HIST2HEX: CONSTRUCTION-HISTORY-GUIDED CAD DECOMPOSITION WITH A LANGUAGE MODEL FOR HEXAHEDRAL MESHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Hexahedral meshing of CAD models is central to simulation-driven design. It requires first decomposing the solid into hex-meshable blocks, which remains a long-standing bottleneck. Existing methods operate on the final boundary representation. This representation omits the construction history and the feature boundaries it records. We observe that replaying the Boolean construction history deterministically associates each final body with its body-specific construction subsequence. We present Hist2Hex, a two-stage framework that exploits this structure for hexahedral meshing. In the first stage, deterministic history replay reconstructs the final bodies and establishes these associations between geometry and history without learning. In the second stage, a language model decomposes each sub-solid into hex-meshable leaf blocks. It selects cut planes conditioned on three sources of information: the serialized history, the boundary-representation (B-rep) geometry, and a table of enumerated candidates. Hist2Hex achieves the highest final-mesh success rates across all three datasets, while a controlled Fusion360 ablation shows that serialized history improves candidate selection with Stage 1 and the candidate pool fixed.

1 INTRODUCTION

The structured tensor-product elements of hexahedral meshes improve numerical conditioning, boundary alignment, and solver efficiency in finite-element simulation Pietroni et al. (2022). For a general CAD solid, however, generating such a mesh typically requires first decomposing the domain into simpler blocks that can be swept or mapped by structured grids. Automatically obtaining such a block decomposition for solids of arbitrary complexity remains difficult, since a single missed cut or misplaced boundary can propagate into unmeshable regions across the whole part.

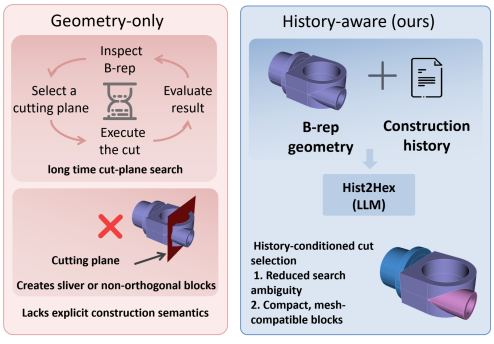


Figure 1: Construction history provides structural cues for hex-meshable decomposition.

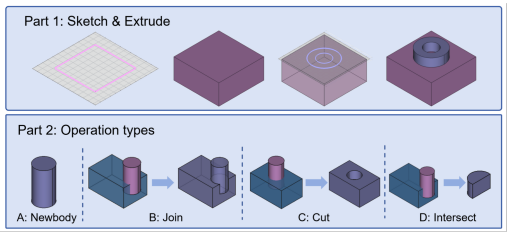


Figure 2: Four Boolean body-interaction modes illustrated for sketch-extrusion events.

In practice, this decomposition is still often performed manually. An analyst inspects the geometry, proposes a cut, executes it, and checks whether the resulting blocks are hex-meshable, repeating

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

this cycle of inspection, cutting, and evaluation until the solid is resolved (Figure 1). This workflow is slow, expertise-dependent, and can still produce sliver or non-orthogonal blocks that degrade downstream mesh quality Lu et al. (2017).

Existing automatic methods operate on the final geometry through global optimization Tarini et al. (2004); Fang et al. (2016); Lyon et al. (2016); Livesu et al. (2020); Brückler et al. (2022), sequential search or learned cut prediction Bourmaud et al. (2025); DiPrete et al. (2024); Zhang et al. (2025); Quadros et al. (2026), or generative polycube construction He et al. (2026). Despite their different mechanisms and outputs, these approaches do not use the original construction history as an input to decomposition.

Operating on the final B-rep alone limits hex decomposition. After export, boundaries created by Boolean operations are fused into a watertight shell, and construction-derived cuts are no longer distinguishable. In contrast, the CAD construction history preserves this information as an ordered sequence of sketches, extrusions, and Boolean operations. It provides decomposition-relevant cues, including target-body identities, sketch profiles, extrusion intervals, and cut/intersect boundaries. As a program-like sequence that can be serialized with B-rep descriptors and candidate cuts, the history enables a language model to condition cut decisions on both design process and current geometry.

We introduce Hist2Hex, a two-stage framework that exploits CAD construction structure for hexahedral decomposition. Stage 1 replays the history to reconstruct the final bodies and associate each with its body-specific construction subsequence. Stage 2 uses a language model to decompose each sub-solid into hex-meshable leaf blocks, conditioned on serialized history, B-rep geometry, and enumerated cut candidates. Across three CAD datasets, Hist2Hex produces hex-meshable decompositions, and removing history causes the largest performance drop, showing that construction history provides complementary cues beyond the final B-rep.

Our contributions are as follows.

- **Problem formulation.** We formulate CAD-history-guided hexahedral decomposition as candidate-indexed cut prediction over history-grouped sub-solids, using body-specific construction histories and OCC-verified hex-meshability as an automatic training signal.
- **Method.** We introduce Hist2Hex, a language-model predictor that conditions jointly on the construction history, the B-rep geometry, and enumerated cut candidates.
- **Experiments.** We evaluate Hist2Hex on three CAD datasets and show that removing the serialized history causes the largest drop in decomposition quality, demonstrating the complementary value of explicit history conditioning.

