STRUCTURAL GEOMETRY CONDITIONING FOR EXECUTABLE MESH-TO-CAD RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

An editable CAD model requires a valid construction program, whereas an observed mesh specifies only the resulting surface. This mismatch makes mesh-to-CAD reconstruction especially fragile for multi-part objects: dense surface samples describe shape but may obscure the profile boundaries and operation choices needed for a short executable program. We use an external part segmenter and study a geometry-conditioned program generator that augments surface tokens with curves extracted from each mesh part at inference, concatenates the two streams, and weights operation-name tokens during supervised training. Training uses reference B-Rep curves, so curve inputs differ between training and evaluation. A compiler prepares canonical training targets from source histories; a deterministic CAD executor converts predicted programs to B-Rep and STEP output. On the reported 12-case multi-part evaluation, the system has Chamfer distance 0.010054, 95th-percentile surface distance 0.032163, and 64^3 voxel IoU 0.580917. On all 1,000 validation part programs, operation weighting raises STEP export success for the all-curve model from 0.894 to 0.958 and reduces penalized mean CD from 0.0198 to 0.0162. Linear curves alone outperform the unfiltered all-curve input on several program metrics, revealing a curve-selection trade-off. These results support structural conditioning for executable reconstruction while leaving broader generalization to larger, fully audited evaluations.

1 INTRODUCTION

Turning a triangle mesh into an editable CAD model is useful for reverse engineering, manufacturing, and downstream design changes. The output must contain more than a geometrically similar surface: a CAD system must be able to parse and execute a sequence of sketches, primitives, extrusions, and Boolean operations. An incorrect operation may change topology or invalidate the entire program even when nearby coordinates look plausible. Multi-part inputs introduce a further challenge because local part geometry must be reconstructed while preserving the assembly's common coordinate frame.

DeepCAD established sequence modeling of CAD programs (Wu et al., 2021); later reconstruction systems condition program generation on points and other observations (Khan et al., 2024a; Rukhovich et al., 2025; Kolodiazhnyi et al., 2025). These reconstruction approaches typically condition a sequence model on dense observations. CADFit instead searches over executable programs with geometric feedback (Nehme et al., 2026). Dense samples faithfully describe visible shape, but sharp edges, profile loops, and other low-dimensional construction cues can be diluted in a surface-token stream. Program likelihood alone also treats a mistaken operation name like an ordinary token error, although that token may control whether the result is executable.

There is no unique inverse from a finished surface to its construction history. A circular cavity, for example, may be represented through a sketch and cut or through a different combination of primitives and Booleans; both may have similar rendered geometry. This ambiguity makes full-sequence exact match useful for checking a chosen canonical target but insufficient as a sole measure of reconstruction. Conversely, a valid program can be far from the observed mesh. Our evaluation therefore separates program agreement, execution success, and final geometric fit. The same distinction guides the method: structural observations constrain what operations are plausible, and execution tests whether a predicted sequence defines a usable CAD artifact.

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.

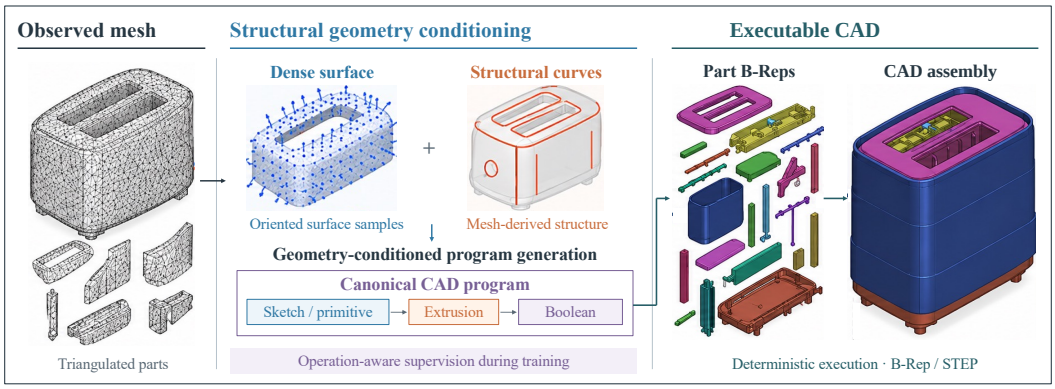


Figure 1: From an observed multi-part mesh to executable CAD geometry. At inference, surface samples and mesh-derived curves condition one program per part; parsing and execution produce part B-Reps and a STEP assembly. The drawing illustrates the task and output representation, while practical editability is not directly evaluated.

We study two interventions at different stages of program recovery: structural curves provide evidence about candidate profiles, while operation-aligned supervision changes the cost of errors in the generated construction sequence. At inference, our system extracts curves from the *input mesh*, encodes them alongside dense surface samples, and generates a canonical CAD program for each part. A weighted program loss emphasizes operation identifiers without introducing a separate prediction head. The resulting programs are executed by a deterministic CAD backend, then transformed into the input object's coordinate frame and assembled. Figure 1 shows the task and Figure 2 shows how the components interact.

The evidence has two complementary scopes. A controlled study on the full 1,000-instance validation split isolates curve inputs and operation-aware supervision using program, executability, and executed-geometry metrics. A separate 12-case multi-part study compares assembled geometry with three existing systems. The latter is a small external comparison with aggregate results; Table 3 instead tests geometry within the controlled variants. We keep the program-level, part-geometry, and assembled-object conclusions separate.

Our contributions are:

1. A part-wise conditioning pipeline that injects mesh-derived structural curves into surface-conditioned CAD program generation without using target B-Rep information at inference time.
2. An operation-aware training objective for canonical executable CAD programs, with a controlled comparison against identical geometry conditioning without that objective.
3. An evaluation spanning program accuracy, parsing, STEP export, and geometric agreement, including curve-type ablations and an explicit account of the method's observed failure modes.

2 Related Work

2.1 Programmatic CAD representations and datasets

Programmatic CAD differs from mesh synthesis because it records construction operations and their parameters. Fusion 360 Gallery provides human design histories and an executable reconstruction environment (Willis et al., 2021), while SketchGraphs represents two-dimensional primitives and constraints (Seff et al., 2020). Complementing these history- and sketch-oriented resources, ABC supplies parametric curves and surfaces for geometric learning (Koch et al., 2019). At the sketch level, SketchGen and Vitruvion generate primitives together with constraints (Para et al., 2021; Seff et al., 2022); Computer-Aided Design as Language combines serialization with language modeling

(Ganin et al., 2021). Learning recurring design concepts further captures modular sketch structure (Yang & Pan, 2022).

