DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

SAND #######
LDRD PROJECT NUMBER:
FY25-0783
LDRD PROJECT TITLE:
Replace Human Intelligence with Fast and Smart Geometric Reasoning and Graph Neural Network to Accelerate Next-Gen ModSim Workflows
PROJECT TEAM MEMBERS:
William Roshan Quadros, Nick Winovich, Corey Ernst, Nya Hardy

ABSTRACT

We present an agent-guided approach to CAD geometry decomposition that automates hex/hybrid meshing with graph neural networks (GNNs) to accelerate next-generation ModSim workflows. Our end-to-end pipeline (i) reduces 3D boundary-representation (B-Rep) models to a 2D chordal-axis skeleton (CAT) and then to a 1D bipartite graph of surface and curve nodes, (ii) assigns per-node labels as Cubit® WebCut actions, (iii) trains a multi-action GNN under supervised learning, and (iv) predicts five surface-node and three curve-node actions on out-of-distribution test geometries. Each graph node carries geometric, topological, and meshing attributes drawn from the B-Rep "skin" and CAT "skeleton," with two-way mappings across 3D↔2D↔1D representations to maintain traceability back to 3D CAD. The supervised learning model exhibits stable convergence of the binary cross-entropy loss and achieves 98.7% accuracy on unseen lattice models. To operationalize decision-making, we rank predicted commands by geometric significance and prototyped the agent-guided workflow through the Cubit® Meshing PowerTool GUI. As a stretch goal, we explore reinforcement learning (RL) to reduce or remove label requirements and to learn policies for action sequences that maximize total reward (e.g., size of hex-meshable regions and resulting hex mesh quality). When all-hex meshing is not feasible, the agent assists in producing hybrid meshes—prioritizing hex in critical regions and transitioning to tetrahedral elements (tets) elsewhere—maintaining fidelity while ensuring robustness. The overarching objective is to replace manual, heuristics-based decomposition with data-driven, reproducible automation, cutting meshing turnaround time by orders of magnitude. We anticipate direct impact on simulation workflows through intelligent, scalable decomposition of complex CAD models into hex-meshable subdomains.

1. INTRODUCTION AND EXECUTIVE SUMMARY OF RESULTS

Automated hexahedral meshing for complex CAD remains a critical bottleneck in many modeling and simulation (ModSim) workflows. Mature tools like Cubit® provide robust meshing operations and geometry decomposition operations but still depend on expert user's reasoning

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

when it comes to deciding where to place WebCuts, which surfaces to sweep, how to preserve key features, and how to maintain high mesh quality—introducing latency, variability, and scalability limits for high-consequence simulations (see Figure 1). The fundamental difficulty is not generating elements inside a known meshable subdomains (a.k.a. blocks) but *discovering a valid decomposition of complex 3D CAD* and an action plan that respects sweepability/feature constraints while maintaining element quality. Hex meshes are especially valued in nonlinear solid mechanics, hydrodynamics, and aerodynamics for their efficiency (fewer elements and faster analyses), accuracy, and convergence; in some tasks, hex is the only viable option [1] . Yet even the best classical methods (mapping, sweeping) are reliable mainly on simpler geometries and typically require substantial decomposition [2] [3] [4].

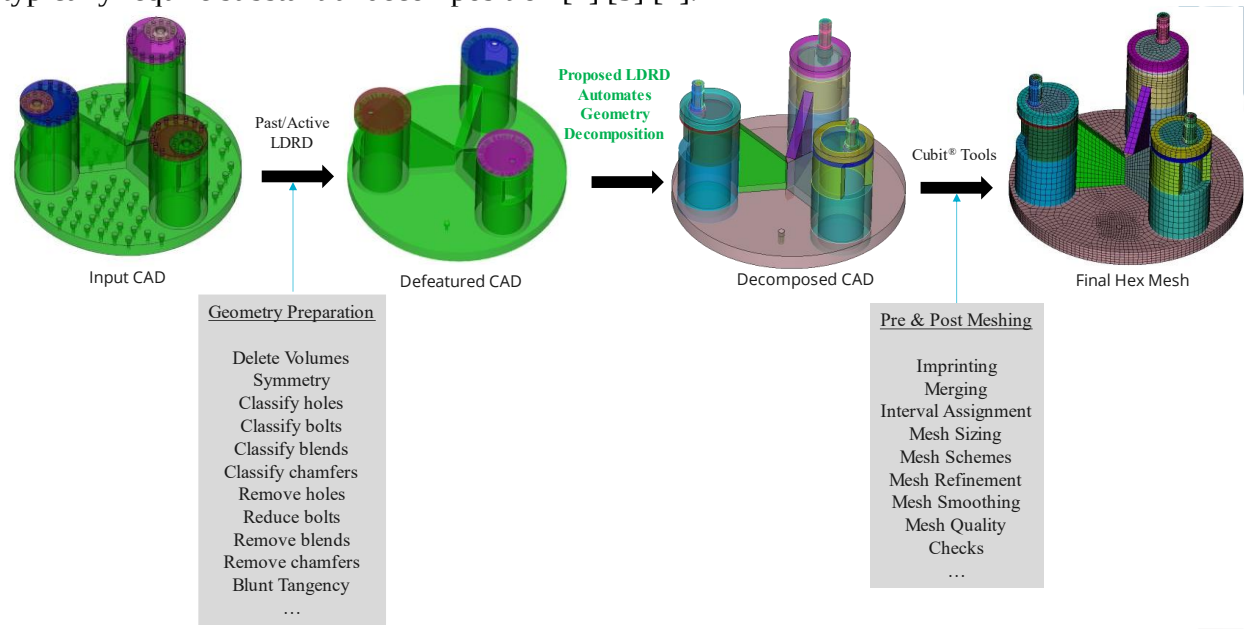


Figure 1 Automating geometry decomposition for hex meshing in Cubit®

Classical hex meshing methods such as mapping and sweeping have long delivered high-quality hexes on model classes that fit their assumptions, and are widely implemented in commercial systems; however, they require user preparation (reference blocks/surfaces) and struggle on arbitrary shapes [2] [3] [4]. Grid-based approaches broaden coverage but face trade-offs: e.g. grid-based Sculpt scale well but does not resolve sharp boundary features [3] [5]. Similarly, Whisker-Weaving and Plastering produce good surface structure yet may leave unmeshable interior cavities [6] [7]. Frame-field and volumetric parameterization yields structured, high-quality elements but often needs careful feature preparation or user-guided fields [8] [9]. Tet-to-Hex conversion via a template is simple but can create high node degrees and local quality issues. Comprehensive

surveys concur that fully automatic, robust all-hex meshing for arbitrary 3D CAD remains an open challenge [1].

Geometry decomposition is a dominant approach in industry and much of the literature embrace geometry decomposition followed by map/sub-map/sweep as the most reliable path to high quality hex meshes. Commercial tools (Cubit®, ICEM CFD, Abaqus) expose powerful geometry decomposition operations but rely heavily on user-created reference objects and Boolean splits. Automatic variants detect swept volumes [10] [11] [12] [13] [14], apply medial axis-based partitioning [15] [16] [17], or use feature recognition and dual/cross-field cutting loops [18] [19]; however, they show success on targeted geometries (e.g. thin-wall solids) but lack universal robustness. Quadros has published a hex-dominant meshing algorithm using continuous, more accurate, skeleton called Medial Axis Transform (MAT) as shown in Figure 2 [20] [16] [21] [22]. This approach takes advantage of the mapping from skin (B-Rep) to skeleton (MA) and non-manifold junctions of the MA (i.e. regions where advancing front cause singularities). MAT is a theoretically well-defined skeletal representation, but it is hard to extract it on complex CAD parts in practice. Therefore, the current approach proposes discrete Chordal Axis Transform (CAT) skeleton; some of the key characteristics of the skeleton such as valency, thickness, number of loops, etc. are used as features to train the GNN. Skeleton-guided and multi-cube strategies (e.g., T-mesh for IGA) further demonstrate the value of an interior "organizing" structure for decomposition [23] [18] [24].Semi-automatic systems combining medial geometry with geometric reasoning illustrate strong human–computer teaming, yet still demand expert input [25] [26].

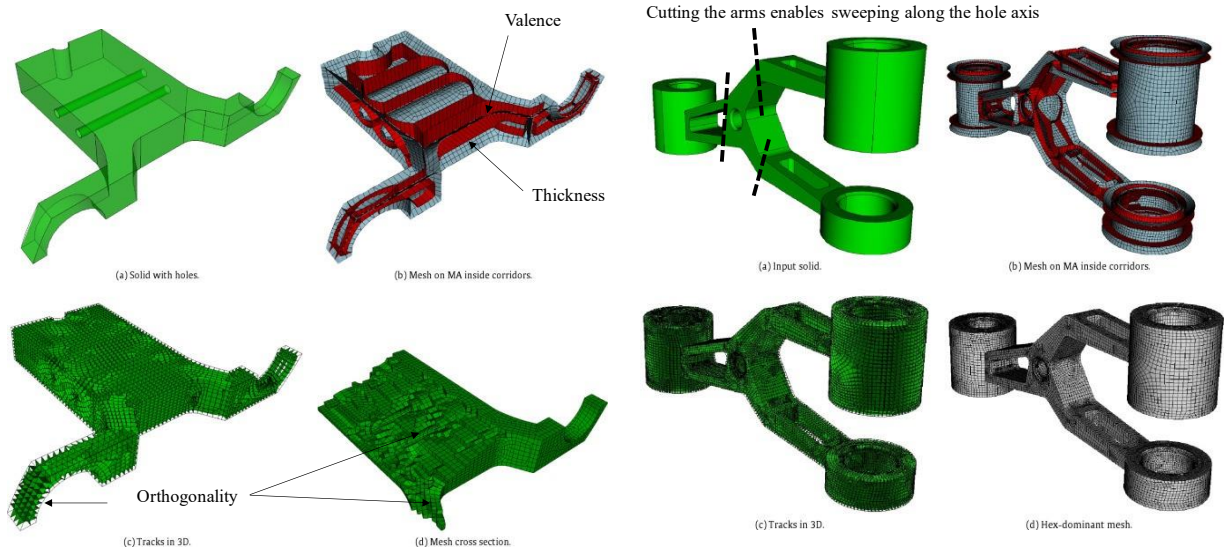


Figure 2 Skeletal features provide inner complexity of 3D CAD

Because ground-truth sequences of decomposition actions are expensive and ambiguous to label at scale, Reinforcement learning (RL) is a natural fit. Quadros in collaboration with Illinois Rocstar LLC researchers [27] introduced Auto-Hex, combining geometric reasoning and artificial intelligence for automatic geometric decomposition and robust hexahedral meshing of complex geometries using a reinforcement-learning loop with the meshing environment Cubit®. Given the boundary representation (B-Rep) and a chordal-axis skeletal object, prominent geometric features (e.g., T-junctions, sweep directions, through-holes) are detected. An RL policy then proposes actions (Cubit® WebCuts) based on B-Rep and skeletal features; rewards from the meshing environment train the agent to favor decompositions that yield hex or hex-dominant meshes over time. DiPrete et al. work explored RL loops with geometric predicates inside meshing environments and demonstrated policy learning on planar shapes [28]. Zhang et al. [29] formulate 3D block decomposition as a sequential decision problem and train a Soft Actor-Critic agent (bootstrapped via imitation learning) to select reference-face operations (planar, extruded-surface, revolution-surface, trimmed-surface) in a tree-structured MDP induced by recursive splits. A face-adjacency graph with discretized surface/curve samples encodes B-Rep state. Trained on ABC and NIST CAD sets, the policy attains high success and meshable ratios showcasing RL's strength in ordering cuts for long-horizon hex objectives [29] [30] [31].

