DRAWING-TO-CAD EDITING: AN END-TO-END BENCHMARK AND A REINFORCEMENT LEARNING APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

Parametric computer-aided design (CAD) editing typically presumes access to underlying source programs. In practical engineering workflows, however, designers frequently need to modify components starting solely from dimensioned 2D engineering drawings and natural language instructions, without access to original CAD models. Existing benchmarks evaluate either drawing-to-CAD reconstruction or text-guided editing from provided source CAD models, leaving this end-to-end task unaddressed. Learning this task poses a fundamental coordination dilemma: two-stage cascaded pipelines suffer from severe error accumulation at inference, direct generation forfeits explicit intermediate geometric feedback, and joint learning with undifferentiated feedback can obscure component-specific quality differences and assign reconstruction credit to editing actions that cannot affect source quality. To tackle these challenges, we introduce CAD-D2E, the first benchmark for drawing-to-CAD editing. CAD-D2E pairs dimensioned orthographic views and textual instructions with executable source and edited programs across 15 edit subtypes and five part families, supporting end-to-end geometric evaluation and intermediate verification. We further propose JORE (Joint Optimization of Reconstruction and Editing), a framework that generates both programs within a unified autoregressive policy. JORE uses CARPO (Causal Advantage Routing for Policy Optimization) to route component-relative advantages according to autoregressive stage dependencies. On CAD-D2E, JORE achieves state-of-the-art performance, surpassing cascaded baselines and standard GRPO by substantial margins, with 78.50% IoU and 87.63% program executability. These results demonstrate the vital role of dependency-aware intermediate supervision in drawing-to-CAD editing.

1 INTRODUCTION

Generative models increasingly support computer-aided design (CAD) (Doris et al., 2025; Guan et al., 2025; Kolodiazhnyi et al., 2026). Existing approaches reconstruct parametric models from dimensioned drawings (Niu et al., 2025; Joglekar et al., 2026) or edit a supplied CAD representation using textual instructions (Yuan et al., 2025; Hu et al., 2026). Yet drawings may be available without their corresponding editable CAD models (Harish & Prasad, 2021). Drawing-to-CAD editing therefore requires recovering the source geometry and applying the requested modification from the views and instruction alone.

Related CAD benchmarks evaluate reconstruction from engineering drawings (Niu et al., 2025; Joglekar et al., 2026) or instruction-guided editing from provided source CAD models (Yuan et al., 2025; Hu et al., 2026), rather than the complete drawing-to-CAD editing workflow (Figure 1(a)). Beyond evaluation, reconstruction errors can compromise the final edit, but accurate reconstruction alone does not guarantee editing success. In a separately trained pipeline, an editor trained on ground-truth source programs must handle predicted sources at inference. Direct generation avoids this interface but provides no explicit source-program supervision (Figure 1(b)). Joint generation enables supervision of both programs, but leaves open how to combine intermediate and final feedback and assign it across stages.

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.

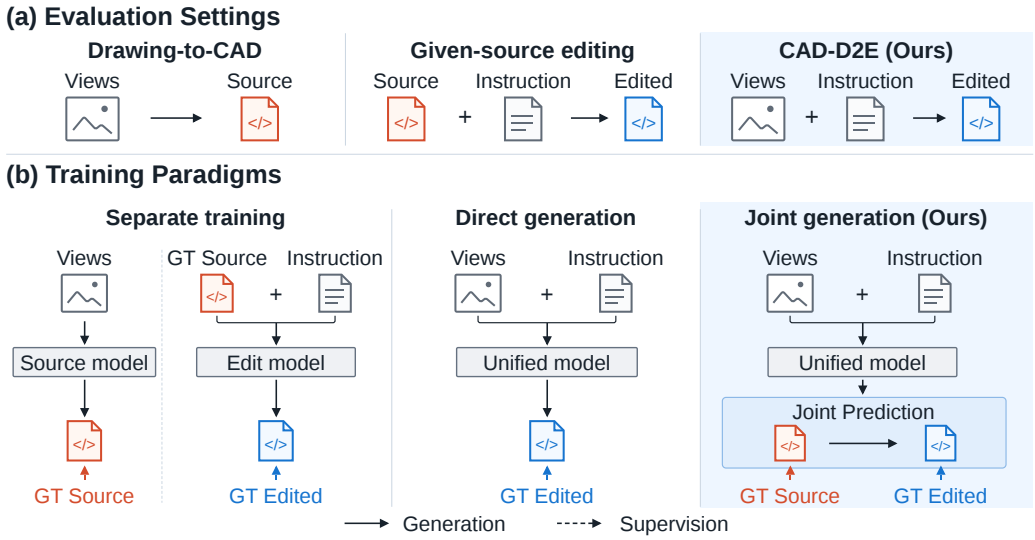

Figure 1: **Comparison of Evaluation Settings and Training Paradigms.** **(a)** Existing settings evaluate reconstruction from drawings or editing a provided source program; CAD-D2E evaluates editing from drawings and instructions without source CAD input. **(b)** Separate training uses independently trained reconstruction and editing models; direct generation supervises only the edited program; joint generation supervises both source and edited programs within a single autoregressive response.

To support end-to-end evaluation, we introduce **CAD-D2E**, a benchmark for Drawing-to-CAD Editing, in which models generate edited CAD programs from three dimensioned orthographic views and a textual editing instruction without source CAD input (Figure 1(a)). Models are evaluated only on the final edited program. CAD-D2E covers 15 edit subtypes and pairs each input with executable source and edited programs, with execution checks and human review assessing consistency among the drawings, instructions, and program pairs. The source annotations additionally support intermediate-state supervision without prescribing a particular generation scheme.

To coordinate reconstruction and editing, we propose **JORE** (Joint Optimization of Reconstruction and Editing), a framework that coordinates intermediate source supervision with final editing feedback. A single model generates source and edited programs in one response from the same views and instruction; during reinforcement learning, editing conditions on the model's own reconstructed source (Figure 1(b)). Joint supervision must account for which stages can affect each outcome. Reconstruction can influence the final edit, whereas editing cannot change the source program already generated. We therefore train the joint policy with CARPO (Causal Advantage Routing for Policy Optimization), which routes component-relative advantages according to these stage dependencies. To limit masking caused by differences in reward dispersion, CARPO computes relative feedback separately for each component before aggregation. It restricts direct source-quality credit to reconstruction while retaining final-edit feedback across both stages.

Experiments on CAD-D2E demonstrate that JORE outperforms the evaluated existing methods and alternative generation baselines in final-program quality. Controlled ablations show that component normalization and stage-aware credit assignment are complementary within CARPO. Further credit-routing comparisons favor retaining final-edit feedback to reconstruction while restricting direct source-quality credit to reconstruction, rather than isolating stage rewards or sharing all rewards across both stages.

Our contributions are threefold:

- We introduce CAD-D2E, a benchmark for Drawing-to-CAD Editing, in which models generate edited CAD programs from three dimensioned orthographic views and a textual editing instruction without source CAD input. Paired executable source and edited programs support end-to-end evaluation and intermediate supervision.

- We propose JORE, a framework that jointly generates source and edited programs within a single autoregressive policy. JORE is trained with CARPO, which routes component-relative advantages according to autoregressive stage dependencies to coordinate reconstruction and editing.
- Experiments on CAD-D2E demonstrate improved final editing over the evaluated existing methods and generation baselines. Controlled studies further support the value of guiding reconstruction with final-edit feedback, beyond supervising source quality alone.

