CADFLOW: RECOVERING THE CONSTRUCTION PROCESS FOR SKETCH-TO-CAD GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating accurate and executable parametric CAD models from raster sketch images remains challenging. Existing methods either rely on individually accessible strokes and learn to group them into CAD operations, requiring additional stroke recovery for raster sketch inputs, or directly generate CAD sequences from raster sketch images, where insufficient intermediate-state context increases the risk of erroneous construction steps and invalid geometries. To address this problem, we propose CADFlow, which formulates sketch-to-CAD generation as construction-process recovery. CADFlow employs CAD Construction Process Recovery (CCPR) to recursively recover preceding-state features from the final input state to an empty state. It then decodes adjacent-state transitions in the forward direction into a parametric CAD sequence. To alleviate error accumulation, CADFlow further employs Error-Aware Redrawing (EAR) to detect and locally redraw erroneous CAD operations and optimize sketch parameters under predicted geometric constraints. Experiments on DeepCAD and Fusion 360 demonstrate that CADFlow generates accurate, executable, and geometrically consistent CAD models while exhibiting strong cross-dataset generalization.

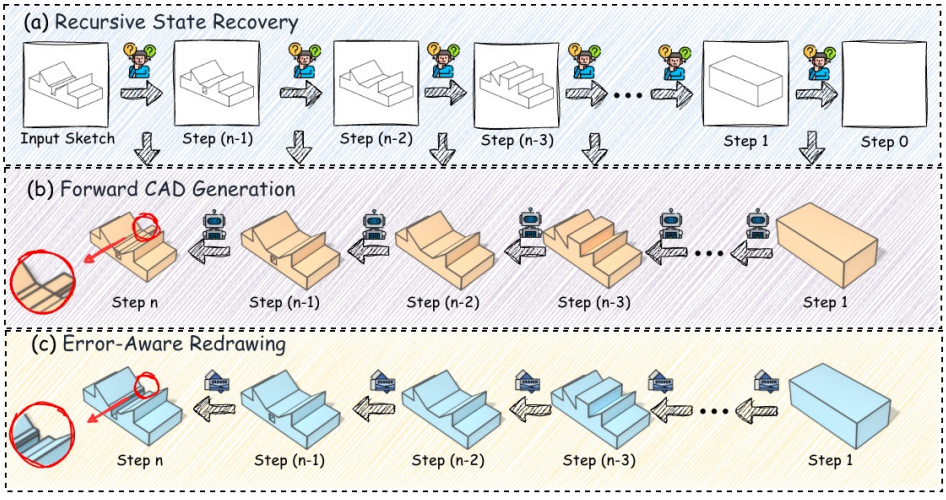

Figure 1: CADFlow reconstructs a CAD model by reasoning about its construction process. Starting from the input sketch, it (a) recovers the features of preceding construction states, (b) decodes the recovered state transitions in the forward direction into a CAD sequence, and (c) detects and locally redraws erroneous CAD operations to produce an accurate and geometrically consistent model.

1 INTRODUCTION

Sketch-to-CAD generation seeks to transform single-view 2D raster sketches depicting 3D designs into parametric 3D models, offering the potential to streamline CAD modeling and content-creation

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.

workflows. However, this inverse problem is fundamentally under-constrained: a raster sketch image of the final design records the 2D geometry accumulated across construction steps, while the intermediate states and operations that produced it remain unobserved. Consequently, faithful reconstruction requires more than matching the final shape; it also demands the recovery of a valid and coherent construction sequence whose operations reproduce the observed geometry. This gap between visible geometric evidence and latent construction history makes parametric CAD reconstruction from raster sketch images particularly challenging.

One line of work, exemplified by Free2CAD (Li et al., 2022), learns to group input strokes according to CAD operations and fits an operation and its parameters to each predicted group. Geometric fitting provides context for subsequent grouping and operation reconstruction, rather than requiring users to specify operation-level groups in advance. However, Free2CAD takes an ordered set of individually accessible strokes as input. When only a flattened raster sketch image is available, stroke identities and drawing order are not provided, so applying such a pipeline requires an additional step to recover stroke-level input information.

Alternatively, recent methods such as GenCAD (Alam & Ahmed, 2025) take the raster sketch image as input and directly generate a CAD sequence with diffusion models. They avoid explicit entity grouping, but treat all sketch geometry as a single conditioning signal. The model is therefore not told which evidence belongs to the current operation or how previous operations constrain the next one. This missing construction-step prior makes it difficult to infer a faithful sequence of CAD operations from composite sketches.

CAD modeling is inherently sequential. Each operation is performed in the context of a preceding construction state, and successive operations form a complete construction process from an empty state to the final design. However, the input sketch depicts only the final state of this process, leaving the intermediate construction states unobserved. As illustrated in Figure 1, we start from the final design depicted in the input sketch and reason backward to recover progressively simpler preceding construction states. The recovered intermediate states expose the construction context of each CAD operation, providing guidance for subsequent forward CAD generation and targeted error correction. Based on this insight, we formulate sketch-to-CAD generation as construction-process recovery and propose CADFlow.

Starting from the input sketch, CADFlow uses CAD Construction Process Recovery (CCPR) to recursively recover the features of preceding construction states, producing a feature trajectory from the final input state to an empty canvas. It then decodes the changes between adjacent states in the forward direction into a parametric CAD sequence. This formulation infers CAD operations directly from a single raster sketch image, without requiring individually accessible strokes or an explicit stroke-grouping stage. To reduce errors accumulated during sequential generation, CADFlow further introduces Error-Aware Redrawing (EAR). It first identifies inaccurate CAD operations and locally redraws only those operations while retaining valid operations. It then predicts geometric relations, such as parallelism and perpendicularity, and optimizes sketch parameters under the predicted constraints. Together, these stages recover a CAD sequence that is faithful to the input sketch and geometrically consistent. Our contributions are as follows:

- We introduce CCPR, which recursively recovers the features of preceding construction states from a single raster sketch image and decodes their forward transitions into a parametric CAD sequence, without requiring stroke-level input at inference time.
- We propose EAR, which detects and locally redraws erroneous CAD operations and refines sketch parameters under predicted geometric constraints, improving reconstruction accuracy and geometric consistency.
- By integrating CCPR and EAR, we propose CADFlow, a unified sketch-to-CAD framework that generates accurate, executable, and geometrically consistent parametric CAD models while exhibiting effective zero-shot cross-dataset transfer from DeepCAD to Fusion 360 and robustness to hand-drawn inputs.

2 Related Work

Parametric CAD Generation. Parametric CAD generation has been approached through several modeling paradigms. Autoregressive methods generate CAD commands and parameters token by

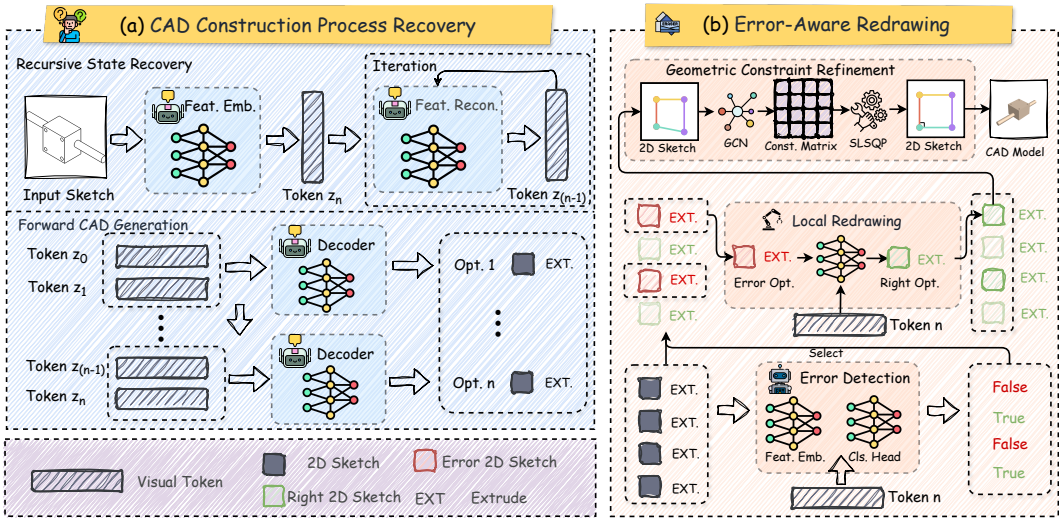

Figure 2: Overview of CADFlow. (a) CCPR recovers preceding-state features and decodes them into an initial CAD sequence. (b) EAR locally redraws erroneous CAD operations and refines sketch parameters under geometric constraints.