In summary, we built an end-to-end, agent-guided pipeline that reduces 3D B-Rep models through a 2D chordal-axis skeleton (CAT) to a 1D bipartite graph of surface and curve nodes with geometric, topological, and meshing attributes, then GNN agent learns to predict multi-action WebCuts directly on this graph via supervised learning (see Figure 3). A supervised heterogeneous GNN, trained with binary cross-entropy across five surface and three curve actions per node, converges smoothly and reaches ~98.7% action accuracy on a synthetic lattice benchmark, substantially outperforming a valency-only MLP baseline. This proves that supervised learning is an effective strategy for meshing complex targeted parts/components by training on similar randomized synthetic data and labels. On a few general CAD graphs, the supervised GNN model produces ~85% accurate WebCut commands, which significantly outperforms the suggestions provided by the currently released product Cubit 17.04 [32]. But further work is required to clearly quantify the accuracy and generality of supervised GNN model. When full all-hex is not feasible, the risk mitigation strategy is to generate hybrid meshes (hex in critical regions, tets elsewhere with transition pyramid elements) to meet required fidelity and ensure timely analyses (see Figure 4). Finally, a prototype reinforcement-learning architecture demonstrates promise for label-free planning and action sequencing toward maximizing hex-meshable volume and downstream mesh quality, setting the stage for broader deployment across general CAD assemblies. Parallel advances in deep learning for B-Rep CAD (graph/sequence models and large CAD datasets such as ABC) further strengthen the foundation for learned geometry understanding and decision-making [30] [31] [33] [34] [35].

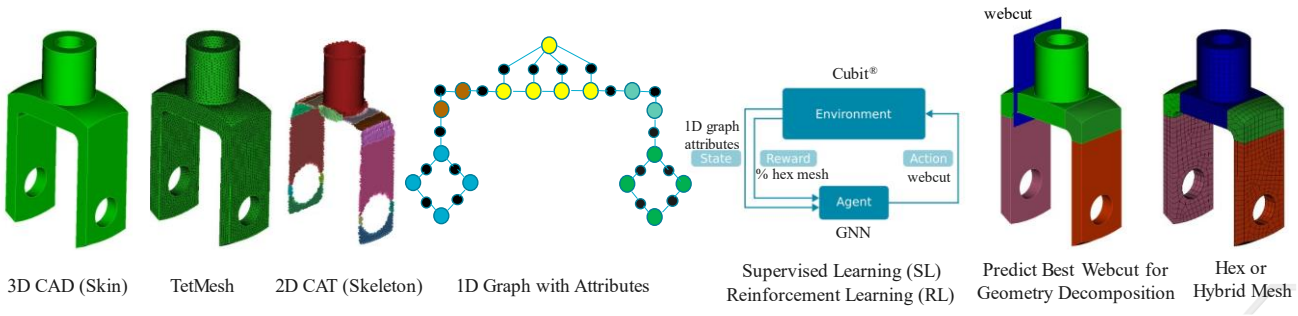


Figure 3 Overview of GNN agent guided decomposition via supervised/reinforcement learning

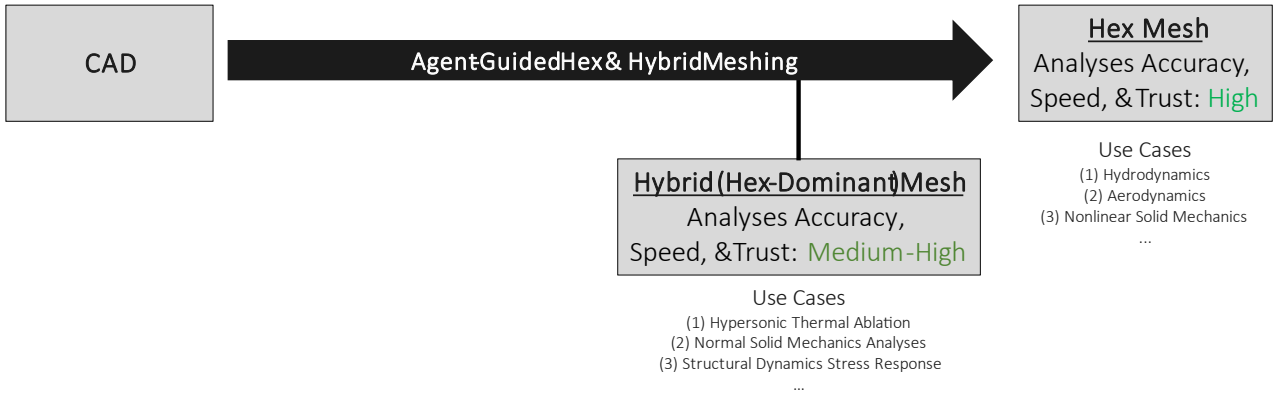


Figure 4 Hex/Hybrid (Hex-dominant) meshes are required in many use cases

2. DETAILED DESCRIPTION OF RESEARCH AND DEVELOPMENT AND METHODOLOGY

2.1. Synthetic Training Data Generation

To train the supervised heterogeneous GNN agent, we procedurally synthesize CAD "lattice" solids with internal structures inducing a wide range of curve valencies, junction types, and sweep patterns. This training/test set is motivated by a mission part that contains lattice-like structure and is also motivated by the fact that these lattice structures contain polycube type Cartesian axes-oriented entities [36]. This is just one exemplar and similar class of training set can be generated for other targeted parts of interest.

Hyperparameters used in creating random lattice training and test data:

```
slab_thickness = 0.1
slab_length = 0.5
num_slab_planes = 2 # Number of values in the coordinates vector
offset_distance = slab_length / 2.0
creation_probability = 0.60 # Probability to create a slab at a given location (default = 0.65)
hole_probability = 0.35  # Probability to create a hole in a slab
hole_radius = slab_length / 8  # Radius of the cylindrical hole (adjust as desired)
hole_axis_factor = 2.0 # increase slab thicking by this factor to create length of hole tool
max_slabs = 300 # Maximum number of slabs to create
N = 10 # Number of lattice files to create
unite_volumes = True  # Option to unite all slabs into a single volume
tolerance = 0.001  # Tolerance for floating-point comparison
curved_lattice = True  # Generate curved lattice
curved_lattice_rotation_ang = 10
normal_holes = True  #If True, hole axis is  always normal to the slab
```

Each lattice in our exemplar is assembled from slab primitives arranged along Cartesian coordinate axes using parameter `num_slab_planes`; the planes are spaced by an `offset_distance = slab_length/2.0`, so that neighboring slabs intersect and create meaningful interior junctions (see Figure 5). Individual slabs are extruded volumes of thickness `slab_thickness` and slab dimensions along length and width is `slab_length`. At each grid location, a probability `creation_probability` decides whether a slab is instantiated; this mechanism yields sparse-to-dense configurations within a single parameterization and prevents overly regular training examples. To introduce inner loops, concave surface curvatures, and through-features, holes are optionally bored through selected slabs with probability `hole_probability`; the cylindrical tool uses `hole_radius = slab_length/8` and is extended along the local normal by a factor `hole_axis_factor` to guarantee complete removal through the volume. When `normal_holes = True`, the drill axis is constrained to be normal to the slab, eliminating grazing cuts and producing well-conditioned boundaries.

The generator caps the total number of created slabs at `max_slabs` to control geometric complexity and runtime. For each dataset call, it produces `N` distinct lattice files, thereby sampling different Bernoulli outcomes for slab and hole creation and diversifying junction layouts. When `unite_volumes = True`, all slab bodies are united into a single watertight volume (see Figure 6); this step, combined with a geometric `tolerance = 0.001`, removes coincident faces and hairline gaps that would otherwise complicate tetmeshing and skeletonization. To further expand geometric diversity, the generator can produce curved assemblies by enabling `curved_lattice = True`, applying a gentle twist of `curved_lattice_rotation_ang = 10` degrees (about a prescribed axis) across the slab stack. This curvature introduces nontrivial surface normals and oblique intersections, enriching the training/test set to resemble industrial CAD parts.

