ED2CAD: PARAMETRIC CAD RECONSTRUCTION FROM DIMENSION-ANNOTATED ENGINEERING DRAWINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reconstructing editable 3D parametric models from dimension-annotated engineering drawings remains a challenging problem. We propose ED2CAD, a dimension-aware framework for reconstructing parametric Computer-Aided Design (CAD) models from multimodal engineering drawings. To support this task, we develop an automated data pipeline that generates paired raster and vector drawings from B-rep geometry, including orthographic views with visible lines, hidden lines, and dimensional annotations. A raster-vector Transformer jointly encodes and fuses multimodal drawing features to autoregressively predict CAD commands and parameters. To improve geometric fidelity, we further introduce a point cloud geometry proxy that provides 3D supervision based on Chamfer distance. The predicted command sequences are instantiated through a CAD kernel to obtain executable 3D models for geometric evaluation. Extensive experiments demonstrate the effectiveness of dimensional annotations, multimodal inputs, geometric supervision, and the proposed architecture.

1 INTRODUCTION

Learning-based CAD has progressed from Constructive Solid Geometry (CSG) and program synthesis to parametricsketches, BRep representations, and editable feature reconstruction Sharma et al. (2017); Kania et al. (2020); Jones et al. (2020); Yu et al. (2022); Para et al. (2021); Seff et al. (2021); Willis et al. (2021); Li et al. (2022); Lambourne et al. (2021); Jayaraman et al. (2020; 2023); Li et al. (2023; 2024); Dupont et al. (2022); Ren et al. (2022),including longer programs and advanced operations Deng et al. (2025); Yin et al. (2026); Qi et al. (2026). Yet industrial engineering drawings, which include orthographic geometry, hidden structure and dimensions, still require costly manual conversion into editable 3D models.

Multi-view and point cloud reconstruction demonstrates the value of structured geometric priors Choy et al. (2016); Kar et al. (2017); Xie et al. (2019); Wang et al. (2020); Uy et al. (2022); Guo et al. (2022); Zhu et al. (2023); Liu et al. (2024), while vector graphics and differentiable rasterization expose complementary precise geometry Carlier et al. (2020); Reddy et al. (2021); Li et al. (2020); Puhachov et al. (2021); Jain et al. (2023); Li et al. (2025a). However, existing CAD datasets Koch et al. (2019); Wu et al. (2021); Willis et al. (2020) do not align general dimensioned drawings, B-rep geometry, and CAD command sequences, and drawing-based methods remain restricted to specific object categories or basic primitives Hu et al. (2023); Wang et al. (2024b); Qin et al. (2025); Niu et al. (2026). Moreover, raster and vector modalities contain complementary semantic information, yet they have not been jointly exploited Qin et al. (2025).

To address these limitations, we propose ED2CAD, a dimension-aware network for reconstructing parametric CAD models from annotated 2D engineering drawings. Based on the CAD command sequences and B-rep geometry provided by WHUCAD Fan et al. (2025), we construct the ED2CAD Dataset through three-view projection, visibility analysis, and dimension annotation, producing aligned PNG, DXF, and SVG drawings. Representative samples are shown in Figure 6. We further introduce a raster-vector Transformer that fuses raster and vector modalities to autoregressively predict CAD commands and parameters, together with point-cloud supervision based on Chamfer distance. The predicted sequences are instantiated using a CAD kernel and evaluated

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.

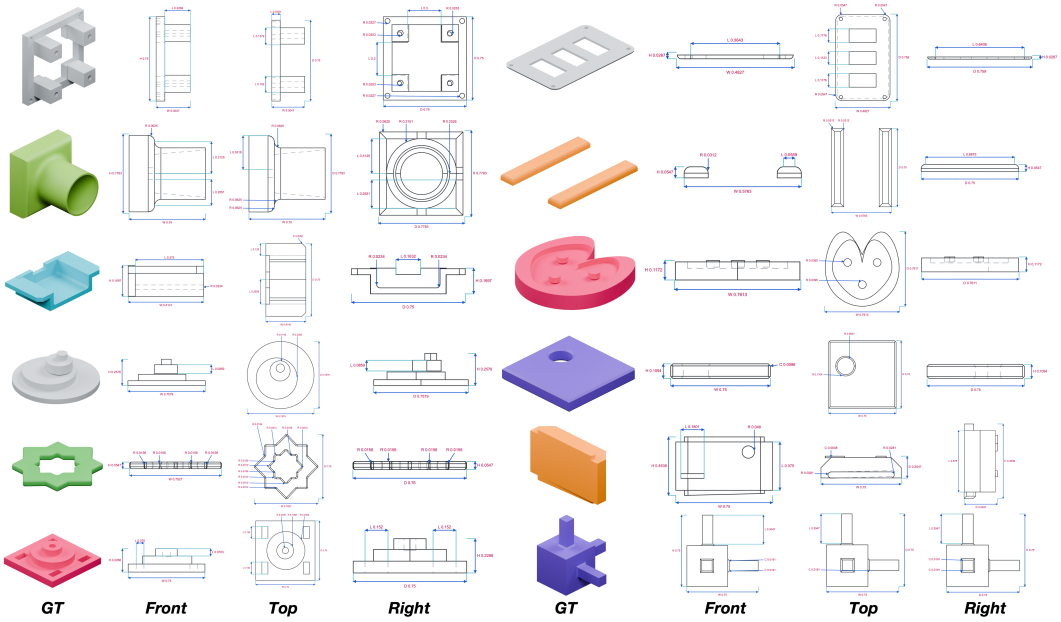

Figure 1: Representative samples from the ED2CAD Dataset. Ground truth shows the original WHUCAD CAD models, while ED2CAD generates the corresponding dimension-annotated three-view engineering drawings. Colored annotations are used for visualization. The samples include advanced primitives, such as parabolas and ellipses, and operations, such as chamfers and fillets.

quantitatively using command accuracy, parameter accuracy, Invalidity Ratio, and Chamfer distance, complemented by qualitative comparisons of the reconstructed CAD models. Ablation studies assess the effects of dimension annotations, input modalities, and geometric supervision.

Our main contributions are summarized as follows:

- We introduce the ED2CAD Dataset, comprising 133,656 samples of aligned raster and vector engineering drawings. An automated pipeline constructs the dataset from B-rep geometry and CAD sequences, incorporating visible lines, hidden lines, and dimensional annotations.
- We propose ED2CAD, a raster-vector Transformer that fuses multimodal engineering drawings to autoregressively predict parametric CAD commands and their parameters.
- We introduce a differentiable geometry proxy that predicts point clouds from the fused global representation, providing auxiliary 3D shape supervision through a Chamfer distance loss.

2 Related Work

2.1 Parametric CAD Sequence Modeling

Parametric CAD modeling commonly represents designs as construction sequences. DeepCAD encodes sketch and extrusion operations as discrete commands Wu et al. (2021), while Fusion 360 derives sketch and extrusion programs from real design histories. SketchGraphs models 2D primitives and constraints as graphs for parametric sketch generation Seff et al. (2020), and SkexGen disentangles sketch topology, geometry, and extrusion through discrete codebooks and autoregressive generation Xu et al. (2022). Recent methods further explore hierarchical vector quantization, state space models, multi scale diffusion, hierarchical BRep to sequence modeling, CAD kernel feedback, and direct entity selection Wang et al. (2024a); Li et al. (2025b); Deng et al. (2025); Zhang et al. (2024); Yin et al. (2026); Qi et al. (2026).

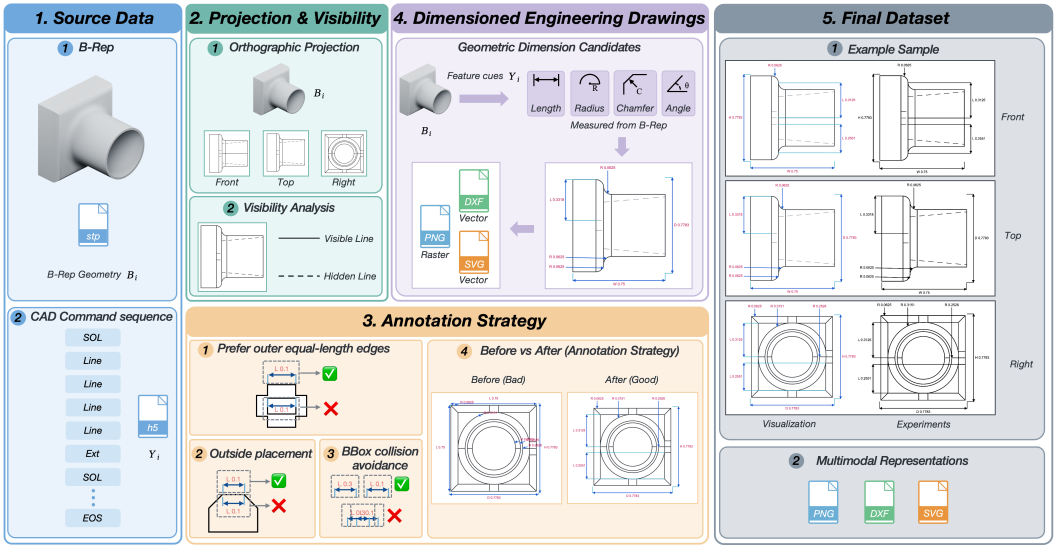

Figure 2: Engineering drawing data generation pipeline. B-rep geometry provides orthographic projections, visibility information, and dimension values. H5 sequence provides the corresponding dimension categories. The generated PNG, SVG, and DXF drawings are paired with their reference CAD sequences and B-rep geometry by sample ID.