token, learning the sequential structure of construction programs (Wu et al., 2021; Xu et al., 2022; Dupont et al., 2024). Diffusion-based methods instead synthesize CAD models through iterative denoising, operating on raster sketches, vector-quantized latents, or structured B-rep representations (Li et al., 2025; Wang et al., 2024; Zhang et al., 2025a; Xu et al., 2024b; Lee et al., 2025; Guo et al., 2025). More recently, large language and multimodal models have been used to generate CAD programs from text or other multimodal conditions (Khan et al., 2024; Zhang et al., 2025b; Guan et al., 2025; Xu et al., 2024a; Pyatov et al., 2026; Li et al., 2026a). A complementary line incorporates planning, program editing, or iterative optimization to improve generated CAD models (Kolodiazhnyi et al., 2025; Kabisov et al., 2026; Yuan et al., 2026). While these methods advance the generation of complete CAD outputs, they generally do not explicitly recover the intermediate construction states underlying the final design depicted in a raster sketch image.

Sketch-conditioned Parametric CAD Reconstruction. Prior work reconstructs CAD models from freehand strokes or raster CAD images. Stroke-based methods such as Free2CAD (Li et al., 2022) group ordered strokes into CAD operations and infer their commands and parameters, but require additional stroke recovery for flattened raster inputs. Raster-based methods directly generate CAD programs using autoregressive, multimodal, or vision-language models, sometimes with spatial or geometric cues such as sketch planes, depth, and normals (Wang et al., 2025b;a; Wu et al., 2024; Li et al., 2025; You et al., 2026). These methods generate final CAD models directly from input sketches without explicitly recovering intermediate construction states, making the generated results prone to structural errors, invalid geometries, or missing local details. In contrast, CADFlow recovers the construction process to provide context for stepwise generation, thereby improving the accuracy, executability, and geometric consistency of the generated parametric CAD models.

3 METHOD

Overview. Given a raster sketch I, our goal is to recover its underlying construction process and generate the parametric CAD sequence. As illustrated in Figure 2, CADFlow consists of two stages. First, CAD Construction Process Recovery (CCPR) starts from the final construction state represented by the input image and recursively recovers the features of preceding construction states until reaching the empty state. It then decodes the transitions between adjacent states in the forward direction into CAD operations, each consisting of a 2D sketch followed by an extrusion command, to generate an initial parametric CAD sequence. Second, Error-Aware Redrawing (EAR) detects erroneous CAD operations in the initial sequence and regenerates only the affected operations through local redrawing while preserving the remaining valid operations. EAR further predicts parallel and perpendicular relations between line segments in the generated 2D sketches and optimizes their endpoint parameters under the predicted constraints, producing the final CAD sequence.

3.1 CAD Construction Process Recovery

CCPR first recovers preceding construction states in the feature space and then decodes the recovered trajectory in the forward direction into a parametric CAD sequence. We represent the CAD sequence as an ordered sequence of sketch–extrude operations, each consisting of a 2D sketch followed by an extrusion command.

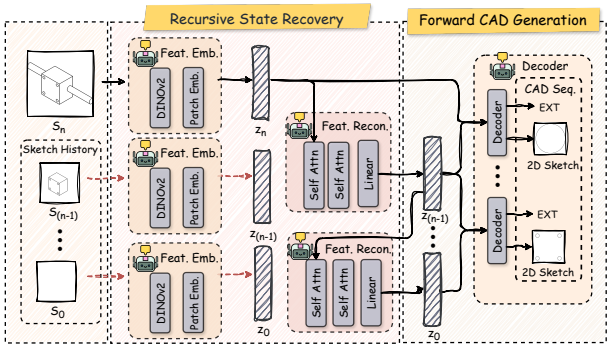

Figure 3: Architecture of CCPR. Red dashed arrows indicate training-only supervision.

CAD Representation. We represent a parametric CAD model as a CAD sequence of operations arranged in execution order. Each CAD operation contains one 2D sketch and one extrusion command. The sketch comprises geometric primitives such as `Line`, `Arc`, and `Circle` and their parameters. The extrusion command specifies the sketch placement, extrusion distances, and Boolean operation. Executing these CAD operations in order forms the complete CAD sequence. See Appendix A.1 for details.

CAD States and State Features. We conceptualize the CAD construction process as a sequence of intermediate CAD states $\mathcal{S} = \{S_0, S_1, \ldots, S_N\}$. Each S_i is the rendered sketch image of the accumulated geometry after the first i CAD operations, where S_0 is an empty canvas and $S_N = I$ is the final input image. Here, S_i is a raster rendering of the accumulated 3D geometry, rather than the parametric 2D sketch within a CAD operation. Each transition $S_{i-1} \rightarrow S_i$ corresponds to one CAD operation: a 2D sketch is first created and then extruded to produce the next state. Rather than directly reconstructing the intermediate state images, CCPR performs recovery in the feature space. As shown in Figure 3, a feature encoder consisting of a pretrained DINOv2 (Oquab et al., 2024) and patch embeddings maps each state image S_i to a state token z_i.

Recursive State Recovery. CCPR starts from the final-state token z_N and recursively predicts the tokens of preceding construction states. At each recovery step, the feature predicted in the previous recovery step serves as the current-state input, with z_N used as the input to the first step. This feature provides the query, key, and value for the first self-attention layer, followed by a second self-attention layer and a linear projection that predict the token of the preceding state. Repeated recovery steps produce a latent trajectory of state features from the final input state to the empty state, rather than directly reconstructing intermediate sketch images. Since S_0 is the blank canvas, its token $z_0 = E(S_0)$ serves as the empty-state reference. Recovery terminates when the cosine similarity between the predicted state feature and z_0 reaches a threshold τ_{stop}, thereby determining the number of CAD operations in the recovered sequence. During training, each recovery step takes the state feature predicted at the previous step as input and is supervised by the feature extracted from the corresponding ground-truth state, as indicated by the red dashed arrows in Figure 3.

Forward CAD Generation. After obtaining the complete feature trajectory, we generate the CAD sequence in the forward direction from the empty state. At each step, the CAD decoder conditions on the features of two adjacent states and autoregressively generates the CAD operation that transforms the former state into the latter. Specifically, the decoder generates the geometric commands and parameters of the 2D sketch, followed by the corresponding extrusion command and its parameters. Once the extrusion command is generated, the decoder advances to the next pair of adjacent states. Applying the decoder sequentially along the recovered trajectory yields the complete parametric CAD sequence. Further details of the CAD sequence representation and autoregressive decoding process are provided in Appendix A.1.

3.2 Error-Aware Redrawing

Although CCPR can reconstruct a complete CAD sequence from an input raster sketch, its autoregressive generation process may cause prediction errors to accumulate and propagate across CAD

operations. Moreover, visual information from an input raster sketch alone may be insufficient to accurately recover design intentions such as parallelism and perpendicularity between geometric primitives. To address these issues, we introduce Error-Aware Redrawing (EAR). As illustrated in Figure 2(b), EAR consists of three stages: CAD operation error detection, local redrawing, and geometric constraint refinement. These stages progressively correct operation errors and geometric parameter deviations in the generated sequence.