2 Related Work

Hexahedral meshing and block decomposition. Existing approaches to automatic hexahedral meshing commonly decompose the domain into swept or mapped blocks that can be meshed in a structured manner Pietroni et al. (2022). A separate fully automatic line uses octree-based all-hexahedral meshing to generate a conforming mesh directly from the input geometry Maréchal (2009). Polycube parametrizations Tarini et al. (2004); Fang et al. (2016), volumetric hexahedral mesh extraction Lyon et al. (2016), frame-field methods that align elements via octahedral fields Liu et al. (2018), feature-aligned block decomposition guided by a cross-field Livesu et al. (2020), and volume parametrization quantization Brückler et al. (2022) pursue this goal through global optimization over the final boundary geometry. PolycubeNet He et al. (2026) instead uses a dual-latent diffusion model to generate a polycube representation from input geometry and then transfers it to a hexahedral mesh. These methods do not use the construction history that produced the final part.

Search- and learning-based decomposition. Sequential methods select decomposition operations from the current geometric state. Bourmaud et al. (2025) formulate planar CAD block decomposition as Monte Carlo Tree Search, exploring partition operations to construct quadrilateral layouts. Their prototype demonstrates structured search as an alternative to learned policies, but targets a 2D setting rather than volumetric blocks for all-hex meshing. DiPrete et al. (2024) train a reinforcement-learning agent for axis-aligned cut selection in planar CAD, while Zhang et al. (2025) extend learned selection to 3D solids. Quadros et al. Quadros et al. (2026) encode B-rep geometry as a heterogeneous graph and predict executable 3D cutting commands with a GNN agent.

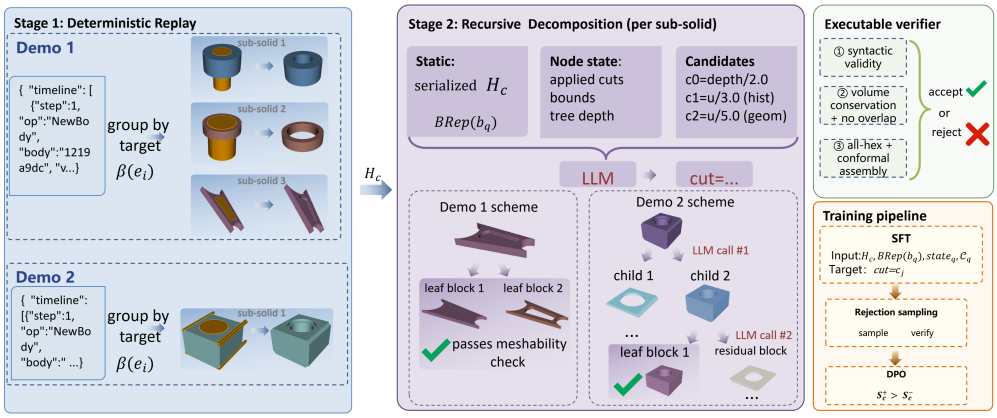


Figure 3: Hist2Hex pipeline. Stage 1 replays the construction history to reconstruct each final body c and associate it with its subsequence H_c. For a Stage 2 tree node, the model receives the static history, current node state, and candidate cuts, then selects one candidate-indexed cut. The executor recursively applies the selected cuts and assembles them into a complete decomposition scheme. The two right panels show example schemes retained by the verifier for preference fine-tuning.

CAD generation and sequence modeling. Large CAD datasets have enabled neural modeling of sketches, constraints, and construction histories Seff et al. (2020); Willis et al. (2021); Wu et al. (2021); Xu et al. (2022); Jayaraman et al. (2022). A related line of work reconstructs construction information from final geometry. Zone graphs support inference of plausible modeling sequences Xu et al. (2021), while CADOps-Net Dupont et al. (2022) jointly segments B-rep faces by CAD operation type and modeling step. Such predictions could provide partial construction cues when the original authoring history is unavailable, although they do not recover the complete executable history assumed by Hist2Hex. More recently, language-conditioned systems have generated CAD programs and design histories from text Wu et al. (2023); Khan et al. (2024). In these prior works, construction history is an output to be reconstructed or generated. Hist2Hex instead uses an available history as input to downstream decomposition.

3 Method

We use *sub-solid* for a Stage 1 output and *block* for any geometric region represented by a node during Stage 2 recursion, with the root block equal to its sub-solid. A block that still requires decomposition is a *residual block*, whereas a block that passes the local hex-meshability check and terminates recursion is a *leaf block*.

3.1 Problem Formulation

Hist2Hex uses the ordered CAD construction history $H = (e_1, \ldots, e_M)$. Each event e_i records its profile and extrusion interval, operation type $o(e_i) \in \mathcal{O}$, and target body $\beta(e_i)$. We categorize sketch-extrusion events by Boolean effect as NewBody, Join, Cut, or Intersect (Figure 2). The broader Stage 1 replay vocabulary also supports other construction features. Figure 3 summarizes the full pipeline. Target-body identities enable deterministic grouping even when earlier features are edited.

Sub-solids from body identity. Stage 1 groups H by the recorded target body $\beta(e_i)$, yielding one sub-solid c for each final body $\mathrm{id}(c)$ and associating it with its construction subsequence

$$H_c = (e_{i_1}, \ldots, e_{i_{m_c}}), \qquad i_1 < \cdots < i_{m_c}, \tag{1}$$

where $i_1, \ldots, i_{m_c}$ are exactly the indices $i \in \{1, \ldots, M\}$ satisfying $\beta(e_i) = \mathrm{id}(c)$. Each H_c begins with the body's `new_body` event and contains all later operations targeting it. Figure 4 illustrates this grouping.