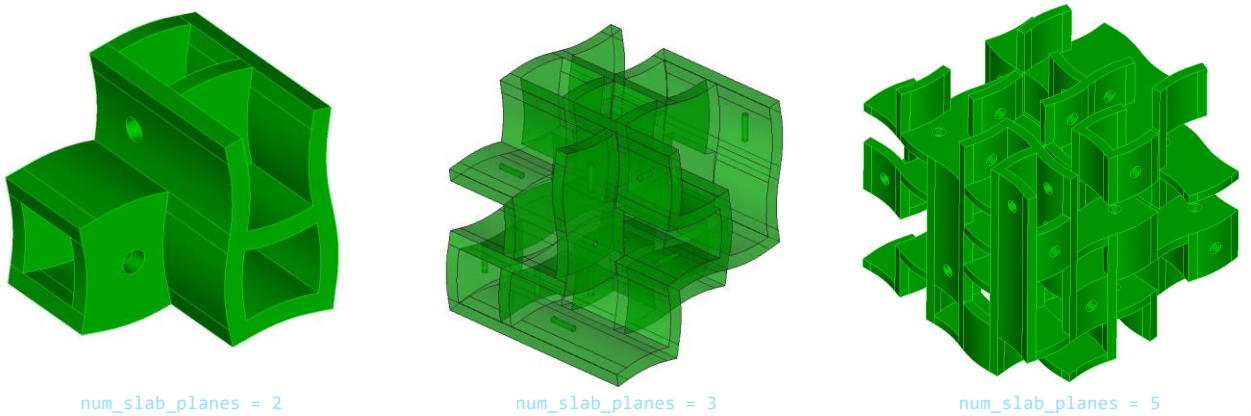


Figure 5 Random lattice data set with different `num_slab_planes`

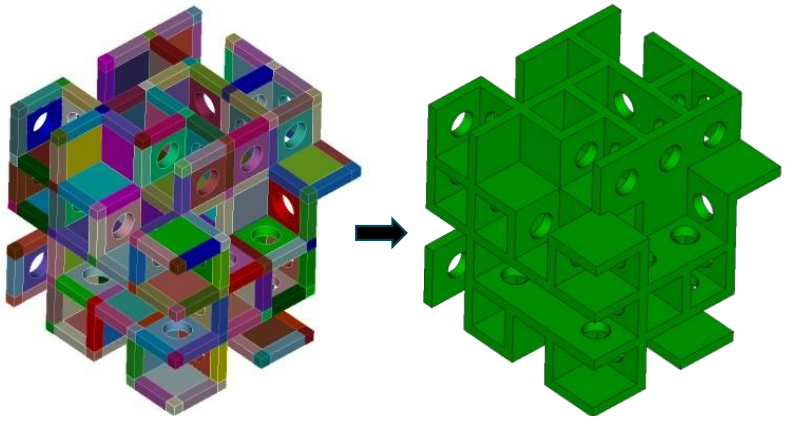


Figure 6 Union of random slabs create a lattice structure

2.2. Reducing 3D CAD to 1D Bipartite Graph via Chordal Axis Transform (CAT)

We begin with a contiguous B-Rep CAD model and generate a discrete Delaunay tetrahedral mesh without seeding any interior points in the volume to ensure that the skeletal structure we extract is driven by the boundary (see Figure 7) [37]. From this conforming tetmesh, we cut internal tetrahedral edges at mid points and assemble the resulting chordal vertices into a discrete chordal-axis complex by grouping across adjacent tetrahedra to form a set of CAT faces composed of triangles and quads that represents the skeleton. We then trim the chordal axis at convex boundary edges to remove unwanted branches near highly convex features. This produces a topologically consistent 2D skeletal surface that aligns with perceived thickness and flow directions inside the solid (see Figure 8).

We then establish a two-way mapping between CAT surfaces and curves and B-Rep surfaces and curves, respectively by traversing tetmesh adjacencies and recording which boundary surfaces influence each CAT faces. This mapping supports traceability during learning and post-processing: any action predicted on a skeleton entity can be projected back to an actionable WebCut commands with BRep IDs. We then take dual graph of the 2D CAT to construct a 1D heterogeneous bipartite graph whose surface nodes represent CAT surfaces and whose curve nodes represent the CAT curves (i.e., interface & junction curves between CAT Surfaces). Incidence relations between CAT surfaces and curves become bidirectional edges in the graph (surface↔curve). Finally, we expose a Cubit® command, create chordal axis, that writes the graph to disk as paired CSV files for surfaces, curves, and their edges; these files include the two-way mapping tables that link graph nodes back to B-Rep entity identifiers and CAT patches for later visualization, auditing, and command synthesis.

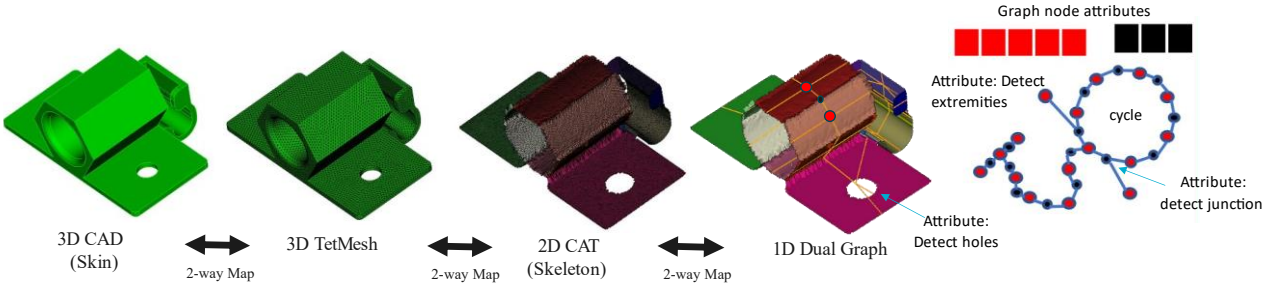


Figure 7 Reduce 3D CAD to 1D graph with attributes via Chordal Axis Transform (CAT)

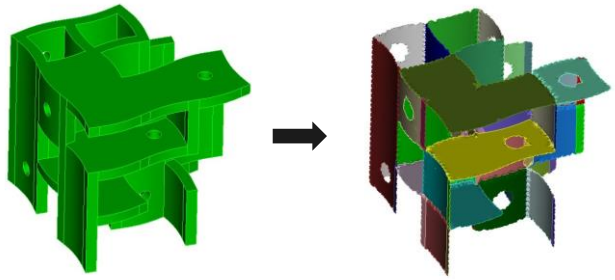


Figure 8 2D Chordal Axis of a random 3D lattice model

2.3. Feature Engineering and Action Labeling

Table 1 and Table 2 tabulate the graph surface and curve node features, respectively to capture geometric, topological, mapping, and meshing characteristics of a 3D CAD. Using the Cubit® API, we compute node attributes related to the B-Rep and CAT entities so that each graph node carries both outer-skin and inner-skeleton characteristics. Table 1 tabulates surface node attributes such as centroid coordinates; area; estimated thickness; principal radial, axial, and circumferential

sweep directions and their directional cosines; planarity flags; average dihedral angle across the CAT patch; orthogonality measures between principal directions; the number of interior loops; and counts/summaries of incident curves and neighboring surfaces. Table 2 tabulates curve node attributes such as start/end vertex coordinates; curve length; linearity flags; curvature proxies; valency; identifiers of incident surfaces (stored as invariants); and local direction vectors from start to end. Both node types include additional meshing cues such as sweepability indicators and neighborhood thickness/angle summaries. In our implementation, each node type contributes more than fifty scalar features after normalization, providing a rich yet compact geometric and topological description suitable for message passing. Figure 9 shows that valence at a curve node and surfaces node identifies junctions and extremities, respectively. Figure 10 shows the three principal sweeping directions at a surface node. Figure 11 shows geometric characteristics of a 3D solid such as average thickness, area, and presence of holes via surface node features.

Table 1 Graph Surface Node Features

Category	Feature	Use
Geometric	`cencentroid_{x,y,z}, area, thickness, angle, is_planar`	Capture geometric characteristics
Topological	`valValence, num_of_loops`	Capture topological characteristics
Map	`BRep Parent IDs; Spatial Info of Parent IDs (e.g., centroid_{x,y,z})`	IDs are used in Actions
Meshing	`Radial, Axial, Circum sweeping direction vectors (spatial info); corresponding B-Rep ID spatial info`	Direction vectors convey sweepability; IDs used to bind WebCut actions/commands

Table 2 Graph Curve Node Features

Category	Feature	Use
Geometric	`start_{x,y,z}, end_{x,y,z}, length, is_linear`	Geometric descriptors of CA curves
Topological	`valence`	Junction descriptor
Map	`Parent Curve IDs for Actions 1-3 (stored as flags); Spatial info from B-Rep: start_{x,y,z}, end_{x,y,z}`	IDs used in Actions
Meshing	`Parent curves' start/end vertex IDs for Actions 1 & 2 (flags); Parent1 curve end surfaces spatial info; Parent1 curve end surfaces B-Rep IDs for Action 3`	Used to resolve plane-normal and sweep targets

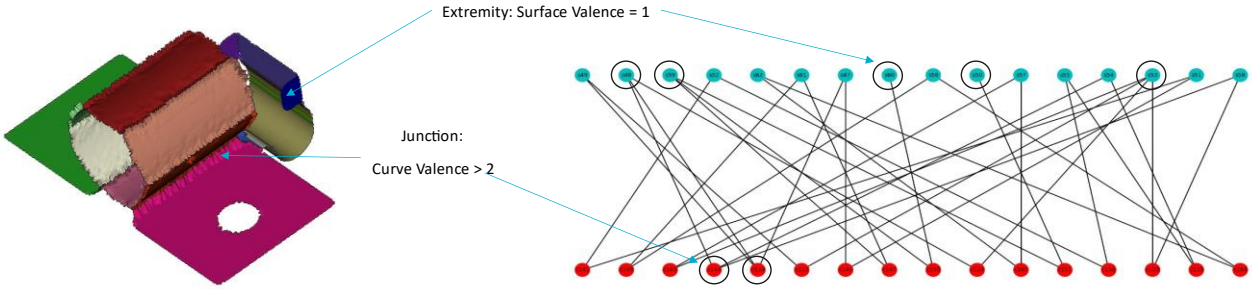


Figure 9 Valence at Curve and Surface nodes identifies junctions and extremities, respectively

