts Anderson, and Fei Sha. Neural ideal large eddy simulation: Modeling turbulence with neural stochastic differential equations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[11] Toby van Gastelen, Wouter Edeling, and Benjamin Sanderse. Energy-conserving neural network for turbulence closure modeling. *Journal of Computational Physics*, 508:113003, 2024.

[12] Romit Maulik, Omer San, Jamey D Jacob, and Christopher Crick. Sub-grid scale model classification and blending through deep learning. *Journal of Fluid Mechanics*, 870:784–812, 2019.

[13] Long Wei, Peiyan Hu, Ruiqi Feng, Haodong Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Diffphycon: A generative approach to control complex physical systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in Neural Information Processing Systems*, 31, 2018.

[15] Pantelis R Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multi-scale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[16] Vansh Sharma, Michael Ullman, and Venkat Raman. A machine learning based approach for statistical analysis of detonation cells from soot foils. *Combustion and Flame*, 274:114026, 2025.

[17] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[18] Vansh Sharma, Andreas H Rauch, and Venkatramanan Raman. Accelerating cfd simulations with super-resolution feedback-informed adaptive mesh refinement. In *AIAA SCITECH 2025 Forum*, page 1467, 2025.

[19] Neiwen Ling, Xuan Huang, Zhihe Zhao, Nan Guan, Zhenyu Yan, and Guoliang Xing. Blastnet: Exploiting duo-blocks for cross-processor real-time dnn inference. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, pages 91–105, 2022.

[20] Ruben Ohana, Michael McCabe, Lucas Thibaut Meyer, Rudy Morel, Fruzsina Julia Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B. Dalziel, Drummond Buschman Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich Kerswell, Suryanarayana Maddu, Jonah M. Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam Holden Parker, Miles Cranmer, and Shirley Ho. The well: a large-scale collection of diverse physics simulations for machine learning. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[21] Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

[22] Mohamed Elrefaie, Florin Morar, Angela Dai, and Faez Ahmed. Drivaernet++: A large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. *Advances in Neural Information Processing Systems*, 37:499–536, 2024.

[23] Neil Ashton, Jordan Angel, Aditya Ghate, Gaetan Kenway, Man Long Wong, Cetin Kiris, Astrid Walle, Danielle Maddix, and Gary Page. Windsorml: High-fidelity computational fluid dynamics dataset for automotive aerodynamics. *Advances in Neural Information Processing Systems 37*, 2024.

[24] J Kim, S Park, N Kim, N Hur, and C Oh. Prediction and minimization of micro deformation on the automobile hood after dipping process. *International Journal of Automotive Technology*, 16:293–300, 2015.

[25] J Kim, Naksoo Kim, Nahmkeon Hur, and C Oh. Micro deformation of automobile hood in dipping process using stiffness. *International Journal of Automotive Technology*, 15:475–482, 2014.

[26] Mohammad Hassan Shojaeefard, Amir Najibi, and Meisam Rahmati Ahmadabadi. Pedestrian safety investigation of the new inner structure of the hood to mitigate the impact injury of the head. *Thin-Walled Structures*, 77:77–85, 12 2013.

[27] Zhijun Yang, Tao Deng, and Zhenfei Zhan. Design and analysis of vehicle active hood for pedestrian protection. *SAE technical papers on CD-ROM/SAE technical paper series*, 12 2021.

[28] C. Thill, J. Etches, I. Bond, K. Potter, and P. Weaver. Morphing skins. *The Aeronautical Journal*, 112(1129):117–139, 3 2008.

[29] Juan Diego Toscano, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. From pinns to pikans: Recent advances in physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 1(1):1–43, 2025.

[30] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[31] Hao Zhongkai, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, et al. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *Advances in Neural Information Processing Systems*, 37:76721–76774, 2024.

[32] Sheikh Md Shakeel Hassan, Arthur Feeney, Akash Dhruv, Jihoon Kim, Youngjoon Suh, Jaiyoung Ryu, Yoonjin Won, and Aparna Chandramowlishwaran. Bubbleml: A multiphase multiphysics dataset and benchmarks for machine learning. *Advances in Neural Information Processing Systems*, 36:418–449, 2023.

[33] Vishal Thengane, Xiatian Zhu, Salim Bouzerdoum, Son Lam Phung, and Yunpeng Li. Foundational models for 3d point clouds: A survey and outlook. *arXiv preprint arXiv:2501.18594*, 2025.

[34] Dingning Liu, Xiaoshui Huang, Yuenan Hou, Zhihui Wang, Zhenfei Yin, Yongshun Gong, Peng Gao, and Wanli Ouyang. Uni3d-llm: Unifying point cloud perception, generation and editing with large language models. *arXiv preprint arXiv:2402.03327*, 2024.

[35] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, 131(8):1909–1963, 2023.

[36] Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm: Empowering large language models to understand point clouds. In *European Conference on Computer Vision*, pages 131–147. Springer, 2024.

[37] Yuan Tang, Xu Han, Xianzhi Li, Qiao Yu, Yixue Hao, Long Hu, and Min Chen. Minigpt-3d: Efficiently aligning 3d point clouds with large language models using 2d priors. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6617–6626, 2024.

[38] Ziyu Guo, Renrui Zhang, Xiangyang Zhu, Yiwen Tang, Xianzheng Ma, Jiaming Han, Kexin Chen, Peng Gao, Xianzhi Li, Hongsheng Li, et al. Point-bind & point-llm: Aligning point cloud with multi-modality for 3d understanding, generation, and instruction following. *arXiv preprint arXiv:2309.00615*, 2023.

[39] Jean Lahoud, Jiale Cao, Fahad Shahbaz Khan, Hisham Cholakkal, Rao Muhammad Anwer, Salman Khan, and Ming-Hsuan Yang. 3d vision with transformers: A survey. *arXiv preprint arXiv:2208.04309*, 2022.

[40] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects, 2023.

[41] Chendi Lin, Heshan Liu, Qunshu Lin, Zachary Bright, Shitao Tang, Yihui He, Minghao Liu, Ling Zhu, and Cindy Le. Objaverse++: Curated 3d object dataset with quality annotations, 2025.

[42] Shiming Wang, Holger Caesar, Liangliang Nan, and Julian FP Kooij. Unibev: Multi-modal 3d object detection with uniform bev encoders for robustness against missing sensor modalities. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 2776–2783. IEEE, 2024.

[43] Patricia Wollstadt, Mariusz Bujny, Satchit Ramnath, Jami J. Shah, Duane Detwiler, and Stefan Menzel. Carhoods10k: An industry-grade data set for representation learning and design optimization in engineering applications. *IEEE Transactions on Evolutionary Computation*, 26(6):1221–1235, 2022.

[44] Yves Dejonghe. Py-madcad. `https://github.com/jimy-byerley/pymadcad`, 2024.