2.2 CAD Sequence Reconstruction

Several studies recover CAD construction sequences from existing geometric or visual representations. CADParser infers construction sequences from B-rep graphs Zhou et al. (2023). For point clouds, MultiCAD, CAD-Diffuser, PS-CAD, and PartCAD introduce cross-modal contrastive learning, outline-first diffusion, residual-geometry prompting, and projection-guided part reasoning, respectively Ma et al. (2023; 2024); Yang et al. (2025); Liu et al. (2025). Other methods generate editable CAD models or command sequences from natural images, product images, and multiview renderings Chen et al. (2025b); Li & Sha (2025). FreeCAD supports text and image prompts, while CAD-GPT and CADCrafter improve image-to-sequence generation through spatial reasoning and preference optimization guided by a code checker Lin et al. (2025); Wang et al. (2025); Chen et al. (2025a). However, these methods do not explicitly exploit dimension-annotated engineering drawings.

2.3 CAD Reconstruction from Engineering Drawings

Few studies directly reconstruct parametric CAD models from 2D engineering drawings. PlankAssembly predicts plank-based shape programs from three orthographic views but supports limited geometries and no dimension annotations Hu et al. (2023). CAD2Program generates parametric programs from rasterized annotated drawings but focuses on cabinets with predefined components Wang et al. (2024b). CME-CAD introduces the CADExpert benchmark, which pairs dimensioned orthographic projections with expert reasoning, executable CADQuery code, and 3D models, and explores collaborative multi-expert reinforcement learning Niu et al. (2026). Drawing2CAD predicts CAD commands from SVG drawings but does not explicitly model dimensions or layer attributes and supports only basic primitives Qin et al. (2025). In contrast, ED2CAD jointly exploits raster and vector modalities, together with dimension annotations, to reconstruct complex CAD models containing advanced primitives and operations.

3 2D Engineering Drawing Dataset Construction

Our task comprises two stages: generating multimodal engineering drawings from each CAD sequence Y_i and its aligned B-rep geometry B_i, and reconstructing an executable CAD sequence from the generated drawings. This section describes the data generation stage, whose pipeline is illustrated in Figure 2.

3.1 Source Audit and Geometry Alignment

We audit the WHUCAD command sequences and align them with the corresponding step geometry.

Sequence audit. For each H5 file, we record its identifier, official split, sequence length, $T \times 33$ command array, command names, End-of-Sequence (EOS) status, and integrity checks. Dataset-level statistics summarize split sizes, command frequencies, sequence lengths, and invalid records, ensuring traceability between generated drawings and command supervision.

Geometry alignment. STEP files are indexed and matched to vector samples by identifier. This process aligns 133,656 of 146,331 samples (91.3%). Unmatched samples are retained for diagnosis but excluded from projection. The official train, validation, and test splits are assigned before filtering to preserve the original data partition. For further details, please see Appendix A.2.

3.2 Visibility-Aware Orthographic Projection

We load each B-rep geometry B_i into FreeCAD, which uses OpenCascade for geometric processing. For the front, top, and right views, denoted by $\mathcal{V} = v_f, v_t, v_r$, We export SVG and DXF drawings with visible and hidden lines. We record geometric attributes of edges and faces, projected polylines, bounding boxes, and visibility evidence, together with view-level metadata and failure information. Only samples with complete SVG/DXF pairs for all three views and valid geometry metadata are retained. Coordinate conventions, visibility states, and export metadata are specified in Appendix A.3.

3.3 Semantic Dimension Annotation

For each view, we generate overall, linear, radial, and command-derived dimension candidates from the projected B-rep geometry. Linear dimensions are retained only when the corresponding edge is sufficiently long and nearly parallel to the projection plane. Feature dimensions, including radii, diameters, and dimensions associated with holes, fillets, and chamfers, are identified from compatible B-rep geometric entities, with the H5 sequence providing the corresponding dimension categories. All displayed dimension values are measured directly from the B-rep geometry.

After removing invalid and duplicate candidates, semantic matches take precedence over geometry-only matches. Each candidate a is ranked by

$$s(a) = b_{\tau(a)} + \lambda_g \bar{g}(a) + \lambda_a A(a) - \lambda_{\text{val}} r_{\text{val}}(a) - \lambda_{\text{pos}} r_{\text{pos}}(a), \tag{1}$$

where $b_{\tau(a)}$ is the prior associated with annotation type $\tau(a)$, $\bar{g}(a)$ measures the normalized geometric size, and $A(a)$ rewards semantic and geometric agreement. The terms $r_{\text{val}}(a)$ and $r_{\text{pos}}(a)$ penalize repeated dimension values and spatially crowded regions, respectively, while λ_g, λ_a, λ_{val}, and λ_{pos} control the contributions of the corresponding terms. The annotation budget is adjusted according to drawing complexity to balance information coverage and visual clutter. Further details are provided in Appendix A.4.

Candidate labels are placed outside the object contour to avoid overlaps:

$$\ell^\star = \arg\min_{\ell \in \mathcal{L}(a)} \left(4 r_{\text{side}}(\ell) + 2 r_{\text{lane}}(\ell)\right) \text{ s.t.} \qquad B(\ell) \cap C_v = \varnothing,\, B(\ell) \cap \mathcal{B}^{+\delta} = \varnothing, \tag{2}$$

where $\mathcal{L}(a)$ is the set of candidate layouts for annotation a, $r_{\text{side}}(\ell)$ and $r_{\text{lane}}(\ell)$ denote the side and lane preference ranks of layout ℓ, respectively, and the coefficients 4 and 2 weight their relative importance. $B(\ell)$ denotes the annotation region, C_v is the object contour, and $\mathcal{B}^{+\delta}$ contains previously occupied regions expanded by a clearance margin δ. Finally, the three annotated views are composed and exported as monochrome PNG, SVG, and DXF drawings, together with color audit, annotation-free, and dimension-only variants. Details of annotation layout and drawing generation are provided in Appendices A.5 and A.6.

4 Architecture of ED2CAD

ED2CAD comprises a raster encoder, two vector encoders, a modality fusion Transformer, an autoregressive CAD decoder, and a geometry proxy for auxiliary 3D supervision. Its overall architecture is illustrated in Figure 3.

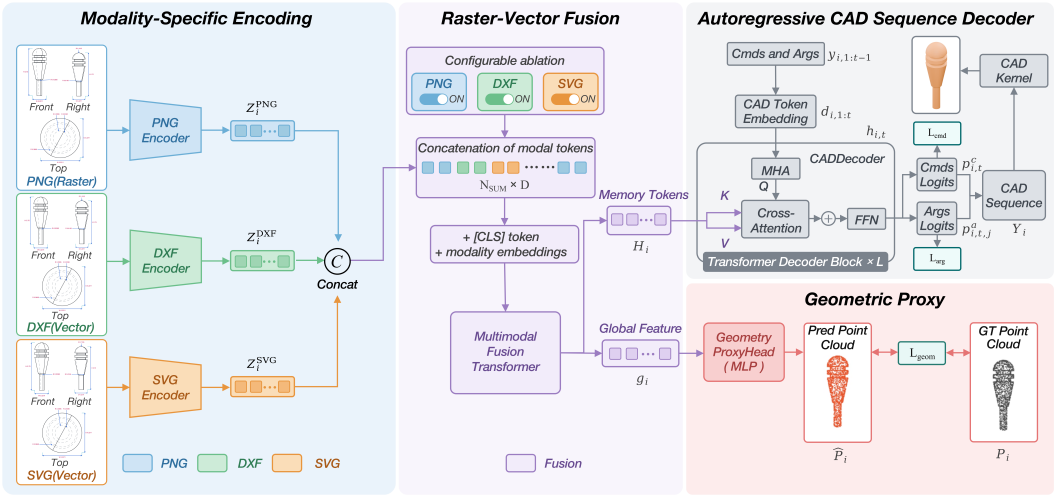

Figure 3: The overview of our ED2CAD framework. The model encodes and fuses PNG, DXF, and SVG inputs, autoregressively predicts a CAD program for kernel execution, and receives auxiliary 3D supervision from a geometry proxy during training.