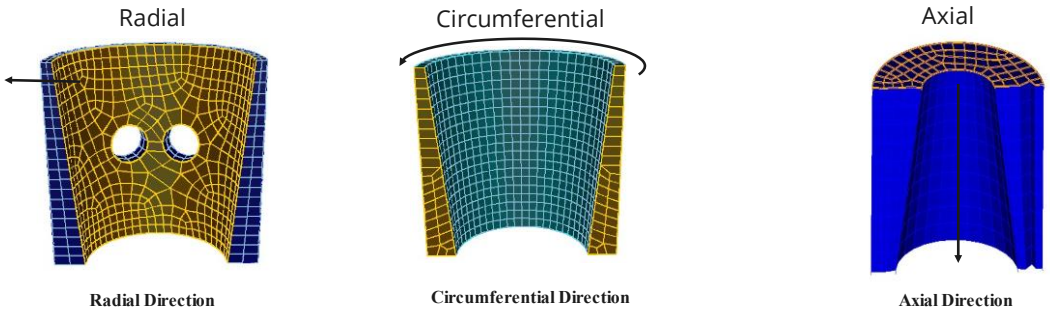
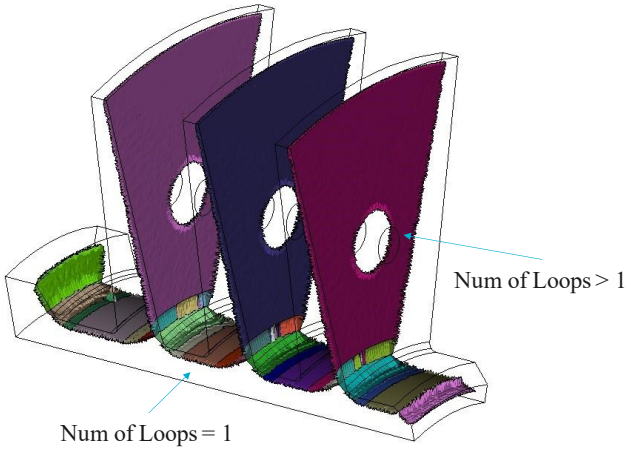
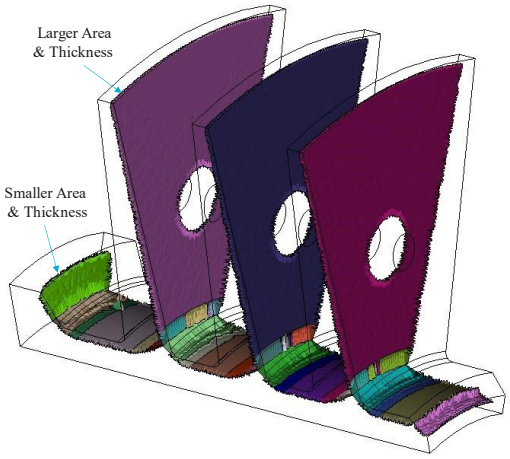


Figure 10 Surface node feature includes three principal sweeping directions



Surface Node Feature: Number of Loops



Surface Node Features: Area & Thickness

Figure 11 Surface node features include area, average thickness, and number of loops

Before training the GNN agent, we sanitize features to prevent overfitting to dataset-specific naming schemes. Wherever the graph stores a B-Rep or CAT identifier for source/target surfaces, linking curves, parent curves, and related entities, we replace raw IDs with binary presence flags and zero-fill any coordinates or direction vectors tied to missing entities. As we normalize all principal sweeping direction vectors to unit length, they reflect orientation without encoding scale.

We then attach multi-action labels to each node by applying rule-based geometric predicates derived from the CAT and B-Rep mapping: five action bits are defined on surface nodes (i.e., Action 1&2: sheet-extended from top and bottom parent surfaces; Action 3,4,&5: sweep along a linking curve in a specified principal direction), and three action bits are defined on curve nodes (i.e., Action 1&2: plane normal to parent curve at start and end vertex; Action 3: sweep end-surface along the parent curve). Each bit is set to 1 when the required availability of B-Rep references (e.g., parent surfaces, linking curves, endpoints) and local geometric thresholds (planarity, angle, and sweepability) are satisfied; otherwise, it is 0. This procedure yields feature and label tables for supervised learning that are invariant to arbitrary ID permutations while preserving all geometry, topology, and meshing characteristics that matter in geometry decomposition.

2.4. Supervised Training of Heterogeneous GNN

We cast geometry decomposition as multi-label node classification on a bipartite graph. The network is a heterogeneous GNN implemented using Pytorch Geometric library with message passing on the bidirectional edges connecting surface and curve nodes. Unlike traditional neural networks (which work on fixed size data/images), GNNs can be applied to data stored in graphs with arbitrary connectivity (Figure 12). K-level compute graphs and aggregation via message passing captures the global landscape/complexity of graph (derived from 3D CAD) through multiple message-passing iterations. We can predict WebCut actions (i.e. labels) at nodes using binary cross entropy (BCE) loss function.

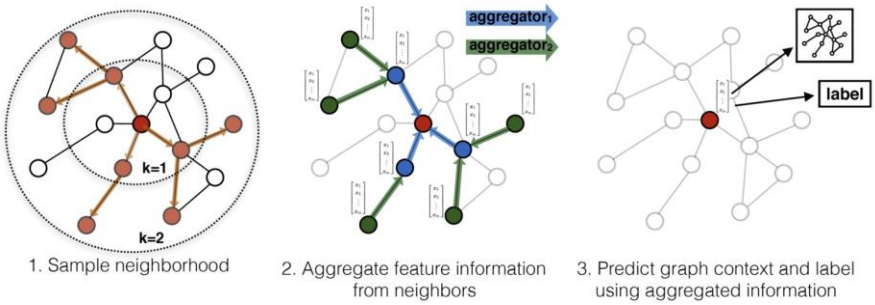


Figure 12 GNN enables message passing on k-level compute graph in a heterogeneous graph

The architecture includes surface and curve node types whose features are processed by a stack of three message passing layers (GraphSAGE/GCN variants) [38] using maximum aggregation, followed by rectified linear (ReLU) activation functions, and culminates in two independent heads: a surface head that outputs five logits and a curve head that outputs three logits (see Figure 13). Sigmoid activations convert logits to probabilities for node classification.

During training, the binary cross-entropy objective is summed across all action channels and both node types (five surface and three curve outputs) and minimized end-to-end. We use a random 80/20 train/test split, train for 2000 epochs with a batch size of 10 graphs for each training step, and optimize with Adam plus weight decay. Learning rates are gradually reduced using an exponential decay schedule until a minimum learning rate is reached, at which point the learning rate remains constant for the remainder of training. We log per-head (surface, curve) and aggregate losses, export TensorBoard traces, and automatically generate loss plots. The observed monotonic decline and stabilization of both per-head and aggregate BCE plots indicate stable learning dynamics and adequate capacity for the task as the training data set size increases (i.e. 10, 100, and 300 lattice CAD models with associated CSV files).

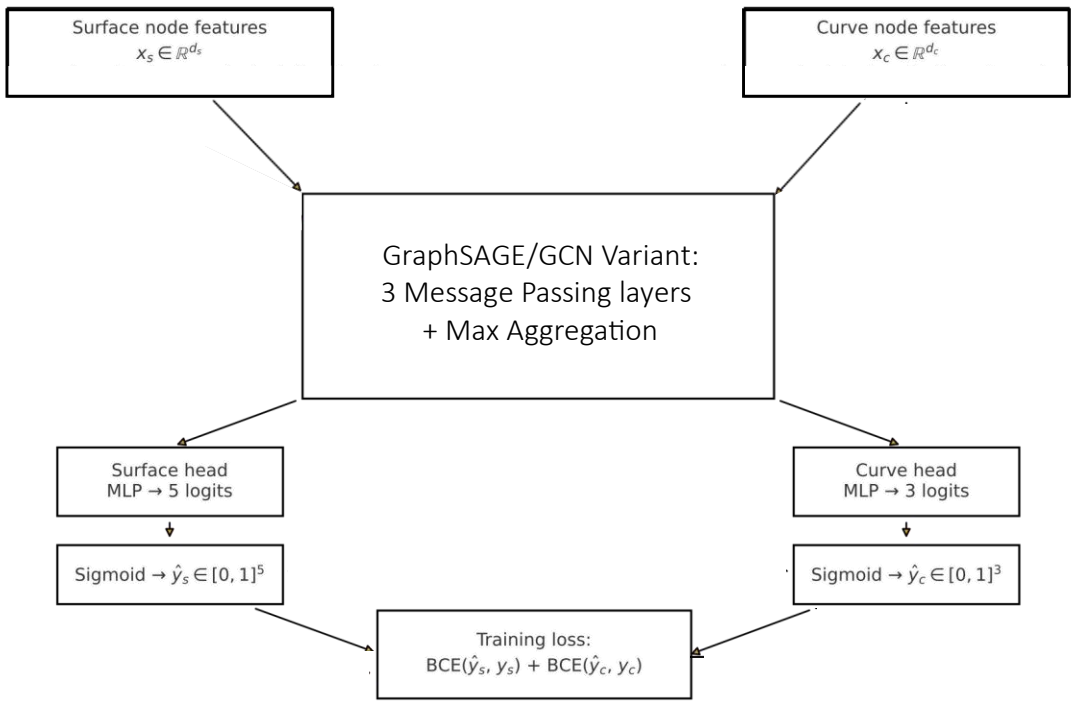


Figure 13 Heterogeneous GNN architecture for multi-label node classification

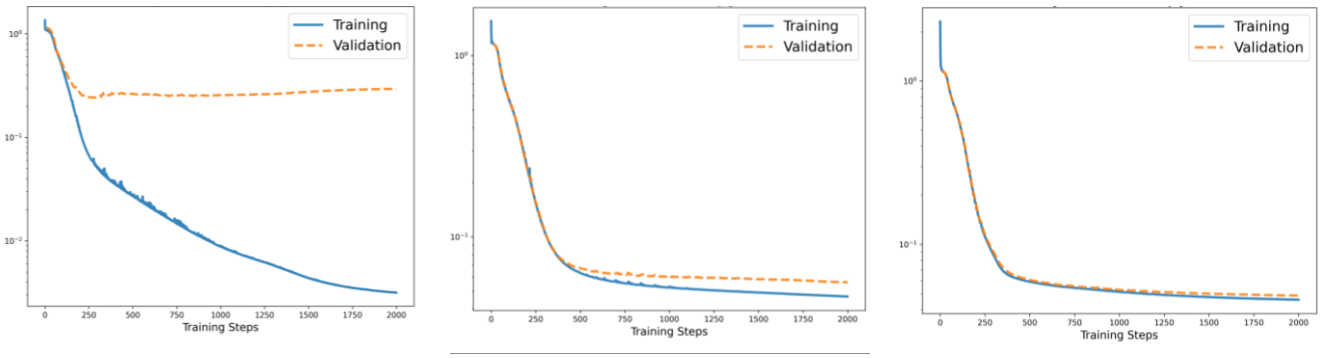


Figure 14 Binary cross entropy (BCE) training loss on 10/100/300 training data

PyTorch-Geometric pseudocode of the core loop:

```
data = HeteroData(...)                # surfaces.x, curves.x,
edge_index[('surfaces','sc','curves')]...
model = BipartiteGNN(in_s=SURF_DIM, in_c=CURVE_DIM, hidden=H, out_s=5, out_c=3)
opt   = torch.optim.Adam(model.parameters(), lr=2.5e-3, weight_decay=5e-4)

for epoch in range(2000):
model.train()
opt.zero_grad()
out_s, out_c = model(data.x_dict, data.edge_index_dict)   # logits
loss  = BCEWithLogits(out_s, y_s).mean() + BCEWithLogits(out_c, y_c).mean()
loss.backward()
torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
opt.step()
# eval on held-out graphs; log BCE_s, BCE_c, BCE_total; save checkpoints
```

2.5. Predicting WebCuts on Test Set

At inference, we first load the trained checkpoint, call Cubit®'s create chordal axis command to export graph in *.csv, and forward once to obtain surface and curve node actions in the new *_predict.csv file as shown in Figure 14. The pipeline achieves 98.7% action accuracy on our lattice test set, with all five surface actions and three curve actions correctly recovered in most cases as shown in Figure 19.