[45] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[46] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[47] Andrew Heather, Mattijs Janssens, Mark Olesen, Prashant Sonakar, Pawan Ghildiyal, Kutalmış Berçin, Matej Forman, Chiara Pesci, Martin Lichtmes, Jiri Polansky, Ann Ronchetti, Fred Mendonça, Swapnil Salokhe, and Venkata Ramana Eaga. OpenFOAM v2406 Software. Available at: https://www.openfoam.com/news/main-news/openfoam-v2312, 2023.

[48] M Kh Strelets. A new version of detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and computational fluid dynamics*, 20:181–195, 2006.

[49] Vilas J. Shinde, Jack J. McNamara, and Datta V. Gaitonde. One-Way response of a flexible panel to shock wave boundary layer interaction. *AIAA Aviation 2019 Forum*, 6 2023.

[50] Elliot Kimmel, Daning Huang, Vansh Sharma, Jagmohan Singh, Venkatramanan Raman, and Peretz P. Friedmann. Evaluation of Shock Wave-Boundary Layer interaction modeling capabilities for use in a hypersonic aerothermoelastic framework. *AIAA SCITECH 2022 Forum*, 1 2024.

[51] Friedrich-Karl Benra, Hans Josef Dohmen, Ji Pei, Sebastian Schuster, and Bo Wan. A comparison of One-Way and Two-Way coupling methods for numerical analysis of Fluid-Structure interactions. *Journal of Applied Mathematics*, 2011(1), 1 2011.

[52] Nora Hagmeyer, Matthias Mayr, Ivo Steinbrecher, and Alexander Popp. One-way coupled fluid–beam interaction: capturing the effect of embedded slender bodies on global fluid flow and vice versa. *Advanced Modeling and Simulation in Engineering Sciences*, 9(1), 6 2022.

[53] G Chourdakis, K Davis, B Rodenberg, M Schulte, F Simonis, B Uekermann, G Abrams, HJ Bungartz, L Cheung Yau, I Desai, K Eder, R Hertrich, F Lindner, A Rusch, D Sashko, D Schneider, A Totounferoush, D Volland, P Vollmer, and OZ Koseomur. preCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]. *Open Research Europe*, 2(51), 2022.

[54] Gerasimos Chourdakis, David Schneider, and Benjamin Uekermann. OpenFOAM-preCICE: Coupling OpenFOAM with external solvers for multi-physics simulations. *OpenFOAM® Journal*, 3:1–25, Feb 2023.

[55] Angelina I. Heft, Thomas Indinger, and Nikolaus A. Adams. Introduction of a new realistic generic car model for aerodynamic investigations. *SAE technical papers on CD-ROM/SAE technical paper series*, 4 2012.

[56] Wadhah Garhuom, Simeon Hubrich, Lars Radtke, and Alexander Düster. A remeshing strategy for large deformations in the finite cell method. *Computers & Mathematics with Applications*, 80(11):2379–2398, 4 2020.

[57] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[58] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[60] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017.

[61] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[62] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[63] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020.

[64] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[65] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[66] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. In *International Conference on Learning Representations*, 2021.

[67] Mark EJ Newman. Mixing patterns in networks. *Physical review E*, 67(2):026126, 2003.

[68] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

[69] Joel H Ferziger and Milovan Perić. *Computational methods for fluid dynamics*. Springer, 2002.

[70] P. Cardiff, Ž. Tuković, H. Jasak, and A. Ivanković. A block-coupled Finite Volume methodology for linear elasticity and unstructured meshes. *Computers & Structures*, 175:100–122, 8 2016.

[71] Roger T Fenner. Engineering elasticity: application of numerical and analytical techniques. *(No Title)*, 1986.

[72] A.K. Slone, C. Bailey, and M. Cross. Dynamic solid mechanics using finite volume methods. *Applied Mathematical Modelling*, 27(2):69–87, 12 2002.

[73] Wolfgang Bangerth, Ralf Hartmann, and Guido Kanschat. deal. ii—a general-purpose object-oriented finite element library. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):24–es, 2007.

[74] Martin Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E Rognes, and Garth N Wells. The fenics project version 1.5. *Archive of numerical software*, 3(100), 2015.

[75] Links Schiller and Z Naumann. A drag coefficient correlation. *Zeit. Ver. Deutsch. Ing.*, 77:318–320, 1933.

[76] Benedikt Schott, Christoph Ager, and Wolfgang A Wall. Monolithic cut finite element–based approaches for fluid-structure interaction. *International Journal for Numerical Methods in Engineering*, 119(8):757–796, 2019.

[77] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[78] Mark D Wilkinson, M Dumontier, IJ Aalbersberg, G Appleton, M Axton, A Baak, N Blomberg, JW Boiten, LB da Silva Santos, PE Bourne, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1), 3 2016.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately capture the scope and key contributions of the article, including the CFD/FEA solver test cases (detailed in the Supplementary Information and on GitHub) and the fully open-source, end-to-end workflow for dataset reproduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated "Limitations and Future Work" section systematically examines the study's constraints and their broader implications.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This is beyond the scope of the current study focused on dataset.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The complete set of codes and the model settings are included in the article and on GitHub. We also include the pre-split dataset for model training and reproducing the benchmarking results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer:[Yes]

   Justification: Links to the dataset and the codes are shared in the article.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The details of the training and parameter selection are shared in the main article. Additional experiment and information on hyperparameters, optimizer, scheduler settings, and model architecture are included in the SI.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: Not relevant to this study.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The discussion regarding computational resources has been included in the "Dataset Workflow" section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors confirm that the research follows the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The authors note that, given the academic emphasis of this work, there is no direct societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The current study does not include high-risk models or data.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The dataset and codes are released under the Creative Commons Attribution-NonCommercial (CC BY- NC 4.0) license, allowing use and modification for noncommercial purposes provided appropriate credit is given. Additional information is included in the article.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Details/instructions of the dataset and code usage are shared in the article and on GitHub.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not in the scope of the current study.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not in the scope of the current study.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: A modality of the dataset, collection of natural language prompts for design variations, was created using a vision-LLM following user-defined prompts and design information. This has been discussed in the “Dataset Workflow" section and SI. The authors clarify that no LLM was employed for methodological innovation or core research development; these contributions are entirely the work of the research team.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Technical Appendices and Supplementary Material