Stage 1 recovers only body-level separation and does not expose latent feature interfaces erased by Boolean operations. When final bodies are represented as disjoint solids, its output matches the body-level partition obtainable from the final B-rep. Its primary role is to associate each replayed sub-solid c with a body-specific construction subsequence H_c, providing the geometry–history pairs required by Stage 2. This association is unavailable from the final B-rep alone. Records of `join_body`, `cut_body`, and `intersect_body` remain in H_c and provide decomposition cues for Stage 2.

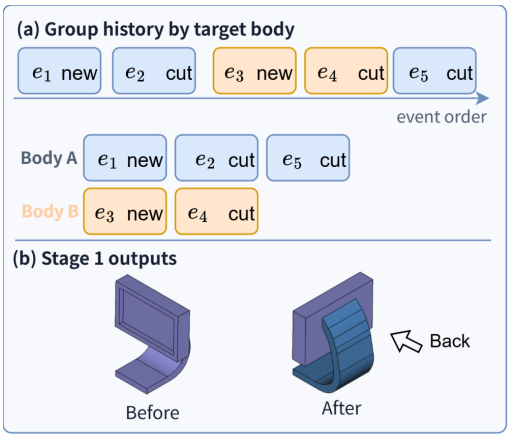


Figure 4: Stage 1 groups events by target-body identity and replays each subsequence to recover one sub-solid.

3.2 Input Representation and Cut-Action Prediction

Given a sub-solid c and its construction subsequence H_c, the remaining difficulty is to select cuts that make its constituent regions hex-meshable. Let q index a node in the binary decomposition tree, with root $q = 0$, and let b_q be its associated block, where $b_0 = c$. We call b_q *locally hex-meshable* when the fixed structured mesher, using the same automatic scheme-selection rules and timeout as final verification, meshes b_q independently into a non-empty pure-hexahedral mesh. Shared-interface division propagation and conformal assembly are applied only after a complete scheme has been generated, so local hex-meshability is necessary but not sufficient for final verifier acceptance.

Every invocation of the model resolves one node. The recursive executor maintains its dynamic state $\mathrm{state}_q = (\text{applied_cuts}, \text{bounds}, d_q^{\mathrm{tree}})$, which records the cuts along the root-to-q path, the bounds of b_q, and its recursion depth d_q^{tree}. We write $\mathrm{BRep}(b_q)$ for the serialized boundary representation of the current residual block. From the history, current B-rep, and node state, the executor constructs the node-specific candidate set

$$\mathcal{C}_q = \mathcal{C}(H_c, \mathrm{BRep}(b_q), \mathrm{state}_q)\,. \tag{2}$$

The model then selects one cut action

$$\hat{a}_q = \arg\max_{a \in \mathcal{C}_q} p_\theta(a \mid H_c, \mathrm{BRep}(b_q), \mathrm{state}_q, \mathcal{C}_q)\,. \tag{3}$$

Figure 3 shows the input components under Static, Node state, and Candidates, while Demo 1 scheme and Demo 2 scheme illustrate recursively assembled decompositions.

The input contains the static history H_c and three node-specific components: $\mathrm{BRep}(b_q)$, state_q, and $\mathcal{C}_q$. The B-rep and candidate table are updated after each accepted cut to describe only the current residual block b_q.

Sub-solid history serialization. The serialization lists each event in H_c in order, recording its step index, operation type, target body, and two geometry-derived quantities: the resulting volume and bounding box. These fields expose the body's evolving extent to the model, rather than operation types alone. The Supplementary Material gives the exact fields and a worked example. Figure 3 under **Static** shows this timeline as part of the input.

Candidate cut table. Complementing the history serialization, the candidate table $\mathcal{C}_q$ supplies a finite set of geometry-grounded cuts and is re-enumerated at each invocation from the static context and state_q. Candidates are derived from recorded feature intervals, planar B-rep faces, and circular or cylindrical geometry. Planar candidates are represented in the fixed CAD/STEP coordinate frame by an axis and offset or by a point and normal, while circular features additionally provide through-center and radial cuts that enable cross- or wedge-like partitions. Each candidate carries a compact identifier, e.g. `c0`, for candidate-indexed prediction. The model therefore selects among executable,

feature-supported cuts rather than generating free-form coordinates. The Supplementary Material gives the full construction.

The serialized timeline, current-block B-rep description, node state, and candidate table are concatenated into a single input $(H_c, \mathrm{BRep}(b_q), \mathrm{state}_q, \mathcal{C}_q)$.

Cut-action prediction. Before invoking the model, the executor checks whether b_q is locally hex-meshable. A passing block becomes a leaf. Otherwise, the model selects a candidate cut using Eq. (3). Applying the cut produces two child blocks, and the same procedure recursively resolves those that remain non-hex-meshable. Let $q_1, \ldots, q_{T_c}$ denote the model-invoked nodes and a_{q_t} their selected cuts. The executor assembles them into the complete decomposition scheme $S_c = ((q_t, a_{q_t}))_{t=1}^{T_c}$. The Supplementary Material specifies the output format, recursion, and termination rules.

3.3 Training Pipeline

We train the model in three steps: supervised fine-tuning (SFT), a rejection-sampling bootstrap, and DPO preference fine-tuning. The executable verifier (Section 3.4) gates both bootstrap acceptance and preference-data construction.