CAD Operation Error Detection. Given the initial CAD sequence generated by CCPR, we first identify CAD operations that may contain prediction errors. The complete CAD sequence and its corresponding recovered state tokens are encoded together with the input-sketch token z_N to obtain a sketch-conditioned global sequence representation. The token associated with each CAD operation is then fused with the global representation through cross-attention (Vaswani et al., 2017), allowing the detector to capture both the overall construction context and operation-specific information. The fused features are processed by self-attention (Vaswani et al., 2017) layers and a binary classification head to predict an error probability for each CAD operation. Operations with probabilities above τ_{err} are selected for redrawing, while the others are preserved.

Local Redrawing. For each CAD operation identified as erroneous, we regenerate only the affected operation rather than the complete CAD sequence. For an erroneous operation at index i, the model conditions on the adjacent state tokens z_{i-1} and z_i, the input-sketch token z_N, and the context provided by the complete CAD sequence. These features are fused through cross-attention (Vaswani et al., 2017), enabling the model to determine the CAD operation corresponding to the state transition $S_{i-1} \rightarrow S_i$. A decoder then regenerates the geometric commands and parameters of the 2D sketch, together with the corresponding extrusion command and parameters. The regenerated operation replaces the original prediction, while unflagged operations remain unchanged.

Geometric Constraint Refinement. Although local redrawing can correct erroneous CAD operations, the generated 2D sketches in the CAD representation may still contain small coordinate deviations. For example, contours in the input raster sketch may visually resemble rectangles, slots, or orthogonal connections, whereas the corresponding line segments in the generated 2D sketch do not strictly satisfy parallel or perpendicular relations. Such deviations may have little effect on visual correspondence with the input image, but cause the generated result to depart from the regular geometric structures commonly used in CAD modeling. Parallel and perpendicular relations are among the most common geometric relations in CAD sketches and are widely used to describe rectangular profiles, slots, holes, and orthogonal structures. We therefore predict parallel, perpendicular, and unconstrained relations between line segments, and use them to optimize the line-segment parameters in the generated 2D sketches while preserving consistency with the input raster sketch.

For each reconstructed 2D sketch, we construct a graph in which each line segment is represented as a node defined by the 2D coordinates of its two endpoints. Every pair of line-segment nodes is connected to form a candidate relation. The GCN aggregates the geometric attributes of line segments and their structural context to classify each candidate relation as parallel, perpendicular, or unconstrained. The predicted relations are organized into a constraint matrix for subsequent parameter optimization. Specifically, we optimize the 2D coordinates of all line-segment endpoints, with endpoints shared by connected line segments represented by the same optimization variable. Let $\mathbf{p}_l^s$ and $\mathbf{p}_l^e$ denote the optimized start and end points of line segment l, and let $\hat{\mathbf{p}}_l^s$ and $\hat{\mathbf{p}}_l^e$ denote their initial predictions. With $\mathbf{d}_l = \mathbf{p}_l^e - \mathbf{p}_l^s$, we use SLSQP to minimize $\sum_l \left(\|\mathbf{p}_l^s - \hat{\mathbf{p}}_l^s\|_2^2 + \|\mathbf{p}_l^e - \hat{\mathbf{p}}_l^e\|_2^2\right)$, subject to $\mathbf{d}_a \times \mathbf{d}_b = 0$ for predicted parallel pairs and $\mathbf{d}_a^\top \mathbf{d}_b = 0$ for predicted perpendicular pairs, where $\times$ denotes the 2D cross product. This optimization adjusts line-segment endpoints only as needed to satisfy the predicted relations while remaining close to the reconstructed sketch. Because shared endpoints move jointly, the original endpoint connectivity and closed contours are preserved throughout optimization. The optimized endpoint coordinates are written back to the corresponding line-segment parameters and CAD operations, producing the final CAD sequence.

3.3 Dataset Construction

CCPR requires complete construction trajectories containing both parametric CAD operations and intermediate geometric states. We therefore construct the training data from DeepCAD and retain its original training split. For each CAD program, we partition the construction process at extrusion operations. The first extrusion and all preceding operations define the initial state, while each

subsequent extrusion produces a new state containing all operations up to that point. This yields an ordered construction trajectory from the initial structure to the final model. As illustrated in Figure 8, trajectory lengths range from one to ten states, depending on the number of extrusion operations. To ensure geometric consistency, we compute a single normalization transform from the final model and apply it to all intermediate states. We then sample eight viewpoints for each model and sequentially render its construction states. In addition, we derive parallel, perpendicular, and unconstrained relations between line segments from the underlying CAD geometry to supervise the GCN used for geometric constraint prediction. All images are rendered using `pythonocc-core`.

3.4 Loss Function

We jointly train CADFlow with five standard objectives. For CCPR, the feature reconstruction loss $\mathcal{L}_{\text{feat}}$ measures the ℓ_2 distance between the predicted and ground-truth features of the preceding state, while the sequence loss $\mathcal{L}_{\text{seq}}$ applies cross-entropy supervision to the CAD tokens conditioned on the recovered state-feature trajectory. For EAR, the error detection loss $\mathcal{L}_{\text{err}}$ uses binary cross-entropy to supervise the CAD-operation error mask. The local redrawing loss $\mathcal{L}_{\text{local}}$ applies CAD-token cross-entropy to erroneous CAD operations conditioned on the initial CAD sequence and the adjacent state tokens z_{i-1} and z_i. Finally, the relation loss $\mathcal{L}_{\text{rel}}$ uses multi-class cross-entropy to supervise the prediction of parallel, perpendicular, and unconstrained relations. The overall objective is

$$\mathcal{L} = \lambda_{\text{feat}}\mathcal{L}_{\text{feat}} + \lambda_{\text{seq}}\mathcal{L}_{\text{seq}} + \lambda_{\text{err}}\mathcal{L}_{\text{err}} + \lambda_{\text{local}}\mathcal{L}_{\text{local}} + \lambda_{\text{rel}}\mathcal{L}_{\text{rel}},$$

where the λ coefficients balance the five objectives.

4 Experiments

4.1 Experimental Setup

Implementation Details. We implement CADFlow in PyTorch and train it with distributed data parallelism on four NVIDIA RTX 4090 GPUs. The input sketches are resized to 224×224. We use a frozen DINOv2 (Oquab et al., 2024) ViT-S/14 encoder to extract visual features, and set the state feature dimension to 192. All components of CADFlow are jointly optimized: CCPR is supervised by feature reconstruction and CAD sequence prediction, while EAR is supervised by CAD operation error detection, local redrawing, and geometric relation prediction. In the joint objective, we set $\lambda_{\text{feat}} = 5.0$, $\lambda_{\text{seq}} = 1.0$, $\lambda_{\text{err}} = 1.0$, $\lambda_{\text{local}} = 1.0$, and $\lambda_{\text{rel}} = 2.0$. We train the model for 100 epochs with a batch size of 32 and an initial learning rate of 1×10^{-4}. We use fixed thresholds of $\tau_{\text{stop}} = 0.95$ and $\tau_{\text{err}} = 0.95$ across all experiments and datasets.

Datasets. We use DeepCAD (Wu et al., 2021) for training and evaluate the models on the held-out DeepCAD test set as well as the Fusion 360 dataset. All methods are trained exclusively on the DeepCAD training split. For cross-dataset evaluation, we directly transfer the trained models to Fusion 360 (Willis et al., 2021).

Evaluation Metrics. For methods that generate CAD sequences, we report command accuracy (ACC_{cmd}) and parameter accuracy ($\text{ACC}_{\text{param}}$). We additionally measure the Invalid Ratio (IR), defined as the percentage of generated CAD sequences that fail to produce valid geometries, and the Chamfer distance (CD) between the generated and ground-truth geometries. Higher ACC_{cmd} and $\text{ACC}_{\text{param}}$ indicate better performance, whereas lower IR and CD are preferred.