2 Related Work

2.1 CAD Code Generation and Editing

Executable CAD representations enable language and vision–language models to generate editable geometry from different input modalities. CAD-Coder models generate CadQuery scripts from images (Doris et al., 2025) or textual descriptions (Guan et al., 2025), while cadrille (Kolodiazhnyi et al., 2026) supports images, point clouds, and text within a unified model. For engineering drawings, CME-CAD (Niu et al., 2025) reconstructs CAD programs from dimensioned orthographic views and introduces CADExpert, pairing drawings with executable programs and reasoning annotations. Ortho2CAD (Joglekar et al., 2026) also studies orthographic drawing-to-code reconstruction, including iterative refinement through execution and rendered-view feedback.

Instruction-guided CAD editing instead modifies an existing design according to a textual request. CAD-Editor (Yuan et al., 2025) locates and infills regions of a supplied CAD representation, with training data synthesized as source–instruction–target triplets. Beyond single-pass editing, IterCAD (Hu et al., 2026) formulates generation and editing as multi-turn interaction with an executable CAD sandbox. It uses execution and visual feedback to refine programs across drawing-to-code, text-to-code, and interactive editing tasks.

2.2 Reinforcement Learning with Verifiable Rewards

Reinforcement learning with verifiable rewards (RLVR) optimizes language models using automatically checked outcomes, such as answer correctness and code execution. GRPO (Shao et al., 2024) estimates advantages by comparing responses sampled for the same input, without a learned critic. DeepSeek-R1 (DeepSeek-AI, 2025) demonstrates the effectiveness of large-scale reinforcement learning with rule-based feedback for reasoning. Subsequent work improves exploration and policy optimization. DAPO (Yu et al., 2025) introduces asymmetric clipping, dynamic sampling, token-level loss aggregation, and overlong reward shaping to improve training stability. Dr. GRPO (Liu et al., 2025) removes response-length and reward-standard-deviation normalization to address length and question-difficulty biases. GSPO (Zheng et al., 2025) instead computes importance ratios from length-normalized sequence likelihoods and applies sequence-level clipping.

Other approaches refine the learning signals derived from rewards. PRIME (Cui et al., 2025) learns implicit process rewards online from policy rollouts and outcome labels, without requiring annotated intermediate steps. VinePPO (Kazemnejad et al., 2025) estimates intermediate-state values through Monte Carlo continuations rather than a learned critic, providing step-level advantages. Segment Policy Optimization (Guo et al., 2025) estimates advantages over contiguous segments, balancing credit granularity with the cost of Monte Carlo value estimation. For multiple reward objectives, GDPO (Liu et al., 2026) normalizes each component separately within a rollout group before weighted aggregation and batch-level normalization, preserving component-specific relative differences that can be obscured by normalizing an aggregate reward.

3 CAD-D2E Benchmark

3.1 Task Definition

Given three dimensioned orthographic views $\mathcal{V}$ and an editing instruction $\mathcal{T}$, a model f_θ must generate a complete CAD program that applies the requested edit while preserving unaffected geometry,

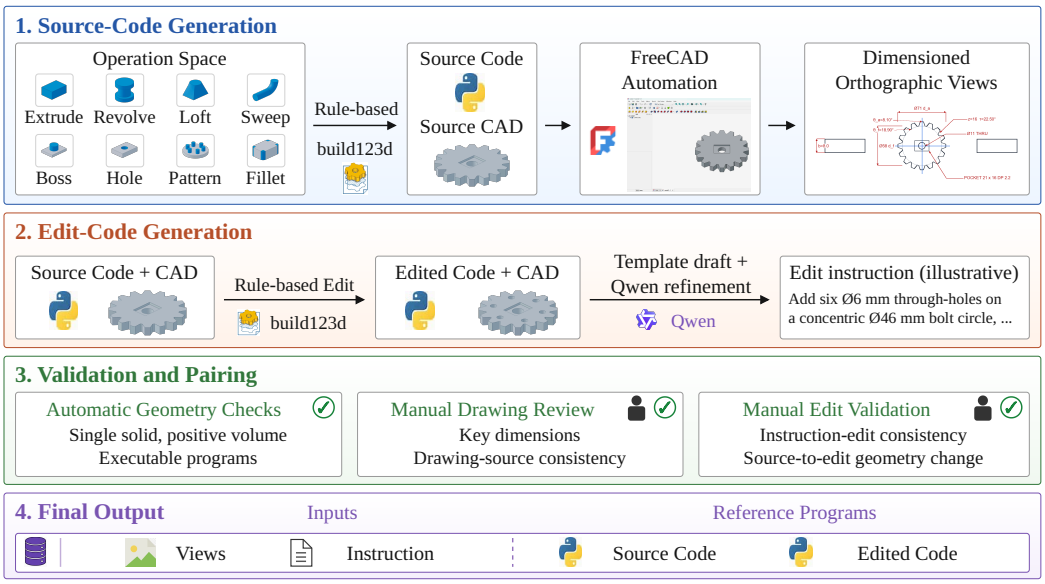

Figure 2: CAD-D2E Data Construction Pipeline.

Subset	Samples	Sources	Edit composition (%)			
Cold-start	3,000	1,028	29.2	8.9	33.3	28.6
RLVR Train	5,932	3,161	27.3	14.5	31.5	26.8
Evaluation	2,692	940	27.8	14.4	30.8	27.0

Feature addition | Feature removal | Geometry modification | Pattern creation

Figure 3: **CAD-D2E subset sizes and edit distributions.** Counts refer to samples and unique sources. Training pools overlap; evaluation is source-disjoint.

without source CAD input:

$$\hat{\mathcal{C}}^1 = f_\theta(\mathcal{V}, \mathcal{T}). \tag{1}$$

Each example $(\mathcal{V}, \mathcal{T}, \mathcal{C}^0, \mathcal{C}^1)$ includes executable `build123d` reference programs for the source part ($\mathcal{C}^0$) and target edited part ($\mathcal{C}^1$). We evaluate the final edited program by its executability and geometric agreement with the reference target.

3.2 Dataset Construction

Figure 2 illustrates the CAD-D2E construction pipeline. We sample sequences of CAD operations, such as extrusion, hole creation, and filleting, together with their geometric parameters. A rule-based compiler converts each specification into a source program $\mathcal{C}^0$; the same specification supplies the geometry and dimensions for its corresponding views $\mathcal{V}$. We then select an edit compatible with the source geometry and existing features. The selected operation and parameters determine both the edited program $\mathcal{C}^1$ and a template-based draft instruction, which Qwen refines individually to produce $\mathcal{T}$. Multiple edits of the same source yield separate examples, each with its own instruction and target program. The resulting pairs undergo automatic checks for program execution, geometric validity, and a nontrivial source-to-target change. Human reviewers assess dimension completeness, agreement between drawings and source geometry, and consistency between the instruction and edited geometry. For the cold-start subset, validated pairs are further augmented with model-generated reasoning for source reconstruction and editing to form supervised generation trajectories. Annotation and assembly details are provided in Appendix C.1.