SFT. We initialize training with the supervised dataset

$$\mathcal{D}_0 = \{(H_c, \mathrm{BRep}(b_q), \mathrm{state}_q, \mathcal{C}_q, a_q^*)\}. \tag{4}$$

Each tuple supervises one model-invoked node, and $a_q^* \in \mathcal{C}_q$ is its reference cut action (see Supplementary Material). Conditioned on $x_q = (H_c, \mathrm{BRep}(b_q), \mathrm{state}_q, \mathcal{C}_q)$, we train the model with standard token-level cross-entropy to emit the serialization `cut=c_j` of a_q^*.

Rejection-Sampling Bootstrap. To extend the model beyond the cases covered by the initial supervision, the current checkpoint θ_r performs K recursive rollouts for each sub-solid. In each rollout, it samples an action at every model-invoked node, which the executor combines in fixed depth-first order into a complete scheme $S_c^{(k)}$. We retain only the verified pool, with validity formally defined in Section 3.4:

$$\mathcal{P}_r(c) = \left\{ S_c \;\middle|\; \begin{array}{c} S_c \sim p_{\theta_r}(\cdot \mid H_c, \mathrm{BRep}(c)), \\ \mathrm{Valid}(H_c, S_c) = 1 \end{array} \right\}. \tag{5}$$

For each sub-solid, we retain up to m accepted schemes with the fewest leaf blocks, then add their node actions to the next-round training set. The resulting verified schemes extend supervision to nodes absent from the initial supervised data, and the final bootstrap checkpoint serves as the reference model for DPO.

DPO Preference Fine-Tuning. Rejection sampling often produces multiple schemes for the same sub-solid. We form preference pairs (S_c^+, S_c^-) in which S_c^+ is verifier-valid and more parsimonious, while S_c^- is either verifier-invalid or verifier-valid but has more leaf blocks. Starting from the bootstrap reference model, we apply the standard DPO objective Rafailov et al. (2023) with $\beta = 0.1$. Because the executor deterministically constructs each node state from the preceding depth-first actions, the probability of a complete scheme factorizes over its ordered model invocations:

$$\log p_\theta(S_c \mid H_c, \mathrm{BRep}(c)) = \sum_{t=1}^{T_c} \log p_\theta(a_{q_t} \mid x_{q_t}). \tag{6}$$

The DPO objective compares complete schemes from recursive execution while retaining the node-level cut interface.

3.4 Verification

At inference time, greedy decoding is invoked on each child node whose residual block fails the local hex-meshability check, until all depth-first invocations terminate (see Supplementary Material). The assembled scheme passes three operational checks: syntactic validity, OCC-based geometric

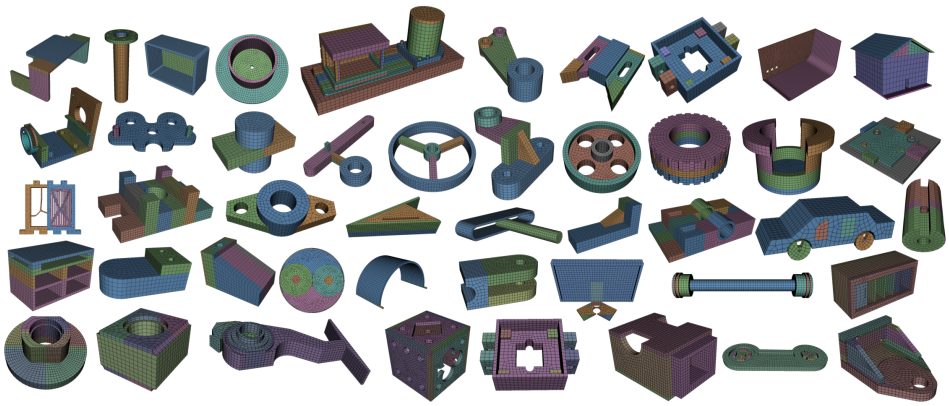


Figure 5: Representative globally conforming all-hexahedral meshes. Colors indicate leaf-block provenance.

reassembly, and globally conforming all-hexahedral mesh generation. The meshing interface propagates compatible divisions across shared leaf-block interfaces, applies an automatically selected structured scheme to each leaf block, and merges coincident interface entities into one mesh. A scheme is accepted only when non-empty leaf blocks conserve the parent sub-solid's volume with limited pairwise overlap and assemble into a non-empty, conforming pure-hexahedral mesh without hanging nodes or non-manifold interfaces. The meshing configuration, tolerances, and timeout are frozen across training-data filtering and evaluation. This cascade filters rejection-sampling rollouts and constructs DPO preferences. The Supplementary Material gives exact predicates, tolerances, and failure handling.

4 Experiments

4.1 Experimental Setup

Datasets and evaluation setup. We evaluate on three public parametric CAD datasets: Fusion 360 Gallery Willis et al. (2021), DeepCAD Wu et al. (2021), and WHUCAD Fan et al. (2025), containing human-authored, large-scale synthetic, and feature-based CAD histories, respectively. Figure 5 shows representative all-hexahedral meshes from the evaluation corpus. For each dataset, we first exclude sub-solids whose unsplit geometry directly produces a non-empty pure-hexahedral mesh under the same structured meshing configuration and timeout used for final evaluation. We then construct a fixed evaluation set of 1,500 sub-solids from the remaining decomposition-required population. Each dataset contains n=1,500 evaluation instances. Dataset construction and partitioning are performed at the case level so that sub-solids derived from the same CAD case cannot cross partitions. All methods use the same frozen dataset-specific manifests and identical verification, interface-conformity, and meshing settings.