4.1 Problem Formulation and Model I/O

For sample i, the available modalities form a nonempty set $\mathcal{M}_i \subseteq \text{PNG}, \text{DXF}, \text{SVG}$, and the input is defined as

$$\mathcal{X}_i = \{\mathcal{X}_i^m\}_{m\in\mathcal{M}_i}, \quad \mathcal{X}_i^m = \begin{cases} I_i, & m = \text{PNG}, \\ X_i^m, & m \in \{\text{DXF}, \text{SVG}\}, \end{cases} \tag{3}$$

where $I_i \in \mathbb{R}^{3\times224\times224}$ captures the global drawing layout, and $X_i^m \in \mathbb{R}^{2048\times128}$ encodes vector entities and their geometric and semantic attributes. Missing modalities are omitted.

The output is a parametric CAD construction sequence:

$$Y_i = (y_{i,1}, \ldots, y_{i,T_i}), \quad y_{i,t} = (c_{i,t}, \mathbf{a}_{i,t}), c_{i,t}\in\{0,\ldots,26\}, \quad \mathbf{a}_{i,t}\in\{0,\ldots,255\}^{32}, \tag{4}$$

where, $T_i \leq 100$, ci, t denotes one of 27 CAD commands, and $\mathbf{a}_{i,t}$ contains 32 quantized parameters. A validity matrix $Q \in 0, 1^{27\times32}$ masks command specific parameters, with unused slots set to 0. Generation begins with SOL and terminates at EOS or the maximum sequence length. The preprocessing and target-sequence conventions are detailed in Appendices B.1 and B.2.

4.2 Modality-Specific Encoding

For each modality, N_m denotes its encoded token count, with $N_{\text{PNG}} = 196$ and $N_{\text{DXF}} = N_{\text{SVG}} = 2048$.

Raster branch. The PNG encoder uses ViT patch embedding to partition I_i into 196 non-overlapping image tokens. The raster tokens are encoded as

$$S_i^{\text{m}} = \mathcal{P}_{16}(I_i) + P^{\text{m}} + e^{\text{m}}, \qquad Z_i^{\text{m}} = \text{LN}(\mathcal{T}_{\text{m}}(S_i^{\text{m}})) \in \mathbb{R}^{N_m\times D}, \quad D = 256, \tag{5}$$

where $m = \text{PNG}$; $\mathcal{P}_{16}$ denotes patch embedding; $P^{\text{PNG}} \in \mathbb{R}^{196\times D}$ is learned positional embedding; $e^{\text{PNG}} \in \mathbb{R}^D$ is learned modality embedding; $\mathcal{T}_{\text{PNG}}$ is six-layer PNG Transformer encoder.

Vector branches. DXF and SVG tokenizers map raw vector entities and their metadata into 128-dimensional tokens, producing $X_i^m \in \mathbb{R}^{2048\times128}$ after padding or truncation. DXF and SVG are then processed by architecturally matched but separately parameterized vector encoders:

$$\begin{aligned} U_i^m &= \text{MLP}_m(X_i^m) + E_{\tau,m}(\boldsymbol{\tau}_i^m) + E_{\ell,m}(\boldsymbol{\ell}_i^m) + e^m \in \mathbb{R}^{N_m\times D}, \\ Z_i^m &= \text{LN}(\mathcal{T}_m(U_i^m)) \in \mathbb{R}^{N_m\times D}. \end{aligned} \tag{6}$$

where $m \in \{\text{DXF}, \text{SVG}\}$; $\boldsymbol{\tau}_i^m$ denotes entity-type; $\boldsymbol{\ell}_i^m$ is layer identifiers; MLP_m, $E_{\tau,m}$, and $E_{\ell,m}$ denote feature, entity-type, and layer embeddings; $e^m \in \mathbb{R}^D$ is learned modality embedding; $\mathcal{T}_m$ is four-layer vector Transformer encoder.

This representation encodes vector-primitive and layer information while matching the raster feature dimension for subsequent fusion. Encoder and token details are provided in Appendices C.1 and C.2.

4.3 Raster-Vector Fusion

We fuse the enabled modality token sequences Z_i^m into a representation H_i as

$$H_i = F_\phi\left(\left[z_{\text{cls}}; \underset{m\in\mathcal{M}_i}{\|}(Z_i^m + \tilde{e}^m)\right]\right) = [g_i; R_i] \in \mathbb{R}^{(1+L_i)\times D}, \tag{7}$$

where $L_i = \sum_{m\in\mathcal{M}_i} N_m$ is the total number of enabled modality tokens for sample i; $\tilde{e}^m \in \mathbb{R}^D$ is a learned fusion-stage modality embedding, distinct from the encoder-stage e^m; $\|$ denotes concatenation along the token dimension in the fixed order PNG-DXF-SVG; $z_{\text{cls}} \in \mathbb{R}^D$ is a learned global token; F_ϕ denotes a four-layer fusion Transformer followed by a final normalization layer, and ϕ denotes its learnable parameters. The first token $g_i \in \mathbb{R}^D$ of H_i is the global feature used by geometry proxy, $R_i \in \mathbb{R}^{L_i\times D}$ is modality fusion tokens. CADDecoder uses H_i as cross-attention memory. Additional fusion-memory details are provided in Appendix C.2.

4.4 Autoregressive CAD Sequence Decoder

For $t = 1, \ldots, T_i$, the command of $t-1$ and its parameters are embedded as

$$d_{i,t} = E_c(c_{i,t-1}) + W_{\text{arg}}\left(\frac{1}{32}\sum_{j=1}^{32} E_a(a_{i,t-1,j})\right) + P_{t-1}^{\text{dec}}, \tag{8}$$

where $d_{i,t} \in \mathbb{R}^D$ is the CADDecoder input; $a_{i,t,j}$ is the jth component of $\mathbf{a}_{i,t}$; E_c and E_a are learned command and parameter embeddings; W_{arg} is a learned parameter-embedding projection; $P_{t-1}^{\text{dec}} \in \mathbb{R}^D$ is a learned decoder positional embedding.

As illustrated in Figure 3, CADDecoder applies masked self-attention to $d_{i,1:t}$ and cross-attention to H_i. The decoding process and predictions of CADDecoder are defined as follows:

$$\begin{gathered} h_{i,t} = \text{CADDecoder}(d_{i,1:t}, H_i) \in \mathbb{R}^D,\ \boldsymbol{p}_{i,t}^c = \text{softmax}(W_c h_{i,t}) \in \mathbb{R}^{27}, \\ \boldsymbol{p}_{i,t,j}^a = \text{softmax}(W_{a,j} h_{i,t}) \in \mathbb{R}^{256}. \end{gathered} \tag{9}$$

where $d_{i,1:t} = (d_{i,1}, \ldots, d_{i,t})$; $h_{i,t}$ is the hidden state; $\boldsymbol{p}_{i,t}^c$ is the distribution over 27 commands; $\boldsymbol{p}_{i,t,j}^a$ is the distribution over 256 values; W_c and $W_{a,j}$ are learned output projections. The sequence is generated autoregressively as

$$p(Y_i \mid \mathcal{X}_i) = \prod_{t=1}^{T_i} p(y_{i,t} \mid y_{i,<t}, H_i), \tag{10}$$

where $y_{i,<t} = (y_{i,1}, \ldots, y_{i,t-1})$ denotes the preceding sequence prefix, and $p(\cdot \mid \cdot)$ denotes a conditional probability distribution. During training, the ground-truth prefix is provided through teacher forcing; during inference, it is replaced by the previously generated prefix. Prefix alignment and inference details are given in Appendix C.3.

4.5 Differentiable Geometry Proxy

To inject three-dimensional shape information into the fused representation, we feed the global feature g_i to an auxiliary geometry proxy head. The head represents each proxy point cloud using 2048 three-dimensional points and predicts their coordinates through

$$\widehat{P}_i = \text{reshape}(\tanh(G_\psi(g_i))) \in [-1, 1]^{2048\times 3}, \tag{11}$$

where $\widehat{P}_i$ is the predicted point cloud for sample i; $G_\psi : \mathbb{R}^D \to \mathbb{R}^{6144}$ denotes the input normalization and three-layer MLP of the geometry proxy head; ψ denotes its learnable parameters; reshape organizes the 6144 outputs into 2048 three-dimensional points. During training, each prediction $\widehat{P}_i$ is paired with a cached target point cloud $P_i \in \mathbb{R}^{2048\times 3}$ sampled from the corresponding B-rep geometry. Point-cloud sampling and proxy gradient paths are described in Appendix D.1.

4.6 Loss Function

We train the model with a sequence reconstruction objective and an auxiliary geometry objective.