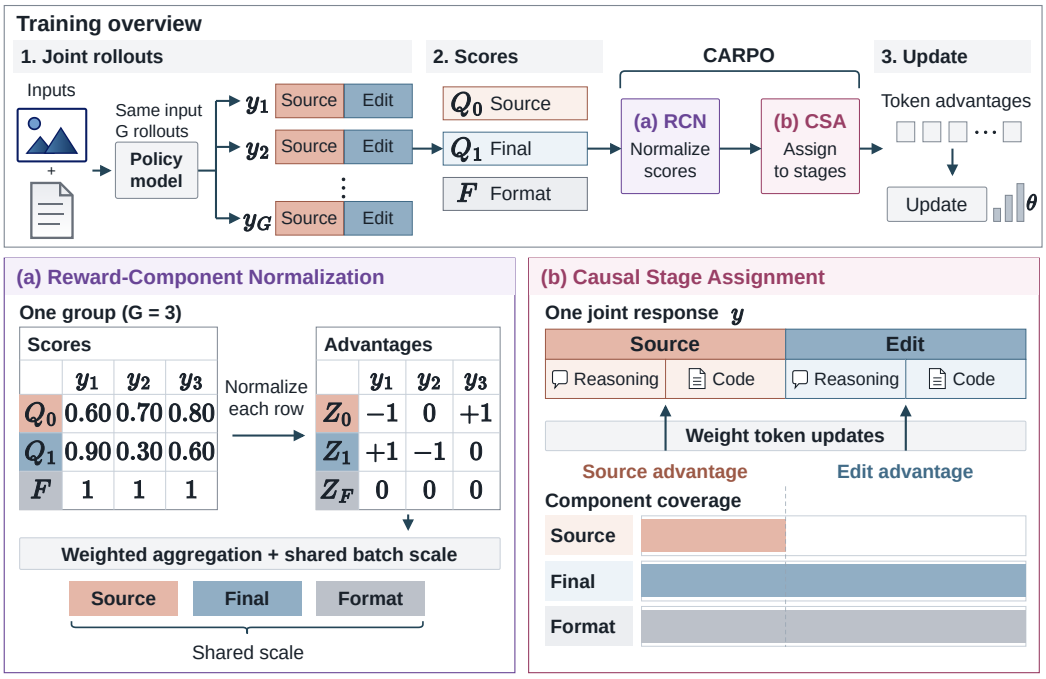

Figure 4: **Overview of JORE.** A single policy generates source and edited CAD programs in one autoregressive response from dimensioned views and an editing instruction. CARPO trains this joint policy using intermediate and final feedback. The lower panels illustrate how component-relative advantages are computed (a) and assigned to response tokens (b): source-quality credit applies only to Source, while final-edit and format credit cover both stages.

3.3 Dataset Statistics

CAD-D2E comprises cold-start and RLVR training pools and an evaluation set. All three subsets cover 15 edit subtypes, organized into four groups: feature addition, feature removal, geometry modification, and pattern creation. Figure 3 summarizes their sizes and edit distributions. The training pools overlap in 2,904 editing examples and together contain 6,028 unique examples from 3,161 source parts, whereas the evaluation set shares neither source parts nor editing examples with either pool. The detailed edit taxonomy is provided in Appendix C.1.

4 Method

Separately trained reconstruction and editing models retain source supervision but introduce a mismatch between the source programs seen during editor training and inference, whereas direct generation omits explicit source-program supervision. JORE combines supervision of both programs within a single autoregressive policy, allowing editing to learn from model-generated sources during reinforcement learning. Joint optimization must also account for the different dependencies of intermediate and final feedback: editing cannot alter the source already generated, whereas reconstruction can affect the final edit. CARPO routes component-relative advantages according to these dependencies (Figure 4).

4.1 Joint Generation and Scoring

For each input $x = (\mathcal{V}, \mathcal{T})$, we sample G complete responses $y_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$. Each response comprises Source and Edit stages:

$$y = \underbrace{r^0 \oplus \hat{\mathcal{C}}^0}_{y_{\text{src}}} \oplus \underbrace{r^1 \oplus \hat{\mathcal{C}}^1}_{y_{\text{edit}}}. \tag{2}$$

Here, $\oplus$ denotes concatenation, and each stage contains reasoning r^k and predicted code $\hat{\mathcal{C}}^k$. The reasoning and code of each stage occupy separate tagged blocks. A single policy generates both stages autoregressively from the shared views and instruction, with Edit conditioned on the complete model-generated Source stage.

Each response receives separate source-quality, final-edit-quality, and format scores $(Q_{0,i}, Q_{1,i}, F_i)$. We independently execute the two predicted programs and define $Q_{k,i}$ as the volumetric IoU between the predicted and reference shapes for $k \in \{0, 1\}$. An unscoreable program receives zero for its geometry component. The binary score F_i indicates adherence to the four-block response protocol.

4.2 Causal Advantage Routing for Policy Optimization

The reward components of a joint response depend on different generation stages. Source quality is determined by reconstruction, whereas final-edit quality can be influenced by both reconstruction and editing. CARPO retains component-specific feedback and assigns its direct credit according to these stage dependencies. To account for differences in within-group reward dispersion, we compute component-relative advantages using reward-component normalization (RCN), following GDPO (Liu et al., 2026) (Figure 4(a)). With component weights w_k and $Q_{F,i} = F_i$, we obtain

$$Z_{k,i} = \frac{Q_{k,i} - \mu_k}{\sigma_k + \varepsilon}, \qquad A_i^{\mathrm{rcn}} = \frac{\widetilde{A}_i - \mu_{\widetilde{A}}}{\sigma_{\widetilde{A}} + \varepsilon}. \tag{3}$$

Here, $k \in \{0, 1, F\}$ and $\widetilde{A}_i = \sum_k w_k Z_{k,i}$. The component statistics μ_k and σ_k are computed within each input's rollout group, while $\mu_{\widetilde{A}}$ and $\sigma_{\widetilde{A}}$ are computed over all responses across inputs in generation batch $\mathcal{B}$. Each pair denotes the corresponding mean and standard deviation; ε is a stabilizer.

For fixed input x and a completed Source stage y_{src}, the reward $Q_0 = q_0(x, y_{\mathrm{src}})$ is constant across editing continuations. The standard score-function identity (Williams, 1992) gives

$$\mathbb{E}_{y_{\mathrm{edit}} \sim \pi_\theta(\cdot | x, y_{\mathrm{src}})} \left[Q_0 \nabla_\theta \log \pi_\theta(y_{\mathrm{edit}} \mid x, y_{\mathrm{src}}) \right] = 0. \tag{4}$$

This zero expected contribution of the raw source reward motivates restricting its direct credit to Source (Appendix A.1). In contrast, Source can influence the expected final-edit reward through its effect on the Edit continuation, so final-edit credit covers both stages. Format credit also covers both stages because protocol validity depends on the complete response.

CARPO applies this dependency-based coverage to the normalized component advantages through causal stage assignment (CSA; Figure 4(b)). For response i and token t, let $m_{i,t} = 1$ for Source and 0 for Edit, covering both reasoning and code within each stage. The token-level advantage is

$$A_{i,t} = \underbrace{\frac{w_1 Z_{1,i} + w_F Z_{F,i} - \mu_{\widetilde{A}}}{\sigma_{\widetilde{A}} + \varepsilon}}_{\text{both stages}} + \underbrace{m_{i,t} \frac{w_0 Z_{0,i}}{\sigma_{\widetilde{A}} + \varepsilon}}_{\text{Source stage}}. \tag{5}$$

Both stages share the batch center and scale without further normalization. Source tokens receive A_i^{rcn}; Edit tokens receive direct credit only from the final-edit and format components.