For solid construction, DeepCAD models command sequences (Wu et al., 2021), SkexGen separates topology, geometry, and extrusion through disentangled codebooks (Xu et al., 2022), and HNC-CAD organizes design variation into hierarchical neural codes (Xu et al., 2023). User-conditioned generation includes freehand drawing interpretation in Free2CAD (Li et al., 2022), text-to-sequence generation in Text2CAD (Khan et al., 2024b), and hierarchy-aware completion in FlexCAD (Zhang et al., 2025). Our study compiles canonical training programs to reconstruct observed mesh parts. Since different histories can yield similar solids, we distinguish sequence agreement, execution success, and geometric fit.

2.2 Reconstruction from geometric observations

Geometry-to-program reconstruction spans several operation vocabularies. CSGNet predicts constructive solid geometry programs (Sharma et al., 2018), while InverseCSG combines geometric processing with program synthesis to recover CSG trees from meshes (Du et al., 2018). CSG-Stump uses a fixed-depth Boolean representation (Ren et al., 2021); CAPRI-Net learns assemblies of quadric primitives with shape-specific adaptation (Yu et al., 2022). For sketch-and-extrude histories, Zone Graphs searches for modeling sequences from an input B-Rep (Xu et al., 2021), and CADParser directly maps B-Rep features to commands (Zhou et al., 2023). Their access to boundary entities differs from reconstruction using unstructured observations. Point2Cyl recovers extrusion cylinders from point clouds (Uy et al., 2022); ExtrudeNet and SECAD-Net learn inverse sketch-and-extrude reconstruction without ground-truth operation sequences (Ren et al., 2022; Li et al., 2023). CNC-Net studies the related but distinct vocabulary of machining operations (Yavartanoo et al., 2024).

Sequence predictors also differ in how they connect observations to construction structure. TransCAD uses hierarchical prediction and sketch-loop refinement (Dupont et al., 2025); CAD-SIGNet uses sketch-instance guided attention (Khan et al., 2024a). CAD-Diffuser instead reconstructs sequences through multimodal masked-token diffusion (Ma et al., 2024). CAD-Recode translates points into executable Python CAD code with a pretrained language decoder (Rukhovich et al., 2025), while Cadrille supports multiple input modalities and online reinforcement learning (Kolodiazhnyi et al., 2025). Execution feedback provides another axis: CADReasoner iteratively edits programs using geometric discrepancies (Kabisov et al., 2026), and CADFit fits and validates structured operations against mesh geometry (Nehme et al., 2026). Our system makes one autoregressive proposal per part from surface and mesh-derived curve features, followed by deterministic execution. These systems differ in input modality, program language, supervision, and inference budget. Our internal ablations fix the generator and target representation to isolate curve conditioning and operation weighting.

2.3 Structural geometry and geometry-conditioned models

PointNet processes unordered point sets (Qi et al., 2017a); PointNet++ adds hierarchical local aggregation (Qi et al., 2017b), and DGCNN learns neighborhood features with EdgeConv (Wang et al., 2019). Point Transformer introduces point-wise attention (Zhao et al., 2021), Point Transformer V3 scales processing through serialized neighborhoods (Wu et al., 2024), and Sonata learns transferable representations through self-supervision (Wu et al., 2025). For language conditioning, PointLLM aligns object point features with a language model (Xu et al., 2025), while 3D-LLM incorporates spatially grounded features for three-dimensional understanding (Hong et al., 2023); SpatialLM targets structured indoor modeling (Mao et al., 2025). These models motivate geometry-to-language interfaces, but do not by themselves specify executable CAD operations.

Supervised primitive fitting recovers analytic elements (Li et al., 2019), ParSeNet extends fitting to parametric patches including B-splines (Sharma et al., 2020), and HPNet combines learned and spectral descriptors for primitive segmentation (Yan et al., 2021). EC-Net preserves edges during point-set consolidation (Yu et al., 2018), whereas PIE-NET directly infers parametric feature curves (Wang et al., 2020). UV-Net and BRepNet learn from B-Rep geometry and topology (Jayaraman et al., 2021; Lambourne et al., 2021). Direct CAD reconstruction includes Point2CAD's surface fitting and intersection-based topology recovery (Liu et al., 2024) and ComplexGen's prediction of boundary entities and incidence relations (Guo et al., 2022); BrepGen further studies direct B-Rep generation using structured latent geometry (Xu et al., 2024). P3-SAM supplies our upstream part

segmentation (Ma et al., 2025). We condition an ordered program decoder on surface samples and mesh-derived curves, and emphasize operation tokens during training. Our claim concerns how structural evidence supports executable programs, rather than new curve representations or direct B-Rep graph prediction. Tessellation and extraction errors remain important limitations.

3 METHOD

Problem and notation. We use the automatic part masks of P3-SAM (Ma et al., 2025) to assign faces of the high-resolution input mesh to J geometric parts, denoted $M = \{M^{(j)}\}_{j=1}^{J}$. Part j has a canonical CAD program $Y^{(j)} = (y_1^{(j)}, \ldots, y_{T_j}^{(j)})$ of length T_j. A shared model conditions on each part separately and predicts $\widehat{Y}^{(j)}$; no other part's tokens enter that prediction. A deterministic CAD executor $\mathcal{E}$ parses each valid program to obtain a boundary representation $\widehat{B}^{(j)} = \mathcal{E}(\widehat{Y}^{(j)})$. The resulting solids are transformed back to the shared input coordinate system and assembled into $\widehat{B}_{\mathrm{obj}}$. Exporting a valid STEP file is an executability measure, while agreement with the reference geometry is evaluated separately.

Pipeline overview. We call the three model components *Structural Curve Encoding* (SCE), *Dual-Stream Geometry Fusion* (DGF), and *Operation-Aware CAD Decoding* (OCD). SCE obtains a variable-length curve representation from each segmented part; DGF concatenates its projected curve tokens with continuous surface tokens; OCD generates one canonical program per part and trains its shared decoder with program cross-entropy plus an operation-weighted term. P3-SAM segmentation, the compiler that prepares training targets, and the CAD executor that evaluates predictions are external interfaces around these components. Figure 2 follows one part through extraction, encoding, fusion, generation, and execution; the last stage restores and assembles the parts.

The components address consecutive decisions in the same reconstruction problem. SCE exposes candidate profile geometry before any program is predicted, DGF makes that evidence available beside the broader surface description, and OCD focuses supervision on the operations that turn a profile into a solid. Their roles are tested at different points in Table 2: curve variants probe the conditioning path, while A3 versus P1 probes the loss with fixed all-curve input.