Contents

1 Introduction **1**

2 Related Work **3**

3 Dataset Workflow **3**

3.1 Base Geometry Generation 3

3.2 Solver Setup 4

3.3 Dataset for Generative AI 5

3.4 Dataset Structure 6

4 ML Benchmarking **6**

4.1 Benchmarking Results 7

4.1.1 In-Distribution Tests 7

4.1.2 Out-of-Distribution Test 7

5 Conclusion **9**

6 Limitations and Future Scope **9**

A Hyperparameter Tuning **23**

A.1 Graph Construction with Max-Neighbor Tuning 23

B Analysis of MLP Performance **24**

B.1 Homophily Test 24

B.2 GraphSAGE mNN Sweep 25

B.3 MLP Hidden-Size Ablation Study 25

C Numerical Solvers **25**

C.1 Fluid Solver 25

C.2 Solid Solver 26

C.3 Solver Validation 27

C.3.1 Fluid Test Case 27

C.3.2 Solid Test Case 27

C.3.3 FSI Test Case 28

D Dataset **29**

D.1 Dataset Structure and URLs 29

D.2 Contents 29

D.3 Hood Geometry Variations 29

D.4 Clustering Cutouts 30

D.5 Automated Simulations 31
D.6 Analysis of Simulation Results 32
D.7 LLM Prompt 33
D.8 Long-term maintenance 33

A Hyperparameter Tuning

We adopt a unified set of hyperparameters that yielded stable performance across all architectures, with the specific parameters of each model (e.g., encoder-decoder configurations, hidden-channel width) tuned within this framework (see Table 5). All models are trained using the AdamW [65] optimizer, paired with a ReduceLROnPlateau scheduler configured as detailed in Table 4. For PointGNNConv, we reduced the base learning rate to 5×10^{-5} to prevent training instabilities observed in early trials, and the step patience for GraphSAGE is set to 9^* . Although the training pipeline supports distributed data-parallel (DDP) execution over the full 16000+ design ensemble, the results presented here are based on a reduced subset of 4500 hood geometries trained using a single H100 GPU.

Weight Decay	Step Factor	Min LR	Step Patience	Epochs	LR Warmup
1.00×10^{-6}	0.1	1.00×10^{-7}	7^*	750	Linear, 1.5%

Table 4: Optimizer and scheduler hyperparameters.

Parameter	PointGNNCon	GraphUNet	GraphSAGE	PointNet	MLP
encoder	[10,128,128,64]	[10,128,128,64]	[10,128,128,64]	[10,256,128,64]	[10,256,128,64]
decoder	[64,128,128,7]	[64,128,128,7]	[64,128,128,7]	[64,128,256,7]	[64,128,256,7]
nb_hidden_layers	4	–	4	–	4
size_hidden_layers	64	64	64	–	256
base_nb	–	–	–	8	–
layer	–	SAGE	–	–	–
pool	–	random	–	–	–
nb_scale	–	5	–	–	–
pool_ratio	–	[0.5,0.5,0.5,0.5]	–	–	–
list_r	–	[0.05,0.2,0.5,1,10]	–	–	–
batch_size	128	128	128	128	128
nb_epochs	750	352	750	750	750
lr	1e-4	1e-4	1e-4	1e-4	1e-4
max_neighbors	4	4	4	–	–
r	0.05	0.05	0.05	–	–

Table 5: Hyperparameter settings for each model.

A.1 Graph Construction with Max-Neighbor Tuning

In the graph model setup, each node in the mesh is connected to all neighbors within a fixed radius, defining the local geometric context, where “max_neighbors" (mNN) caps the number of edges per node. By building the full mesh graph once (so no contour detail is lost to sub-sampling mesh points or repeated graph construction) and keeping the radius constant, we promote “max_neighbors" to the only complexity control. To measure its effect on training loss and per-epoch runtime, we train GraphSAGE (using same settings as mentioned above except the eopchs set to 150) on the complete mesh, preserving every cutout, and systematically vary only max_neighbors (testing 4, 8, 16, and 32), thus identifying the optimal neighborhood size that balances accuracy and speed. Here, the entire dataset is reloaded at each epoch, maintaining a constant memory footprint. This optimal setting captures critical geometric detail without incurring unnecessary computational cost.

Table 6 demonstrates that increasing the GraphSAGE neighborhood size (mNN) steadily reduces MSE across all fields but at the expense of both runtime and data-storage footprint. With mNN=4, the model trains in only 147 s per epoch on a 201 GB dataset, producing acceptable ID errors. Doubling to 8 neighbors halves certain displacement errors (e.g. D_y) but more than doubles per-epoch time and dataset size. By mNN=32, further accuracy gains are marginal while per-epoch time reaches $\sim$ 2361

mNN	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)	Time per Epoch	Dataset Size (GB)
4	4.65	1.31	2.00	4.13	14.56	3.78	6.56	147 ± 2	201.1
8	3.10	1.19	1.97	5.15	9.95	1.76	3.09	345 ± 7	290.4
16	2.93	1.07	1.71	4.30	10.49	4.28	8.53	443 ± 15	469.2
32*	1.91	0.98	1.40	3.35	8.10	1.51	2.83	2361 ± 245	826.6

Table 6: mNN Test: Mean squared error using the ID testing dataset for the different normalized fields using different mNN settings. *For mNN=32 setting, the model results are reported at 130 epochs due to longer per-epoch runtime.

s and the storage exceeds 800 GB, this configuration was stopped at 130 epochs due to intractability. Thus, mNN=4 delivers the most favorable speed–accuracy–size balance in this case, especially for larger graph models where high neighbor counts become prohibitively slow and memory intensive.

B Analysis of MLP Performance

Based on results for in-distribution tests in Sect. 4.1, MLP model tends to show better performance on multiple output variables (such as U_x, p) compared to more complex graph models. In this section, our aim is to explore the underlying reasons for these results, which could potentially be attributed to overfitting. Another possible explanation is that "MLPs can incorporate geometric information, given their encoder's width being four times that of any other model." While it is true that the MLP's hidden layer width is four times the dimensionality of its encoder output—in our case, a 64 to 256 configuration—this does not necessarily imply that the MLP is aggregating information from neighboring points. Instead, each vertex's feature vector (comprising its coordinates, normals, and distance field) is processed independently through fully connected layers, without any explicit pooling or message passing over spatial neighbors. Thus, MLP's unexpectedly strong in-distribution performance cannot be directly attributed to local geometric aggregation in the same way as with graph models. To further investigate this, we performed the following analysis:

Hypothesis	Description	Result
A. **Non-homophily**	If adjacent nodes differ greatly, graph aggregation may harm performance.	**Rejected**: high homophily found (see Table 8), so GNNs should benefit from local pooling.
B. **Sensitivity to neighbor count (k)**	GraphSAGE performance depends on neighborhood size.	**Confirmed**: larger k reduces prediction error but at steep compute and I/O cost.
C. **MLP Capacity & Overfitting**	The wide MLP memorizes training-set patterns, inflating in-distribution scores.	**Supported**: ablation shows ID error rises with hidden-dim reduction while OOD error improves (Table 10).

Table 7: Summary of hypotheses, their motivations, and empirical findings.