4.2 Comparisons

We compare CADFlow with representative CAD generation methods on DeepCAD and Fusion 360. Free2CAD (Li et al., 2022), GenCAD (Alam & Ahmed, 2025), Img2CAD (You et al., 2025), and Drawing2CAD (Qin et al., 2025) generate CAD sequences; Img2CADSeq (Tan et al., 2026), CADrawer (Li et al., 2026b), BrepDiff (Lee et al., 2025), and HiFi-BRep (Hou et al., 2026) generate B-rep models; and CADRILLE (Kolodiazhnyi et al., 2025) and CADReasoner (Kabisov et al., 2026) generate CADQuery programs. Further details on data preparation, method adaptations, and training settings are provided in Appendix A.2.

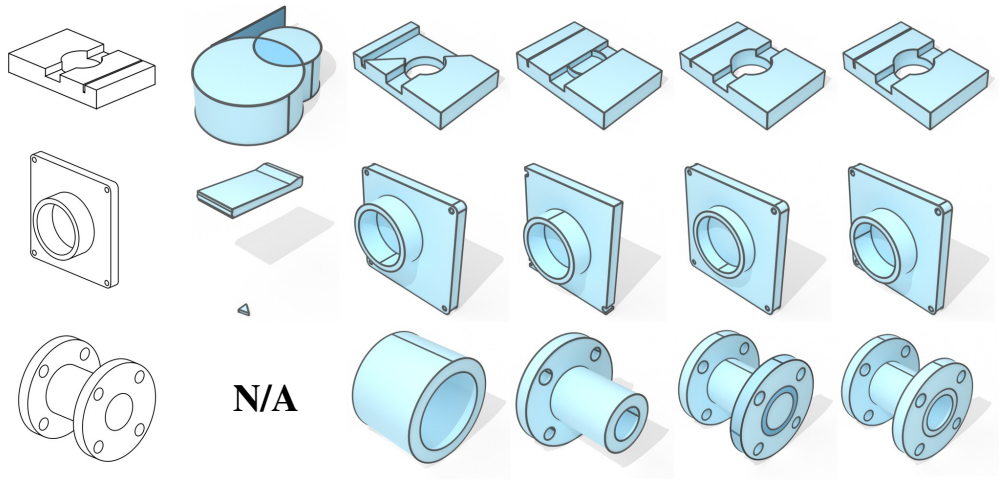

Figure 4: Visual comparison of sketch-to-CAD reconstructions on the DeepCAD test set.

Table 1: CAD reconstruction accuracy, geometric quality, and average per-sample inference time on the DeepCAD test set across different output representations. For each metric, the best and second-best results are highlighted in green and yellow, respectively.

Category	Method	ACC_{cmd} (%)↑	$\text{ACC}_{\text{param}}$ (%)↑	IR (%)↓	CD ($\times 10^2$)↓	Avg. Time (s)↓
CADQuery	CADRILLE	–	–	36.1	19.4	5.966
	CADReasoner	–	–	31.3	17.2	2.350
B-rep	Img2CADSeq	–	–	19.6	17.5	0.126
	BrepDiff	–	–	20.7	17.6	0.690
	CADrawer	–	–	22.9	20.1	0.158
	HiFi-BRep	–	–	31.2	22.8	1.861
CAD Sequence	Free2CAD	65.1	69.1	29.2	16.1	**0.110**
	GenCAD	54.6	52.2	15.5	33.8	0.175
	Img2CAD	72.8	71.0	7.9	18.9	4.800
	Drawing2CAD	80.1	77.7	21.4	16.9	0.156
	Ours	**82.3**	**82.4**	**3.9**	**13.7**	0.173

DeepCAD Dataset. As shown in Table 1, CADFlow achieves the best results across all reconstruction metrics, with a parameter accuracy of 82.4% and an IR of 3.9%, demonstrating its advantages in parameter prediction and geometric validity. The visual comparisons in Figure 4 are consistent with these quantitative results: GenCAD (Alam & Ahmed, 2025) produces extraneous structures or incorrect topology, while Drawing2CAD (Qin et al., 2025) may miss holes, grooves, or local cuts. In contrast, CADFlow preserves local details and their connections to the body. These results support the effectiveness of construction-process modeling. By decomposing reconstruction into successive state transitions, CADFlow uses the context provided by adjacent intermediate states to infer local operations, helping reduce ambiguity caused by overlapping geometry in the input raster sketch image.

Fusion 360 Dataset. When trained exclusively on DeepCAD and directly transferred to Fusion 360, CADFlow outperforms the compared methods on all metrics, achieving command and parameter accuracies of 75.6% and 77.8%, respectively, with an IR of 9.7% (Table 2). CADFlow also consistently outperforms Img2CAD (You et al., 2025). The visual comparisons in Figure 5 show that some baseline predictions exhibit incorrect component compositions or omit holes, steps, and connecting structures, whereas CADFlow better preserves both the overall shape and local details. Together, these quantitative and qualitative results demonstrate the cross-dataset reconstruction capability of CADFlow. This is consistent with its construction-process modeling, which focuses on local geometric changes and their corresponding operations, allowing the model to leverage local construction patterns when the overall geometric composition changes.

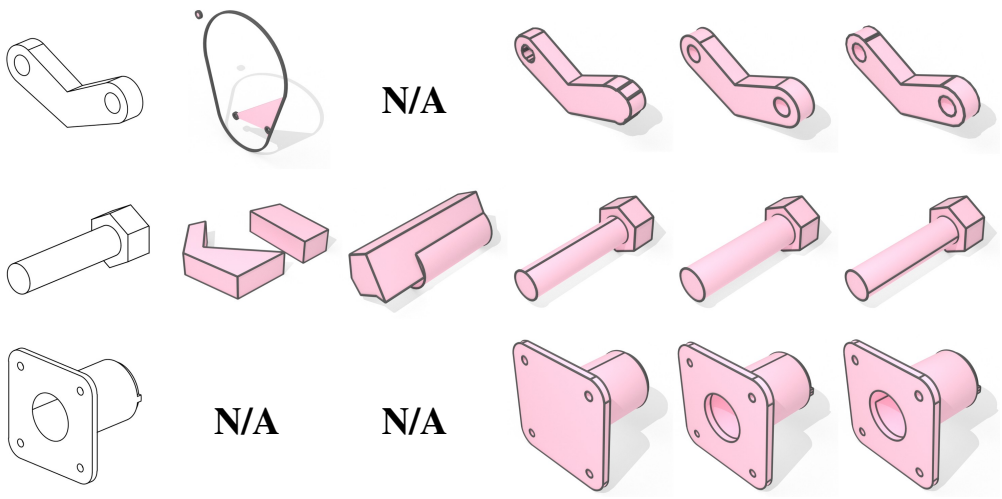

Figure 5: Cross-dataset reconstruction examples on Fusion 360 using models trained on DeepCAD.

Table 2: Cross-dataset CAD reconstruction performance on Fusion 360.

Category	Method	ACC_{cmd} (%)↑	$\text{ACC}_{\text{param}}$ (%)↑	IR (%)↓	CD ($\times 10^2$)↓
CADQuery	CADRILLE	–	–	43.5	24.7
	CADReasoner	–	–	28.5	23.2
B-rep	Img2CADSeq	–	–	25.5	33.2
	BrepDiff	–	–	42.4	29.5
	CADrawer	–	–	40.7	49.5
	HiFi-BRep	–	–	41.7	27.8
CAD Sequence	Free2CAD	49.9	51.4	42.2	43.6
	GenCAD	35.4	45.1	17.4	43.3
	Img2CAD	68.6	67.3	16.9	61.9
	Drawing2CAD	62.3	66.2	36.1	35.8
	Ours	**75.6**	**77.8**	**9.7**	**21.1**

4.3 Ablation Studies and Discussion

Component-wise Analysis. As shown in Table 3, CCPR reduces IR from 23.21% to 4.81% compared with direct generation, demonstrating the benefit of construction-process recovery for geometric validity. It also achieves lower IR and CD than both forward recovery with intermediate-state supervision and backward recovery without it, supporting the combined use of backward recovery and state supervision. Step-wise recovery results across construction lengths are reported in Appendix A.3.1 and A.3.2. Applying local redrawing once to every CAD operation without error detection improves parameter accuracy and CD but reduces command accuracy, indicating that indiscriminate redrawing may alter originally correct operations. Adding error detection enables selective redrawing, recovering command accuracy while further improving geometric validity and quality. Geometric constraint refinement provides additional gains in parameter accuracy and geometric quality, highlighting the complementary roles of these modules.