Baselines and metrics. We compare Hist2Hex with a greedy heuristic and a uniform-grid method using the same geometry-coordinate candidate pool, as well as the RL policy of Zhang et al. (2025). We evaluate LoopyCuts Livesu et al. (2020) separately under its native success criterion in Table 2. We additionally evaluate prompt-only Qwen2-1.5B-Instruct Yang et al. (2024) and DeepSeek-Coder-6.7B-Instruct Guo et al. (2024) baselines. Both receive the same construction history, B-rep description, candidate table, and candidate-indexed output contract as Hist2Hex, but use frozen pretrained weights without SFT, LoRA, DPO, or test-time parameter updates. We also evaluate *Classifier-hist*, a 1.6M-parameter three-layer MLP that replaces only the LM's candidate-selection step and conditions on candidate and node features together with a mean-pooled embedding of the serialized history H_c. The Supplementary Material provides its full architecture and training details. We retrain the RL policy on the same dataset-specific training splits with seeds 42/43/44, while retaining the original cut-action parameterization of Zhang et al. (2025) rather than substituting Hist2Hex's candidate-indexed action pool. Seeds 42/43/44 index three independent Hist2Hex

Table 1: Benchmark on three CAD datasets. SR is the fraction of cases producing an accepted final mesh. Mesh-quality metrics are computed on successful final meshes. Decomposition methods are executed and reassembled before final-mesh acceptance. Values are mean±std over three runs. Bold marks the best per dataset.

	Method	MinSJ ↑	MeanSJ ↑	Aspect ↓	SR ↑
DeepCAD	Greedy heuristic	-0.70 ± 0.00	0.82 ± 0.00	6.50 ± 0.00	0.05 ± 0.00
	Uniform grid	0.03 ± 0.00	0.58 ± 0.00	8.00 ± 0.00	0.12 ± 0.00
	RL policy Zhang et al. (2025)	-0.72 ± 0.18	0.86 ± 0.03	3.40 ± 0.15	0.28 ± 0.03
	Qwen2 3-shot	0.08 ± 0.08	0.22 ± 0.07	**2.10 ± 0.70**	0.10 ± 0.03
	DeepSeek-Coder 3-shot	0.32 ± 0.08	0.42 ± 0.06	2.80 ± 0.50	0.35 ± 0.05
	Verifier random search	0.45 ± 0.18	0.62 ± 0.12	3.80 ± 0.45	0.46 ± 0.04
	Classifier-hist	0.60 ± 0.03	0.84 ± 0.04	3.00 ± 0.15	0.62 ± 0.02
	Hist2Hex	**0.68 ± 0.02**	**0.95 ± 0.01**	2.60 ± 0.10	**0.72 ± 0.01**
Fusion360	Greedy heuristic	-0.90 ± 0.00	0.88 ± 0.00	2.40 ± 0.00	0.05 ± 0.00
	Uniform grid	0.76 ± 0.00	**0.94 ± 0.00**	**2.00 ± 0.00**	0.07 ± 0.00
	RL policy Zhang et al. (2025)	-0.85 ± 0.12	0.84 ± 0.03	3.90 ± 0.18	0.11 ± 0.04
	Qwen2 3-shot	0.42 ± 0.10	0.55 ± 0.08	3.30 ± 0.65	0.08 ± 0.03
	DeepSeek-Coder 3-shot	0.38 ± 0.07	0.52 ± 0.07	3.60 ± 0.55	0.26 ± 0.05
	Verifier random search	0.55 ± 0.15	0.72 ± 0.10	3.90 ± 0.40	0.28 ± 0.04
	Classifier-hist	0.70 ± 0.03	0.90 ± 0.03	3.60 ± 0.18	0.44 ± 0.02
	Hist2Hex	**0.77 ± 0.02**	0.93 ± 0.01	2.30 ± 0.12	**0.60 ± 0.02**
WHUCAD	Greedy heuristic	-0.92 ± 0.00	0.84 ± 0.00	3.60 ± 0.00	0.04 ± 0.00
	Uniform grid	-0.94 ± 0.00	0.92 ± 0.00	1.90 ± 0.00	0.08 ± 0.00
	RL policy Zhang et al. (2025)	-0.21 ± 0.12	0.91 ± 0.03	2.20 ± 0.16	0.12 ± 0.04
	Qwen2 3-shot	0.02 ± 0.05	0.10 ± 0.05	2.40 ± 0.60	0.04 ± 0.02
	DeepSeek-Coder 3-shot	0.08 ± 0.06	0.15 ± 0.05	3.50 ± 0.80	0.12 ± 0.04
	Verifier random search	0.40 ± 0.20	0.58 ± 0.14	3.40 ± 0.50	0.18 ± 0.03
	Classifier-hist	0.55 ± 0.04	0.82 ± 0.05	2.60 ± 0.20	0.32 ± 0.02
	Hist2Hex	**0.62 ± 0.02**	**0.94 ± 0.01**	**1.80 ± 0.08**	**0.45 ± 0.01**

training runs and, for the prompt-only baselines, three distinct demonstration sets shared across both backbones. Decomposition-producing methods additionally undergo recursive execution and reassembly before the final-mesh check. We report mean±std over the three runs. Deterministic non-learning baselines are run once and assigned std$= 0$. Success Rate (SR) is the fraction of cases producing an accepted final mesh. We report SR, MinSJ, MeanSJ, and Aspect for successful final meshes. For each successful final mesh, MinSJ is the minimum element-wise scaled Jacobian and MeanSJ is its mean over all hexahedral elements. The table reports these per-mesh values averaged over successful cases. For a successful final mesh M, let $\mathrm{AR}(e) = \ell_{\max}(e)/\ell_{\min}(e)$ denote the aspect ratio of hexahedral element e, using its largest and smallest characteristic dimensions. For the successful-mesh set $\mathcal{M}^+$ of one evaluation run, we report

$$\text{Aspect} = \frac{1}{|\mathcal{M}^+|} \sum_{M \in \mathcal{M}^+} \max_{e \in M} \mathrm{AR}(e), \tag{7}$$

where lower values indicate better shape. Checkpoints and seed-level results are frozen before test evaluation.