B.1 Homophily Test

The hood surface often exhibits cutout edges and curvature changes that could cause non-homophily. In such cases, neighborhood pooling of GNNs can oversmooth fine features—whereas an MLP's point-wise mapping avoids this [66], potentially explaining its ID test errors. Therefore, to measure the non-homophily, we computed numeric assortativity ($\bar{r}$) [67] and a standard edge-wise multi-dimensional cosine similarity [68] over the entire in-distribution test data/graphs:

Field	U_x	U_y	U_z	p	D_x	D_y	D_z	Multi-dim Cosine
$\bar{r}$	+0.9756	+0.9739	+0.9711	+0.9650	+0.9910	+0.9933	+0.9877	+0.9449
σ_r	0.0037	0.0106	0.0059	0.0058	0.0301	0.0154	0.0322	0.0109

Table 8: Mean assortativity ($\bar{r}$) and its standard deviation for each physical field with cosine similarity.

Interpretation: All fields exhibit strong homophily ($\bar{r} > 0.96$), so neighbor aggregation should be beneficial for GNNs—not detrimental—if effectively leveraged. This can be done by carefully selecting the "max neighbors" (mNN), as the quality of feature aggregation in GNNs is significantly influenced by this choice. This is the focus of Sec. A.1, where we explore the impact of mNN selection on GNN performance in more detail.

B.2 GraphSAGE mNN Sweep

In a separate experiment (see Table 6), we varied GraphSAGE's neighbor count (mNN = 4,8,16,32). Increasing the neighborhood size from 4 to 32 reduces ID MSE by up to 40-50%, but:
Compute Time: per-epoch time grows by an order of magnitude.
Data I/O: dataset size inflates proportionally to neighborhood features stored.
Higher mNN reduced in-distribution errors at the expense of increased computation time, illustrating the classic trade-off of neighborhood aggregation.

B.3 MLP Hidden-Size Ablation Study

Our ablation study compared MLPs with hidden-layer widths of 256, 128 and 64—keeping all other hyperparameters consistent with Table 5. The following is the complete set of results:

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP 256	0.25	0.27	0.44	0.37	2.80	0.51	0.76
MLP 128	0.32	0.30	0.49	0.47	4.59	0.75	0.91
MLP 064	0.39	0.37	0.53	0.58	7.63	0.74	1.02

Table 9: ID Test: Mean squared error on the different normalized fields for all the MLP models.

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP 256	1.49	1.89	2.97	4.96	181.02	5.87	19.81
MLP 128	1.58	1.60	2.68	4.65	175.28	7.58	22.91
MLP 064	2.14	1.09	1.59	3.38	124.11	8.50	20.55

Table 10: OOD Test: Mean squared error on the different normalized fields for all the MLP models.

ID Trend: error increases steeply as hidden dimension shrinks, confirming that the 256-dim model leverages high capacity to fit training data.
OOD Trend: the smaller MLPs generalize marginally better, indicating the largest network is more susceptible to overfitting the data.
Together, these studies suggest (Table 7) that the advantage of MLPs on ID tests is not directly due to hidden neighbor aggregation; rather, their wider layers facilitate point-wise memorization of global patterns. On the other hand, graph-based models take advantage of the existing homophily within the data, but they must strike a careful balance between neighborhood size and the risk of oversmoothing, as well as the associated computational overhead. As such, the choice of model for practical surrogate design will ultimately depend on the specific application: MLPs are better suited for high-throughput, in-distribution tasks, while GNNs—when tuned for maximum neighborhood size (mNN)—are more effective for robust generalization, assuming sufficient computational resources are available.

C Numerical Solvers

The following subsections detail the configuration of each solver and the corresponding test cases.

C.1 Fluid Solver

UM_pimpleFoam is a CPU-based solver built on OpenFOAM's pimpleFoam solver. Unlike the standard pimpleFoam solver, UM_pimpleFoam incorporates an additional transient acceleration

source term in the momentum-conservation equation to capture the inertial body force arising from the prescribed fluid acceleration. For this study, LES simulation is conducted with the Spalart-Allmaras Delayed Detached Eddy Simulation (DDES) model [48]. UM_pimplefoam solves the incompressible Navier-Stokes equations on a nonuniform grid. Following are the system of equations along with the additional acceleration term (A_i). The mass-conservation equation is:

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1}$$

where x_i = (x, y, z) are the Cartesian coordinates, and the components of the velocity vector are u_i = (u, v, w). The fluid is treated as incompressible with constant density ρ, therefore the momentum equation is normalized by ρ [69]. The resulting form is:

$$\underbrace{\frac{\partial u_i}{\partial t}}_{\text{local (unsteady) acceleration}} + \underbrace{u_j \frac{\partial u_i}{\partial x_j}}_{\text{convective momentum transport}} = \underbrace{-\frac{1}{\rho}\frac{\partial p}{\partial x_i}}_{\text{pressure-gradient force}} + \underbrace{\nu \frac{\partial^2 u_i}{\partial x_j \partial x_j}}_{\text{viscous diffusion}} + A_i \tag{2}$$

Here, the left-hand side collects the inertial terms, while the right-hand side comprises pressure, viscous, and body-force contributions. The term A_i explicitly represents the time-varying acceleration applied to each fluid cell. Therefore, due to the incompressible formulation, the fluid density ρ is implicitly incorporated through kinematic viscosity ($\nu = \mu/\rho$).

C.2 Solid Solver

UM_solidDisplacementFoam is a CPU-based solver built on OpenFOAM's solidDisplacementFoam. We chose OpenFOAM's structural solver for its seamless integration with the fluid solver and streamlined mesh generation, since the solid mesh derives directly as the complementary subset of the fluid-domain mesh. The solver uses a small-strain elastic deformation model to solve the stress and displacement of the hood, and solves the stress-strain relationship defined by Hooke's law. Since the deformations during the dip process are not large [24, 25], the linearity assumption between stress and strain remains valid. However, for larger deformations, different solvers can be incorporated using the preCICE adapter [53]. Unlike the standard implementation, our solver embeds the force patch calculation routine [70] directly within its code, eliminating the need for any external function. Following equation is the transient linear momentum balance for a deformable solid [71]:

$$\rho \frac{\partial^2 D_i}{\partial t^2} = \underbrace{\frac{\partial \sigma_{ij}}{\partial x_j}}_{\text{divergence of stress}} + \underbrace{b_i}_{\text{body-force density}}, \tag{3}$$

where D_i are the displacement vector components and σ_{ij} is the Cauchy stress. The small-strain tensor is defined as:

$$\varepsilon_{ij} = \underbrace{\frac{1}{2}\Big(\frac{\partial D_i}{\partial x_j} + \frac{\partial D_j}{\partial x_i}\Big)}_{\text{small-strain tensor}}. \tag{4}$$

And the Hooke's law for a linear elastic, isotropic material:

$$\sigma_{ij} = \underbrace{\lambda\, \varepsilon_{kk}\, \delta_{ij}}_{\text{volumetric response}} + \underbrace{2\,\mu\, \varepsilon_{ij}}_{\text{shear response}}, \tag{5}$$

expressing the stress tensor (σ_{ij}) in terms of volumetric response (through Lamé's first parameter λ) and shear response (through the shear modulus μ) [72]. Thermal stresses are neglected in our formulation, and the corresponding terms are therefore omitted from the governing equations. The deformation of the solid region is solved based on the pressure distribution at the interface for the previous timestep.