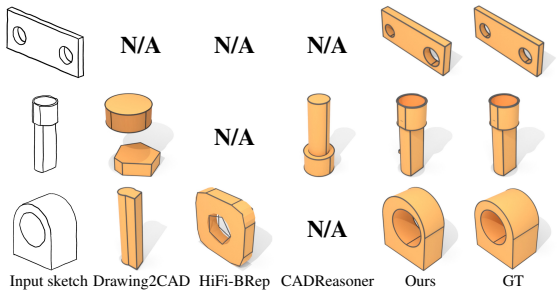

Figure 6: Visual comparison on hand-drawn sketches.

Hand-drawn Sketch Evaluation. To evaluate the applicability of CADFlow to hand-drawn sketches, we selected 200 3D CAD models from the DeepCAD test set and invited five design-

Table 3: Component ablation on DeepCAD. ED, LR, and GCR denote error detection, local redrawing, and geometric constraint refinement, respectively. "All-operation LR" denotes locally redrawing all CAD operations without error detection. Best results are shown in bold.

Variant	CCPR	ED	LR	GCR	$\mathrm{ACC}_{\mathrm{cmd}}$ (%)↑	$\mathrm{ACC}_{\mathrm{param}}$ (%)↑	IR (%)↓	CD ($\times 10^2$)↓
Direct generation					80.2	76.6	23.21	23.1
Forward + state sup.					80.7	80.4	21.0	20.8
Backward w/o state sup.	✓				80.9	80.5	19.5	19.1
CCPR	✓				81.3	81.7	4.81	18.7
CCPR + All-operation LR	✓		✓		78.6	82.0	4.53	15.1
CCPR + ED+LR	✓	✓	✓		**82.3**	82.3	3.96	14.3
CADFlow (full)	✓	✓	✓	✓	**82.3**	**82.4**	**3.88**	**13.7**

Table 4: CAD reconstruction performance on the hand-drawn sketch dataset.

Method	$\mathrm{ACC}_{\mathrm{cmd}}$ (%)↑	$\mathrm{ACC}_{\mathrm{param}}$ (%)↑	IR (%)↓	CD ($\times 10^2$)↓
Drawing2CAD	44.2	40.9	45.5	53.4
HiFi-BRep	–	–	17.6	28.2
CADReasoner	–	–	39.1	29.8
Ours	**66.0**	**76.3**	**5.9**	**16.2**

ers with CAD modeling experience to create freehand sketches of the corresponding models using Procreate on an iPad. Figure 6 presents qualitative comparisons. As shown in Table 4, CADFlow achieves good reconstruction accuracy and geometric validity on these hand-drawn inputs. It exceeds Drawing2CAD in parameter accuracy by 35.4 percentage points and achieves an IR of 5.9%, compared with 17.6% for HiFi-BRep. The visual results show that CADFlow preserves holes and connected component structures despite irregular hand-drawn contours, while some baselines produce invalid, distorted, or disconnected geometry.

Failure Case Analysis and Limitations. As shown in Figure 7, CADFlow recovers the visible overall shape but misses local details on the back side, including the circular feature surrounding the central hole of the disk and the rectangular cutouts on the cylinder. These features lack visible evidence in the input sketches, making it difficult to determine their geometry and corresponding construction operations. Although construction-process recovery and error correction improve reconstruction, they cannot guarantee accurate recovery of unobserved details. Consequently, a generated model consistent with the input view may still differ from the ground truth in its back-side geometry.

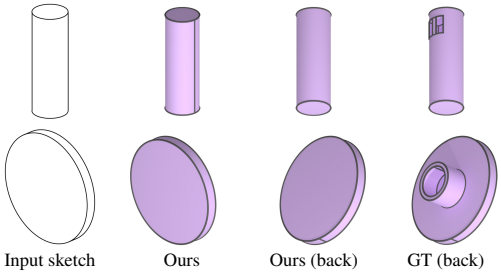

Figure 7: Failure cases of CADFlow.

5 Conclusion

We presented CADFlow, a sketch-to-CAD framework that recovers construction processes to generate accurate and executable parametric CAD models. By explicitly modeling intermediate construction states, CADFlow provides operation-level context for interpreting composite sketches and predicting CAD sequences. Experiments on DeepCAD and Fusion 360 demonstrate improvements over the compared methods in command and parameter accuracy, geometric validity, and reconstruction similarity. These results support construction-process recovery as an effective approach to reducing structural ambiguity in sketch-to-CAD generation. Future work will explore combining sketches with textual descriptions to better express design intent and guide construction-process recovery.

AI USE STATEMENT

In this work, ChatGPT was used solely after the initial draft had been completed to improve the grammar, wording, translation, and overall flow of the manuscript. All AI-assisted revisions were carefully reviewed and verified by the authors. ChatGPT was not used to generate any algorithms, proofs, experimental results, or technical contributions, and no proprietary or sensitive data were shared with the tool. All technical content, analyses, and conclusions are entirely the authors' own work, and the authors take full responsibility for the complete content of the manuscript.

REFERENCES

Md Ferdous Alam and Faez Ahmed. Gencad: Image-conditioned computer-aided design generation with transformer-based contrastive representation and diffusion priors. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=e817c1wEZ6.

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 Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXI*, volume 15119 of *Lecture Notes in Computer Science*, pp. 19–36. Springer, 2024. doi: 10.1007/978-3-031-73030-6_2. URL https://doi.org/10.1007/978-3-031-73030-6_2.

Yandong Guan, Xilin Wang, Ximing Xing, Jing Zhang, Dong Xu, and Qian Yu. Cad-coder: Text-to-cad generation with chain-of-thought and geometric reward. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/564a224e88f2490f3c1deaae877d37e3-Abstract-Conference.html.

Hao Guo, Xiaoshui Huang, Jiacheng Hao, Yunpeng Bai, Hongping Gan, and Yilei Shi. Brepgiff: Lightweight generation of complex b-rep with 3d GAT diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 26587–26596. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.02476. URL https://openaccess.thecvf.com/content/CVPR2025/html/Guo_BrepGiff_Lightweight_Generation_of_Complex_B-rep_with_3D_GAT_Diffusion_CVPR_2025_paper.html.

Junhao Hou, Chenqi Luo, Pufan Wang, Jiaying Lu, Yusheng Liu, Feiwei Qin, Meie Fang, and Kun Zhou. Hifi-brep: High-fidelity latent representation for robust b-rep generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27199–27208, June 2026.

Soslan Kabisov, Vsevolod Kirichuk, Andrey Volkov, Marina Barannikov, Gennadiy Savrasov, Anton Konushin, Andrey Kuznetsov, and Dmitrii Zhemchuzhnikov. Cadreasoner: Iterative program editing for cad reverse engineering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 6143–6153, June 2026.

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 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/0e5b96f97c1813bb75f6c28532c2ecc7-Abstract-Conference.html.

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. *CoRR*, abs/2505.22914, 2025.

doi: 10.48550/ARXIV.2505.22914. URL https://doi.org/10.48550/arXiv.2505.22914.

Mingi Lee, Dongsu Zhang, Clément Jambon, and Young Min Kim. Brepdiff: Single-stage b-rep diffusion model. In Ginger Alford, Hao (Richard) Zhang, and Adriana Schulz (eds.), *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH Conference Papers 2025, Vancouver, BC, Canada, August 10-14, 2025*, pp. 103:1–103:11. ACM, 2025. doi: 10.1145/3721238.3730698. URL https://doi.org/10.1145/3721238.3730698.

Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Free2cad: parsing freehand drawings into CAD commands. *ACM Trans. Graph.*, 41(4):93:1–93:16, 2022. doi: 10.1145/3528223.3530133. URL https://doi.org/10.1145/3528223.3530133.

Jiahao Li, Qingwang Zhang, Qiuyu Chen, Guozhan Qiu, Yunzhong Lou, and Xiangdong Zhou. Towards high-fidelity CAD generation via LLM-driven program generation and text-based b-rep primitive grounding. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=jfOhTGs5G5.

Pu Li, Wenhao Zhang, Jianwei Guo, Jinglu Chen, and Dong-Ming Yan. Revisiting CAD model generation by learning raster sketch. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 4869–4877. AAAI Press, 2025. doi: 10.1609/AAAI.V39I5.32515. URL https://doi.org/10.1609/aaai.v39i5.32515.

Yuanbo Li, Gilda Manfredi, Henro Kriel, Chengye Hao, Xianghao Xu, Adrien Bousseau, and Daniel Ritchie. Cadrawer: Autoregressive CAD generation from 3d sketches. *Comput. Graph. Forum*, 45(2), 2026b. doi: 10.1111/CGF.70352. URL https://doi.org/10.1111/cgf.70352.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

Vladislav Pyatov, Gleb Bobrovskikh, Saveliy Galochkin, Nikita Boldyrev, Oleg Voynov, Alexander Filippov, Gonzalo Ferrer, Peter Wonka, and Evgeny Burnaev. Cadfs: A big cad program dataset and framework for computer-aided design with large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10176–10186, June 2026.

Feiwei Qin, Shichao Lu, Junhao Hou, Changmiao Wang, Meie Fang, and Ligang Liu. Drawing2cad: Sequence-to-sequence learning for CAD generation from vector drawings. In Cathal Gurrin, Klaus Schoeffmann, Min Zhang, Luca Rossetto, Stevan Rudinac, Duc-Tien Dang-Nguyen, Wen-Huang Cheng, Phoebe Chen, and Jenny Benois-Pineau (eds.), *Proceedings of the 33rd ACM International Conference on Multimedia, MM 2025, Dublin, Ireland, October 27-31, 2025*, pp. 10573–10582. ACM, 2025. doi: 10.1145/3746027.3755782. URL https://doi.org/10.1145/3746027.3755782.

Shiyu Tan, Zixuan Zhao, Hao Gao, Zhiheng Chen, Xiaolong Yin, and Enya Shen. Img2cadseq: Image-to-cad generation via sequence-based diffusion. In Christopher P. Redmann, Justin Solomon, and Mirela Ben-Chen (eds.), *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH Conference Papers 2026, Los Angeles, CA, USA, July 19-23, 2026*, pp. 19:1–19:11. ACM, 2026. doi: 10.1145/3799902.3811174. URL https://doi.org/10.1145/3799902.3811174.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von

Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Hanxiao Wang, Mingyang Zhao, Yiqun Wang, Weize Quan, and Dong-Ming Yan. VQ-CAD: computer-aided design model generation with vector quantized diffusion. *Comput. Aided Geom. Des.*, 111:102327, 2024. doi: 10.1016/J.CAGD.2024.102327. URL https://doi.org/10.1016/j.cagd.2024.102327.

Siyu Wang, Cailian Chen, Xinyi Le, Qimin Xu, Lei Xu, Yanzhou Zhang, and Jie Yang. CAD-GPT: synthesising CAD construction sequence with spatial reasoning-enhanced multimodal llms. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 7880–7888. AAAI Press, 2025a. doi: 10.1609/AAAI.V39I8.32849. URL https://doi.org/10.1609/aaai.v39i8.32849.

Xilin Wang, Jia Zheng, Yuanchao Hu, Hao Zhu, Qian Yu, and Zihan Zhou. From 2d CAD drawings to 3d parametric models: A vision-language approach. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 7961–7969. AAAI Press, 2025b. doi: 10.1609/AAAI.V39I8.32858. URL https://doi.org/10.1609/aaai.v39i8.32858.

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 Trans. Graph.*, 40(4): 54:1–54:24, 2021. doi: 10.1145/3450626.3459818. URL https://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 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 6752–6762. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00670. URL https://doi.org/10.1109/ICCV48922.2021.00670.

Sifan Wu, Amir Hosein Khasahmadi, Mor Katz, Pradeep Kumar Jayaraman, Yewen Pu, Karl D. D. Willis, and Bang Liu. Cadvlm: Bridging language and vision in the generation of parametric CAD sketches. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXX*, volume 15128 of *Lecture Notes in Computer Science*, pp. 368–384. Springer, 2024. doi: 10.1007/978-3-031-72897-6_21. URL https://doi.org/10.1007/978-3-031-72897-6_21.

Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. CAD-MLLM: unifying multimodality-conditioned CAD generation with MLLM. *CoRR*, abs/2411.04954, 2024a. doi: 10.48550/ARXIV.2411.04954. URL https://doi.org/10.48550/arXiv.2411.04954.

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 Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 24698–24724. PMLR, 2022. URL https://proceedings.mlr.press/v162/xu22k.html.

Xiang Xu, Joseph G. Lambourne, Pradeep Kumar Jayaraman, Zhengqing Wang, Karl D. D. Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent

geometry. *ACM Trans. Graph.*, 43(4):119:1–119:14, 2024b. doi: 10.1145/3658129. URL https://doi.org/10.1145/3658129.

Yang You, Mikaela Angelina Uy, Jiaqi Han, Rahul Krishna Thomas, Haotong Zhang, Yi Du, Hansheng Chen, Francis Engelmann, Suya You, and Leonidas J. Guibas. Img2cad: Reverse engineering 3d CAD models from images through vlm-assisted conditional factorization. In Taku Komura, Michael Wimmer, and Hongbo Fu (eds.), *Proceedings of the SIGGRAPH Asia 2025 Conference Papers, SA Conference Papers 2025, Hong Kong, December 15-18, 2025*, pp. 95:1–95:12. ACM, 2025. doi: 10.1145/3757377.3763891. URL https://doi.org/10.1145/3757377.3763891.

Zhenguo You, Dong Du, Jinghao Liu, Lingyan Cheng, Meng Wu, and Weiming Wang. Selfsketch2cad: Self-supervised CAD sequence learning from single-view sketches. *Comput. Aided Geom. Des.*, 127:102562, 2026. doi: 10.1016/J.CAGD.2026.102562. URL https://doi.org/10.1016/j.cagd.2026.102562.

Meng Yuan, Dawei Lin, Hongxia Xie, Tieru Wu, and Rui Ma. Cad-refiner: A unified framework for cad generation and iterative editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3244–3253, 2026.

Aijia Zhang, Weiqiang Jia, Qiang Zou, Yixiong Feng, Xiaoxiang Wei, and Ye Zhang. Diffusioncad: Controllable diffusion model for generating computer-aided design models. *IEEE Trans. Vis. Comput. Graph.*, 31(12):10188–10199, 2025a. doi: 10.1109/TVCG.2025.3535797. URL https://doi.org/10.1109/TVCG.2025.3535797.

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 *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=Z0eiiV3Yyh.

A APPENDIX

In this appendix, we provide additional methodological and experimental details and analyses. In Sec. A.1, we detail the CAD sequence representation and autoregressive decoding process. In Sec. A.2, we describe data preparation (Sec. A.2.1), adaptations of the compared methods (Sec. A.2.2), and training and inference settings (Sec. A.2.3). In Sec. A.3, we present step-wise construction accuracy (Sec. A.3.1), generation results for construction trajectories of different lengths (Sec. A.3.2), and the computational cost and performance trade-offs of the ablation variants (Sec. A.3.3).