4.3 Policy Optimization

CARPO uses the stage advantages in Equation 5 to weight token updates under the clipped GRPO objective (Shao et al., 2024). For response y_i given x_i, the token-level importance ratio is

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x_i, y_{i,<t})}{\pi_{\theta_{\mathrm{old}}}(y_{i,t} \mid x_i, y_{i,<t})}, \tag{6}$$

where $\pi_{\theta_{\mathrm{old}}}$ is the rollout policy. With $\bar{\rho}_{i,t} = \mathrm{clip}(\rho_{i,t}, 1 - \epsilon, 1 + \epsilon)$ and clipping threshold ϵ, we minimize

$$\mathcal{L}_{\mathrm{CARPO}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \left[-\min\left(\rho_{i,t} A_{i,t}, \bar{\rho}_{i,t} A_{i,t} \right) + \beta \mathcal{K}_{i,t} \right]. \tag{7}$$

Here, $\mathcal{K}_{i,t}$ is the sampled KL penalty to the frozen reference policy π_{ref}, with weight β. The loss averages over valid response tokens within each completion, then over responses in $\mathcal{B}$; $|y_i|$ excludes prompt and padding tokens.

Table 1: End-to-end CAD editing results on CAD-D2E. All metrics evaluate the final edited program. General-purpose VLMs receive no CAD-D2E fine-tuning. Cascades use predicted source programs; + denotes reconstruction followed by editing and * our implementations. CD denotes normalized Chamfer distance. Best results are bold.

System	IoU (%) ↑	Mean CD ↓	Med CD ↓	Exec. (%) ↑
General-purpose VLMs				
Qwen3.8-Max (Qwen Team, 2026b)	64.46	0.31410	0.02582	71.00
Kimi K3 (Kimi Team, 2026)	63.41	0.32370	0.02613	70.00
Cascaded reconstruction and editing				
Source-GRPO + Edit-GRPO	57.62	0.32381	0.01955	68.95
Ortho2CAD* (Joglekar et al., 2026) + CAD-Editor* (Yuan et al., 2025)	68.21	0.31240	0.01854	72.25
CME-CAD* (Niu et al., 2025) + CAD-Editor* (Yuan et al., 2025)	68.01	0.31250	0.01837	71.84
IterCAD* (Hu et al., 2026)	75.01	0.14320	0.01621	87.00
Direct edited-program generation				
Direct-C1 SFT	73.61	0.17499	0.01657	83.80
Direct-C1 GRPO	76.22	0.14617	0.01645	86.74
Joint source–edit generation				
Joint SFT	74.13	0.18257	0.01623	82.99
Joint GRPO (Shao et al., 2024)	76.54	0.15617	0.01596	85.66
JORE (Ours)	**78.50**	**0.13677**	**0.01588**	**87.63**

4.4 Training and Inference

We first apply standard autoregressive supervised fine-tuning (SFT) to a pretrained vision-language model using the cold-start subset's complete Source–Edit responses (Appendix C.1). The resulting checkpoint initializes both the trainable policy and the frozen reference policy. We then optimize the policy on online joint rollouts from the RLVR subset using Equation 7. At inference, the model generates a complete joint response from the views and editing instruction, and we extract the edited program $\hat{\mathcal{C}}^1$ as the final output.

5 Experiments

5.1 Experimental Setup

Data and training. JORE and our controlled generation baselines use Qwen3.5-9B (Qwen Team, 2026a) with LoRA. Joint GRPO, JORE, and the component and routing ablations share the same cold-start checkpoint, RLVR training data, execution verifier, and 741-update budget. Full training configurations are provided in Appendix A.2.

Evaluation metrics. We evaluate final edited programs on all 2,692 test examples using volumetric IoU (%), mean and median normalized Chamfer distance (Mean CD and Med CD), and executability (Exec., %). IoU and CD measure geometric agreement with the reference shape; Exec. is the percentage of programs that execute successfully and produce valid shapes. Detailed evaluation procedures are provided in Appendix A.2.

5.2 Main Results

JORE achieves the best end-to-end editing performance among the evaluated systems, leading all four metrics in Table 1 with 78.50% IoU and 87.63% executability. Against IterCAD, the strongest external CAD system in this comparison, JORE improves IoU by 3.49 percentage points (pp), while also attaining lower Mean and Med CD and higher executability. JORE also outperforms Qwen3.8-

Table 2: Ablations of CARPO within JORE. RCN: reward-component normalization; CSA: causal stage assignment. Best results are bold.

Variant	RCN	CSA	IoU (%) ↑	Mean CD ↓	Med CD ↓	Exec. (%) ↑
GRPO	×	×	76.54	0.15617	0.01596	85.66
RCN-only	✓	×	76.94	0.15505	0.01597	85.77
CSA-only	×	✓	76.48	0.15629	0.01602	85.66
JORE (Ours)	✓	✓	**78.50**	**0.13677**	**0.01588**	**87.63**

Table 3: Source reconstruction and final editing performance on CAD-D2E. * denotes our implementations. Best results in each column are bold.

	Source $\hat{\mathcal{C}}^0$		Edited $\hat{\mathcal{C}}^1$	
System	IoU (%) ↑	Exec. (%) ↑	IoU (%) ↑	Exec. (%) ↑
Cascaded systems				
Ortho2CAD* + CAD-Editor*	80.79	84.26	68.21	72.25
CME-CAD* + CAD-Editor*	81.21	85.43	68.01	71.84
IterCAD*	**84.56**	**90.11**	75.01	87.00
Joint generation				
GRPO	82.80	87.13	76.54	85.66
RCN-only	82.95	87.22	76.94	85.77
JORE (Ours)	84.52	88.85	**78.50**	**87.63**

Max and Kimi K3, indicating that source-then-edit prompting of general-purpose VLMs alone does not match the learned joint policy in this setting. The controlled generation comparisons suggest the benefit of learning reconstruction and editing together: our Source-GRPO + Edit-GRPO cascade reaches only 57.62% IoU despite reinforcement learning in both stages. A likely limitation of this cascade is that the editor learns from GT source programs but must process predicted ones at test time, while reconstruction receives no final-edit feedback. Joint RLVR instead trains editing on model-generated sources and allows final-edit outcomes to guide reconstruction. JORE also surpasses Direct-C1 GRPO by 2.28 pp in IoU. This result is consistent with the benefit of intermediate supervision: an explicit source program provides verifiable feedback on the recovered initial geometry, while final-edit reward links this reconstruction to the requested modification; Direct-C1 receives geometric reward only for the final program. The gains over Joint GRPO further indicate that joint generation alone does not account for JORE's advantage. With the same joint outputs, reward components, and training settings, JORE improves IoU by 1.96 pp and executability by 1.97 pp. These results support CARPO's coordination of intermediate and final feedback within the joint policy.

5.3 Ablation Studies

Within CARPO, RCN and CSA exhibit complementary effects, with their combination outperforming either component alone. Under matched initialization, rewards, and training budgets in Table 2, removing RCN lowers IoU by 2.02 pp, while removing CSA lowers it by 1.56 pp; both removals also reduce executability and increase Mean and Med CD. Either component alone yields IoU close to GRPO. RCN-only preserves component-relative signals, but still uses source-quality feedback to update Edit tokens, which cannot affect the already generated source program. CSA-only restricts this credit to Source, but aggregation without component normalization can still allow larger reward fluctuations to obscure informative differences in other components. This pattern is consistent with their complementary roles: preserving component-relative feedback provides an effective basis for assigning direct credit according to stage dependencies.