Sequence loss. Let Ω_c contain all valid command positions, and let $\Omega_a = \{(i,t,j) : (i,t) \in \Omega_c,\ Q_{c_{i,t},j} = 1\}$ contain the valid parameter positions. The command and parameter losses are

$$\mathcal{L}_{\text{cmd}} = -\frac{1}{|\Omega_c|} \sum_{(i,t)\in\Omega_c} \log \boldsymbol{p}^c_{i,t}[c_{i,t}],\ \mathcal{L}_{\text{arg}} = -\frac{1}{|\Omega_a|} \sum_{(i,t,j)\in\Omega_a} \log \boldsymbol{p}^a_{i,t,j}[a_{i,t,j}], \tag{12}$$

where $\boldsymbol{p}^c_{i,t}[c_{i,t}]$ and $\boldsymbol{p}^a_{i,t,j}[a_{i,t,j}]$ select the probabilities assigned to the corresponding ground-truth classes. The sequence loss is

$$\mathcal{L}_{\text{seq}} = \lambda_c \mathcal{L}_{\text{cmd}} + \lambda_a \mathcal{L}_{\text{arg}} + \lambda_e \mathcal{L}_{\text{eos}}, \tag{13}$$

where $\mathcal{L}_{\text{eos}}$ is the loss of EOS positions, and λ_c, λ_a, and λ_e weight the three terms.

Geometry loss. The geometry loss is defined as

$$\mathcal{L}_{\text{geom}} = \frac{1}{2048}\left(\sum_{\widehat{p}\in\widehat{P}_i} \min_{p\in P_i} \|\widehat{p} - p\|_2 + \sum_{p\in P_i} \min_{\widehat{p}\in\widehat{P}_i} \|p - \widehat{p}\|_2 \right), \tag{14}$$

where $\widehat{p}$ and p denote individual predicted and target points.

Total loss. The final training objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{seq}} + \lambda_g \mathcal{L}_{\text{geom}}, \tag{15}$$

where λ_g controls the geometry contribution. Loss reductions and training-update details are provided in Appendix D.2.

5 Experiments

We present the framework for reconstructing parametric CAD models from dimension annotated vector engineering drawings. We evaluate ED2CAD through baseline comparisons, ablations of input modalities, dimension annotations, and model components, and qualitative failure analysis.

5.1 Experimental Setups

Implementation Details. We train on two NVIDIA GeForce RTX 4090 GPUs for 200 epochs with a global batch size of 32 and two gradient accumulation steps, yielding an effective global batch size of 64. AdamW is used with a learning rate of 10^{-4}, weight decay of 2×10^{-2}, five epoch warmup from 10^{-6}, and cosine decay to 10^{-6}. We apply dropout rates of 0.1 in the encoders, 0.15 in the fusion Transformer, and 0.2 in the CAD decoder, gradient clipping at 1.0, and mixed precision training, and evaluate the checkpoint with the lowest validation loss. The expanded optimization settings are listed in Appendix D.3.

Table 1: Ablation over all non-empty input modality subsets.

PNG	DXF	SVG	$A_c \uparrow$	$A_p \uparrow$	IR↓	MCD↓
✓	–	–	89.89	79.39	19.14	8.58
–	✓	–	92.91	80.58	18.31	6.89
–	–	✓	92.49	80.24	18.74	7.54
✓	✓	–	93.37	80.79	17.48	5.97
✓	–	✓	93.24	80.70	18.08	6.54
–	✓	✓	93.61	80.93	17.41	5.90
✓	✓	✓	94.03	81.05	17.36	5.81

Datasets. We use WHUCAD samples with aligned CAD sequences, B-rep geometry, PNG/DXF/SVG drawings, and valid point clouds. Filtering the official split yields 120,274 training, 6,703 validation, and 6,679 test samples. PNG inputs are resized to 224×224; vector inputs are limited to 2048 tokens, and CAD sequences are padded or truncated to 100 steps. All ablations use identical samples and splits. The filtering and split construction are summarized in Appendix A.2.

Evaluation Metrics. We report command accuracy A_c, parameter accuracy A_p, invalidity ratio IR, and minimum Chamfer distance MCD. The exact token-level definitions and execution-level conventions are provided in Appendix E.1.

5.2 Multimodal Input Ablation

Table 4: Component ablation with fully annotated PNG+DXF+SVG inputs. Direct fusion follows Eq. equation 16.

Variant	Fusion	$\tilde{e}^m$	$\mathcal{L}_{\text{geom}}$	$A_c \uparrow$	$A_p \uparrow$	IR↓	MCD↓
w/o fusion	Concat.+LN	✓	✓	90.71	77.68	18.62	6.48
w/o modality embedding	F_ϕ	–	✓	92.14	78.96	17.94	6.18
w/o geometry proxy	F_ϕ	✓	–	92.28	79.37	19.07	6.91
Full ED2CAD	F_ϕ	✓	✓	94.03	81.05	17.36	5.81

To evaluate the effectiveness of each modality, we conduct ablation experiments with annotated inputs and the geometry proxy enabled, while keeping the data split, model architecture, and optimization schedule fixed. The results are shown in Table 1. DXF achieves the strongest single-modality performance, followed by SVG and PNG. Combining two modalities consistently improves the results, with DXF+SVG slightly outperforming PNG+DXF. The complete multimodal setting achieves the best performance across all metrics, indicating that raster appearance and the two vector representations provide complementary evidence for CAD reconstruction. Further details are provided in Appendix E.2.

Table 2: Ablation of dimension annotations under fixed multimodal input.

Drawing variant	$A_c \uparrow$	$A_p \uparrow$	IR↓	MCD↓
Unannotated	93.23	79.81	18.26	6.72
Annotated	94.03	81.05	17.36	5.81

5.3 Dimension Annotation Ablation

We isolate the contribution of dimension annotations using paired drawings of the same CAD models. All three input modalities and the geometry proxy are enabled, while the drawings are provided under two settings: without and with dimension annotations. The results are shown in Table 2. Removing dimension annotations causes a substantially larger reduction in parameter accuracy than in command accuracy, as they primarily provide explicit numerical constraints. They can also disambiguate similar geometric primitives and construction operations, such as a shallow arc and a straight line. The annotated setting further reduces IR and MCD, indicating improved program validity and geometric fidelity.

Table 3: Comparison under matched input and output protocols. Adapted methods are retrained on the same WHUCAD split.

Method	Input	$A_c \uparrow$	$A_p \uparrow$	IR↓	MCD↓
DeepCAD-vector	SVG	78.96	72.68	25.17	13.46
Drawing2CAD	SVG	80.47	73.81	22.84	12.92
ED2CAD	SVG	88.49	78.23	19.96	8.61

5.4 Comparison with Existing Methods

We adapt Drawing2CAD and its DeepCAD-vector baseline to the WHUCAD output representation. Photo2CAD is excluded because its raster input is incompatible with our matched SVG-only protocol. All methods are retrained on the same WHUCAD split using dimension-annotated SVG input, the same CAD kernel, and identical metrics; the geometry proxy is disabled for ED2CAD. As shown in Table 3, ED2CAD outperforms both baselines. Figure 4 further shows that DeepCAD-vector captures only coarse geometry, whereas Drawing2CAD often misses high-level features such as chamfers and fillets. In contrast, ED2CAD more closely matches the ground truth.

Table 5: Results on the DeepCAD-derived drawing benchmark.

Input	Method	$A_c \uparrow$	$A_p \uparrow$	IR↓	MCD↓
PNG	DeepCAD-raster	78.52	74.31	28.07	16.84
	ED2CAD	90.54	80.17	18.67	8.32
SVG	DeepCAD-vector	83.18	78.24	20.68	10.63
	Drawing2CAD	84.06	79.02	18.75	10.29
	ED2CAD	93.05	81.24	17.65	7.29
All	ED2CAD	94.69	81.32	16.77	5.15

5.5 Component Ablation

Under the annotated multimodal setting, we ablate the fusion Transformer, fusion-stage modality embeddings, and geometry proxy while keeping all other settings fixed.

For direct fusion, we replace F_ϕ with token concatenation, normalization, and mean pooling:

$$\bar{R}_i = \mathrm{LN}\left(\Big\| _{m \in \mathcal{M}_i} (Z_i^m + \tilde{e}^m) \right), \ \bar{g}_i = \frac{1}{L_i} \sum_{\ell=1}^{L_i} \bar{R}_{i,\ell}, \ \bar{H}_i = [\bar{g}_i; \bar{R}_i]. \tag{16}$$

This preserves the decoder memory size but removes cross-modal self-attention. We ablate fusion-stage modality embeddings by setting $\tilde{e}^m = \mathbf{0}$, and ablate the geometry proxy by removing G_ψ and $\mathcal{L}_{\text{geom}}$.

As shown in Table 4, all ablations degrade performance. Direct fusion causes the largest accuracy drop, modality embedding removal weakens modality distinction, and geometry proxy removal notably increases IR and MCD, confirming its contribution to structural validity and geometric fidelity.

5.6 Evaluation on DeepCAD Dataset

We apply the same pipeline to DeepCAD while retaining its native six-command, 16-argument, 60-step representation and official 90%/5%/5% split. All methods are retrained and evaluated under the same protocol. As shown in Table 5, ED2CAD outperforms modality-matched baselines on both PNG and SVG inputs, while multimodal fusion further improves performance, demonstrating generalization across CAD datasets.