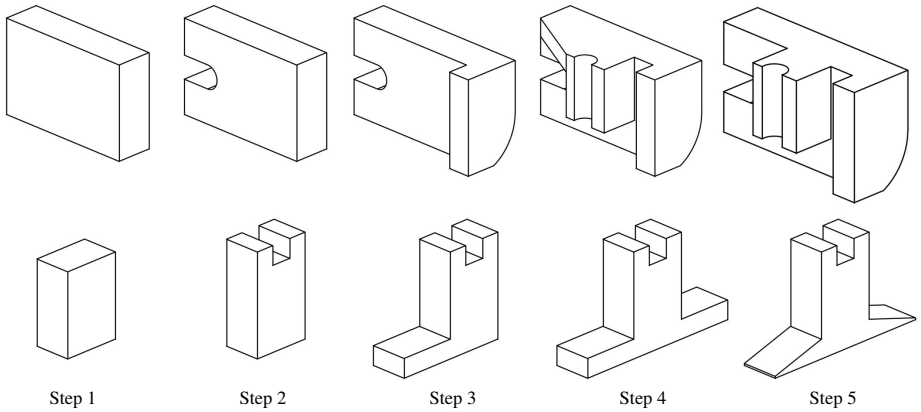

Figure 8: Construction-trajectory data generation.

A.1 CAD REPRESENTATION AND AUTOREGRESSIVE DECODING

CAD Representation. We represent a parametric CAD model as an ordered sequence of CAD operations, each consisting of a 2D sketch followed by an extrusion command. A 2D sketch is composed of geometric commands including `Line`, `Arc`, and `Circle`. `Line` records a line endpoint, `Arc` records an arc endpoint, central angle, and direction, and `Circle` records its center and radius. The extrusion command `Ext` specifies the orientation, position, and scale of the sketch plane, together with the extrusion distances, Boolean operation, and extrusion type.

In the sequence representation, each command consists of a command type and its quantized parameters. Different commands use different subsets of parameters. To support unified prediction, all commands share a fixed-length parameter representation, while parameters irrelevant to the current command are masked during training and decoding. The valid parameters and their meanings are listed below.

Category	Command	Valid parameters	# Params.	Description
Sketch	`Line`	(x, y)	2	2D coordinates of the line endpoint
Sketch	`Arc`	(x, y, α, f)	4	Arc endpoint, central angle, and direction flag
Sketch	`Circle`	(x, y, r)	3	Circle center and radius
Extrusion	`Ext`	$(\theta, \phi, \gamma, p_x, p_y, p_z, s, e_1, e_2, b, u)$	11	Sketch placement, scale, and extrusion attributes

The parameters of the extrusion command `Ext` are defined as follows.

Parameter	Description
(θ, ϕ, γ)	Orientation of the sketch plane
(p_x, p_y, p_z)	Origin coordinates of the sketch plane
s	Sketch scale
(e_1, e_2)	Extrusion distances along the plane normal and its opposite direction
b	Boolean operation: new body, join, cut, or intersect
u	Extrusion type: one-sided, symmetric, or two-sided

Each CAD operation ends with an `Ext` command, so no additional sketch-start or operation-end command is required. Once `Ext` is generated, the current CAD operation is complete and the decoder proceeds to the next operation. The number of operations in the complete CAD sequence is determined by the recovered state-feature trajectory, eliminating the need for an explicit sequence-end command.

Autoregressive Decoding. The CAD decoder generates a CAD sequence in the forward direction along the recovered state-feature trajectory. Each pair of adjacent state features corresponds to one CAD operation, which consists of a set of 2D sketch commands followed by an extrusion command. Each sketch command draws a single geometric primitive. For example, a square is typically represented by four `Line` commands, with each command recording the endpoint parameters of one edge. Circles and arcs are represented by `Circle` and `Arc` commands, respectively. After the 2D sketch is completed, the `Ext` command specifies the sketch plane, extrusion distances, and Boolean operation required to produce the corresponding 3D geometry.

At each decoding step, the model predicts the next command and its parameters conditioned on the current pair of adjacent state features and the commands and parameters generated so far. The adjacent state features describe the geometric change associated with the current CAD operation, while the previously generated commands provide the context of the partially constructed sketch. The predicted command and parameters are then used as input to the next decoding step. Because different commands require different parameters, a command-dependent mask retains only the parameters valid for the predicted command; the training loss is likewise computed only over these valid parameters.

The decoder first generates a sequence of `Line`, `Arc`, and `Circle` commands that form the 2D sketch. Generating an `Ext` command completes the current CAD operation, after which the decoder proceeds to the next pair of adjacent state features and generates the next operation. Repeating this process along the entire recovered trajectory produces the complete parametric CAD sequence.

A.2 Supplementary Experimental Details

A.2.1 Data Preparation

Building on the data construction procedure described in Sec. 3.3, we additionally prepare input and supervision formats required by the compared methods. We retain stroke-layer information when constructing the sketch data, with each layer containing a single stroke. This provides individually accessible strokes for stroke-based methods, whereas CADFlow uses raster sketch images as input. For methods that generate CADQuery programs, we convert the existing DeepCAD (Wu et al., 2021) CAD sequences into CADQuery programs using the CADQuery library, obtaining the corresponding program supervision from the same source data. We check the geometric consistency between the models produced by the converted CADQuery programs and the original DeepCAD models.

For methods that generate B-rep models, we construct ground-truth supervision from the same source CAD models. Specifically, we generate face, edge, and topology data according to the representation requirements of each compared method. Thus, the CAD sequences, B-rep training data, and CADQuery programs correspond to the same source CAD models while providing the supervision required by each output representation.

For the hand-drawn sketch evaluation, we directly use the collected sketches as model inputs without any preprocessing, such as denoising, stroke extraction, vectorization, or geometric correction.

A.2.2 Compared Methods and Adaptations

To ensure a fair comparison, we retrain all compared methods on the same DeepCAD training split and evaluate them on the DeepCAD test set and Fusion 360, without using any Fusion 360 data for training. For GenCAD (Alam & Ahmed, 2025), Img2CAD (You et al., 2025), and Img2CADSeq (Tan et al., 2026), which natively support raster-image inputs, we retain their original input settings. For CADrawer (Li et al., 2026b), BrepDiff (Lee et al., 2025), HiFi-BRep (Hou et al., 2026), and Drawing2CAD (Qin et al., 2025), which do not natively support raster-image inputs, we replace only their input modules with the same DINOv2 (Oquab et al., 2024) image encoder and

feed the extracted raster-sketch features into their original downstream networks. Apart from this input adaptation, their output representations and generation pipelines remain unchanged.

Free2CAD (Li et al., 2022) is treated as a special case. It requires an ordered set of individually accessible strokes and therefore cannot be directly applied to flattened raster sketches, in which stroke identities and drawing order are unavailable. We retain its original input format and provide the stroke-layer information preserved during data construction, as described in Sec. A.2.1.

The CAD-sequence, B-rep, and CADQuery supervision required by the different methods is constructed from the same source CAD models. In particular, CADRILLE (Kolodiazhnyi et al., 2025) and CADReasoner (Kabisov et al., 2026) are trained using CADQuery programs converted from the source CAD sequences. This setup allows all methods to be retrained on the same source data while preserving their original output representations and generation procedures.

A.2.3 Training and Inference Details

This section provides further details on the training and inference of CADFlow. We first describe how predicted and ground-truth sequences of different lengths are aligned for CAD sequence optimization, followed by the construction of CAD operation error labels from actual CCPR predictions. We then specify the recovery termination criterion used during inference and the settings for geometric constraint optimization.

CAD Sequence Optimization. During joint training, CCPR generates an initial CAD sequence as the input to the subsequent EAR optimization. Because the predicted and ground-truth sequences may have different lengths, we align the two sequences to their maximum length and pad the shorter one with -1. This padding is used only to organize variable-length training sequences, and positions containing padding are excluded from the error-detection and local-redrawing losses. Ground-truth sequence lengths are used only for alignment during training, whereas the sequence length at inference is determined by the empty-state termination mechanism.