Table 4: Cross-stage credit routing ablation of CARPO. ✓ and × indicate enabled and disabled paths. Within-stage and format credit remain fixed. Best results are bold.

	Cross-stage credit		Final editing			
Variant	$Q_0 \to$ Edit	$Q_1 \to$ Source	IoU (%) ↑	Mean CD ↓	Med CD ↓	Exec. (%) ↑
Reverse routing	✓	×	75.80	0.16252	0.01612	85.03
Stage-local	×	×	75.92	0.16325	0.01608	84.96
All-to-all	✓	✓	76.94	0.15505	0.01597	85.77
JORE (Ours)	×	✓	**78.50**	**0.13677**	**0.01588**	**87.63**

5.4 Source and Edit Quality

Better reconstruction does not necessarily translate into better final editing. In Table 3, IterCAD and JORE attain similar mean source IoU (84.56% and 84.52%), yet their final-program IoU differs substantially (75.01% and 78.50%). IterCAD also has higher source executability, although its edited-program executability is lower. Source metrics measure recovery of the initial shape; successful editing additionally requires applying the requested modification while preserving unaffected geometry. JORE also improves reconstruction while strengthening final editing in the matched joint-generation comparisons. It raises source IoU and executability by 1.72 pp each over GRPO alongside its final-edit gains, and outperforms RCN-only at both stages. The final-edit improvements therefore do not come at the expense of source reconstruction under matched training.

5.5 Cross-Stage Credit Assignment

Preserving final-edit feedback to Source while excluding direct source-quality credit to Edit gives the best performance among the four routing configurations. With RCN fixed, Table 4 varies these two cross-stage paths. Retaining $Q_1 \to$Source improves IoU by 2.58 pp over Stage-local, which jointly generates both stages but provides no direct final-edit credit to Source tokens. Stage-local retains the same shared policy, source supervision, and conditioning of Edit on the generated Source. The comparison therefore distinguishes access to an intermediate program from direct reward credit: allowing Edit to use the reconstructed source is not equivalent to weighting Source-token updates with final-edit feedback. The latter connects reconstruction learning to downstream editing success. Removing $Q_0 \to$Edit from All-to-all improves IoU by 1.56 pp: source quality evaluates the completed reconstruction rather than the success of subsequent editing. In the absence of $Q_1 \to$Source, however, the same removal changes IoU only slightly, from 75.80% for Reverse routing to 75.92% for Stage-local. This pattern suggests that suppressing source-to-edit feedback alone does not account for the full gain. Its combination with final-edit feedback to Source connects reconstruction updates to the downstream task while keeping source-quality credit local. Reverse routing and JORE each retain one cross-stage path, yet the latter performs better. Together, these comparisons support assigning feedback according to stage dependencies, rather than simply reducing the number of credit paths.

6 Conclusion

We studied end-to-end CAD editing from dimensioned drawings and textual instructions without source CAD input. We introduced CAD-D2E to evaluate this setting, with paired executable source and edited programs that also support intermediate supervision. JORE coordinates intermediate reconstruction supervision with final-edit feedback through joint generation trained with CARPO. CARPO routes component-relative advantages according to stage dependencies, restricting direct source-quality credit to reconstruction while retaining final-edit feedback across both stages. Experiments on CAD-D2E show improved final editing over the evaluated existing systems and alternative generation baselines, while matched ablations support the normalization and stage-routing design of CARPO. These findings support combining component-relative signals with dependency-aware intermediate supervision for end-to-end CAD editing, and suggest that explicit stage structure can improve learning when intermediate reconstruction quality directly affects downstream edits.

AI USE STATEMENT

Generative AI tools, including ChatGPT, were used in this work for two purposes. First, they were used to aid and polish the writing of author-written text, including improving grammar, clarity, and readability. The tools were not used to independently draft substantive sections of the paper. All AI-assisted text was reviewed, revised, and approved by the authors. Second, generative AI was used to generate synthetic data used in our experiments. The generated data were subsequently reviewed and validated by the authors using the same quality-control and evaluation procedures described in the paper. The authors verified the resulting data and experimental outputs before including them in the study. Generative AI was not used for literature retrieval, research ideation, proving mathematical claims, or independently conducting the research. The authors take full responsibility for the content of the paper, the validity of the generated data, and the reported results.

ETHICS STATEMENT

This work aims to support research on automated CAD editing and assist engineering design. CAD-D2E is constructed from procedurally generated mechanical parts, synthetic engineering drawings, and editing instructions. The dataset is intended solely for research on CAD generation and editing. Human review is used to assess the consistency and quality of these examples. We encourage responsible use of the benchmark and method, including appropriate engineering validation of generated designs before practical use.

REFERENCES

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025. URL https://arxiv.org/abs/2502.01456.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Anna C. Doris, Md Ferdous Alam, Amin Heyrani Nobari, and Faez Ahmed. CAD-Coder: An open-source vision-language model for computer-aided design code generation. *arXiv preprint arXiv:2505.14646*, 2025. URL https://arxiv.org/abs/2505.14646.

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 *Advances in Neural Information Processing Systems*, volume 38, pp. 59765–59789, 2025. URL https://arxiv.org/abs/2505.19713.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 114399–114431, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a6536243037d1e32c20de85137d478da-Paper-Conference.pdf.

Ajay Bangalore Harish and Abhishek Rajendra Prasad. Automated 3D solid reconstruction from 2D CAD using OpenCV. *arXiv preprint arXiv:2101.04248v1*, 2021.

Tao Hu, Jiaxin Ai, Licheng Wen, Xueheng Li, Shu Zou, Siqi Li, Nianchen Deng, Xinyu Cai, Hongbin Zhou, Pinlong Cai, Daocheng Fu, Yu Yang, Hairong Zhang, Botian Shi, and Xuemeng Yang. IterCAD: An iterative multimodal agent for visually-grounded CAD generation and editing. *arXiv preprint arXiv:2606.13368*, 2026. URL https://arxiv.org/abs/2606.13368.

Aditya Joglekar, Amit Regmi, Kenji Shimada, and Levent Burak Kara. Ortho2CAD: 3D CAD generation from orthographic drawings using vision language models. *arXiv preprint arXiv:2607.08891*, 2026. URL https://arxiv.org/abs/2607.08891.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590. PMLR, 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

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 reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2505.22914.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: Group reward-decoupled normalization policy optimization for multi-reward RL optimization. *arXiv preprint arXiv:2601.05242*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025. URL https://arxiv.org/abs/2503.20783.

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. *arXiv preprint arXiv:2512.23333*, 2025. URL https://arxiv.org/abs/2512.23333.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.8-Max, 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Yu Yuan, Shizhao Sun, Qi Liu, and Jiang Bian. CAD-editor: A locate-then-infill framework with automated training data synthesis for text-based CAD editing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73588–73603. PMLR, 2025. URL https://proceedings.mlr.press/v267/yuan25g.html.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025. URL https://arxiv.org/abs/2507.18071.