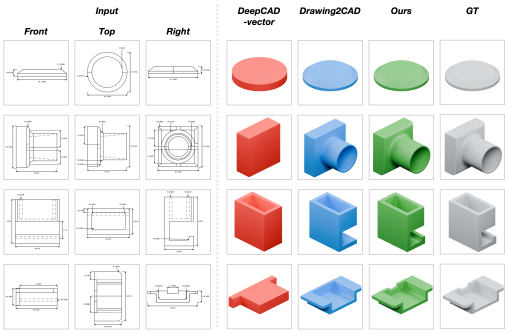

Figure 4: Qualitative comparison under the matched SVG-only protocol. The predicted H5 command sequences are reconstructed using the same CAD kernel.

5.7 Limitations

Reconstruction is constrained by a fixed command vocabulary, argument schema, and sequence length, while autoregressive decoding may accumulate errors on complex models, leading to missing or inaccurate features, as shown in Figure 5. Moreover, point cloud supervision does not explicitly enforce command validity, parametric constraints, or exact B-rep geometry. Future work will incorporate real industrial drawings.

6 Conclusion

We present ED2CAD, a dimension-aware multimodal framework for reconstructing editable CAD programs from PNG, DXF, and SVG engineering drawings. The accompanying data-generation pipeline aligns three-view raster and vector drawings with B-rep geometry and CAD command sequences, while preserving visible structure, hidden structure, and dimension cues. The raster-vector Transformer combines complementary visual and geometric representations to predict executable CAD programs, and the point-cloud geometry proxy supplies an additional training signal for three-dimensional fidelity. Experiments on WHUCAD and DeepCAD show consistent benefits from dimension annotations, multimodal fusion, and geometric supervision across modality ablations, component ablations, and matched baseline comparisons. These results indicate that combining explicit engineering annotations with structured vector inputs is an effective direction for reconstructing editable CAD models, while the remaining failures on complex programs motivate stronger constraint-aware decoding and more robust support for real industrial drawings.

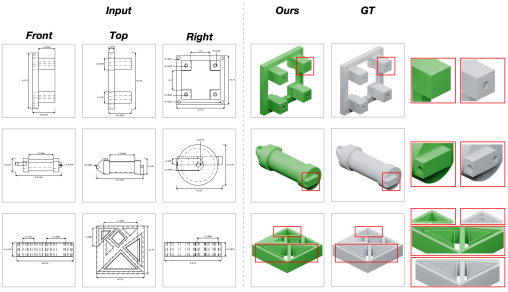

Figure 5: Representative failure cases on complex CAD models. Each row shows dimension-annotated three-view inputs, the reconstruction produced by ED2CAD, and the ground-truth model.

7 AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, assist in the retrieval and discovery of related work, and draft parts of the manuscript. All AI-assisted text and retrieved information were reviewed and verified by the authors, and relevant literature was checked against the original sources. For the development of the research contributions and experimental design, we used AI tools only to assist with literature search and retrieval. Al l research contributions and experimental settings were independently developed by the authors through literature review and independent reasoning, rather than generated by AI. We also used AI tools to assist with limited code generation, such as scripts for batch rendering CAD models in Blender for visualization. We take full responsibility for the final content of this work.

8 ETHICS STATEMENT

This research aims to automate parametric CAD reconstruction from engineering drawings to assist professional design in the CAD domain. The data used in this work are derived from existing CAD datasets through an automated pipeline that generates engineering drawings and corresponding annotations. This process does not involve human participants or personal data, and the generated data are intended solely for research on CAD reconstruction and generation.

9 REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the data generation pipeline, preprocessing procedures, model architecture, training objectives, evaluation metrics, and experimental protocols in the main paper and appendices. The appendices further specify the drawing generation and dimension annotation procedures, modality encoders, CAD decoder, geometric supervision, optimization settings, and evaluation protocols required to reproduce our experiments. Dataset splits, model configurations, and implementation details are also documented. Code will be made available to reviewers during the review process upon request.

REFERENCES

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 16351–16361. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/bcf9d6bd14a2095866ce8c950b702341-Paper.pdf.

Cheng Chen, Jiacheng Wei, Tianrun Chen, Chi Zhang, Xiaofeng Yang, Shangzhan Zhang, Bingchen Yang, Chuan-Sheng Foo, Guosheng Lin, Qixing Huang, and Fayao Liu. Cadcrafter: Generating computer-aided design models from unconstrained images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11073–11082, June 2025a.

Tianrun Chen, Chunan Yu, Yuanqi Hu, Jing Li, Tao Xu, Runlong Cao, Lanyun Zhu, Ying Zang, Yong Zhang, Zejian Li, and Lingyun Sun. Img2CAD: Conditioned 3-d CAD model generation from single image with structured visual geometry. *IEEE Transactions on Industrial Informatics*, 21(11):8539–8549, 2025b.

Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *CoRR*, abs/1604.00449, 2016. URL http://arxiv.org/abs/1604.00449.

Liyuan Deng, Yunpeng Bai, Yongkang Dai, Xiaoshui Huang, Hongping Gan, Dongshuo Huang, Hao Jiacheng, and Yilei Shi. Mamtiff-cad: Multi-scale latent diffusion with mamba+ for complex parametric sequence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10517–10526, October 2025.

Elona Dupont, Kseniya Cherenkova, Anis Kacem, Sk Aziz Ali, Ilya Arzhannikov, Gleb Gusev, and Djamila Aouada. CADOps-Net: Jointly Learning CAD Operation Types and Steps from Boundary-Representations . In *2022 International Conference on 3D Vision (3DV)*, pp. 114–123, Los Alamitos, CA, USA, September 2022. IEEE Computer Society. doi: 10.1109/3DV57658.2022.00024. URL https://doi.ieeecomputersociety.org/10.1109/3DV57658.2022.00024.

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

Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Trans. Graph.*, 41(4), July 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530078. URL https://doi.org/10.1145/3528223.3530078.

Wentao Hu, Jia Zheng, Zixin Zhang, Xiaojun Yuan, Jian Yin, and Zihan Zhou. Plankassembly: Robust 3d reconstruction from three orthographic views with learnt shape programs. 2023. URL https://arxiv.org/abs/2308.05744.

Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1911–1920, June 2023.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Thomas Davies, Hooman Shayani, and Nigel J. W. Morris. Uv-net: Learning from curve-networks and solids. *CoRR*, abs/2006.10211, 2020. URL https://arxiv.org/abs/2006.10211.

Pradeep Kumar Jayaraman, Joseph G. Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. Solidgen: An autoregressive model for direct b-rep synthesis. 2023. URL https://arxiv.org/abs/2203.13944.

R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *CoRR*, abs/2009.08026, 2020. URL https://arxiv.org/abs/2009.08026.

Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. Ucsg-net - unsupervised discovering of constructive solid geometry tree. *CoRR*, abs/2006.09102, 2020. URL https://arxiv.org/abs/2006.09102.

Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/9c838d2e45b2ad1094d42f4ef36764f6-Paper.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)*, June 2019.

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 (CVPR)*, pp. 12773–12782, June 2021.

Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Free2cad: parsing freehand drawings into cad commands. *ACM Trans. Graph.*, 41(4), July 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530133. URL https://doi.org/10.1145/3528223.3530133.

Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. SECAD-Net: Self-supervised CAD reconstruction by learning sketch-extrude operations. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16816–16826, 2023.

Pu Li, Jianwei Guo, Huibin Li, Bedrich Benes, and Dong-Ming Yan. SfmCAD: Unsupervised CAD reconstruction by learning sketch-based feature modeling operations. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4671–4680, 2024.

Pu Li, Wenhao Zhang, Jianwei Guo, Jinglu Chen, and Dong-Ming Yan. Revisiting cad model generation by learning raster sketch. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 4869–4877, 2025a.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph.*, 39(6), November 2020. ISSN 0730-0301. doi: 10.1145/3414685.3417871. URL https://doi.org/10.1145/3414685.3417871.

Xingang Li and Zhenghui Sha. Image2cadseq: Computer-aided design sequence and knowledge inference from product images. 2025. URL https://arxiv.org/abs/2501.04928.

Xueyang Li, Yunzhong Lou, Yu Song, and Xiangdong Zhou. Mamba-CAD: State space model for 3d computer-aided design generative modeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(5):5013–5021, 2025b.

Dawei Lin, Meng Yuan, Ziming Wang, Tieru Wu, and Yuanning Liu. Freecad: A multimodal framework for 3d cad model generation from free-form prompts. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 1948–1956, 2025.

Yang Liu, Daxuan Ren, Yijie Ding, Jianmin Zheng, and Fang Deng. Learning CAD modeling sequences via projection and part awareness. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

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 (CVPR)*, pp. 3763–3772, June 2024.

Weijian Ma, Minyang Xu, Xueyang Li, and Xiangdong Zhou. MultiCAD: Contrastive representation learning for multi-modal 3d computer-aided design models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pp. 1766–1776, 2023.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27144–27153, 2024.