3.1 STRUCTURAL CURVE ENCODING

At validation and deployment, for each part $M^{(j)}$, a deterministic extractor $\mathcal{R}$ identifies candidate boundary or high-curvature edge segments, links compatible adjacent segments, and fits retained chains to geometric curves:

$$\widehat{\mathcal{C}}^{(j)} = \mathcal{R}(M^{(j)}) = \{\widehat{C}_k^{(j)}\}_{k=1}^{K_j}, \qquad \widehat{C}_k^{(j)} = \mathrm{Fit}(Q_k^{(j)}). \tag{1}$$

Here $Q_k^{(j)}$ is an ordered candidate chain and $K_j \leq 128$ is the number of retained curves for part j. We distinguish linear and non-linear candidates for the controlled ablation. The extractor uses no target operation labels or reference B-Rep at inference. Candidate-edge detection, chain filtering, curve fitting, and duplicate suppression are deterministic and shared across controlled variants. During training, the curve stream instead uses curves from the reference B-Rep $B^{(j)}$ of the source design; K_j denotes the active curve count in either phase and may differ between phases. This creates a train–inference gap in the curve inputs; the validation results therefore test whether the learned conditioning transfers to mesh-derived curves.

A fixed sampler maps each retained curve to pointwise geometric descriptors. A pretrained, frozen C2 encoder E_c maps these descriptors to $\mathbf{z}_k^{(j)} = E_c(C_k^{(j)}) \in \mathbb{R}^{256}$, where $C_k^{(j)}$ is a reference B-Rep curve during training and $\widehat{C}_k^{(j)}$ from Eq. 1 during inference. Stacking gives $Z_c^{(j)} \in \mathbb{R}^{K_j \times 256}$. Frozen training features are cached; at inference, the same sampler and encoder operate on curves extracted from the input part. When no curve survives, the curve stream is empty and the model receives surface conditioning only. No reference B-Rep curve is passed to the predictor during validation or deployment.

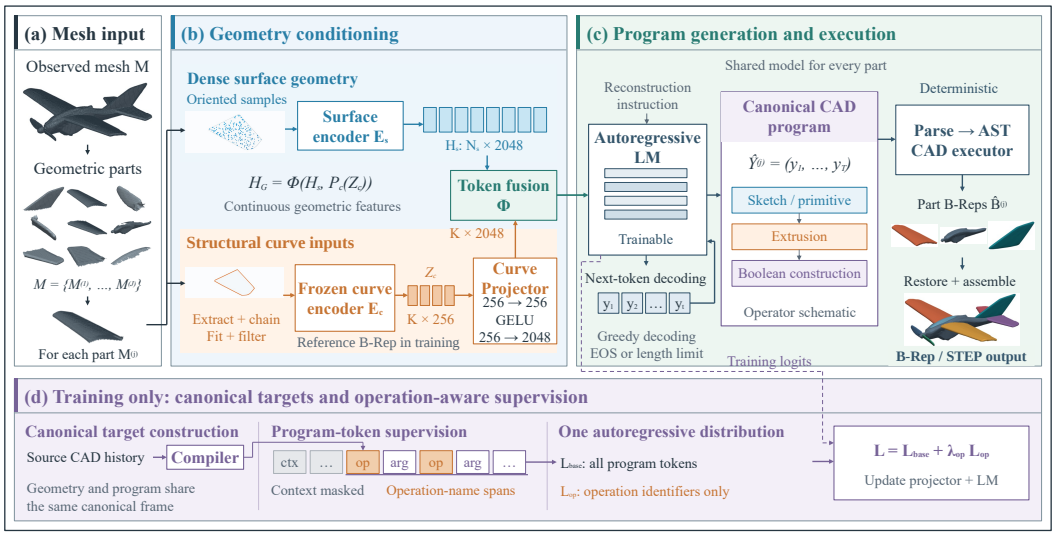


Figure 2: Mesh-to-CAD pipeline. (a) P3-SAM segments the input into geometric parts. (b) At inference, Structural Curve Encoding embeds mesh-derived candidates, and Dual-Stream Geometry Fusion concatenates curve and surface tokens. Training instead supplies reference B-Rep curves to the frozen curve encoder, as marked in the diagram. (c) A shared Operation-Aware CAD Decoder generates one program per part; deterministic parsing and execution yield part B-Reps, which are restored and assembled. (d) During training, source histories are compiled into canonical targets and operation-name spans receive additional loss weight.

3.2 Dual-Stream Geometry Fusion

We sample oriented surface points $P^{(j)} = \{(\mathbf{x}_i^{(j)}, \mathbf{n}_i^{(j)})\}_{i=1}^{N_j}$ from $M^{(j)}$, where $\mathbf{x}_i^{(j)} \in \mathbb{R}^3$ is position and $\mathbf{n}_i^{(j)} \in \mathbb{R}^3$ is normal. A surface encoder E_s returns $H_s^{(j)} = E_s(P^{(j)}) \in \mathbb{R}^{N_s \times 2048}$, where N_s is the number of surface tokens and 2048 is the language-model feature width. A shared coordinate normalization maps the part and its target program together: if s is the uniform scale making the input bounding-box diagonal six and $\mathbf{t}$ is a translation, then positions obey $\mathbf{x}' = s\mathbf{x} + \mathbf{t}$, lengths obey $l' = sl$, and unit directions are unchanged. The inverse transform is applied after CAD execution.

Each curve embedding is projected independently into the same feature width as the surface tokens:

$$\widetilde{Z}_c^{(j)} = P_c(Z_c^{(j)}) \in \mathbb{R}^{K_j \times 2048}, \qquad P_c(\mathbf{z}) = W_2\,\mathrm{GELU}(W_1\mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2, \tag{2}$$

where $P_c : \mathbb{R}^{256} \to \mathbb{R}^{2048}$, $W_1 \in \mathbb{R}^{256\times256}$, $W_2 \in \mathbb{R}^{2048\times256}$, $\mathbf{b}_1 \in \mathbb{R}^{256}$, and $\mathbf{b}_2 \in \mathbb{R}^{2048}$ are learned parameters. The two streams enter the decoder by direct token-sequence concatenation:

$$H_G^{(j)} = \Phi\big(H_s^{(j)}, \widetilde{Z}_c^{(j)}\big) = [H_s^{(j)}; \widetilde{Z}_c^{(j)}] \in \mathbb{R}^{(N_s+K_j)\times2048}, \tag{3}$$

where the semicolon means that the N_s surface tokens precede the K_j projected curve tokens. Both streams remain continuous features of width 2048; no curve coordinates are serialized as natural-language text. If $K_j = 0$, $H_G^{(j)} = H_s^{(j)}$. The same concatenation rule and decoder parameters are reused across parts. Thus, any object-level interaction happens when executed solids are restored and assembled, rather than through a joint object program.