Training setup. We fine-tune Qwen2-1.5B Yang et al. (2024) with LoRA (rank 32, $\alpha = 64$, dropout 0.05) while freezing base weights on one RTX 4090. The seed SFT trains for 12 epochs, followed by four epochs of post-bootstrap SFT. Rejection-sampling bootstrap uses $K = 8$ and retains $m = 3$ valid schemes. DPO uses 3,055 mined preference pairs for four epochs. Full candidate-construction details are provided in the Supplementary Material.

Inference and verification setup. At inference time, Hist2Hex performs one greedy rollout per sub-solid ($K = 1$, pass@1). The executor evaluates candidates in descending model probability and rejects cuts producing empty or non-manifold fragments. Geometry execution and the volume and overlap tests use Open CASCADE Technology (OCC) 7.5.1 through pythonocc-core 7.5.1. Relative volume and overlap tolerances are 10^{-3}. The structured meshing interface automatically selects mapped, submapped, swept, or template meshing and then performs conformal stitching. The full meshing and assembly check uses a 60 s timeout. These settings are fixed for training-data filtering and evaluation. Exact predicates and failure handling are provided in the Supplementary Material.

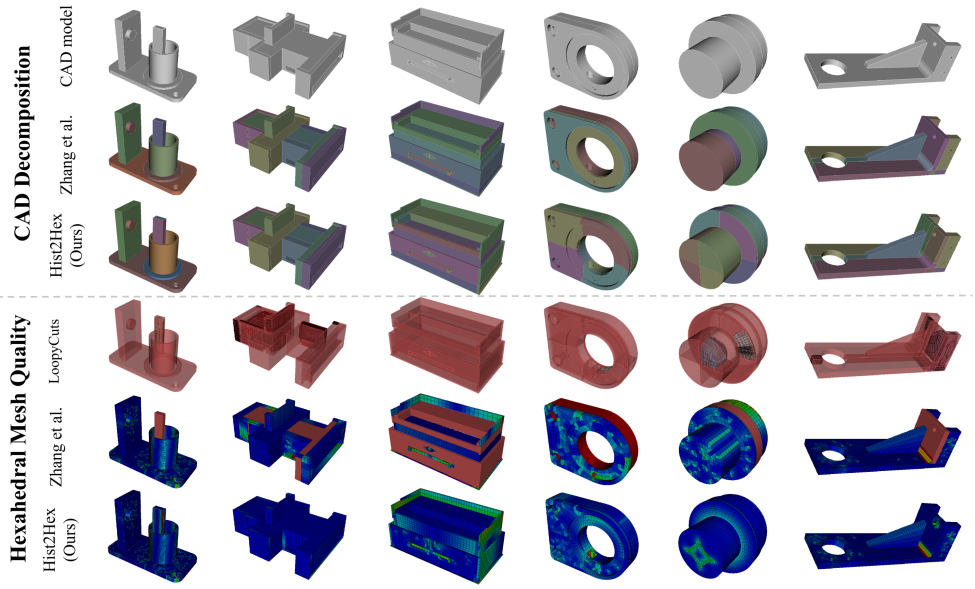


Figure 6: Qualitative comparison on six CAD models. Rows show the input, Zhang et al. Zhang et al. (2025), Hist2Hex, and mesh quality. Red indicates meshing failure and blue indicates high-quality elements.

4.2 RQ1: Quantitative Benchmark on Three Datasets

Table 1 compares Hist2Hex with geometry-based and learning-based baselines across three datasets. Hist2Hex achieves the highest SR on all three datasets. On Fusion360, it also obtains the best MinSJ. The uniform grid yields slightly better MeanSJ and Aspect but substantially lower SR. On DeepCAD, Hist2Hex achieves the best MinSJ, while on WHUCAD it obtains the best Aspect. Table 2 separately compares Hist2Hex with LoopyCuts under the latter's native success criterion.

4.3 RQ2: Qualitative Decomposition Quality

Figure 6 compares Hist2Hex with Zhang et al. Zhang et al. (2025) on six structurally diverse CAD models. Hist2Hex selects geometry-grounded planar and radial cuts that isolate cylindrical intersections, symmetric regions, and local fillets into more regular, hex-meshable blocks. In these examples, construction history helps choose among candidates derived from feature boundaries, planar faces, and circular geometry.

Table 2: Comparison under the LoopyCuts Livesu et al. (2020) native criterion: a non-empty, non-degenerate all-hexahedral volume mesh conforming to the input surface, without B-rep leaf-block or conformal-interface assembly requirements. MinSJ is computed over successful meshes.