Figure 16 shows a deep dive on curve node C33 and action 3 (i.e. sweep the end surface along the parent curve), which relies on correct identification of the end surface, reliable orientation of the parent curve tangent, and adequate local thickness; the model's precision remains high even near multi-valent junctions where curve orientation is ambiguous.

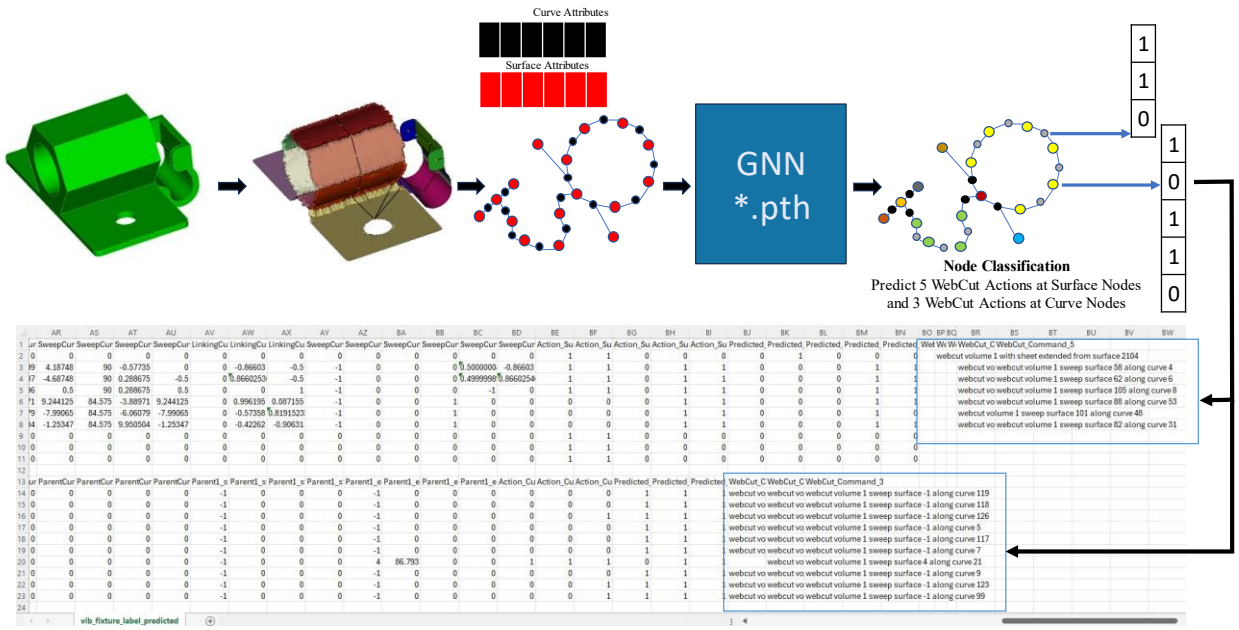


Figure 15 Workflow for predicting WebCut solutions using GNN model

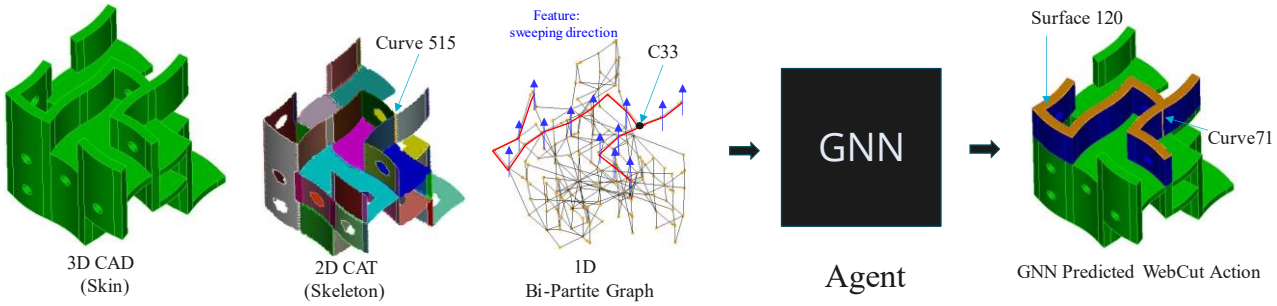


Figure 16 Deep dive at Curve node C33 with GNN predicted action

For transparency, the test script writes surface and curve prediction tables to CSV, with columns for true action bits, predicted bits, probabilities, and the corresponding global IDs that tie back to the B-Rep skin; this enables straightforward reproduction of the previews and rapid triage of any mispredictions.

2.6. Integrate Agent-Guided Solutions in Cubit® Meshing PowerTool GUI

To productionize the workflow, we propose integrating the trained GNN's predictions as a ranked, constraint-checked command plan inside the Cubit® Meshing PowerTool (see Figure 17). The GUI loads the per-node CSVs, restores the local↔global mapping, and renders a prioritized action list, i.e., the agent-suggested WebCut commands under "Possible Solutions" (e.g., sheet-extend from a parent surface, plane-normal-to-curve at a specific endpoint, sweep-surface-along-curve). The user may accept or skip any solution; the powertool repopulates a new set of solutions after executing current selected solution. Because every action originates from the graph with preserved two-way mappings, any generated script is auditable and can be tied back to both the CAT skeleton and the original B-Rep. This integration closes the loop from graph reasoning to executable all-hex decomposition without requiring users to memorize command syntax or manage the ordering constraints manually.

We experimented with a ranking scheme based on geometric significance: larger surface area or curve length yield higher priority. This can be formalized via a small reinforcement learning value network to sort candidate commands based on accumulated reward (e.g. faster mesh convergence, fewer manual overrides).

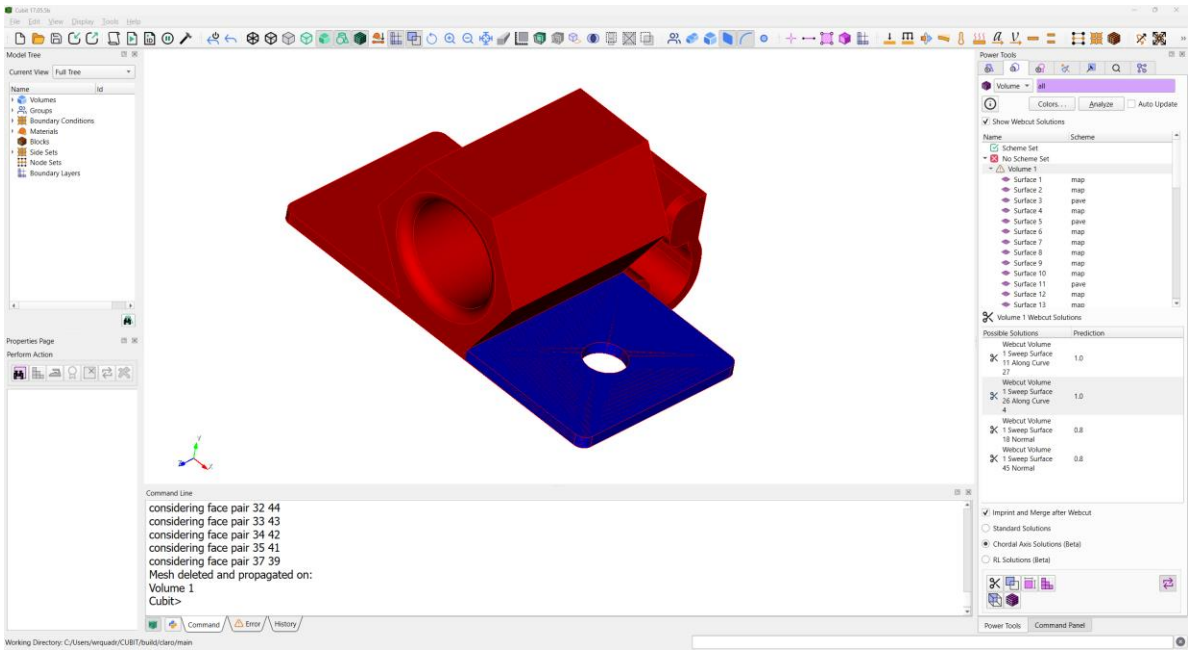


Figure 17 Prototype of proposed decomposition workflow in Cubit® Meshing Powertool

3. EXTENSION TO REINFORCEMENT LEARNING

To move beyond per-node action prediction and learn action ordering on general CAD, we cast geometry decomposition as a sequential decision process and train an RL agent in the same environment used for supervised learning. The state is the bipartite surface↔curve graph (CAT-derived skeleton plus B-Rep "skin") with the identical feature set used in SL; we maintain the local↔global maps so every decision targets a concrete B-Rep face/curve. The action space comprises parameterized WebCut operations anchored at graph nodes (e.g., radial/axial/circum sweeps on surfaces; cut/sweep actions at curves). At each step, the agent proposes a distribution over valid WebCuts; we mask infeasible actions using geometric predicates and sample a valid command, apply it in Cubit®, and update the graph/state. The reward combines (i) the incremental increase in hex-meshable volume percentage and (ii) downstream mesh quality proxies (e.g., sweepability, Jacobian/angle surrogates), with penalties for non-manifold cuts, degraded quality, or dead-end decompositions; a small step penalty encourages shorter plans. Episodes terminate when no further improvement is possible or a hex-meshability threshold is met. We train with modern policy optimization (e.g., PPO/SAC) and warm-start the policy from the supervised model (behavior cloning on our synthetic lattice training set and the held-out test cases), which stabilizes exploration and accelerates convergence. This RL extension preserves data, features, and instrumentation from SL, but adds the closed loop—propose → validate → apply → mesh → reward—that lets the agent learn the sequence of WebCuts required to achieve all-hex, high-quality decompositions on previously unseen, complex CAD models.