Extending FEA Leveraging preCICE's [53] solver-agnostic coupling, our FSI framework can be extended to incorporate non-linear structural solvers, native to OpenFOAM or through external libraries such as deal.II [73] and FEniCS [74], facilitating multiphysics analyses across various engineering domains and requirements.

C.3 Solver Validation

Following validation cases are presented to support the simulation settings and to justify the selection of parameters such as mesh refinement levels and solver specifications. The cases are provided along with the associated codes and workflows.

C.3.1 Fluid Test Case

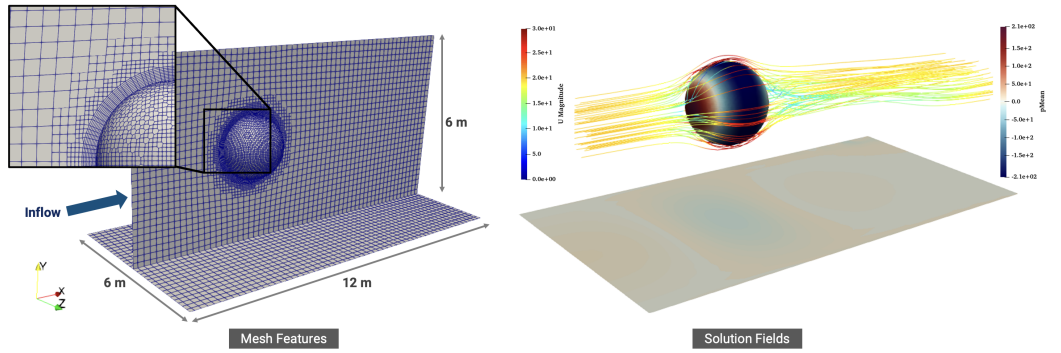

Figure 7: Computational setup and results for flow over a sphere. (Left) Mesh features showing domain dimensions, refinement near the sphere, and inflow direction. (Right) Solution fields illustrating velocity streamlines colored by velocity magnitude, and pressure contours on the sphere and ground plane, highlighting the pressure distribution.

This case considers the simulation of a 3D sphere placed in a domain of dimensions 12×6×6 m, with a uniform flow field to evaluate the pressure drag force, as shown in Fig.7. Inflow consists of atmospheric air at a velocity of U = 20 m/s and a temperature T = 293 K, corresponding to a density ρ = 1.21041 kg/m^3. The sphere has a radius of 1 m, resulting in a frontal area of A = 3.14 m^2. Based on flow conditions, the Reynolds number is calculated as Re = 2.47×10^6 , placing it well within the turbulent regime where the drag coefficient C$_d$ for a smooth sphere is approximately 0.47 (based on [75]). The computational mesh is constructed with a characteristic grid spacing of $\Delta x \simeq 0.039$ m to ensure adequate resolution of the flow features around the sphere. The analytical force can be computed as $F_d = \frac{1}{2}\,\rho\,u^2\,C_d\,A$, and using the provided values, it is found to be 355.582 N. Using the simulation results, the drag force is calculated as $F_d^{Sim} = \sum(p \times \vec{n})\rho A$, which yields 357.049 N, corresponding to an error of 0.41%. The simulation accurately predicts the drag force on the sphere, with a computed error of only 0.41% compared to the analytical value. This close agreement validates the mesh quality and the overall solver setup.

C.3.2 Solid Test Case

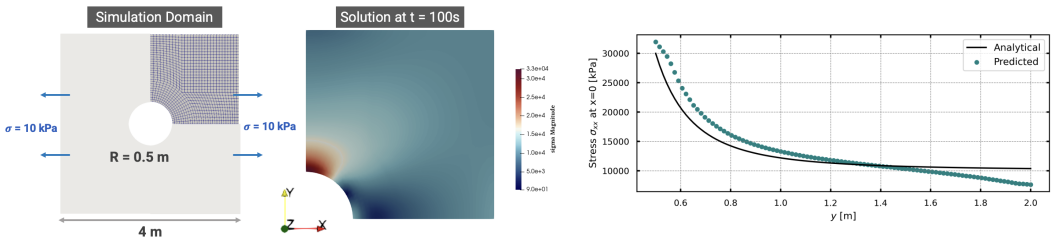

Figure 8: Computational setup and results for time varying flow over a flap. (Left) Mesh features showing domain dimensions, refinement near the hole, and traction directions. Solution fields illustrating stress contour on the plate at time $t = 100s$ (middle) and normal stress along the vertical symmetry plotted on the right.

To test the solid solver, a 2D problem focused on steady-state linear elastic stress analysis on a square plate with a circular hole at its center is used. The dimensions of the plate are: side length 4 m and

radius R = 0.5 m. It is loaded with a uniform traction of σ = 10 kPa on its left and right faces, as shown in Figure 8. Due to dual symmetry, only a quarter of the domain is meshed. We adopt a plane-stress formulation, neglecting all out-of-plane stress and strain components. For an infinitely thin plate with a central circular hole under remote uniaxial tension, there is an exact analytical solution. In particular, the normal stress on the symmetry plane is given by

$$(\sigma_{xx})_{x=0} = \begin{cases} \sigma\left(1 + \frac{R^2}{2y^2} + \frac{3R^4}{2y^4}\right) & \text{for } |y| \geq R \\ 0 & \text{for } |y| < R \end{cases} \tag{6}$$

The middle plot in Figure 8 presents the von Mises stress ($|\sigma|$) field at t=100 s on the quarter-domain mesh, illustrating the stress concentration around the circular hole under a uniform 10 kPa tensile load. Along the symmetry line x=0, the computed normal stress is plotted against the analytical solution derived above. Green markers represent numerical predictions, which closely follow the solid black curve of the analytical solution, with slight overprediction near the hole rim and good agreement farther away.

C.3.3 FSI Test Case

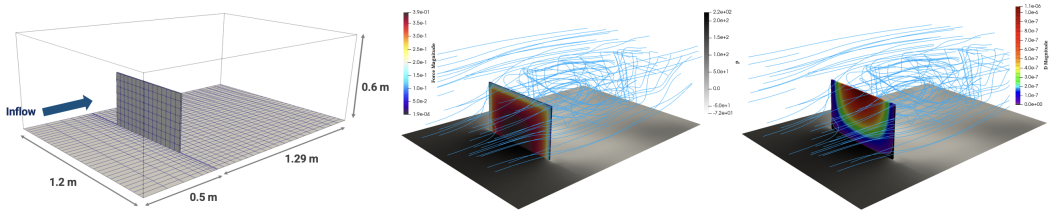

Figure 9: Computational setup and results for time varying flow over a flap. (Left) Mesh features showing domain dimensions, refinement near the flap, and inflow direction. Solution fields illustrating velocity streamlines, and pressure contours on the ground plane with force contour (middle) and displacement contour (right) on the flap.