A Additional Method Details

A.1 Stage-Dependency Derivation

Autoregressive stage dependencies. The joint policy factorizes as

$$\pi_\theta(y \mid x) = \pi_\theta(y_{\text{src}} \mid x)\pi_\theta(y_{\text{edit}} \mid x, y_{\text{src}}). \tag{8}$$

The stage assignment in CARPO uses this directional dependency to determine the stages receiving direct reward credit. The following form of the score-function identity (Williams, 1992) applies to rewards fixed by a generated prefix.

Proposition 1 (Prefix-determined reward) *For fixed input x and prefix u, let $v \sim \pi_\theta(\cdot \mid x, u)$ and $R = r(x, u)$. If the conditional policy has differentiable probabilities, parameter-independent support, and permits differentiation under the probability sum, then R has zero expected score-function contribution to the continuation.*

Proof. With (x, u) fixed, R is constant across continuations. Thus,

$$\begin{aligned}
\mathbb{E}_{v\sim\pi_\theta(\cdot|x,u)}\left[R\nabla_\theta \log \pi_\theta(v \mid x, u)\right] &= R\sum_v \pi_\theta(v \mid x, u)\nabla_\theta \log \pi_\theta(v \mid x, u) \\
&= R\sum_v \nabla_\theta \pi_\theta(v \mid x, u) \\
&= R\nabla_\theta \sum_v \pi_\theta(v \mid x, u) \\
&= R\nabla_\theta 1 = 0.
\end{aligned} \tag{9}$$

Raw-reward gradient decomposition. For fixed input x, define the stage scores $g_{\text{src}} = \nabla_\theta \log \pi_\theta(y_{\text{src}} \mid x)$ and $g_{\text{edit}} = \nabla_\theta \log \pi_\theta(y_{\text{edit}} \mid x, y_{\text{src}})$. With fixed weights and verifier rewards that have no explicit dependence on θ, the raw-reward objective is

$$J_x(\theta) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[w_0 Q_0 + w_1 Q_1 + w_F F]. \tag{10}$$

All expectations below are under π_θ. Assume the completed Source prefix fixes the program scored for Q_0, so $Q_0 = q_0(x, y_{\text{src}})$. Applying Proposition 1 conditional on y_{src} and then averaging over Source prefixes gives $\mathbb{E}[Q_0 g_{\text{edit}}] = 0$. Applying the policy factorization in Equation 8 yields

$$\nabla_\theta J_x = \mathbb{E}\left[(w_0 Q_0 + w_1 Q_1 + w_F F)g_{\text{src}} + (w_1 Q_1 + w_F F)g_{\text{edit}}\right]. \tag{11}$$

Final-edit credit across stages. Let $V_1^\theta(x, y_{\text{src}})$ denote the expected final-edit reward conditional on the generated Source. By the law of total expectation,

$$\begin{aligned}
V_1^\theta(x, y_{\text{src}}) &= \mathbb{E}_{y_{\text{edit}}\sim\pi_\theta(\cdot|x,y_{\text{src}})}[Q_1], \\
\mathbb{E}[Q_1 g_{\text{src}}] &= \mathbb{E}[V_1^\theta(x, y_{\text{src}})g_{\text{src}}].
\end{aligned} \tag{12}$$

This term need not vanish because the generated source affects the continuation distribution and its expected editing quality. Final-edit credit therefore covers both stages. Format credit also covers both stages because protocol validity depends on the complete response. Credit assignment weights the stage-specific score terms; both stages update the same policy parameters. The reward-to-stage dependencies identified in Equation 11 motivate the assignment of RCN-normalized component advantages in Equation 5.

A.2 Implementation and Evaluation Details

Training configuration. We apply LoRA (rank 8, $\alpha = 32$, dropout 0.05) to Qwen3.5-9B's language-model linear layers, freezing the vision tower and visual aligner. Training uses bfloat16 and AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon_{\text{Adam}} = 10^{-8}$, and weight decay 0.1. Table 5 summarizes the remaining settings. Matched joint RLVR variants initialize fresh adapters from the same merged cold-start model, also used as the frozen reference policy, and perform one policy update per rollout batch. Training uses A100 80GB GPUs.

Table 5: Training configuration for joint source–edit generation.

Setting	Cold-start (SFT)	RLVR
Training epochs	1	1
Global batch size	48 examples	8 inputs $\times$ 4 responses
Learning rate	10^{-4}	10^{-6}
Scheduler / warmup	Cosine / none	Cosine / 3%
Maximum sequence length (tokens)	6,144	10,240
Maximum response length (tokens)	–	4,096
Sampling temperature / top-p	–	$0.8/0.95$
KL coefficient / clipping threshold	–	$0.04/0.2$
Reward weights (w_0, w_1, w_F)	–	$(0.2, 0.6, 0.2)$

Inference and metrics. We use greedy decoding with vLLM for our task-trained models, with a 6,144-token context limit and up to 4,096 generated tokens. Predictions are evaluated without code repair or candidate selection; geometry evaluation has a 120-second timeout.

For normalized Chamfer distance, we sample 1,000 points from each predicted and reference surface to obtain P_n and T_n. The symmetric mean nearest-neighbor distance is normalized by the sum of their axis-aligned bounding-box diagonals:

$$D_n = \frac{\frac{1}{|P_n|}\sum_{p\in P_n}\min_{q\in T_n}\|p-q\|_2 + \frac{1}{|T_n|}\sum_{q\in T_n}\min_{p\in P_n}\|q-p\|_2}{d(P_n)+d(T_n)}, \tag{13}$$

where $d(\cdot)$ denotes the bounding-box diagonal. IOU, Mean CD and Medium CD are computed over successful executions.

B Additional Experimental Results

B.1 Source-Reward Weight Sensitivity

Table 6: Source-reward weight sensitivity, with $(w_1, w_F) = (0.6, 0.2)$. Best results are bold.

w_0	IoU (%) ↑	Mean CD ↓	Med CD ↓	Exec. (%) ↑
0.0	76.69	0.15542	0.01599	85.74
0.2	**78.50**	**0.13677**	**0.01588**	**87.63**
0.4	77.11	0.15076	0.01609	86.22

Direct source reward provides useful supervision beyond final-edit feedback in this setting. Table 6 varies the relative source weight w_0 with $(w_1, w_F) = (0.6, 0.2)$ fixed, retaining joint generation, cold-start supervision, RCN, and CSA. At $w_0 = 0$, the model still generates and scores the source program, but only final-edit and format rewards contribute to RL updates. Relative to the default $w_0 = 0.2$, this removal lowers final IoU by 1.81 pp (paired 95% confidence interval: $[-2.91, -0.72]$ pp) and executability by 1.89 pp. Thus, final-edit feedback to Source does not fully substitute for direct intermediate-state reward under the evaluated configuration.

Further increasing source-reward weight does not improve final editing. At $w_0 = 0.4$, final IoU falls by 1.39 pp relative to the default, with a paired 95% confidence interval of $[-2.48, -0.32]$ pp. Because RCN computes its shared scale from the weighted aggregate, changing w_0 also rescales final-edit and format contributions rather than strengthening source feedback in isolation. This coupling offers a possible explanation for the non-monotonic response, with the default $w_0 = 0.2$ performing best among the three tested settings. The default leads all four final-edit metrics, although Med CD varies only slightly.