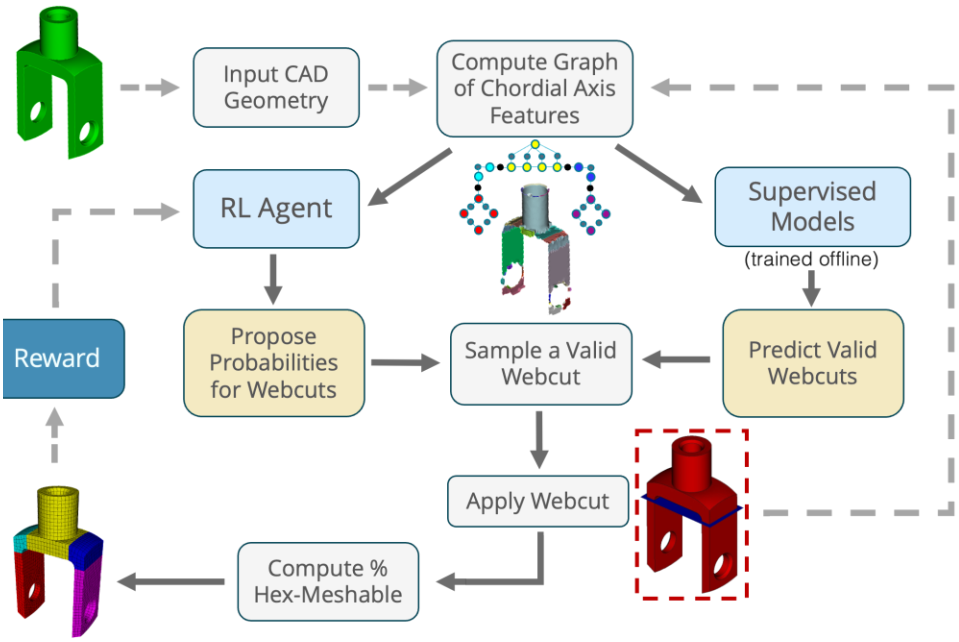


Figure 18 Training workflow for Reinforcement Learning agent

4. RESULTS AND DISCUSSION

The prediction pipeline was tested on a hold-out dataset consisting of 20% of the available CAD files that were not seen during training. The predicted WebCut commands were structurally viable, had correct syntax, and aligned functionally with true labels as shown in Figure 19. A very small percentage of discrepancies arose due to label noise or previously unseen geometric configurations. To improve, we scaled training data up to 500+ randomly synthesized CAD models and observed ~ 3–5% accuracy gains. More enhanced features, larger hidden dimensions, and more message-passing layers could further help improve the accuracy and robustness of predicted WebCut commands.

We benchmarked workflow speed, i.e., a typical CAD graph of ~100 nodes take ~0.1s for feature read and inference, and ~0.05 s to generate commands. The end-to-end prediction significantly reduces manual modeling effort, i.e., significantly reduces trial-and-error decomposition efforts. Most mesh automation methods rely on heuristics or human supervision. Our agent-guided approach enables generalized, learned decision-making from data across different targeted CAD geometries via supervised learning, with better adaptability to new geometries via reinforcement learning.

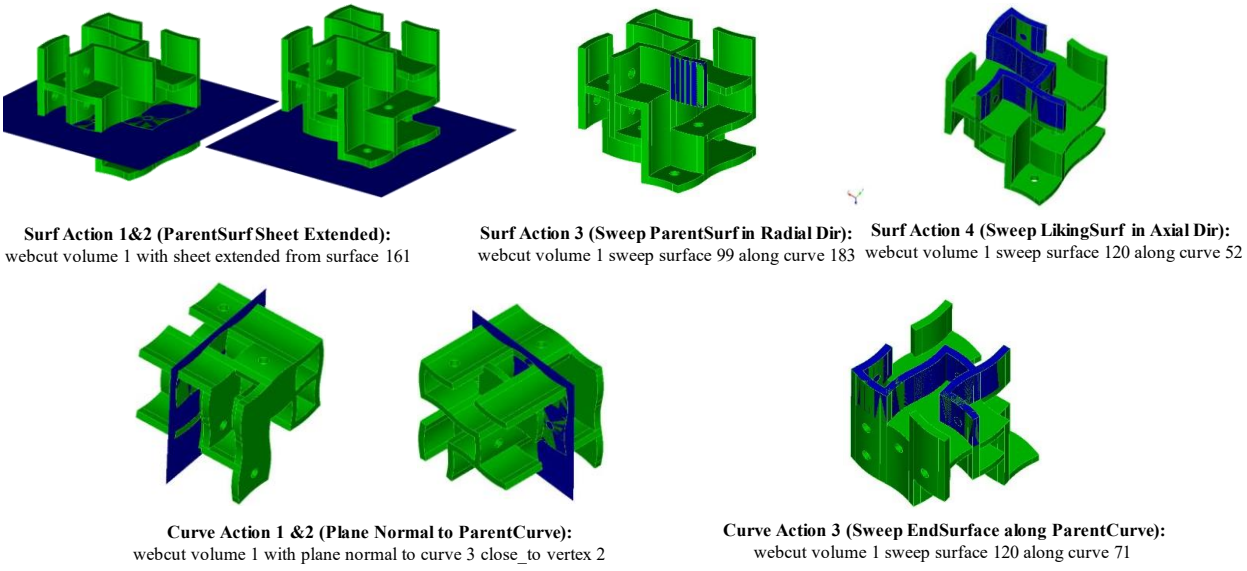
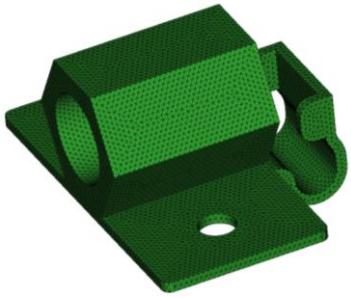
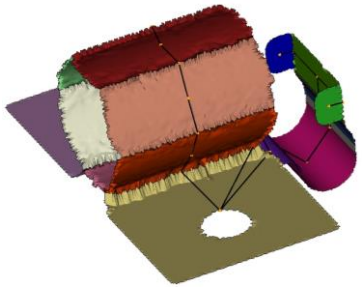


Figure 19 WebCut previews highlighting five surface actions and three curve actions.

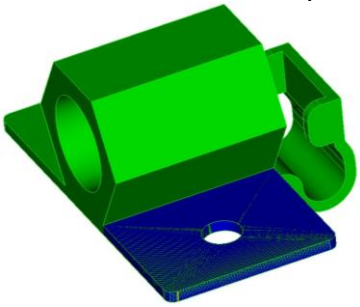
We further validated the supervised learning approach on general 3D CAD models Browell, PlumbingFixture, and JetBlade by previewing the predicted WebCuts in Cubit®. The model produces sensible decomposition WebCut commands as shown in Figure 20 to Figure 22 with greater than 85% accuracy.



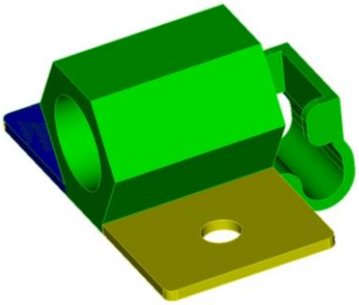
(a) Tetmesh with no interior points



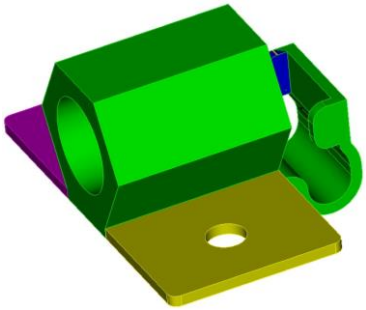
(b) Reduced 2D CAT and 1D Graph



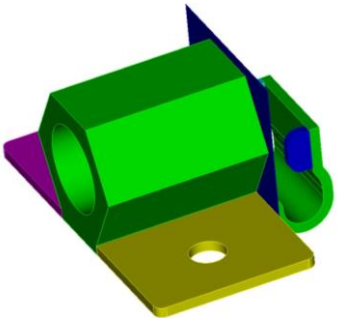
(c) GNN predicted action at surface node



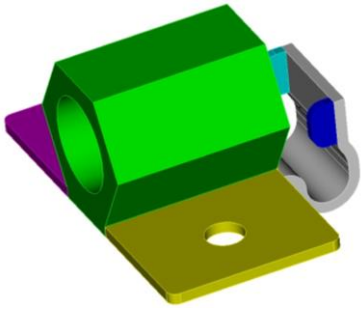
(d) GNN predicted action at surface node



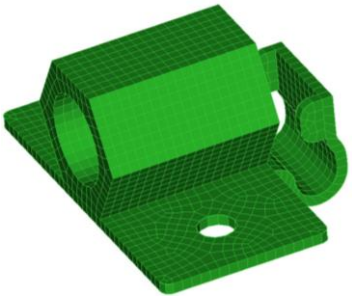
(e) GNN predicted action at surface node



(f) GNN predicted action at curve node

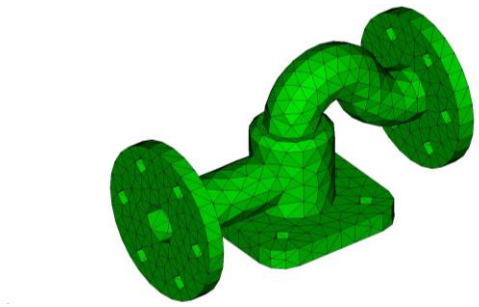


(g) Hex meshable subdomains

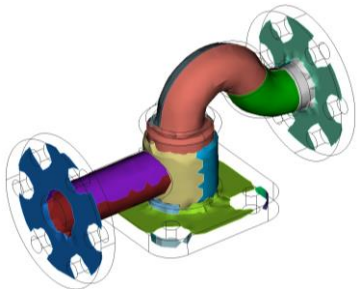


(h) Hex mesh

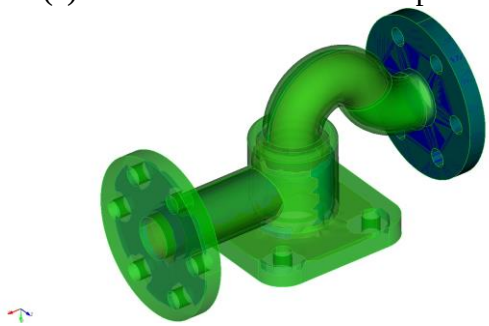
Figure 20 Agent-guided WebCuts and hex mesh on Browell