3.3 Operation-Aware CAD Decoding

Source CAD histories are compiled into a common executable vocabulary of sketches, primitives, extrusions, and Boolean operations. In this representation, a profile-creation operation precedes the extrusion or Boolean action that consumes it; nested inner profiles become explicit subtraction where supported. This canonicalization reduces superficial sequence variation and defines the target of exact-match evaluation; it does not make construction history identifiable from geometry alone. A source history and its target program are normalized by the same spatial map used for the input mesh.

For each part j, the language model with shared parameters θ generates its canonical program from the part-specific context:

$$p_\theta(Y^{(j)} \mid H_G^{(j)}) = \prod_{t=1}^{T_j} p_\theta\big(y_t^{(j)} \mid y_{<t}^{(j)}, H_G^{(j)}\big). \tag{4}$$

For a training part j, only target-program positions $\Omega_j \subseteq \{1, \ldots, T_j\}$ contribute to the base loss; prompt, system, and geometry-placeholder positions are masked. Let $m_t^{(j)} \in \{0, 1\}$ identify subword tokens belonging to CAD operation names in the canonical target, and let $c_t^{(j)} = -\log p_\theta(y_t^{(j)} \mid y_{<t}^{(j)}, H_G^{(j)})$. The per-part training objective, averaged over training parts in a batch, is

$$\mathcal{L}^{(j)} = \underbrace{\frac{\sum_{t\in\Omega_j} c_t^{(j)}}{|\Omega_j|}}_{\mathcal{L}_{\text{prog}}^{(j)}} + \lambda_{\text{op}} \underbrace{\frac{\sum_{t\in\Omega_j} m_t^{(j)} c_t^{(j)}}{\max(1, \sum_{t\in\Omega_j} m_t^{(j)})}}_{\mathcal{L}_{\text{op}}^{(j)}}. \tag{5}$$

The nonnegative coefficient λ_{op} sets the extra weight on operation identifiers; the denominator handles programs with no marked operation tokens. Thus every target token has base coefficient $1/|\Omega_j|$, and a marked operation token receives the additional coefficient $\lambda_{\text{op}}/\max(1, \sum_{t\in\Omega_j} m_t^{(j)})$. The same output vocabulary and decoder are used for both terms. A0–A3 use $\lambda_{\text{op}} = 0$, while P1 uses $\lambda_{\text{op}} = 1.0$. Greedy generation stops at EOS or the maximum token budget. Valid per-part programs are parsed, executed, placed by the inverse coordinate transform, assembled, and exported to STEP.

Let $\mathcal{T}_j^{-1}$ denote the inverse of the part's normalization and placement map, and let $\mathcal{A}$ denote deterministic assembly of the resulting solids. The final output is

$$\widehat{B}_{\text{obj}} = \mathcal{A}\left(\left\{\mathcal{T}_j^{-1}\Big(\mathcal{E}(\widehat{Y}^{(j)})\Big)\right\}_{j=1}^{J}\right). \tag{6}$$

Execution is outside the training loss: Eq. 5 optimizes token probabilities, while parse and STEP rates measure the downstream effect of those predictions. This separation explains why operation weighting is a targeted but limited intervention. It increases the cost of errors in operation-name spans; it does not directly optimize geometry or guarantee a correct Boolean topology.

4 Experiments

4.1 Datasets, implementation, and protocol

The EXACT9998 collection contains 9,998 program records derived from DeepCAD and Fusion 360 Gallery sources (Wu et al., 2021; Willis et al., 2021). It is split into 7,998 training records (after excluding two records originally assigned to training), 1,000 validation records, and 1,000 test records. Tables 2 and 3 use *all* 1,000 validation part-program instances; their denominator is predicted programs, including failures, rather than assembled objects. A separate set of 12 multi-part objects measures executed geometry against a common reference for CADFit, CAD-Recode, Cadrille, and our system. Its relation to the EXACT9998 splits is not established by the available record. The latter protocol evaluates final solids rather than agreement with a particular program history. Table 1 contains aggregate values only, with no case identifiers, per-case scores, or selection rule. We therefore report its values descriptively and do not claim paired significance or component attribution from it. The visual gallery is not mapped to those 12 objects. Our pipeline includes P3-SAM segmentation; baseline part handling is not documented in the aggregate record, so only the internal variants constitute controlled changes to the proposed model.

The five variants share P3-SAM preprocessing, a frozen point-cloud tower, a geometry-pretrained SpatialLM-Llama backbone, the canonical program vocabulary, and greedy decoding (Mao et al., 2025). All variants start from the same backbone weights. A0 uses surface tokens; A1 adds linear curves; A2 adds non-linear curves; A3 adds all retained curves; P1 uses A3 geometry with $\lambda_{\text{op}} = 1.0$. A frozen C2 curve encoder produces 256-dimensional features for at most 128 curves per example; the 256–256–GELU–2048 projector has 592,128 learned parameters. Training uses reference B-Rep curves, whereas validation and deployment use mesh-extracted curves.

Table 1: Reported mean geometry on 12 multi-part cases. Lower CD and H95 and higher IoU_{64} are better. Per-case values and uncertainty estimates are unavailable.

Method	CD ↓	H95 ↓	IoU_{64} ↑
Cadrille	0.020054	0.070462	0.300782
CAD-Recode	0.013433	0.045794	0.386169
CADFit	0.013692	0.038845	0.371324
Ours	**0.010054**	**0.032163**	**0.580917**

Table 2: Controlled ablation on all 1,000 EXACT9998 validation instances. A0–A3 vary curve conditioning under the base loss; P1 adds operation weighting to A3. Higher is better for Exact, Op-F1, Parse, STEP, and EOS; lower is better for MaxLen and UnderGen.

Variant	Exact ↑	Op-F1 ↑	Parse ↑	STEP ↑	EOS ↑	MaxLen ↓	UnderGen ↓
A0: Surface	0.047	0.6318	0.882	0.874	0.964	0.036	0.333
A1: + Linear	0.052	0.7578	0.941	0.929	0.969	0.031	0.290
A2: + Non-linear	0.041	0.6842	0.875	0.868	0.929	0.071	0.327
A3: + All curves	0.046	0.7390	0.904	0.894	0.933	0.067	0.258
P1: A3 + Op. Loss	**0.098**	**0.8119**	**0.963**	**0.958**	**0.990**	**0.010**	**0.206**

All variants use AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, zero weight decay), initial learning rate 2×10^{-5}, a cosine schedule with 0.03 warmup ratio, per-device batch size 1, and gradient accumulation 4 (effective batch size 4 on one GPU). The reported runs use 20 epochs, 39,980 optimizer steps, bfloat16, gradient clipping at 1.0, and seed 42; evaluation and checkpoint saving occur every 250 steps. The selected checkpoint minimizes validation base cross-entropy, excluding the operation term and downstream metrics: step 6,000 for A0–A3 and step 8,000 for P1. Thus Tables 2 and 3 report the same validation split used for checkpoint selection, not untouched test-set performance. At inference, the shared model decodes each part until EOS or its token limit and the CAD backend executes and assembles valid outputs.