For the 3D FSI test case, the simulation consists of a thin 3D flap anchored at the bottom in a channel with a time-varying inflow from the left, as shown in Fig. 9. One-way coupling is used between the fluid and the solid solvers as described in the previous sections. The overall dimensions of the domain are 1.8 $\times$ 0.6 $\times$ 1.2 m with a prescribed no slip ($u = 0$) condition in the top and bottom channel walls, with inflow velocity described as $u = (10, 0, 0)$. The case is adapted from [76], with the properties of the selected material set as: density (ρ_s) = 2700 kg/m^3, Poison ratio (ν) = 0.33 and Young's modulus (E) = 68.9e9 N/m. The flap has dimensions of 0.01 $\times$ 0.35 $\times$ 0.6 m and clamped to the bottom. Three different levels of mesh refinements are simulated (cases 1–3 employ successive 2$\times$ mesh refinements), with the fluid solver settings adapted from the previous section. The force on the flap is measured at the location [0.5,0.35,0.0]. In Figure 10, the force time histories for the

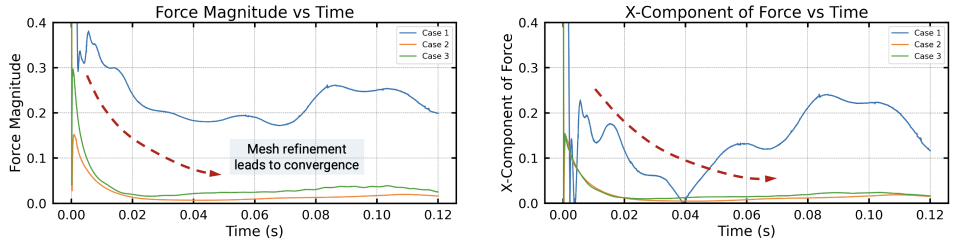

Figure 10: Force response at a flap location under uniform loading for three mesh refinements: Case 1 (baseline), Case 2 (2× refinement), and Case 3 (4× refinement). Left: total force magnitude vs. time. Right: X-component of force vs. time. The red dashed arrow highlights convergence toward a smooth solution as mesh density increases.

two finer meshes (Cases 2 and 3) nearly coincide, demonstrating convergence as the mesh resolution doubles twice. In contrast, the coarsest mesh (case 1) exhibits pronounced oscillations, particularly in the X-component trace, due to under-resolved pressure gradients and numerical interpolation

noise across the flap interface. As resolution increases, these spurious fluctuations decrease, and the solution smoothly decays to its steady-state value.

D Dataset

D.1 Dataset Structure and URLs

- Code available at : `https://github.com/vanshs1/AutoHood3D/`
- Dataset of SFT LLM prompts (in 06_LLM_Generation/Final_consolidated_prompts.jsonl): `https://github.com/vanshs1/AutoHood3D/`
- Dataset of hood base shells/skins: `https://doi.org/10.7910/DVN/9268BB`
- Dataset of 4k hoods
  - STLs: `https://doi.org/10.7910/DVN/HEILMB`
  - Simulation Data (raw): `https://doi.org/10.7910/DVN/VCKOK5`
  - Processed for ML task: `https://doi.org/10.7910/DVN/6OAFF8`
  - Processed for ML task (Graphs): `https://doi.org/10.7910/DVN/WODNWY`
  - Test ML workflow**: `https://doi.org/10.7910/DVN/FSYRJA`
- Dataset of 12k hoods
  - STLs: `https://doi.org/10.7910/DVN/ZOVXLI`
- Dataset of 12k hoods (randomized)
  - STLs: `https://doi.org/10.7910/DVN/OJXIS1`

The updated URLs for the datasets, particularly the simulation data for the 12k case due to its large size, and the corresponding Croissant metadata for each dataset are shared on the GitHub repository. A compact ML test set** of 100 preprocessed cases is included for the rapid validation of training workflows. From these, users can easily sample smaller subsets (e.g., 12 or 32 batch sizes) to perform local testing of the ML pipelines.

D.2 Contents

The dataset contents are as follows:

- 10000+ randomized hoods cut from 108 unique base hood shells.
- 12000+ clustered hoods cut from 108 unique base hood shells.
- An additional 4500+ clustered hoods.
- JSON files for 2500+ prompts.
- 108 base shells for making new geometries.
- 1750 different curve cut-out files.

In Figure 11, the naming convention of a standard clustered geometry is shown. The base shell geo_010 refers to the standard outer shell from which the cutouts are carved. The “clusterID" 5 is assigned based on the strata of the perimeter and the area of the cuts. Further details on clustering are provided in D.4. All geometries are symmetrical, with “crvCount" referring to the number of cutouts on either side of the hood. “Curve ID" 0290_0694_0710_0627 functions to parameterize the geometry. The center distance “cd _0.030" refers to the minimum distance between the cutouts across the mirror plane, and the “md_0.015" refers to the minimum distance between the center of two cutouts on either side of the hood.

D.3 Hood Geometry Variations

Starting from a single inner-hood CAD model, we leverage a suite of Python scripts built on pyMadCAD [44] to automatically generate the full exterior shell and project engineered features onto a reference plane. This end-to-end, workflow transforms one base geometry into numerous distinct hood variants, capturing a wide spectrum of cutout topologies and enabling rich exploration of design and fluid–structure responses. The detailed steps are as follows:

File Format	Description
Hood geometries	
geo_***_clusterID_x_...stl	Surface mesh of the hood geometry from hood of base shell ***, cluster x (see Fig.11)
geo_***_...stl	Surface mesh of the hood geometry from hood of base shell *** with random cutouts (no cluster sampling) (see Fig.11)
Results and other data	
solid (or fluid)_...vtp	Raw CFD and FEA data, contains field values for the physical quantities. VTP is a poly-data format derived from VTK data type.
batch_**.pt	Torch dataset format containing fields for ML training.
x.json	JSON file containing LLM prompts or structured geometry list for creating ML training dataset.