Dataset	Method	Succ.↑	MinSJ↑
DeepCAD	LoopyCuts	0.23	0.30
	Hist2Hex	0.76	0.66
Fusion360	LoopyCuts	0.12	0.35
	Hist2Hex	0.79	0.62
WHUCAD	LoopyCuts	0.16	0.26
	Hist2Hex	0.68	0.48

4.4 RQ3: Ablation Study

Table 3 evaluates the training stages, history content, and representation and search components. The shuffled-history control permutes only the operation-type and target-body labels while preserving event order, per-event volume and bounding box, the Stage 1 partition, and the candidate pool. The history-context-removed control deletes the serialized timeline H_c from the model input while retaining the same Stage 1 partition and candidate pool. These controls separate the contribution of serialized history content from the decomposition prior induced by history-derived candidates.

Table 3: Ablation on Fusion360 (pass@1, n=1,500). All metrics are measured under the same final-mesh acceptance criterion as Table 1. The shuffled-history control keeps Stage 1, the candidate pool, and the prompt length identical, permuting only operation-type and target-body labels.

Variant	SR↑	Aspect↓
Full (Qwen2-1.5B, SFT+RS+DPO)	**0.60** ± **0.02**	**2.30** ± **0.12**
Training stages		
w/o DPO	0.57 ± 0.02	2.50 ± 0.18
SFT only	0.52 ± 0.03	2.80 ± 0.22
History content (Stage 1 and candidate pool held fixed)		
history op/body labels shuffled	0.54 ± 0.02	2.60 ± 0.18
history context removed	0.40 ± 0.03	2.95 ± 0.22
Representation / search		
w/o history-derived candidates	0.47 ± 0.04	3.20 ± 0.25
w/o geometric planes	0.53 ± 0.03	5.50 ± 0.32
w/o recursion	0.49 ± 0.03	2.55 ± 0.19
w/o verifier gate	0.44 ± 0.04	4.20 ± 0.35

4.5 RQ4: Practical Deployment Analysis

We additionally test whether a generated pure-hexahedral decomposition can be consumed directly by a standard downstream solver. Using the bracket in Figure 7, we model steel as an isotropic linear-elastic material, constrain its large left pin bore, and apply two equal loads in the positive x direction to the two smaller right bores. Across coarse, medium, and fine HEX8 meshes, the maximum displacement and strain energy converge, while the volume-weighted 95th-percentile von Mises stress remains stable. This experiment demonstrates solver-ready use of the assembled all-hexahedral mesh. The Supplementary Material gives the detailed settings.

We evaluate Stage 1 replay success by feature type on WHUCAD models. Figure 8 reports the SR. Chamfer, NewBody, and Fillet achieve the highest reliability, whereas body-selection- and topology-dependent features remain challenging due to stronger geometric context dependence.

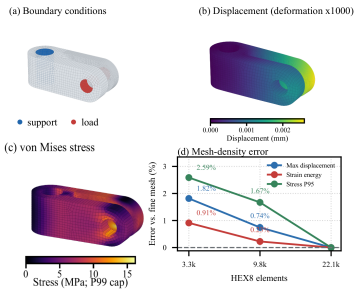


Figure 7: Downstream linear-elastic FEA. (a) Boundary conditions. (b) Fine-mesh displacement, magnified 1000×. (c) Von Mises stress. (d) Mesh-density errors relative to the 22,104-element mesh.

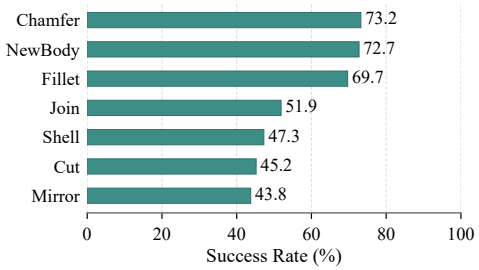


Figure 8: Stage 1 replay success rate by construction-feature type on WHUCAD. Chamfer, NewBody, and Fillet replay most reliably, whereas body-selection- and topology-dependent features have lower success rates.

5 Conclusion

Hist2Hex combines construction-history cues, candidate-indexed cut prediction, and verifier-gated training to produce executable decompositions for hexahedral meshing. Across three CAD datasets, history provides complementary structural information beyond the final B-rep, while rejection-sampling bootstrap and DPO improve the validity of predicted block structures.

Limitations. Hist2Hex assumes access to a complete, executable CAD construction history and therefore does not directly cover imported or legacy B-reps whose histories are unavailable. Whether partially reconstructed histories provide sufficient cues remains untested. The single-case FEA study demonstrates downstream solver compatibility rather than broad simulation validity.

REFERENCES

Paul Bourmaud, Rao Garimella, Navamita Ray, Cristina C. Garcia, Jean-Christophe Janodet, and Franck Ledoux. Monte carlo tree search based block decomposition of CAD models. In *Proceedings of the 2025 SIAM International Meshing Roundtable*, pp. 14–23. Society for Industrial and Applied Mathematics, 2025. doi: 10.1137/1.9781611978575.2. URL https://doi.org/10.1137/1.9781611978575.2.

Hendrik Brückler, David Bommes, and Marcel Campen. Volume parametrization quantization for hexahedral meshing. *ACM Transactions on Graphics (TOG)*, 41(4):1–19, 2022.

Benjamin C. DiPrete, Rao Garimella, Cristina Garcia Cardona, and Navamita Ray. Reinforcement learning for block decomposition of planar cad models. *Engineering with Computers*, 40(5): 2807–2817, Feb 2024. ISSN 1435-5663. doi: 10.1007/s00366-023-01940-6. URL http://dx.doi.org/10.1007/s00366-023-01940-6.