4.2 Metrics and external geometric comparison

Full-program exact match tests agreement with the canonical target. Operation F1 (Op-F1) compares predicted and target operation sequences. Parse and STEP are the fractions of outputs that parse and export successfully. EOS and MaxLen denote normal termination and generation-budget exhaustion; in the reported evaluation they sum to one and are two views of the same termination event. UnderGen is the fraction flagged as prematurely incomplete by the sequence-completeness evaluator. For geometry, CD is symmetric surface Chamfer distance, H95 is the 95th-percentile bidirectional surface distance, and IoU_{64} is volumetric intersection-over-union on a 64^3 grid. In the controlled geometry ablation, a fixed CD and H95 penalty is assigned if and only if STEP export fails. Thus the respective penalized counts for A0, A1, A2, A3, and P1 are 126, 71, 132, 106, and 42 of 1,000; the summaries combine execution and fit effects. The numerical penalty constants are absent from the available evaluation record. Metric implementations are held fixed across reported variants.

Table 1 reports the lowest aggregate CD and H95 and the highest IoU_{64} for our system on these 12 cases. Compared with the strongest listed baseline for each metric, CD is 25.2% lower than CAD-Recode, H95 is 17.2% lower than CADFit, and IoU_{64} is 0.194748 higher than CAD-Recode. The IoU comparison is an absolute difference in the unit interval, not a claim of 19.5% relative improvement. Given the small case count and missing paired scores, these numbers describe this evaluation only.

4.3 Controlled ablations

Table 2 contains the existing controlled ablations: A0–A3 compare curve inputs under the same base loss, and A3 versus P1 tests operation weighting with the same all-curve input. These comparisons support the corresponding component-level conclusions below.

Table 3: Penalized executed-geometry ablation on the same 1,000 EXACT9998 validation part-program instances as Table 2. STEP-export failures receive fixed CD/H95 penalties and remain in the mean and median. Lower is better.

Variant	CD Mean ↓	CD Median ↓	H95 Mean ↓	H95 Median ↓
A0: Surface	0.0248	0.0105	0.0745	0.0302
A1: + Linear	0.0233	0.0089	0.0765	0.0248
A2: + Non-linear	0.0233	0.0095	0.0799	0.0237
A3: + All curves	0.0198	0.0094	0.0665	0.0214
P1: A3 + Op. Loss	**0.0162**	**0.0068**	**0.0502**	**0.0165**

Structural curve encoding. A1 raises Op-F1 from 0.6318 to 0.7578 and STEP success from 0.874 to 0.929 relative to A0. Its 55 additional STEP successes per 1,000 examples indicate that linear structural evidence can help the generator under this shared training protocol. A2 improves Op-F1 by 0.0524 but reduces STEP success by 0.006 relative to A0; a curve stream is therefore not uniformly beneficial. A3 is weaker than A1 on Op-F1 (0.7390 versus 0.7578) and STEP (0.894 versus 0.929), although its UnderGen rate is lower (0.258 versus 0.290). These non-monotone results show a trade-off across metrics and motivate selecting structural candidates by relevance rather than treating all extracted curves as equally useful.

Operation-aware supervision. P1 and A3 share the all-curve input, making their comparison the direct test of Eq. 5. Operation weighting raises Exact by 0.052, Op-F1 by 0.0729, Parse by 0.059, and STEP by 0.064. MaxLen falls from 0.067 to 0.010, and UnderGen from 0.258 to 0.206. Since EOS is the complement of MaxLen in this table, we count this as one termination improvement. STEP export is not geometric correctness: only 0.098 of P1 programs exactly match the canonical target.

Executed geometry under controlled changes. Table 3 uses the same 1,000 instances as the program ablation. Relative to A0, A3 lowers penalized mean CD from 0.0248 to 0.0198 and mean H95 from 0.0745 to 0.0665. Holding all-curve input fixed, P1 further lowers them to 0.0162 and 0.0502; its medians are also the lowest in the table. Relative to A0, P1 has 34.7% lower penalized mean CD and 32.6% lower penalized mean H95, although this comparison changes both the curve input and the loss. The effect of curve type is not uniform: A1 and A2 have higher mean H95 than A0 despite lower H95 medians, and A1 has a lower median CD than A3. A3 has lower penalized mean CD/H95 than A1 but lower STEP success (0.894 versus 0.929). These aggregates combine executability and fit among valid outputs; without the actual penalty constants and per-case scores, we cannot recover valid-output error or assess paired uncertainty.

4.4 Qualitative analysis and limitations

Figure 3 illustrates differences in profiles, part proportions, and the ability to reconstruct an executable object. The gallery is qualitative evidence only: its relation to the 12-case benchmark is unspecified. Curved boundaries, ambiguous sharp edges, and long programs remain visible failure modes. The extractor may miss a curve that corresponds to a desired CAD operation or add a misleading tessellation edge. Accordingly, neither an exported STEP file nor a visually plausible rendering alone verifies a faithful construction history.

The larger gallery in Appendix Figure 4 adds objects with distinct part counts, profile types, and overall scales. Its role is to show what the reported aggregate geometric metrics conceal: a reconstruction can preserve the main silhouette while altering a local cavity or connection, and two executions with similar surfaces may have different part structure. These differences are important for editability and are only partially reflected by surface CD. H95 emphasizes localized surface outliers, while IoU_{64} responds to coarse volumetric agreement. Reading all three metrics alongside the renderings gives a more informative picture than any single number, although it still does not establish that the predicted construction tree matches the designer's intent.