Ke Niu, Haiyang Yu, Zhuofan Chen, Zhengtao Yao, Weitao Jia, Xiaodong Ge, Jingqun Tang, Benlei Cui, Bin Li, and Xiangyang Xue. CME-CAD: Heterogeneous collaborative multi-expert reinforcement learning for CAD code generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39272–39281, June 2026.

Wamiq Reyaz Para, Shariq Farooq Bhat, Paul Guerrero, Tom Kelly, Niloy J. Mitra, Leonidas J. Guibas, and Peter Wonka. Sketchgen: Generating constrained CAD sketches. *CoRR*, abs/2106.02711, 2021. URL https://arxiv.org/abs/2106.02711.

Ivan Puhachov, William Neveu, Edward Chien, and Mikhail Bessmeltsev. Keypoint-driven line drawing vectorization via polyvector flow. *ACM Trans. Graph.*, 40(6), December 2021. ISSN 0730-0301. doi: 10.1145/3478513.3480529. URL https://doi.org/10.1145/3478513.3480529.

Dacheng Qi, Chenyu Wang, Jingwei Xu, Tianzhe Chu, Zibo Zhao, Wen Liu, Wenrui Ding, Yi Ma, and Shenghua Gao. Pointer-CAD: Unifying B-Rep and command sequences via pointer-based edges & faces selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17377–17387, 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 *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, pp. 10573–10582, 2025.

Pradyumna Reddy, Michaël Gharbi, Michal Lukác, and Niloy J. Mitra. Im2vec: Synthesizing vector graphics without vector supervision. *CoRR*, abs/2102.02798, 2021. URL https://arxiv.org/abs/2102.02798.

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*, volume 13662 of *Lecture Notes in Computer Science*, pp. 482–498, Cham, 2022.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *CoRR*, abs/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. *CoRR*, abs/2109.14124, 2021. URL https://arxiv.org/abs/2109.14124.

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. *CoRR*, abs/1712.08290, 2017. URL http://arxiv.org/abs/1712.08290.

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, June 2022.

Hanxiao Wang, Mingyang Zhao, Yiqun Wang, Weize Quan, and Dong-Ming Yan. Vq-cad: Computer-aided design model generation with vector quantized diffusion. *Computer Aided Geometric Design*, 111:102327, 2024a.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7880–7888, 2025.

Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. PIE-NET: parametric inference of point cloud edges. *CoRR*, abs/2007.04883, 2020. URL https://arxiv.org/abs/2007.04883.

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. 2024b. URL https://arxiv.org/abs/2412.11892.

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 reconstruction. *CoRR*, abs/2010.02392, 2020. URL https://arxiv.org/abs/2010.02392.

Karl D.D. Willis, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2105–2114, June 2021.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. volume abs/2105.09492, 2021. URL https://arxiv.org/abs/2105.09492.

Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, Shengping Zhang, and Xiaojun Tong. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. *CoRR*, abs/1901.11153, 2019. URL http://arxiv.org/abs/1901.11153.

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. 2022. URL https://arxiv.org/abs/2207.04632.

Bingchen Yang, Haiyong Jiang, Hao Pan, Guosheng Lin, Jun Xiao, and Peter Wonka. PS-CAD: Local geometry guidance via prompting and selection for CAD reconstruction. *ACM Transactions on Graphics*, 44(3):1–21, 2025.

Xiaolong Yin, Xingyu Lu, Jiahang Shen, Jingzhe Ni, Hailong Li, Ruofeng Tong, Min Tang, and Peng Du. RLCAD: Reinforcement learning training gym for revolution involved CAD command sequence generation. *Computer-Aided Design*, 192:104027, 2026.

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, June 2022.

Shuming Zhang, Zhidong Guan, Hao Jiang, Tao Ning, Xiaodong Wang, and Pingan Tan. Brep2Seq: A dataset and hierarchical deep learning network for reconstruction and generation of computer-aided design models. *Journal of Computational Design and Engineering*, 11(1):110–134, 2024.

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*, pp. 1804–1812, 2023.

Xiangyu Zhu, Dong Du, Weikai Chen, Zhiyou Zhao, Yinyu Nie, and Xiaoguang Han. Nerve: Neural volumetric edges for parametric curve extraction from point cloud. 2023. URL https://arxiv.org/abs/2303.16465.



Figure 6: Representative CAD models from WHUCAD used as source data for constructing the ED2CAD Dataset, including examples with advanced modeling operations such as chamfer, fillet, and mirror.

A Details of engineering-drawing construction

A.1 Additional dataset examples

Additional examples illustrating the construction and visual characteristics of the ED2CAD Dataset are provided in Figs. 6 and 7. Figure 6 shows representative source CAD models from WHUCAD, including models with advanced modeling operations, while Fig. 7 presents the corresponding style of dimension-annotated engineering drawings generated by our pipeline, with both visualization and training renderings.

A.2 Source audit, sample pairing, and dataset splits

WHUCAD contains 146,331 source samples. We retain samples with a CAD sequence, matching B-rep geometry, complete PNG/DXF/SVG drawings, and a valid point cloud for geometric supervision. After filtering, 133,656 samples remain. We preserve their original training, validation, and test assignments, as summarized in Table 6.

To ensure consistent pairing throughout this filtering process, we link all data associated with a sample through a shared identifier. Each identifier combines a group ID and a model ID, such as `0000/00000000`, linking the H5 command sequence, source STEP/B-rep geometry, three-view drawings, annotations, and cached point cloud. The H5 audit records the sequence length, the $T \times 33$ command array, command frequencies, and the presence of EOS, while flagging non-integer values and sequences exceeding 100 rows. Geometry matching first uses the full identifier and falls back to the model basename when necessary. The fallback selects the first indexed match and therefore requires unique basenames for unambiguous pairing; it does not verify geometric equivalence. Unmatched records are retained for diagnosis but excluded from subsequent drawing generation.

Table 6: Dataset sizes after filtering.

Split	Number of samples
Training	120,274
Validation	6,703
Test	6,679
Total	133,656

A.3 Orthographic coordinates and visibility

The generation backend reads source geometry through FreeCAD and uses OpenCascade hidden-line removal (HLR). The drawing coordinates are $(x, -z)$ for the front view, $(x, -y)$ for the top view, and $(y, -z)$ for the right view. The corresponding projector directions are $+Y$, $+Z$, and $+X$.