Table 11: Summary of the dataset contents

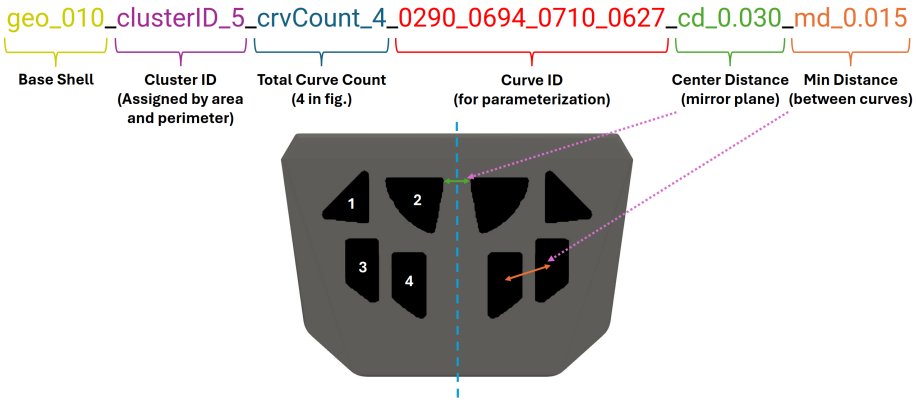

Figure 11: Example hood showing the physical definitions of the geometry naming convention

- **Convex-Hull Shell Reconstruction:** Starting from the inner hood mesh, compute its convex hull to form an outer envelope that closes all recesses. Extrude a uniform offset (e.g. 10 mm) normal to the hull surface to produce the complete 3D hood shell.
- **Planar Projection:** Translate and rotate the shell so that its engineered features face a reference plane perpendicular to the inlet flow. Project the 3D surface orthogonally onto this plane to obtain a binary silhouette highlighting potential cutout regions.
- **Segmentation and Mask Extraction:** Apply the SAM-2 [45] segmentation model to the 2D projection to isolate individual cutout masks. Post-process masks to remove noise and enforce connectivity, ensuring that each mask corresponds to a single-engineered opening.
- **Boundary Sampling and Curve Database:** Trace the contour of each mask and uniformly sample boundary points to generate an ordered point-cloud curve. Store these curves in a parameter database for subsequent re-embedding.
- **Cutout Reintegration:** Map selected boundary curves back onto the 3D shell surface and subtract them to carve precise openings. Enforce symmetry and user-defined spacing constraints to create the final engineered hood variations.

D.4 Clustering Cutouts

Figure 13 delineates nine clusters of engineered cutouts stratified by perimeter and area. Cluster 5 comprises the most elaborate shapes, with a mean perimeter of approximately 0.84 and mean area around 0.045. In contrast, clusters 2 and 7 represent the most compact designs, exhibiting mean perimeters between 0.43 and 0.49 and mean areas near 0.015. In contrast, several intermediate

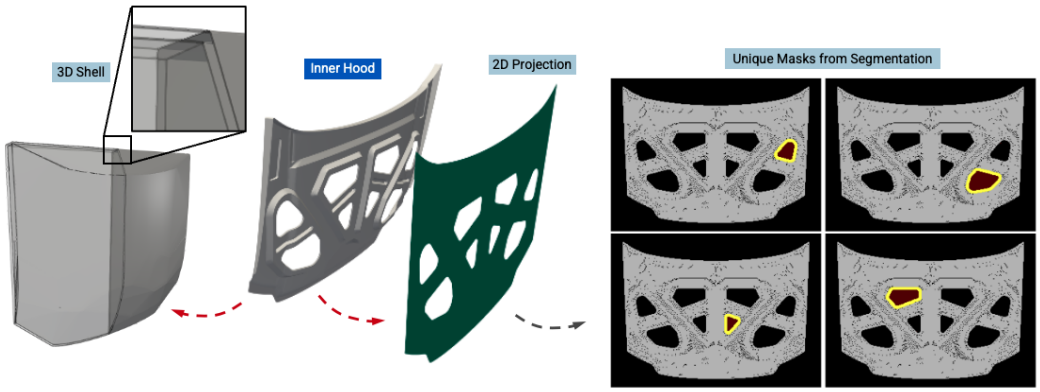

Figure 12: Generation of engineered hood shells from inner-hood CAD. The process begins with the inner-hood surface (second panel), which is convex-hulled and offset to form the complete 3D shell (first panel). This shell is projected onto a 2D plane (third panel) and segmented via SAM-2 [45] to extract unique cutout masks (right panels), each of which defines a boundary curve used to parameterize and re-embed the engineered openings into the final shell geometry.

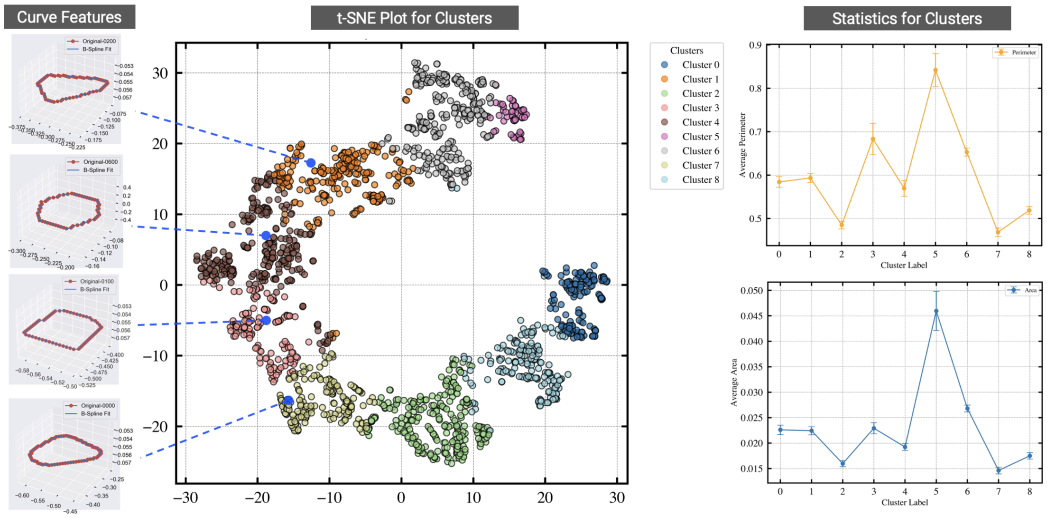

Figure 13: Clustering of extracted cutout curves. Left: representative boundary-curve features (original and B-spline fits) for four sample clusters. Center: t-SNE projection of all curves, colored by cluster label (0–8), illustrating distinct groupings based on perimeter and area metrics. Right: mean perimeter (top) and area (bottom) with error bars for each cluster, highlighting the geometric diversity captured across the nine classes.

clusters (e.g., Clusters 1, 3, and 4) exhibit partial overlap, indicating gradual transitions in perimeter and area metrics and demonstrating that certain mid-range shapes share similar feature compositions. This overlap underscores the continuous nature of our design manifold and suggests opportunities for interpolation between cluster centroids when generating new variants. This calibrated partitioning of cutout topologies provides the dataset with a comprehensive representation of feature scales and intricacies, supporting systematic investigation of geometry-driven variations in fluid–structure response.