Evaluation scope and uncertainty. The external comparison contains only 12 multi-part cases and reports aggregate metrics. Per-case geometric results, the case-selection rule, and paired uncertainty estimates would be required to determine whether the apparent advantage is broad or driven by a few

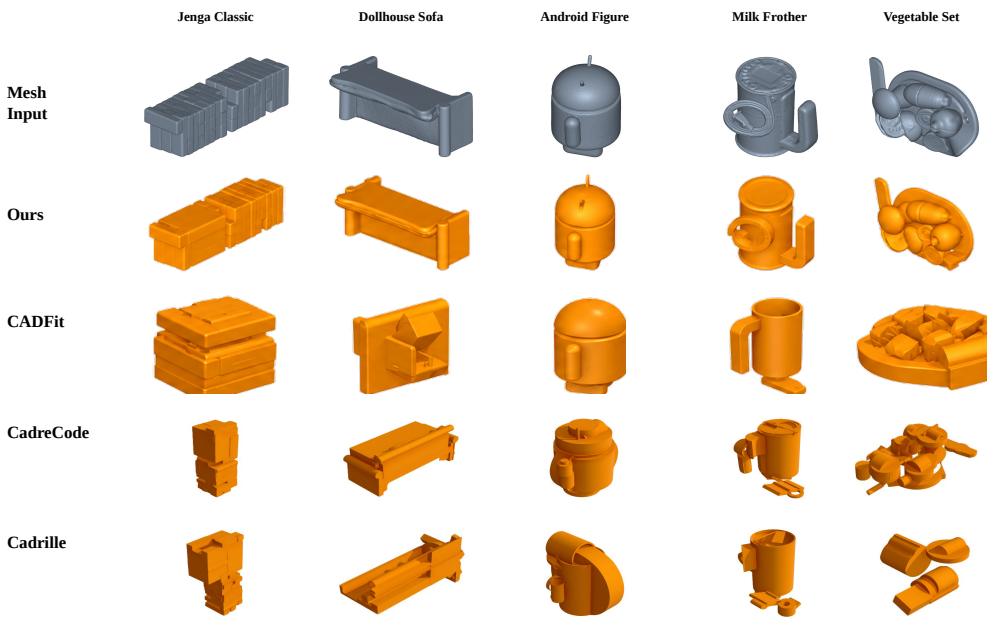


Figure 3: Five visual comparisons of input geometry and executable CAD reconstructions from the evaluated methods. The gallery contains 19 labeled cases; their overlap with the 12 quantitative cases is unspecified.

examples. The visualization gallery cannot substitute for those records: its 19 labeled cases are not mapped to the 12 quantitative cases, and the gallery selects visible views rather than measuring all surfaces. We therefore treat Table 1 as descriptive evidence for the tested set. A larger evaluation with explicit object categories, part counts, and reference-program complexity would be needed to test out-of-distribution behavior.

Component attribution. The controlled variants link program-level and geometric summaries on the same set, but they do not reveal the interaction between curve choice and loss choice. An A1 model with operation weighting would test whether all curves help beyond lines; equal-token-count random curves would test whether added structural content, rather than conditioning capacity, explains the difference. The curve variants train on reference B-Rep curves and evaluate on mesh-extracted curves, so a matched-source training control would also be needed to isolate the effect of this input shift. Paired per-instance geometry would separate broadly shared gains from outlier-driven means.

5 Conclusion

We presented a part-wise mesh-to-CAD pipeline that concatenates structural curves with dense surface tokens and trains an executable program decoder with additional weight on CAD operation identifiers. On the same 1,000 validation instances, the controlled results show that linear curves improve program structure and STEP success over surface-only conditioning, while operation weighting improves the all-curve model on program and penalized geometry metrics. The 12-case assembled-object comparison favors the resulting system on the reported aggregate metrics. Non-linear curves do not consistently help; the train–inference curve-source shift and missing failure penalties and per-case records limit the geometric claim. Reliable curve selection and a fully audited test-set evaluation are the next steps.

AI USE STATEMENT

Before this revision, generative AI was used only for grammar checking and correction. In preparing this revision, generative AI assisted with manuscript structure and prose, mathematical exposition, literature identification, critical review, and interpretation of previously reported aggregate results. Three AI reviewer agents provided multiple rounds of simulated reviews that helped identify missing evidence and internal inconsistencies; these are not official ICLR reviews. No new experimental measurements or figures were generated by AI in this revision. The authors retain responsibility for every claim, citation, numerical result, figure, and AI-assisted change, and will provide a matching disclosure in the submission form.

REPRODUCIBILITY STATEMENT

Section 3 defines the data flow and objective, and Section 4 specifies the reported evaluation groups, split sizes, and training configuration. Independent replication additionally requires the training and evaluation code, split identifiers and object-grouping policy, curve extraction settings, exact backbone and curve-encoder weight versions, executor version, numerical failure penalties, per-case results, and baseline commands. The penalty constants are not recorded in the materials used for this revision, so Table 3 cannot be independently recomputed from the manuscript alone.

REFERENCES

Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. InverseCSG: Automatic Conversion of 3D Models to CSG Trees. *ACM Trans. Graph.*, 37(6), 2018. doi: 10.1145/3272127.3275006. URL https://doi.org/10.1145/3272127.3275006.

Elona Dupont, Kseniya Cherenkova, Dimitrios Mallis, Gleb Gusev, Anis Kacem, and Djamila Aouada. TransCAD: A Hierarchical Transformer for CAD Sequence Inference from Point Clouds. In *Computer Vision – ECCV 2024*, pp. 19–36, 2025.

Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-Aided Design as Language. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5885–5897, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/2e92962c0b6996add9517e4242ea9bdc-Paper.pdf.

Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics*, 41(4), 2022. doi: 10.1145/3528223.3530078.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3D World into Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 20482–20494, 2023. doi: 10.52202/075280-0900. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/413885e70482b95dcbeeddc1daf39177-Paper-Conference.pdf.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D.D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. UV-Net: Learning From Boundary Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11703–11712, 2021.

Soslan Kabisov, Vsevolod Kirichuk, Andrey Volkov, Gennadii Savrasov, Marina Barannikov, Anton Konushin, Andrey Kuznetsov, and Dmitrii Zhemchuzhnikov. Cadreasoner: Iterative program editing for cad reverse engineering. *arXiv preprint arXiv:2603.29847*, 2026. URL https://arxiv.org/abs/2603.29847.

Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024a. URL https://openaccess.thecvf.com/content/CVPR2024/papers/Khan_CAD-SIGNet_CAD_Language_Inference_from_Point_Clouds_using_Layer-wise_Sketch_CVPR_2024_paper.pdf.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2CAD: Generating Sequential CAD Designs from Beginner-to-Expert Level Text Prompts. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7552–7579, 2024b. doi: 10.52202/079017-0242. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0e5b96f97c1813bb75f6c28532c2ecc7-Paper-Conference.pdf.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A Big CAD Model Dataset for Geometric Deep Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Maksim Kolodiazhnyi, Denis Tarasov, Dmitrii Zhemchuzhnikov, Alexander Nikulin, Ilya Zisman, Anna Vorontsova, Anton Konushin, Vladislav Kurenkov, and Danila Rukhovich. cadrille: Multi-modal cad reconstruction with online reinforcement learning. *arXiv preprint arXiv:2505.22914*, 2025. URL https://arxiv.org/abs/2505.22914.

Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/papers/Lambourne_BRepNet_A_Topological_Message_Passing_System_for_Solid_Models_CVPR_2021_paper.pdf.

Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Free2CAD: Parsing Freehand Drawings into CAD Commands. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2022)*, 41(4):93:1–93:16, 2022. doi: 10.1145/3528223.3530133.

Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J. Guibas. Supervised Fitting of Geometric Primitives to 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. SECAD-Net: Self-Supervised CAD Reconstruction by Learning Sketch-Extrude Operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16816–16826, 2023.

Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. Point2cad: Reverse engineering cad models from 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3763–3772, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/papers/Liu_Point2CAD_Reverse_Engineering_CAD_Models_from_3D_Point_Clouds_CVPR_2024_paper.pdf.

Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-sam: Native 3d part segmentation. *arXiv preprint arXiv:2509.06784*, 2025. URL https://arxiv.org/html/2509.06784v3.

Weijian Ma, Shuaiqi Chen, Yunzhong Lou, Xueyang Li, and Xiangdong Zhou. Draw Step by Step: Reconstructing CAD Construction Sequences from Point Clouds via Multimodal Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27154–27163, 2024.

Yongsen Mao, Junhao Zhong, Chuan Fang, Jia Zheng, Rui Tang, Hao Zhu, Ping Tan, and Zihan Zhou. Spatiallm: Training large language models for structured indoor modeling. *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.07491.

Ghadi Nehme, Eamon Whalen, and Faez Ahmed. Cadfit: Precise mesh-to-cad program generation with hybrid optimization. *arXiv preprint arXiv:2605.01171*, 2026. URL https://arxiv.org/abs/2605.01171.

Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas Guibas, and Peter Wonka. SketchGen: Generating Constrained CAD Sketches. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5077–5088, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/28891cb4ab421830acc36b1f5fd6c91e-Paper.pdf.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*, volume 30, 2017b. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. CSG-Stump: A Learning Friendly CSG-Like Representation for Interpretable Shape Parsing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12478–12487, 2021.

Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. ExtrudeNet: Unsupervised Inverse Sketch-and-Extrude for Shape Parsing. In *Computer Vision – ECCV 2022*, pp. 482–498, 2022.

Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/papers/Rukhovich_CAD-Recode_Reverse_Engineering_CAD_Code_from_Point_Clouds_ICCV_2025_paper.pdf.

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

Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A Generative Model of Parametric CAD Sketches. In *International Conference on Learning Representations*, 2022.

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural Shape Parser for Constructive Solid Geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds. In *Computer Vision – ECCV 2020*, pp. 261–276, 2020.

Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G. Lambourne, Tolga Birdal, and Leonidas J. Guibas. Point2Cyl: Reverse Engineering 3D Objects From Point Clouds to Extrusion Cylinders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11850–11860, 2022.

Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. PIE-NET: Parametric Inference of Point Cloud Edges. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20167–20178, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/e94550c93cd70fe748e6982b3439ad3b-Paper.pdf.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)*, 2019.

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), 2021. doi: 10.1145/3450626.3459818.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/papers/Wu_DeepCAD_A_Deep_Generative_Network_for_Computer-Aided_Design_Models_ICCV_2021_paper.pdf.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler Faster Stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4840–4851, 2024.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Wu_Sonata_Self-Supervised_Learning_of_Reliable_Point_Representations_CVPR_2025_paper.pdf.

Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. PointLLM: Empowering Large Language Models to Understand Point Clouds. In *Computer Vision – ECCV 2024*, pp. 131–147, 2025.

Xiang Xu, Karl D.D. Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. SkexGen: Autoregressive Generation of CAD Construction Sequences with Disentangled Codebooks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 24698–24724, 2022. URL https://proceedings.mlr.press/v162/xu22k.html.

Xiang Xu, Pradeep Kumar Jayaraman, Joseph George Lambourne, Karl D.D. Willis, and Yasutaka Furukawa. Hierarchical Neural Coding for Controllable CAD Model Generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 38443–38461, 2023. URL https://proceedings.mlr.press/v202/xu23f.html.

Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024.

Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D.D. Willis, and Daniel Ritchie. Inferring CAD Modeling Sequences Using Zone Graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6062–6070, 2021.

Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qixing Huang. HPNet: Deep Primitive Segmentation Using Hybrid Representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2753–2762, 2021.

Yuezhi Yang and Hao Pan. Discovering Design Concepts for CAD Sketches. In *Advances in Neural Information Processing Systems*, volume 35, pp. 28803–28814, 2022. doi: 10.52202/068431-2088. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b9432d0f94275f0571c6cc99cf8b1664-Paper-Conference.pdf.

Mohsen Yavartanoo, Sangmin Hong, Reyhaneh Neshatavar, and Kyoung Mu Lee. CNC-Net: Self-Supervised Learning for CNC Machining Operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9816–9825, 2024.

Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. CAPRI-Net: Learning Compact CAD Shapes With Adaptive Primitive Assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11768–11778, 2022.

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. EC-Net: an Edge-aware Point set Consolidation Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Zhanwei Zhang, Shizhao Sun, Wenxiao Wang, Deng Cai, and Jiang Bian. FlexCAD: Unified and Versatile Controllable CAD Generation with Fine-tuned Large Language Models. In *International Conference on Learning Representations*, volume 2025, pp. 3204–3227, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/090b23d52bc2722eef2fbf79c5ebf9ec-Paper-Conference.pdf.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16259–16268, 2021.

Shengdi Zhou, Tianyi Tang, and Bin Zhou. CADParser: A Learning Approach of Sequence Modeling for B-Rep CAD. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 1804–1812, 2023. doi: 10.24963/ijcai.2023/200. URL https://doi.org/10.24963/ijcai.2023/200.