Elona Dupont, Kseniya Cherenkova, Anis Kacem, Sk Aziz Ali, Ilya Arzhannikov, Gleb Gusev, and Djamila Aouada. CADOps-Net: Jointly learning CAD operation types and steps from boundary-representations. In *2022 International Conference on 3D Vision (3DV)*, pp. 114–123. IEEE, 2022. doi: 10.1109/3DV57658.2022.00024. URL https://doi.org/10.1109/3DV57658.2022.00024.

Rubin Fan, Fazhi He, Yuxin Liu, Yupeng Song, Linkun Fan, and Xiaohu Yan. A parametric and feature-based cad dataset to support human-computer interaction for advanced 3d shape learning. *INTEGRATED COMPUTER-AIDED ENGINEERING*, 32(1):73–94, 2025. ISSN 1069-2509. doi: 10.3233/ICA-240744.

Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. All-hex meshing using closed-form induced polycube. *ACM Transactions on Graphics (TOG)*, 35(4):1–9, 2016.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming—the rise of code intelligence. *CoRR*, abs/2401.14196, 2024. doi: 10.48550/ARXIV.2401.14196. URL https://doi.org/10.48550/arXiv.2401.14196.

Lu He, Qitao Deng, Junjiang Deng, Liangbin Deng, Yanjun Liang, Wenting Yang, Guoqiang Wang, and Na Lei. Polycubenet: A dual-latent diffusion model for polycube-based hexahedral mesh generation, 2026. URL https://arxiv.org/abs/2605.20274.

Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. Solidgen: An autoregressive model for direct b-rep synthesis, 2022.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad models from beginner-to-expert level text prompts, 2024. URL https://arxiv.org/abs/2409.17106.

Heng Liu, Paul Zhang, Edward Chien, Justin Solomon, and David Bommes. Singularity-constrained octahedral fields for hexahedral meshing. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018. doi: 10.1145/3197517.3201344.

Marco Livesu, Nico Pietroni, Enrico Puppo, Alla Sheffer, Paolo Cignoni, et al. Loopycuts: practical feature-preserving block decomposition for strongly hex-dominant meshing. *ACM Trans. Graph.*, 39(4):121, 2020.

Jean Hsiang-Chun Lu, William Roshan Quadros, and Kenji Shimada. Evaluation of user-guided semi-automatic decomposition tool for hexahedral mesh generation. *Journal of Computational Design and Engineering*, 4(4):330–338, 2017. doi: 10.1016/j.jcde.2017.05.001. URL https://doi.org/10.1016/j.jcde.2017.05.001.

Max Lyon, David Bommes, and Leif Kobbelt. Hexex: Robust hexahedral mesh extraction. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016. doi: 10.1145/2897824.2925976.

Loïc Maréchal. Advances in octree-based all-hexahedral mesh generation: Handling sharp features. In *Proceedings of the 18th International Meshing Roundtable*, pp. 65–84. Springer, 2009.

Nico Pietroni, Marcel Campen, Alla Sheffer, Gianmarco Cherchi, David Bommes, Xifeng Gao, Riccardo Scateni, Franck Ledoux, Jean-François Remacle, and Marco Livesu. Hex-mesh generation and processing: A survey. *ACM Transactions on Graphics*, 42(2):1–44, Oct 2022. ISSN 1557-7368. doi: 10.1145/3554920. URL http://dx.doi.org/10.1145/3554920.

William Roshan Quadros, Nick Winovich, Corey Ernst, and Nya Hardy. Agent-guided geometry decomposition for hex meshing via heterogeneous graph neural network. In *Proceedings of the 2026 SIAM International Meshing Roundtable*, pp. 245–258. Society for Industrial and Applied Mathematics, 2026. doi: 10.1137/1.9781611979138.19. URL https://doi.org/10.1137/1.9781611979138.19.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design, 2020. URL https://arxiv.org/abs/2007.08506.

Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. Polycube-maps. *ACM transactions on graphics (TOG)*, 23(3):853–860, 2004.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics*, 40(4):1–24, Jul 2021. ISSN 1557-7368. doi: 10.1145/3450626.3459818. URL http://dx.doi.org/10.1145/3450626.3459818.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6752–6762. IEEE, Oct 2021. doi: 10.1109/ICCV48922.2021.00670. URL http://dx.doi.org/10.1109/ICCV48922.2021.00670.

Sifan Wu, Amir Khasahmadi, Mor Katz, Pradeep Kumar Jayaraman, Yewen Pu, Karl Willis, and Bang Liu. Cad-llm: Large language model for cad generation. In *NeurIPS 2023 Workshop on Machine Learning for Creativity and Design*, 2023. URL https://neurips.cc/virtual/2023/75064.

Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks, 2022.

Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D. D. Willis, and Daniel Ritchie. Inferring cad modeling sequences using zone graphs. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6058–6066. IEEE, Jun 2021. doi: 10.1109/CVPR46437.2021.00600. URL http://dx.doi.org/10.1109/CVPR46437.2021.00600.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jingyang Bai, Jinze He, Jinzheng Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Shuming Zhang, Zhidong Guan, Xiaodong Wang, Pingan Tan, and Hao Jiang. Reinforcement learning based automatic block decomposition of solid models for hexahedral meshing. *Computer-Aided Design*, 182:103850, 2025. ISSN 0010-4485. doi: 10.1016/j.cad.2025.103850. URL https://doi.org/10.1016/j.cad.2025.103850.