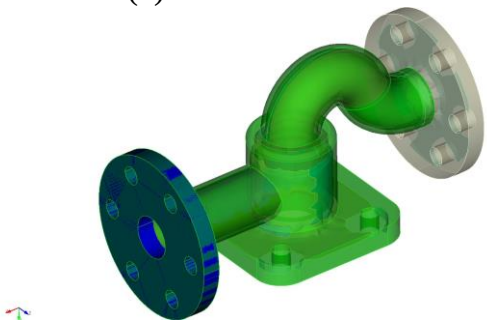
(a) Tetmesh with no interior points



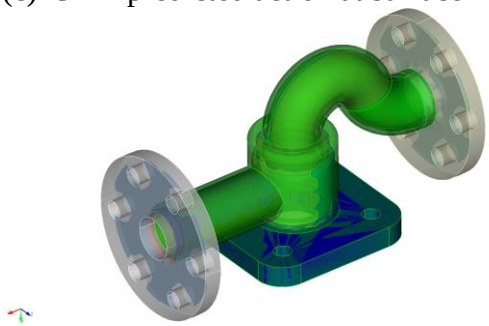
(b) Reduced 2D CAT



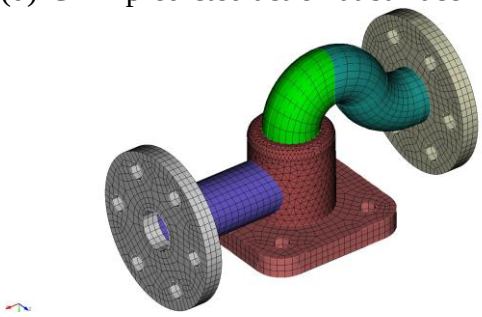
(c) GNN predicted action at surface node



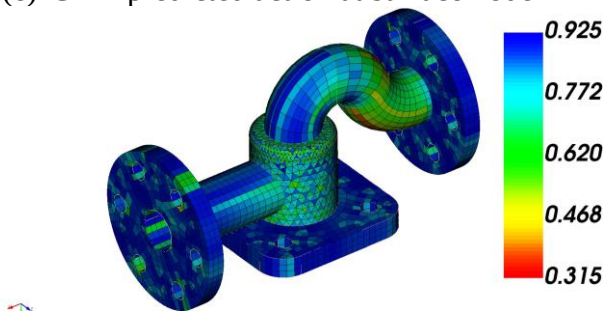
(d) GNN predicted action at surface node



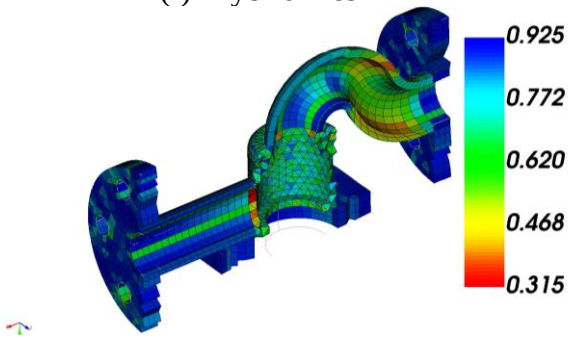
(e) GNN predicted action at surface node



(f) Hybrid mesh

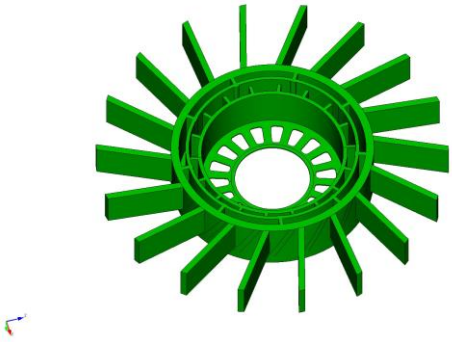


(g) Scaled Jacobian of hybrid mesh

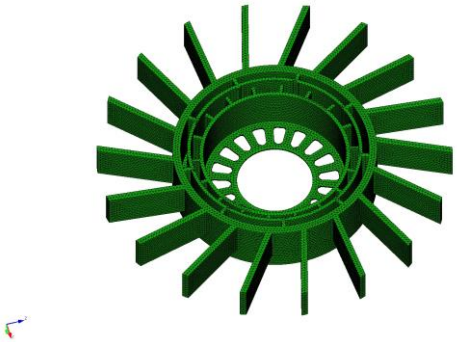


(h) Cross-section of hybrid mesh

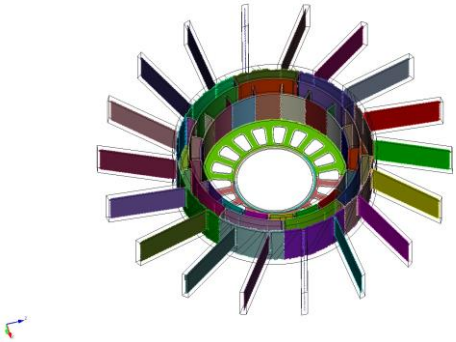
Figure 21 Agent-guided WebCuts and hybrid mesh on PlumbingFixture



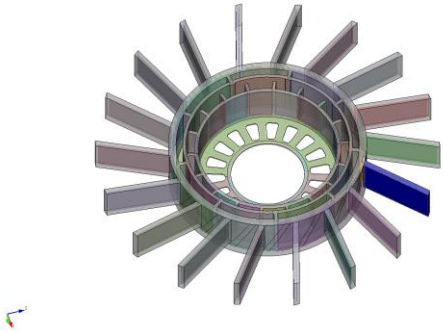
(a) Input 3D CAD



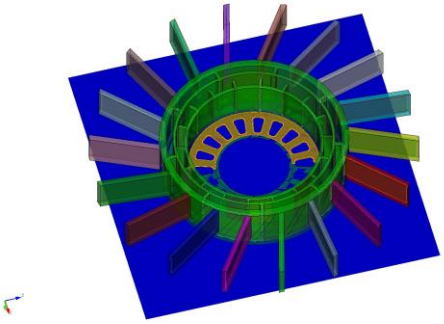
(b) Tetmesh with no interior points



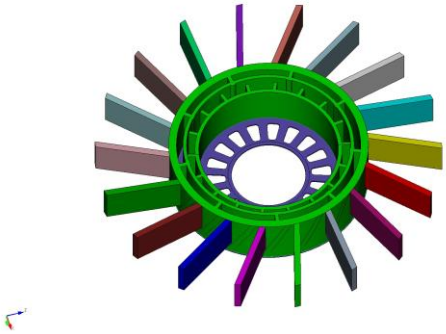
(c) Reduced 2D CAT



(d) GNN predicted action at surface node



(e) GNN predicted action at surface node



(f) Decomposed subdomains

Figure 22 Agent predicted WebCut actions on JetBlade

5. ANTICIPATED OUTCOMES AND IMPACTS

This research is expected to materially reduce mesh-preparation time in multiple ModSim workflows from months/weeks of manual intervention to hours/minutes of agent-guided, automated preprocessing by suggesting intelligent decomposition choices into reproducible, executable command plans. Because the pipeline operates on a CAD-agnostic 3D→2D→1D reduction with invariant features, it is designed to scale and transfer across CAD architectures and to integrate into broader meshing workflows, including future adaptive simulation loops in which decomposition and meshing respond to solution feedback. Our strategy is to fallback to hybrid meshing if all-hex meshing cannot be achieved by the agent, i.e., some workflows prefer hex meshes in critical regions of interest and tet meshing in the remainder of the domain.

With regard to publishing, we have presented 3D CAD to 1D graph reduction at USNCCM-18, July 20-24, 2025 [39] and we will submit a full paper on agent guided geometry decomposition via supervised learning on targeted geometries to SIAM International Meshing Roundtable Workshop (IMR) on or before Oct 1, 2025. We are also planning to present architecture and early results of reinforcement learning as a Research Notes to IMR before Dec 21, 2025. We have also submitted a tri-lab proposal on agent-guided hex meshing using SL/RL to ASC/AI4NS call.

Our future work and R&D roadmap focus on production robustness and extensions of current ML framework. For training data, we will expand the synthetic generator to reflect other Sandia specific parts. On CAT, we will harden the “create chordal axis” export so that the CAT graph and its mappings are reliable across edge cases. For graph features, we will normalize geometry by scaling the bounding box to a parametric space (so centroids, lengths, and thicknesses are comparable across models); add thickness statistics (min/mean/max) to distinguish tapered from constant-section slabs; record principal sweeping directional cosines explicitly; compute surface-node valency as the number of incident interface/junction curves; and explore a B-Rep–first graph augmented with CAT cues to reduce reliance on skeleton extraction. For actions, we will address the failure of the surface-node radial sweep on curved slabs by introducing a loft-based operation that uses the top and bottom parent surfaces as tools; extend the curve-node action set (actions 4–8) to include sweep-curve WebCuts between parent curves; and broaden both supervised and reinforcement-learning coverage to additional Cubit® WebCut commands. In reward shaping, we will penalize WebCuts that introduce sharp angles or otherwise degrade downstream mesh quality. In the GUI, we will integrate the GNN-predicted solutions into the Cubit® Meshing PowerTool for interactive review, constraint checks, and one-click execution. For reinforcement learning, we will train on general CAD models (e.g., curated ABC subsets) without action labels, using imitation warm starts from the supervised model where appropriate.
As mentioned in Figure 1, this LDRD focuses on addressing only the geometry decomposition task for hex meshing; however, geometry preparation, mesh generation, and post-meshing shown in Figure 23 is an iterative process. In the future, we need an orchestration agent to manage the input and output of various steps shown in Figure 23

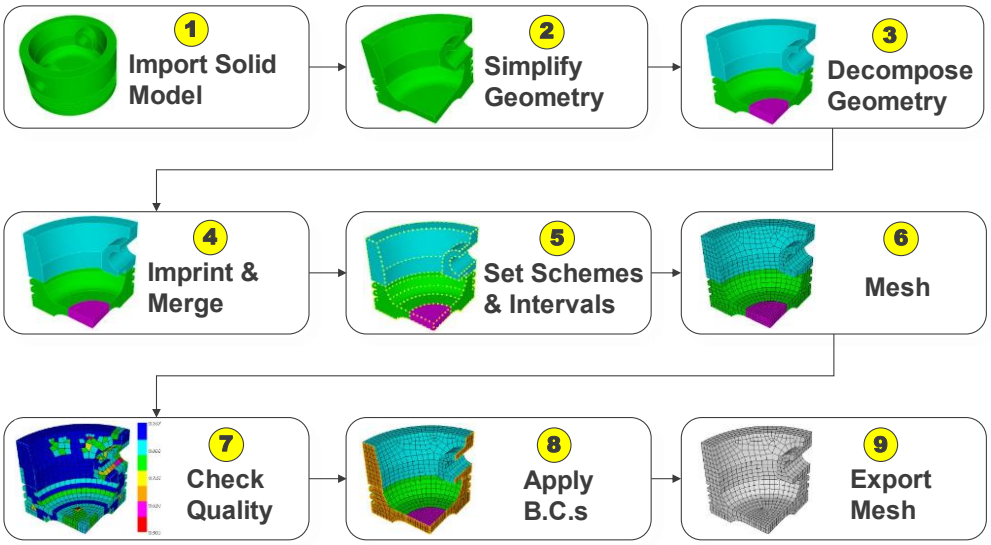


Figure 23 Orchestration agent is required for geometry processing, meshing, and post-meshing

6. CONCLUSION

We present a novel, end-to-end pipeline for agent-guided geometry decomposition using a reduced bi-partite graph representation of 3D CAD and a multi-action Graph Neural Network. By encoding CAD-derived surface and curve features into a bi-partite graph via Chordal Axis Transform and training a multi-action heterogeneous GNN, our system learns to predict WebCut actions that historically required domain expert intervention. Training on hundreds of synthetic 3D CAD models of lattice structure with labels, we observe very high WebCut action prediction accuracy (98.7% on unseen lattice test data set and greater than 85% on general CAD models) and rapid inference. We prototyped integrating Cubit® WebCut commands into Cubit® Meshing Powertool GUI for early productionization, user testing, and distribution in the future.