A SENSITIVITY OF PENALIZED GEOMETRY MEANS

For variant v and distance metric $d \in \{\mathrm{CD}, \mathrm{H95}\}$, let $\bar{d}_v$ be the reported all-instance mean, r_v the fraction of instances that do not receive a failure penalty, C_d the fixed penalty, and $\bar{d}_v^{\mathrm{valid}}$ the mean among unpenalized instances. Their exact relationship is

$$\bar{d}_v = r_v \bar{d}_v^{\mathrm{valid}} + (1 - r_v) C_d. \tag{7}$$

By the confirmed failure rule, r_v equals the STEP-success fraction in Table 2; only C_d remains unreported. Substituting the Table 3 means into Eq. 7, A3 and P1 have equal conditional valid-output means at $C_{\mathrm{CD}} \approx 0.0701$ and $C_{\mathrm{H95}} \approx 0.2942$. For either metric, P1 has lower conditional error below its break-even penalty and higher conditional error above it. These thresholds use rounded reported means and are not estimates of the actual penalties. Therefore the lower penalized mean does not establish which variant fits better conditional on successful execution.

B ADDITIONAL QUALITATIVE EXAMPLES

These panels extend the visualization gallery. They are not used to derive the aggregate scores in Table 1.

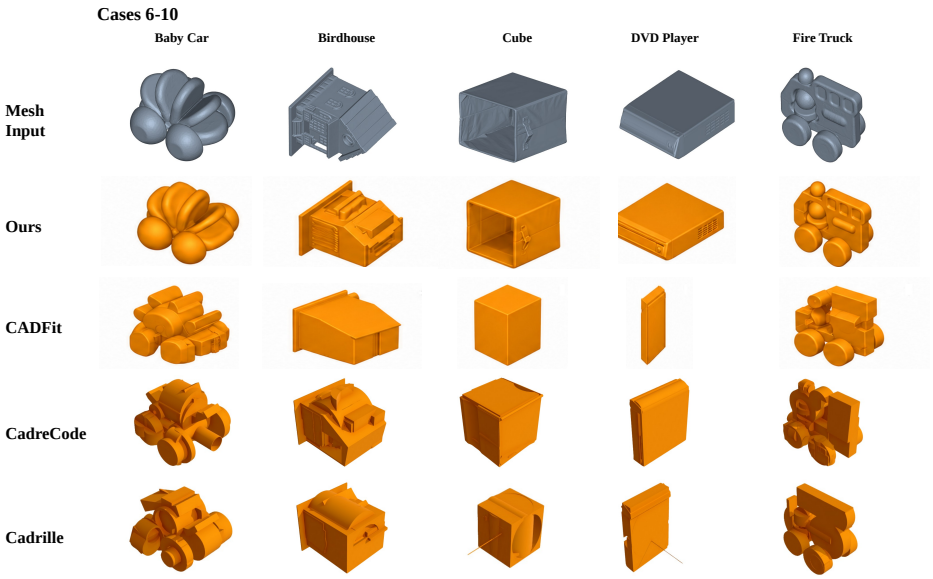


Figure 4: Additional visual comparisons, cases 6–10. The printed *CadreCode* label denotes the CAD-Recode baseline; these cases are not used as additional numerical measurements.

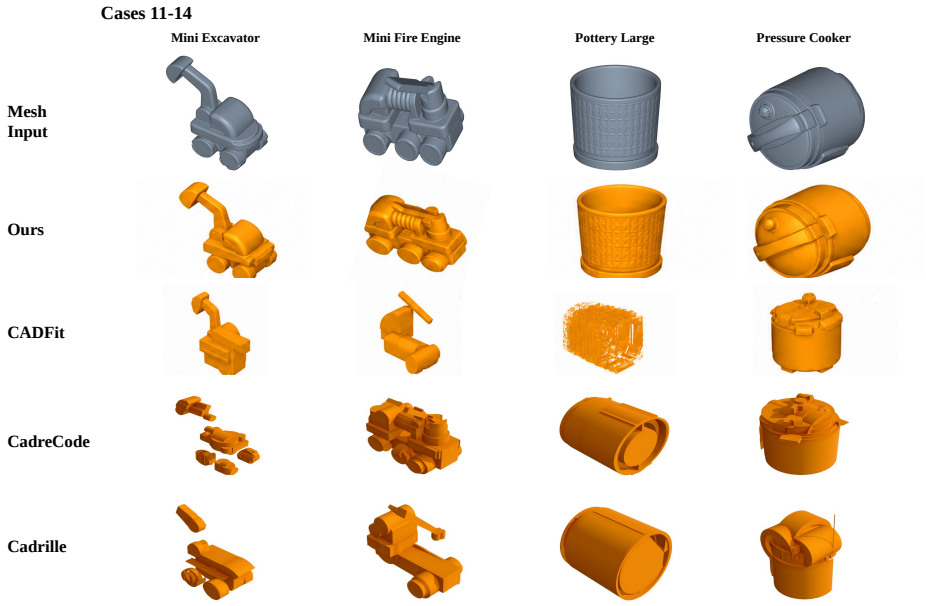


Figure 5: Additional visual comparisons, cases 11–14. This gallery panel is qualitative and has no reported per-case score.

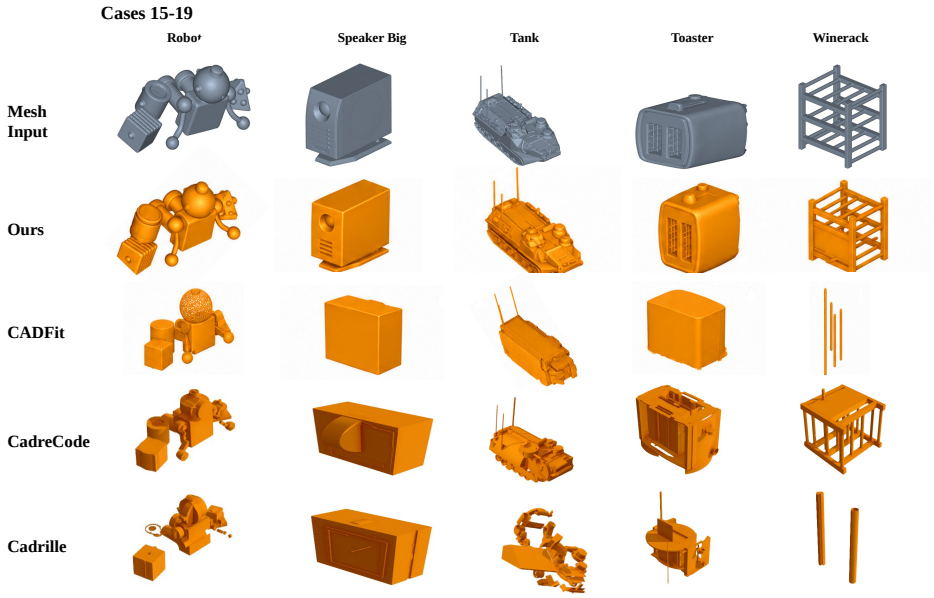


Figure 6: Additional visual comparisons, cases 15–19.