C Dataset Details and Visualizations

C.1 Data Construction and Quality Control

A shared operation specification determines the source program and the geometry and dimensions of its orthographic views. Editing operations are selected according to their compatibility with the source geometry and existing features; Table 7 lists the available edit subtypes within the four groups shown in Figure 3. The chosen operation and parameters determine both the reference edited program and a template-based instruction. Qwen rewrites each draft instruction for linguistic clarity, while the editing specification defines the target geometry.

Automatic checks assess program execution, geometric validity, and a nontrivial change from the source to the edited shape. Human reviewers inspected 30% of the examples in each of the cold-start and RLVR training pools and all evaluation examples. The review assessed dimension completeness, consistency between drawings and source geometry, and agreement between editing instructions and target geometry. Identified inconsistencies were corrected, and the revised examples were retained in the dataset.

For the cold-start subset, Qwen3-VL-Plus generates source-reconstruction and edit reasoning offline using reference-conditioned annotation prompts. Source reasoning is shared across examples of the same source part, whereas edit reasoning is specific to each source–instruction pair. The resulting explanations are paired with the unchanged reference programs to form the four-block supervision sequence $(r^0, \mathcal{C}^0, r^1, \mathcal{C}^1)$. Student inputs contain only the views, instruction, and protocol requirements. The assembled targets are checked for protocol structure and program syntax.

Table 7: Edit groups and subtypes in CAD-D2E.

Edit Group	Edit Subtypes
Feature addition	Add boss; add hole; add pocket; add rib; add slot
Feature removal	Remove boss; remove hole; remove pocket; remove slot
Geometry modification	Increase thickness; hole to counterbore; hole to polygon cutout; add chamfer; add fillet
Pattern creation	Add bolt circle

C.2 Dataset Visualizations

Table 8: Drawing notation in Figures 6 and 7. Lengths are in millimeters; angles are in degrees.

Notation	Meaning
THRU	Through the full local material thickness.
CBORE	Counterbore: a coaxial, flat-bottomed cylindrical recess.
DP	Depth of the specified feature (here, the counterbore).
PCD	Pitch circle diameter of the hole pattern.
Ø, R	Diameter and radius, respectively.
nX	Number of repeated features (e.g., 6X denotes six holes).
SLOT, LUG	Elongated cutout and projecting tab, respectively.
t, b	Part thickness and gear face width, respectively.
d_a, d_f	Tooth-tip and tooth-root diameters, respectively.
z, τ	Tooth count and angular pitch ($\tau = 360^\circ/z$).
θ_a, θ_f	Angular tooth widths at the tip and root, respectively.

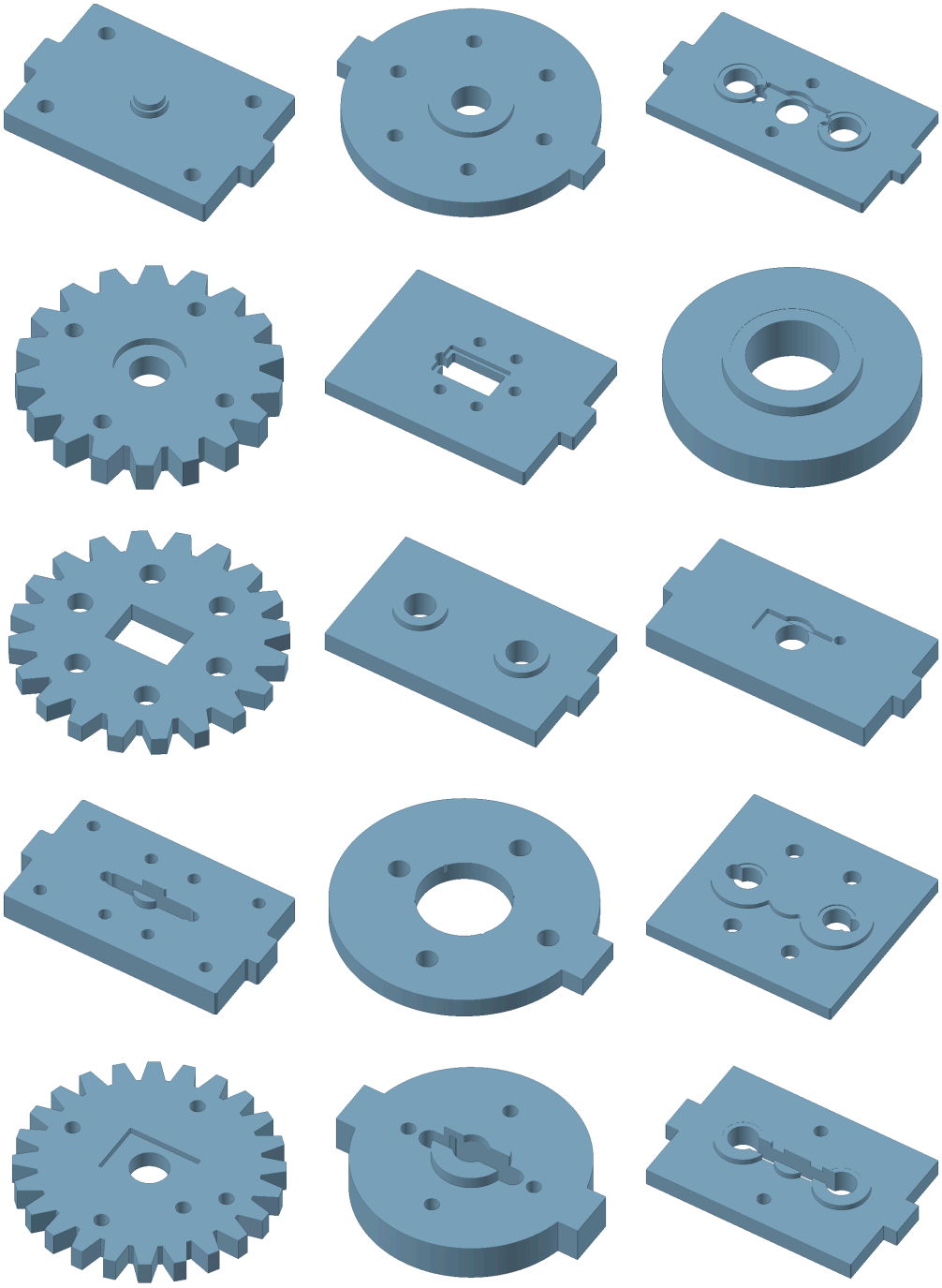

Figure 5: **Edited reference shapes from CAD-D2E.** Fifteen selected examples are rendered from ground-truth edited programs C^1 and independently scaled for visualization.