CAD Operation Error Label Construction. CAD operation error labels are constructed directly from the initial sequences generated by the current CCPR model rather than from artificially perturbed sequences. At each aligned position containing both a predicted operation and a ground-truth operation, we compare the two and label whether the predicted operation requires regeneration. Operations labeled as erroneous use their corresponding ground-truth CAD operations as the targets for local redrawing. EAR therefore refines the commands and parameters of operations already present in the initial predicted sequence, while the sequence length is determined by the construction-state trajectory recovered by CCPR.

Recovery Termination. At inference time, CCPR starts from the state feature corresponding to the input sketch and recursively recovers preceding construction states. We encode a blank canvas with the same feature encoder to obtain the empty-state reference feature and compute its cosine similarity with each predicted state feature. Recovery stops when this similarity reaches the threshold $\tau_{\text{stop}} = 0.95$. CAD operations are then decoded in the forward direction along the recovered trajectory.

Geometric Constraint Optimization. We initialize the optimization variables with the sketch parameters predicted by the local redrawing module and use SLSQP to minimize their squared deviations from the initial predictions subject to the predicted parallel and perpendicular constraints. All sketch parameters are optimized in the normalized coordinate space. We use a maximum of 100 iterations and a convergence tolerance of 10^{-6}. If the solver fails to converge, we retain the initial predicted parameters.

A.3 More Experimental Results

A.3.1 Step-wise Construction Accuracy

We evaluate cumulative construction-prefix accuracy at each extrusion step K over the complete DeepCAD test set. For a ground-truth sequence containing more than K extrusion blocks, the evaluation prefix extends from the beginning of the sequence through the K-th extrusion operation, inclusive. If the sequence contains no more than K extrusion blocks, its complete sequence through the final extrusion operation is evaluated. Each prediction is aligned to the correspond-

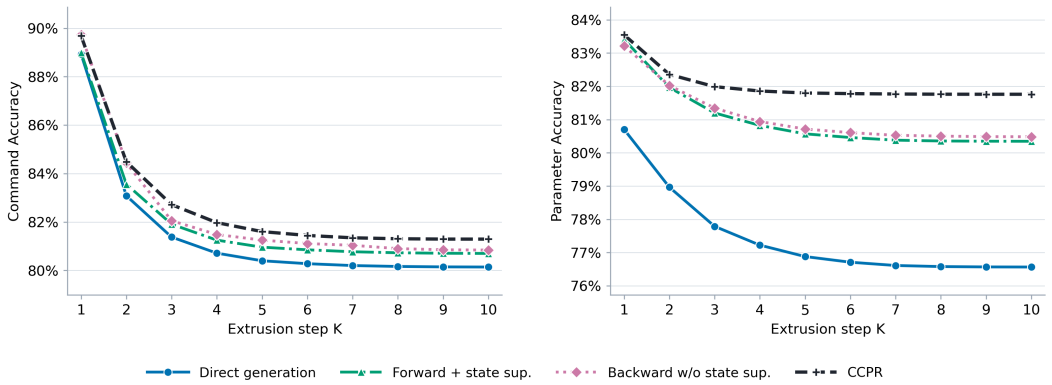

Figure 9: Cumulative command accuracy (left) and parameter accuracy (right) of construction prefixes evaluated up to different extrusion steps on DeepCAD. The first four variants in Table 3 are compared.

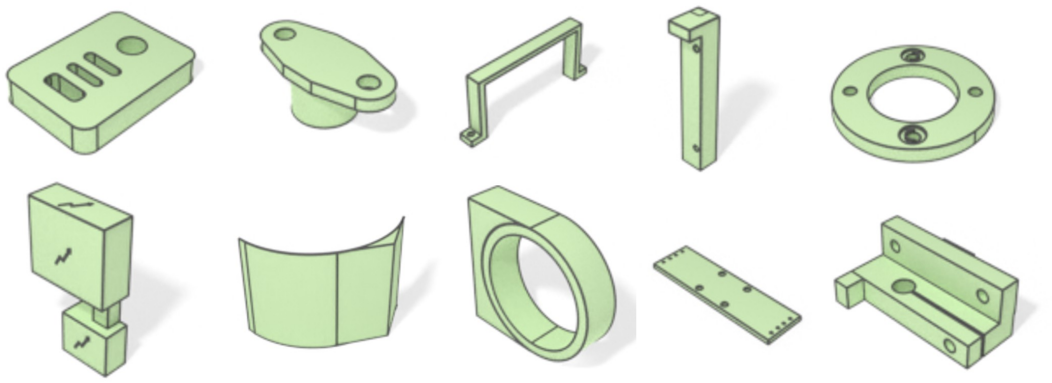

Figure 10: CADFlow results for construction trajectories of different lengths.

ing ground-truth prefix length, and missing predicted positions are counted as incorrect. Command and parameter accuracies are accumulated over all corresponding tokens in each evaluation prefix. Consequently, the result at the maximum extrusion step is equivalent to the final accuracy on the complete test set.

As shown in Figure 9, the cumulative accuracies of all variants decrease as the evaluated prefix covers more construction operations. Nevertheless, CCPR consistently achieves the highest command and parameter accuracies across different prefix lengths and maintains its advantage for longer construction sequences. Compared with direct generation, forward recovery with intermediate-state supervision, and backward recovery without intermediate-state supervision, these results show that combining backward recovery with intermediate-state supervision improves prediction quality throughout the construction process, rather than only improving the final CAD model.

A.3.2 Maximum Construction Length

Figure 10 presents CAD models generated by CADFlow with one to ten CAD operations. From left to right, the first row shows one to five operations and the second row shows six to ten operations, with each operation consisting of a 2D sketch followed by an extrusion command. The examples include perforated plates, brackets, annular parts, and connected block structures, covering a range of geometric compositions. The longer sequences also produce models containing multiple local features, such as holes on different faces and stepped connections. Together, these examples show

Table 5: Computational cost and reconstruction performance of the ablation variants in Table 3. Inference time is reported as the median per-sample latency.

Variant	Params (M)↓	GFLOPs↓	Median Time (ms)↓	$\mathrm{ACC}_{\mathrm{cmd}}$ (%)↑	$\mathrm{ACC}_{\mathrm{param}}$ (%)↑	IR (%)↓	CD ($\times 10^2$)↓
Direct generation	55.54	16.47	14.16	80.2	76.6	23.21	23.1
Forward + state sup.	49.22	33.61	135.77	80.7	80.4	21.0	20.8
Backward w/o state sup.	49.22	33.61	128.71	80.9	80.5	19.5	19.1
CCPR	49.22	33.61	130.42	81.3	81.7	4.81	18.7
CCPR + All-operation LR	66.40	53.61	183.70	78.6	82.0	4.53	15.1
CCPR + ED+LR	76.31	37.80	132.44	**82.3**	82.3	3.96	14.3
CADFlow (full)	77.63	49.44	173.15	**82.3**	**82.4**	**3.88**	**13.7**

that CADFlow can generate models with different construction lengths and geometric compositions, rather than being limited to simple structures requiring only a few operations.

A.3.3 Computational Cost Analysis

Table 5 reports the computational cost of the ablation variants corresponding to the rows in Table 3. Direct generation has the lowest inference cost because it predicts the CAD sequence without recovering intermediate construction states, but it exhibits substantially lower geometric validity, as shown in Table 3. Among the state-recovery variants, both forward and backward recovery require 33.61 GFLOPs, with median inference times of 135.77 ms and approximately 130 ms, respectively. Removing feature supervision does not change the parameter count or GFLOPs because this supervision is used only during training.

Locally redrawing all CAD operations without error localization increases the cost to 53.61 GFLOPs and 183.70 ms. In contrast, error detection and selective local redrawing reduce the cost to 37.80 GFLOPs and 132.44 ms by regenerating only operations predicted to be erroneous. The full CADFlow model additionally performs parallel and perpendicular constraint optimization, resulting in 77.63M parameters, 49.44 GFLOPs, and a median inference time of 173.15 ms. Together with the accuracy improvements in Table 3, these results show that error-aware selective redrawing controls the computational overhead of refinement, while geometric constraint optimization introduces additional cost in exchange for improved parameter accuracy and geometric quality.