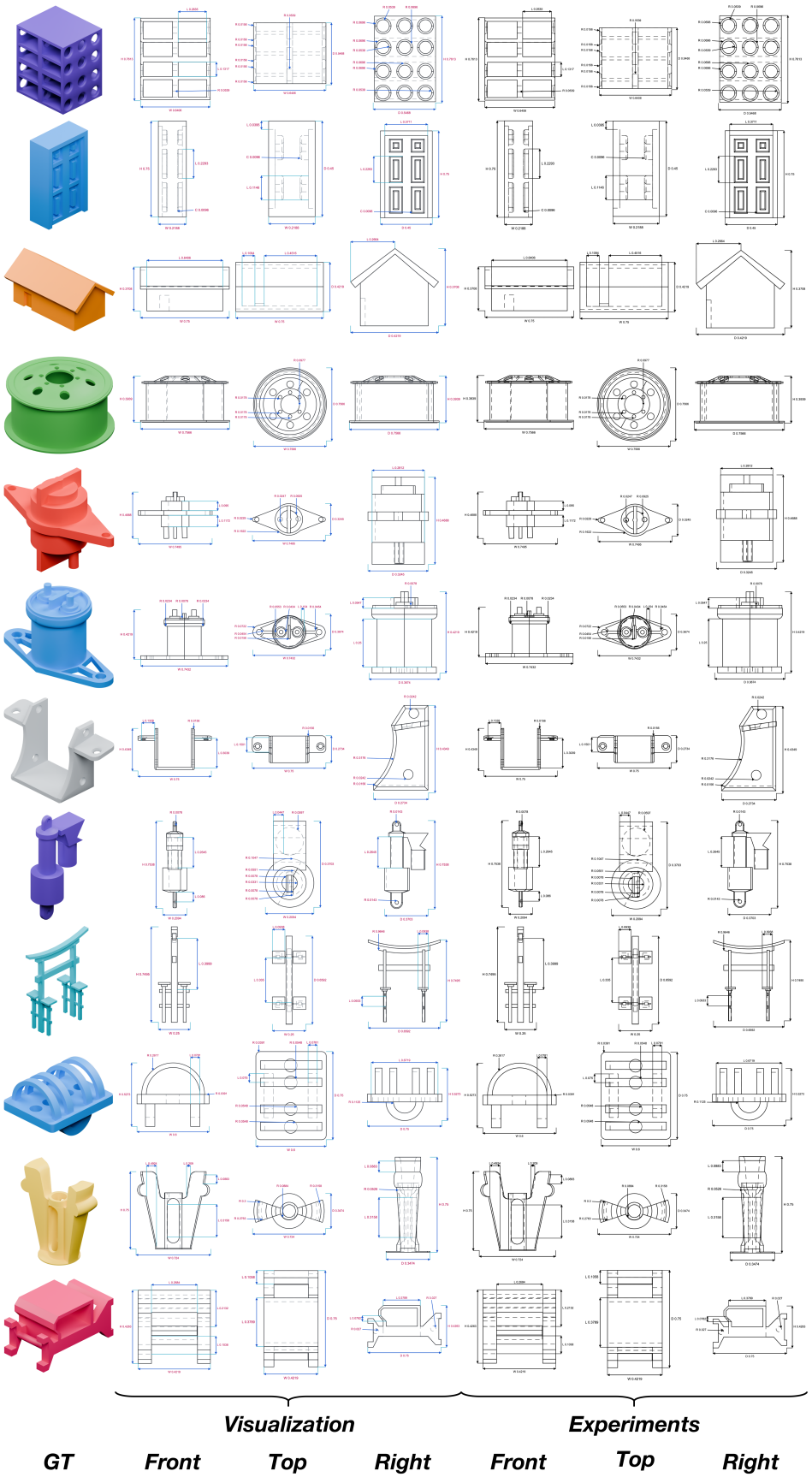

Figure 7: Additional examples from the ED2CAD Dataset. Each sample is shown in two rendering styles: the visualization version uses colored dimension annotations to improve visual distinction, while the training version uses black annotations and corresponds to the engineering drawings used for model training.

The sign changes establish a downward vertical axis for SVG drawing coordinates. Direct B-rep projection preserves source numerical units; the compatibility argument named `--scale` does not rescale this geometry. The DXF writer sets the unit declaration to unitless, so these files alone do not establish a millimetre convention.

HLR output is separated into visible and hidden polylines. When a view's HLR computation fails, direct projection is used and its visibility is marked `unknown`; these contours are drawn as solid lines rather than assigned an unsupported hidden/visible label. A fallback view is not automatically discarded. The exporter also records curve-discretization failures and can skip an individual edge while continuing the sample. Completeness therefore means that the expected files and metadata exist, not that every original edge was successfully discretized. View metadata stores visibility summaries, edge counts, skipped-edge counts, face attributes, and projection information.

The clean per-view SVG uses a bounding-box margin of 6% of its maximum span. Visibility is preserved through SVG attributes/groups and DXF layers `VISIBLE`, `HIDDEN`, and `UNKNOWN`. At composition time, hidden contours are drawn first, uncertain contours second, and visible contours last. Approximately coincident hidden contours covered by visible contours are removed. This drawing-order rule affects visual appearance and should not be interpreted as an additional geometric visibility test.

A.4 Dimension candidates and semantic cues

Let s denote the maximum width or height of the projected object bounding box. Overall dimensions are obtained from the B-rep bounding-box extents. A straight edge becomes a linear candidate only if both its spatial and projected lengths are at least $0.08s$ and $\|e_{\mathrm{proj}}\|/\|e\| \geq 0.96$. Under orthographic projection, this ratio corresponds to an angle of at most $\arccos(0.96) \approx 16.3^\circ$ from the projection plane. Coincident projected endpoints are deduplicated on a grid of size $\max(10^{-5}s, 10^{-9})$.

A circular candidate requires radius at least $0.012s$ and an absolute normal-to-view-direction dot product of at least 0.85. A closed circle is treated as a diameter only when both projected bounding-box extents agree with the measured diameter within 35%; otherwise, accepted circular edges produce radius labels. Arcs with radius at most $0.18s$ may provide fillet candidates.

The H5 sequence supplies offline cues for Fillet, Hole, and Chamfer candidates. For fillets, cylindrical faces are considered only when no Hole command is present, and faces with radius below $0.006s$ are rejected. For holes, view-aligned cylindrical faces with diameter at least $0.012s$ produce diameter candidates. These rules use command presence and geometric compatibility; they do not establish exact feature-history correspondence.

Chamfer candidates have length in $[0.01s, 0.18s]$, projected-to-spatial length ratio at least 0.7, and projected component ratio $|\Delta x|/|\Delta y| \in [0.45, 2.2]$. When this ratio lies in $[0.65, 1.55]$, the displayed value is $l/\sqrt{2}$; otherwise it is l. At most two chamfer candidates are retained per view.

Candidates are ranked according to the score defined in Eq. equation 1. The implementation uses the following type-specific rules:

Table 7: Type-specific dimension ranking rules.

Candidate family	Ranking rule
Overall extent	Mandatory; score 1000.
Linear edge	$520 + 130\min(l/s, 1.5)$; add 110 for horizontal or vertical orientation, subtract 100 otherwise, and subtract 120 if $l < 0.14s$.
Repeated linear evidence	Subtract 90 for repeated value/orientation and 140 for a nearby centre; the position grid is $0.06s$.
Closed-circle diameter	$880 + 50\min(d/s, 1)$; subtract 180 for repetition.
Radius candidate	Base score 850 or 820, plus $40\min(r/s, 1)$; subtract 180 for repetition.
Command-assisted features	Fillet score at least 900; hole diameter score 910; chamfer score $880 + 40(0.18s - l)/(0.18s)$.

Non-finite candidates are removed. Final deduplication uses dimension family, view, value rounded to four decimal places, and centre quantized on a $0.035s$ grid. Command-assisted candidates take

precedence over geometry-only duplicates. Mandatory overall extents are placed first, followed by the remaining candidates in descending score order.

A.5 Complexity budget and collision-aware placement

The view complexity score is the number of B-rep edges plus 1.5 times the number of arc or circle edges and 0.5 times the number of B-spline edges. An additional 5 is added when the view aspect ratio exceeds 4. Scores at most 24, at most 80, and above 80 define simple, medium, and complex views. The resulting annotation targets are respectively eight times a deterministic multiplier clipped to $[6, 10]$, 20% of the candidate count, and 27.5% of the candidate count. Targets are bounded by the available candidates and 12 annotations per view.

A *side* is the direction relative to the object contour, such as left, right, top, or bottom. Its rank r_{side} records the preference order of that direction. A *lane* is an outward offset level within a side, and r_{lane} records its order. Linear labels use six lanes with offsets $s(0.075 + 0.065k)$ for $k = 0, \ldots, 5$. Leader-based labels use four sides and ordered shifts $(0, 1, -1, 2, -2, 3, -3)$.

The layout candidates are evaluated using the objective in Eq. equation 2. The fixed coefficients 4 and 2 are heuristic weights: one side-rank increment is twice as costly as one lane-rank increment. They are implementation constants rather than learned parameters.

The text size is $0.035s$, the arrow size is $0.018s$, and the collision clearance is $\delta = 0.012s$. The outside-contour test is applied to the text box. Collision checking uses the union of the text box, extension lines, leader lines, and arrows; line regions receive an additional $0.004s$ padding. A layout is feasible when its text box is outside the contour, its complete annotation region does not intersect previously occupied regions expanded by δ, and its leader or extension lines can reach the associated feature.

Optional annotations without a feasible layout are discarded. Mandatory overall dimensions are retained by selecting the available layout with the smallest overlap and recording a collision-override flag. Thus the procedure reduces overlap.

A.6 Composition, export variants, and generation procedure

The final canvas is 1600×1100. The front, top, and right views occupy 420×420 panels beginning at $(80, 80)$, $(600, 80)$, and $(1120, 80)$, with 24-pixel internal padding. Each view is fitted independently with a uniform scale. DXF export uses the vertical-coordinate conversion $y_{\text{DXF}} = 1100 - y_{\text{canvas}}$.

The selected layout is reused for monochrome PNG, SVG, and DXF training files, as well as colored audit, annotation-free, and dimension-only variants. The training loader reads the selected drawing files and the target CAD sequence; the annotation metadata is used during data generation rather than as a separate decoder input. The complete executable-style generation procedure is provided in Listing 1. Candidate generation and semantic filtering are defined in Appendix A.4, while budget allocation and collision-aware placement are described in Appendix A.5.