D.5 Automated Simulations

We developed a suite of six interconnected Python scripts to fully automate the end-to-end HPC workflow. A primary “manager" script handles mesh generation, solver invocation, and directory I/O for each design. Two “replicator" scripts clone and distribute case directories across compute

pools. Two SLURM [77] utilities generate and submit batch jobs for each case. Finally, a top-level orchestration script monitors node availability, cleans temporary files, assigns pools to nodes, and invokes the subordinate scripts in sequence.

D.6 Analysis of Simulation Results

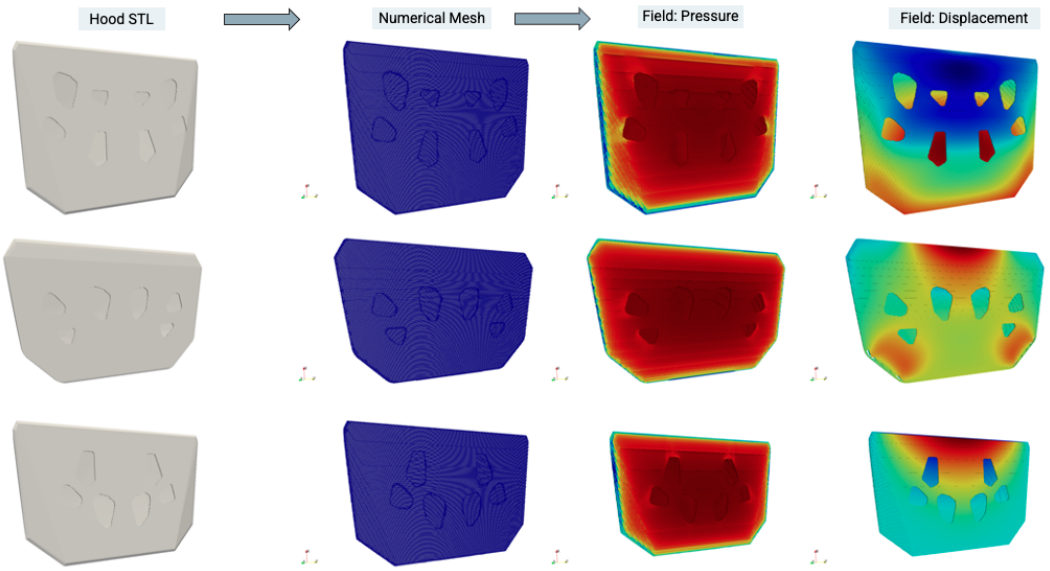

Figure 14: Representative hood variants (left) with corresponding CFD meshes (second column), pressure distributions (third column), and structural displacement fields (right column) for three sample geometries

Each row in Figure 14 illustrates how distinct cutout geometries yield different deformation patterns under identical loading. As the cutout topology varies across the three samples, so does the displacement pattern despite similar looking pressure fields. The large central openings of the top design drive pronounced edge deflections, the smaller scattered apertures of the middle design produce localized deformation around each edge, and the uniformly distributed holes of the bottom variant produce a more evenly distributed displacement profile in the lower half but a higher concentrated deflection on the top. Next, we visualize the simulation results for all the 4k dataset to characterize the global variation in pressure and deformation.

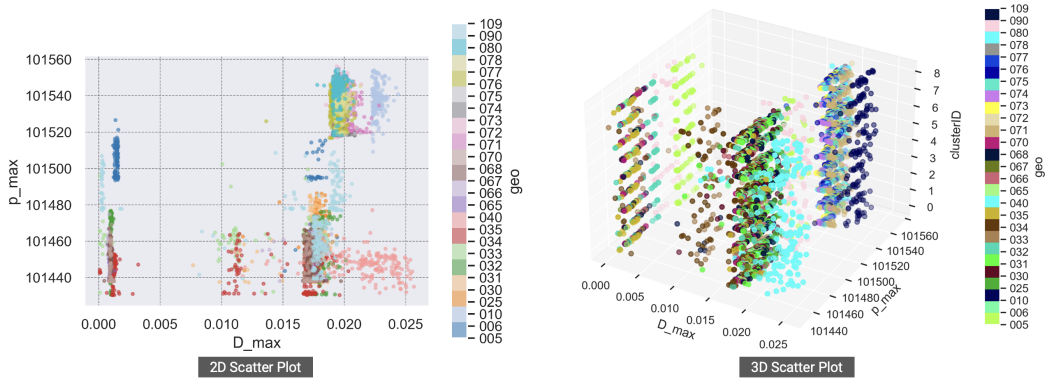

Figure 15: Raw CFD results: maximum pressure p_{max} (Pa) vs. maximum deflection D_{max} (m). Left: colored by geometry index; right: stratified by cluster ID.

The plots in Figure 15 illustrate how geometric variability drives coupled pressure–deflection responses across the dataset. In the 2D scatter (left), each point's color encodes its base-geometry

index, revealing that certain designs, such as “geo 0", consistently exhibit low D_{max} even under moderate p_{max}, whereas others (e.g., “geo 7") manifest higher deflection at similar pressures. The 3D plot (right) layers these same data by cluster ID, showing that extremes in both deflection and pressure occur predominantly in Cluster 5 (IDs around 070–090). Within each cluster band, the spread of geometry-indexed colors demonstrates that individual hood shapes produce unique pressure–deflection signatures. This interplay between cluster-level typologies and specific geometric variations highlights the sensitivity of the structural response to cutout details and underscores the importance of a thorough design exploration.

D.7 LLM Prompt

We designed a structured meta-prompt to guide Gemma3’s vision-LLM into producing paired text–point-cloud training examples. For each hood image and the corresponding variation specification, the prompt instructs the model to generate: (1) a User Input section that formalizes the design request from the base description and variation details; (2) a Chain-of-thought section that explicitly reasons through curve counts, spacing, symmetry, and surface orientation; and (3) a Solution placeholder for the resulting point-cloud output. By enforcing this three-part schema, we automatically extract self-documented examples, complete with transparent intermediate reasoning, suitable for supervised fine-tuning of downstream LLMs on CAD text-to-geometry tasks. The code along with necessary details is shared online and the prompt is shown below.

Meta Prompt

You are a CAD-focused Vision-LLM. Analyze the image provided along with these guidelines:
Description of the base geometry without cuts: description_template
Variation Details: variation_details
Produce exactly three labeled sections—no additional commentary:
1. **User Input** Formulate the user request using the base description and variation details.
2. **Chain of Thought** Show internal reasoning step by step: - Start with: “Let me think through the requirements. . . " - Parse counts, distances, symmetry from the Variation Details. - Identify inner vs. outer face features from the base description. - Plan point-cloud density and cut placement.
3. **Solution** Provide a placeholder for the point-cloud output: “Solution: point_cloud"

D.8 Long-term maintenance

The APCL Lab at the University of Michigan manages the dataset and tracks user-reported issues through the GitHub repository. For long-term accessibility, AutoHood3D is archived on Harvard Dataverse, and our data management practices adhere to the FAIR principles [78] to ensure findability, accessibility, interoperability, and reusability.