Our pipeline not only reduces manual meshing effort but also enhances consistency and scalability in simulation workflows. While initial results are promising, future enhancements such as learned ranking, reinforcement learning for command ordering, data augmentation, and developing a new orchestration agent for geometry preparation, meshing, and post-meshing will further improve the system's utility.

This work sets the foundation for replacing manual geometric reasoning with learned, data-driven strategies embedded in simulation workflows. We expect that continued development and deployment will significantly accelerate mission-critical CAD model preparation and offer a new AI-enabled CAD decomposition for hex and hybrid meshing to accelerate ModSim workflows across different disciplines for various missions.

REFERENCES

[1] N. Pietroni, M. Campen, A. Sheffer, G. Cherchi, D. Bommes, X. Gao, R. Scateni, F. Ledoux, J.-F. Remacle and M. Livesu, "Hexmesh generation and processing: a survey," *ACM Transactions on Graphics,* vol. 42, no. 2, p. 1–44, 2023.

[2] W. J. Gordon and C. A. Hall, "Construction of curvilinear coordinate systems and applications to mesh generation," *International Journal for Numerical Methods in Engineering,* vol. 7, no. 4, p. 461–477, 1973.

[3] R. Schneiders, "A grid-based algorithm for the generation of hexahedral element meshes," *Engineering with Computers,* vol. 12, no. 3–4, p. 168–177, 1996.

[4] T. Blacker, "The Cooper tool," in *Proceedings of 5th International Meshing Roundtable*, Pittsburgh, PA, 1996.

[5] S. J. Owen and M. L. Staten, "Parallel hex meshing from volume fractions," *Engineeringt with Computers,* vol. 30, no. Dec 19, pp. 301-313, 2012.

[6] T. J. Tautges, T. D. Blacker and S. A. Mitchell, "The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes," *International Journal for Numerical Methods in Engineering,* vol. 39, no. 19, p. 3327–3349, 1996.

[7] S. A. Canann, "Plastering (paving): A new approach to automated all-hexahedral mesh generation," in *33rd Structures, Structural Dynamics and Materials Conference*, Dalas, TX, April, 1992.

[8] Y. Li, H. Bao and J. Huang, "Singularity-restricted fields for hex meshing," *ACM Transactions on Graphics,* vol. 31, no. 6, p. Article 177, 2012.

[9] M. Nieser, U. Reitebuch and K. Polthier, "CubeCover: Parameterization of 3D volumes via harmonic fields," *Computer Graphics Forum,* vol. 30, no. 5, p. 1397–1406, 2011.

[10] Y. Lu, R. Gadh and T. J. Tautges, "Feature-based hex meshing methodology: Feature recognition and volume decomposition," *Computer-Aided Design,* vol. 33, no. 3, p. 221–232, 2001.

[11] L. Wu, Y. Zhang, W. Yu and G. Xu, "Fuzzy clustering-based pseudo-swept-volume decomposition," *Computer-Aided Design,* vol. 98, p. 30–45, 2018.

[12] S. H. Shih Bih-Yaw, "Automatic decomposition of solid models into swept volumes," in *Proceedings of 5th International Meshing Roundtable*, 1996.

[13] C. Liu and R. Gadh, "Automatic hexahedral mesh generation by parametric decomposition," *Engineering with Computers,* vol. 13, no. 3, p. 221–238, 1997.

[14] F. Boussuge, R. Raffin, B. Lévy and P. Alliez, "Idealized models for FEA derived from generative modeling processes based on extrusion primitives," *Engineering with Computers ,* vol. 31, no. 28 October 2014, p. 513–527, 2015.

[15] C. G. Armstrong, H. J. Fogg, C. M. Tierney and T. T. Robinson, "Common themes in multi-block structured quad/hex mesh generation," in *24th International Meshing Roundtable*, https://doi.org/10.1016/j.proeng.2015.10.123, 2015.

[16] W. R. Quadros, "LayTracks3D: Meshing general solids using medial axis transform," *Computer-Aided Design,* vol. 72, p. 102–117, 2016.

[17] C. S. Chong, L. Fuchs and V. Shapiro, "Medial surfaces for feature recognition and volumetric decomposition," *Computer-Aided Design,* vol. 91, p. 38–50, 2017.

[18] Z. Zheng, S. Gao and C. Shen, "Dual loops and progressive dual-surface decomposition for block-structured hex meshing," *Engineering with Computers,* vol. 37, p. 2625–2646, 2021.

[19] Z. Zheng, S. Gao and C. Shen, "Progressive dual-surface decomposition for block meshing," *Engineering with Computers,* vol. 38, p. 217–233, 2022.

[20] W. R. Quadros, K. Ramaswami, F. B. Prinz and B. Gurumoorthy, "LayTracks: a new approach to automated geometry adaptive quadrilateral mesh generation using medial axis transform," *International Journal for Numerical Methods in Engineering,* vol. 61, no. 2, pp. 209-237, 2004.

[21] C. G. Armstrong, D. J. Robinson, R. McKeag, T. Li, S. Bridgett, R. Donaghy and M. CA., "Medials for meshing and more," in *Proceedings of 4th international meshing roundtable.*, 1995.

[22] L. Sun, C. M. Tierney, C. G. Armstrong and T. T. Robinson, "Decomposing complex thin-walled CAD models for hexahedral-dominant meshing," *Computer Aided Design,* vol. 31, no. http://dx.doi.org/10.1016/j.cad.2017.11.004, pp. 103-118, 2018.

[23] M. Livesu, A. Sheffer, N. Vining and R. Scateni, "LoopyCuts: Cutting loops for surface segmentation," *Computer Graphics Forum,* vol. 32, no. 5, p. 155–166, 2013.

[24] X. Liu, Y. J. Zhang, W. Wang and L. Liu, "Skeleton-based multi-cube generation for T-spline control mesh construction," *Computer-Aided Design,* vol. 58, p. 162–176, 2015.

[25] J.-H. C. Lu, W. R. Quadros and K. Shimada, "Evaluation of user-guided semi-automatic decomposition tool for hex meshing," *Journal of Computational Design and Engineering,* vol. 4, no. 4, p. 330–338, 2017.

[26] J.-H. C. Lu, I. Song, W. R. Quadros and K. Shimada, "Medial-object-guided volumetric decomposition with pen-based user interface for hexahedral mesh generation," in *20th International Meshing Roundtable*, 2012.

[27] A. Patel, S. Pemberton, M. Safdari and W. R. Quadros, "Autonomous hexahedral meshing using artificial intelligence," in *NAFEMS World Congress*, Oct 25–29, 2021.

[28] B. C. DiPrete, R. V. Garimella, C. Garcia-Cardona and N. Ray, "Reinforcement learning for block decomposition of planar CAD models," *Engineering with Computers,* p. 1–11, 2024.

[29] S. Zhang, Z. Guan, X. Wang, P. Tan and H. Jiang, "Reinforcement learning based automatic block decomposition of solid models for hexahedral meshing," *Computer-Aided Design,* vol. 182, p. 103850, 2025.

[30] G. Z. J. H. W. X. T. P. Zhang Shuming, "BrepMFR: Enhancing machining feature recognition in B-Rep models through deep learning and domain adaptation.," *Comput Aided Geometric Design,* vol. 111, no. https://doi.org/10.1016/j.cagd.2024.102318, 2024.

[31] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, D. Zorin and D. Panozzo, "ABC: A big CAD dataset for geometric deep learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.*, 2019.

[32] S. J. Owen, B. W. Clark, D. J. Melander, M. Brewer, J. F. Shepherd, K. Merkley, C. Ernst and R. Morris, "An Immersive Topology Environment for Meshing," in *Proceedings of 16th International Meshing Roundtable*, 2007.

[33] A. Colligan, Deep learning for boundary representation CAD models, Queen's University Belfast: Doctor of Philosophy Thesis, 2022.

[34] K. D. D. Willis, Y.-C. Pu, S. Luo, H. Chu, T. Du and W. Matusik, "Fusion 360 Gallery: A dataset and environment for programmatic CAD construction from human design sequences," *ACM Transactions on Graphics,* vol. 40, no. 4, p. Article 86, 2021.

[35] H. Cao, H. Yang, M. Shah and N. J. Mitra, "Sequence modeling of parametric CAD with B-Rep Transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024)*, 2024.

[36] . S. Z. James Gregson, "All-Hex Mesh Generation via Volumetric PolyCube Deformation," *Computer Graphics Forum,* vol. 30, no. 5, pp. 1407-1416, Aug 2011.

[37] W. R. Quadros, "An approach for extracting non-manifold mid-surfaces of thin-wall solids using chordal axis transform," *Engineering with Computers,* vol. 24, no. 3, pp. 305-319, 2008.

[38] W. L. Hamilton, R. Ying and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[39] W. R. Quadros, C. Ernst, C. Stimpson, N. Winovich and W. Ziehe, "A bi-partite graph representation of 3D solids for geometric reasoning and GNN," in *18th US National Congress on Computational Mechanics*, Chicago, 2025.