Listing 1: Dimension-aware drawing generation

```
def generate_drawing(sample_id: str,
                     h5_sequence: H5Sequence,
                     brep: BRepGeometry) -> DrawingPackage:
    # Initialize output containers.
    annotations = []
    views = ["front", "top", "right"]

    for view in views:
        # Project the B-rep and use direct projection if HLR fails.
        projected = project_with_hlr(brep, view)

        if projected is None:
            projected = direct_projection(brep, view)
            record_visibility_fallback(sample_id, view)

        # Generate candidates using the rules in Appendix A.3.
```

```
        candidates = generate_candidates(projected, h5_sequence)

        # Remove invalid values and duplicate candidates.
        candidates = remove_invalid_candidates(candidates)
        candidates = remove_duplicate_candidates(candidates)

        # Rank candidates and determine the view-specific budget.
        candidates = rank_candidates(candidates)
        budget = allocate_budget(projected, candidates)

        # Retain the highest-ranked candidates within the budget.
        selected = select_candidates(candidates, budget)

        for candidate in selected:
            # Enumerate preferred sides and outward lanes.
            layouts = enumerate_layouts(candidate, projected)

            # Reject contour and previously occupied-region collisions.
            layout = choose_feasible_layout(
                layouts,
                occupied_regions=annotations,
                contour=projected.contour
            )

            if layout is not None:
                annotations.append(layout)

            elif candidate.is_mandatory:
                # Mandatory dimensions use the least-overlapping fallback
.
                layout = least_overlapping_layout(
                    layouts,
                    occupied_regions=annotations
                )
                layout.collision_override = True
                annotations.append(layout)

            else:
                # Optional dimensions are removed if no valid layout
    exists.
                continue

    # Compose the three views using the selected annotation layouts.
    drawing = compose_three_views(
        brep=brep,
        views=views,
        annotations=annotations
    )

    # Export the variants used for training and inspection.
    export_png(drawing, variant="monochrome")
    export_svg(drawing, variant="monochrome")
    export_dxf(drawing, variant="monochrome")
    export_png(drawing, variant="color_audit")
    export_png(drawing, variant="annotation_free")
    export_dxf(drawing, variant="dimension_only")

    # Store metadata and update the training manifest.
    save_metadata(sample_id, annotations)
    update_manifest(sample_id, drawing, h5_sequence)

    return drawing
```

B INPUT REPRESENTATIONS AND CAD TARGETS

B.1 RASTER AND VECTOR PREPROCESSING

The PNG branch converts each drawing to RGB, resizes it to 224×224, and scales pixel values to $[0, 1]$. The raster encoder then forms 16×16 patches, producing 196 image tokens. The inspected preprocessing does not use geometric augmentation or dataset mean/variance normalization.

DXF and SVG drawings are converted into at most 2048 entity tokens. Each token has 128 features describing the entity type, layer and style information, coordinates, bounding-box statistics, dimension-text attributes, visibility, and presence. Coordinates are normalized by the retained drawing bounding box; streams shorter than 2048 entities are zero-padded and longer streams are truncated after entity filtering and priority sorting. A disabled modality is omitted from the input set rather than replaced by a learned token.

B.2 CAD TARGETS AND MASKING

Each target row contains one of 27 command classes and 32 argument slots. Each argument is represented by a categorical value in $\{0, \ldots, 255\}$, while the command-specific validity matrix determines which slots contribute to the argument loss. Sequences are limited to 100 rows. Stored sequences are padded with EOS rows and zero arguments when shorter than the limit; at export, inactive argument slots are written as -1.

Training uses a shifted ground-truth prefix beginning with SOL. The initial SOL row is consumed by the decoder but is not included as a predicted target. EOS terminates an exported sequence, and inference additionally stops when all items in a batch predict EOS at the same decoding step. These prefix and stopping conventions are used by the decoder details in Appendix C.3.

C NETWORK ARCHITECTURE AND GENERATION

C.1 MODULE CONFIGURATIONS

All Transformer modules use width $D = 256$, eight attention heads, GELU activations, dropout 0.1 for encoders, 0.15 for fusion, and 0.2 for the decoder, pre-normalization, and a final LayerNorm. The reference configuration is:

Module	Layers	Width	Heads	FFN width
PNG encoder	6	256	8	1024
DXF encoder	4	256	8	1024
SVG encoder	4	256	8	1024
Fusion Transformer	4	256	8	1024
CAD decoder	6	256	8	1024

Table 8: Reference Transformer configuration used by ED2CAD.

C.2 MODALITY ENCODERS AND FUSION

The PNG branch applies a convolutional patch projection followed by the six-layer raster Transformer. DXF and SVG use separate four-layer vector Transformers. Their 128-dimensional input tokens are projected to 256 dimensions and supplemented with entity-type, layer, position, and modality embeddings. The two vector branches have the same dimensions but do not share parameters.

The enabled streams are fused by the representation in the main paper, Eq. equation 7. Tokens are concatenated in PNG–DXF–SVG order after adding fusion-stage modality embeddings, and a learned global token is prepended. The complete normalized output $H_i = [g_i; R_i]$ is supplied to the decoder as cross-attention memory; only g_i is sent to the geometry proxy. With all modalities enabled, the memory contains $1 + 196 + 2048 + 2048 = 4293$ tokens.

C.3 Decoder embedding and inference

The decoder uses causal self-attention over the shifted CAD prefix and cross-attention to the complete fusion memory. The previous command is embedded with all 32 previous argument values and a learned decoder position. Argument embeddings are shared across slots and averaged before projection; inactive slots represented by zero therefore remain part of this prefix embedding.

The command head produces 27 logits. A second linear head produces 32×256 argument logits, reshaped into one 256-way distribution per argument slot. Inference uses argmax for both commands and arguments, without beam search or CAD-kernel feedback. The sequence is trimmed at its first EOS before H5 export. The implementation details of prefix alignment and output heads are summarized here so that the main-paper autoregressive factorization remains the only decoder equation.

D Geometry supervision and training

D.1 Point-cloud targets and geometry proxy

Target clouds are sampled from a FreeCAD tessellation with deflection 0.1. Triangles are sampled in proportion to surface area and points are sampled uniformly within each selected triangle. Degenerate meshes fall back to vertex sampling, and a trimesh sampler is used if the FreeCAD route fails. Each cloud contains 2048 points, is centred at the midpoint of its axis-aligned bounding box, and is divided by the diagonal length of that box. The normalized cloud is cached as float32 under an ID-derived filename.

The geometry proxy applies LayerNorm and a three-layer MLP with dimensions $256 \to 256 \to 256 \to 6144$, followed by tanh and a reshape to 2048 three-dimensional points. It branches from the global fusion feature and therefore updates the proxy, fusion Transformer, and enabled input encoders through the geometry loss. It has no direct gradient path to the CAD decoder.

D.2 Loss reductions and training update

Command loss is mean cross-entropy over valid command positions. Argument loss is mean cross-entropy over active argument slots selected by the target command mask; it does not require the predicted command to be correct. The EOS term is computed on valid target EOS positions. The total objective and its weights are given in the main paper by Eq. equation 15; the geometry term is averaged only over examples with valid point-cloud targets. The reference sequence weights are $\lambda_c = 1.0$, $\lambda_a = 1.25$, and $\lambda_e = 0.2$.

One training update encodes the configured modalities, supplies the shifted ground-truth prefix, computes command/argument/EOS losses, optionally computes the proxy loss, and updates the model with mixed-precision AdamW. Gradients are unscaled and clipped before the optimizer step.

D.3 Optimization settings

Setting	Value
Hardware	Two NVIDIA GeForce RTX 4090 GPUs
Training budget	200 epochs
Optimizer	AdamW
Learning rate	10^{-4}, with warmup from 10^{-6}
Weight decay	2×10^{-2}
Schedule	Cosine decay to 10^{-6}
Dropout	0.1 (encoders), 0.15 (fusion), 0.2 (decoder)
Gradient clipping	Norm threshold 1.0
Precision	Automatic mixed precision with gradient scaling
Checkpoint	Lowest total validation loss

Table 9: Optimization settings corresponding to the experimental protocol.

E EVALUATION CONVENTIONS AND EXPERIMENTAL CONTROLS

E.1 METRICS AND EXECUTION LEVELS

Command and parameter accuracy are accumulated over valid token positions under teacher-forced prefixes:

$$A_c = \frac{1}{|\Omega_c|} \sum_{(i,t)\in\Omega_c} \mathbb{I}[\widehat{c}_{i,t} = c_{i,t}], \qquad A_p = \frac{1}{K} \sum_{(i,t,j)\in\Omega_a} \mathbb{I}[\widehat{c}_{i,t} = c_{i,t}]\mathbb{I}[\widehat{a}_{i,t,j} = a_{i,t,j}], \tag{17}$$

where K counts active argument positions attached to correctly predicted commands. This teacher-forced convention should be distinguished from free-running sequence accuracy, which uses previously generated tokens. Likewise, point-cloud Chamfer from the proxy is different from a distance computed after executing the predicted CAD program.

E.2 EXPERIMENTAL PROTOCOLS

For the modality ablation, all seven nonempty PNG/DXF/SVG subsets use the same sample IDs, split, preprocessing limits, optimizer, and geometry supervision. For the annotation ablation, annotated and unannotated drawings are paired by sample ID while the model and split remain fixed. The baseline comparison uses annotated SVG under a matched single-modality protocol; adapted baselines are retrained with the WHUCAD representation and the same CAD kernel and metrics.

For component ablations, direct fusion replaces the fusion Transformer with normalized token concatenation and mean pooling, the fusion-stage modality embedding is removed in the corresponding variant, and the geometry proxy and its loss are removed in the no-proxy variant. For DeepCAD, models are retrained with its native six-command, 16-argument, 60-step representation and official 90%/5%/5% split. The numerical results and all experimental figures remain in the main paper.