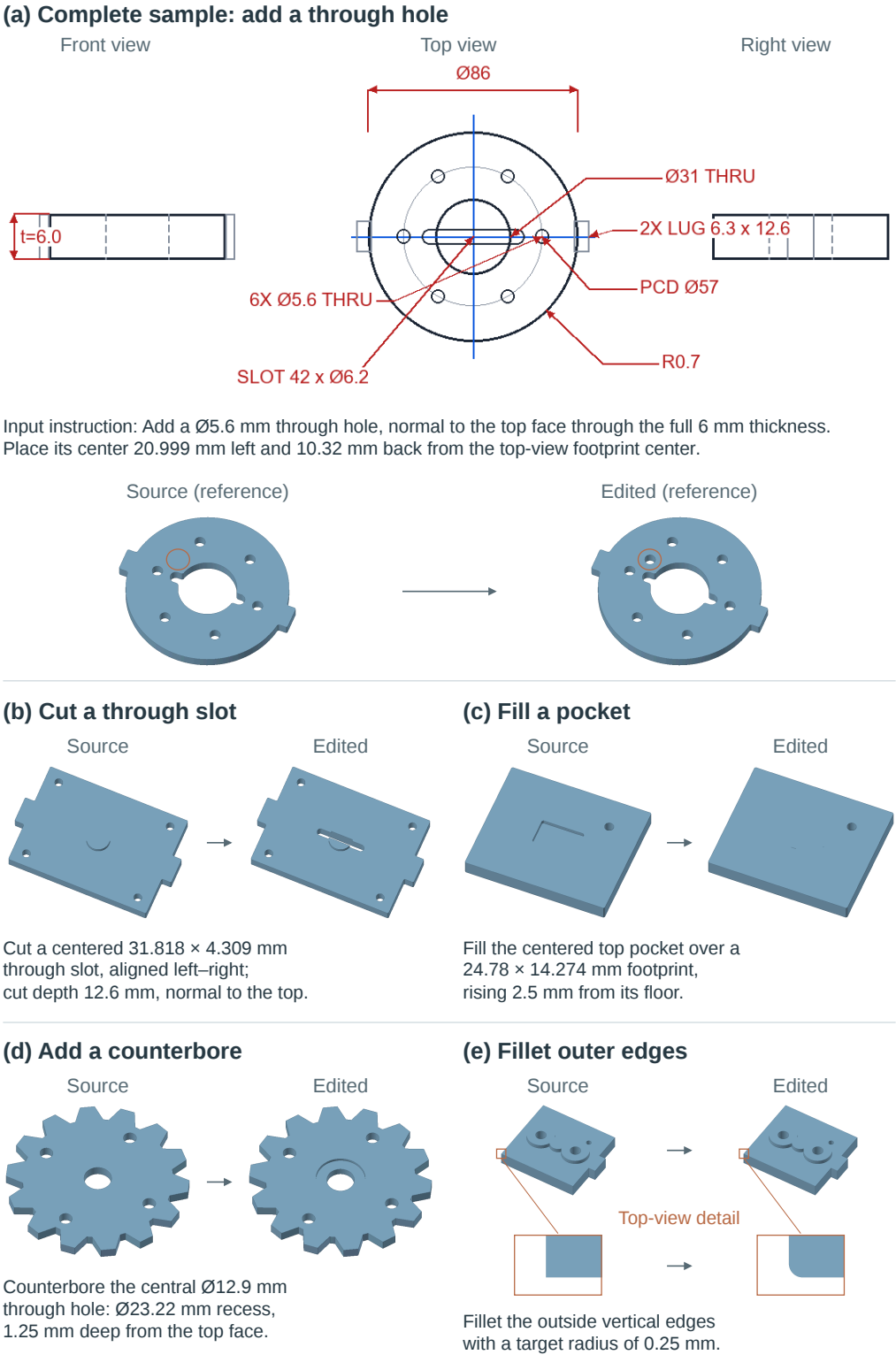

Figure 6: **Editing examples from CAD-D2E.** (a) Input views and instruction with reference shapes. (b–e) Additional reference edit pairs. Instructions are abbreviated; dimensions are in millimeters. Each pair shares the same view and scale; the insets show the marked corner from above.

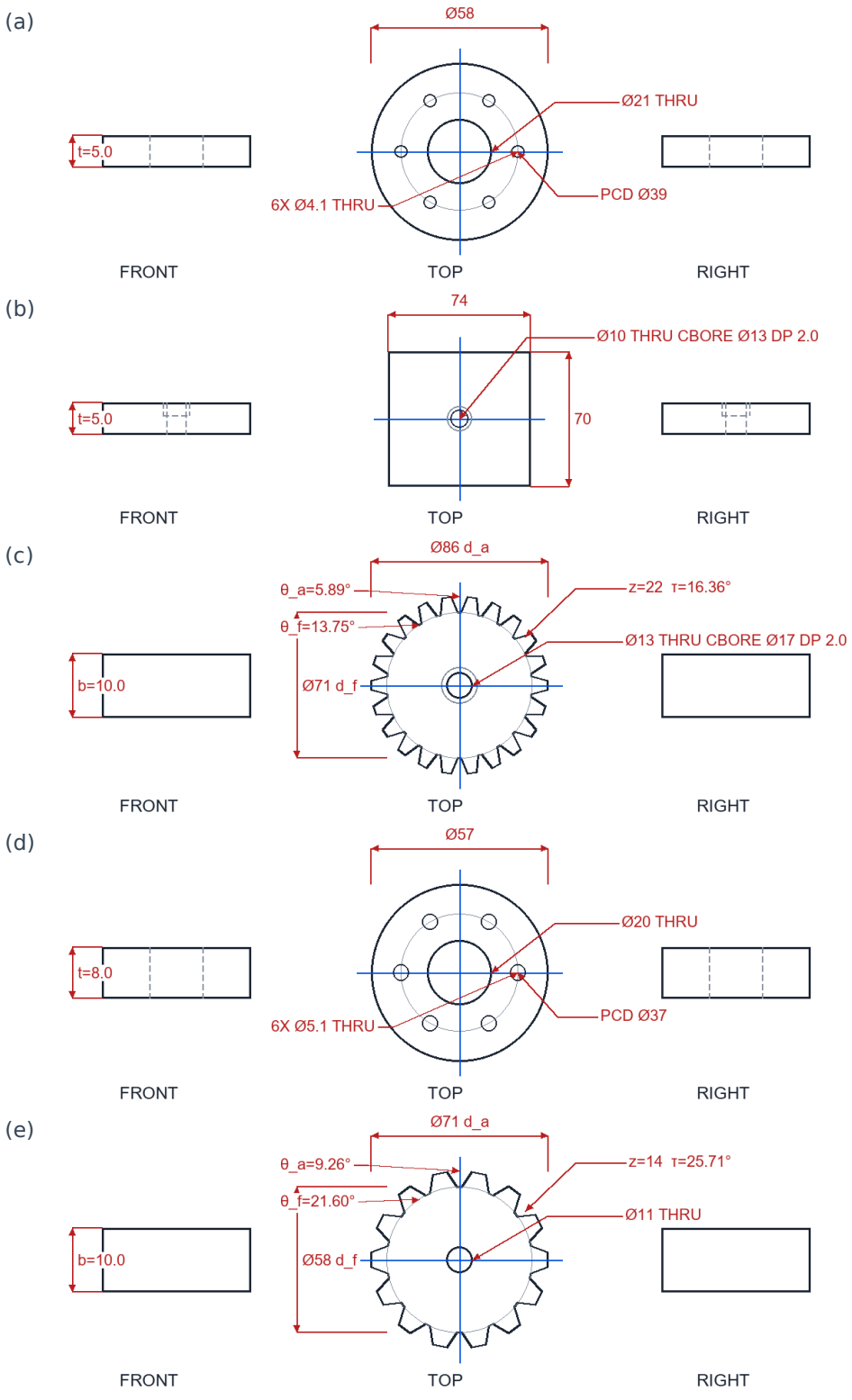

Figure 7: **Dimensioned input drawings from CAD-D2E.** Five examples illustrate hole patterns, counterbores, and tooth profiles. Blank vertical bands are removed while preserving the original three-view arrangement and all dimension annotations. Table 8 defines the drawing notation.

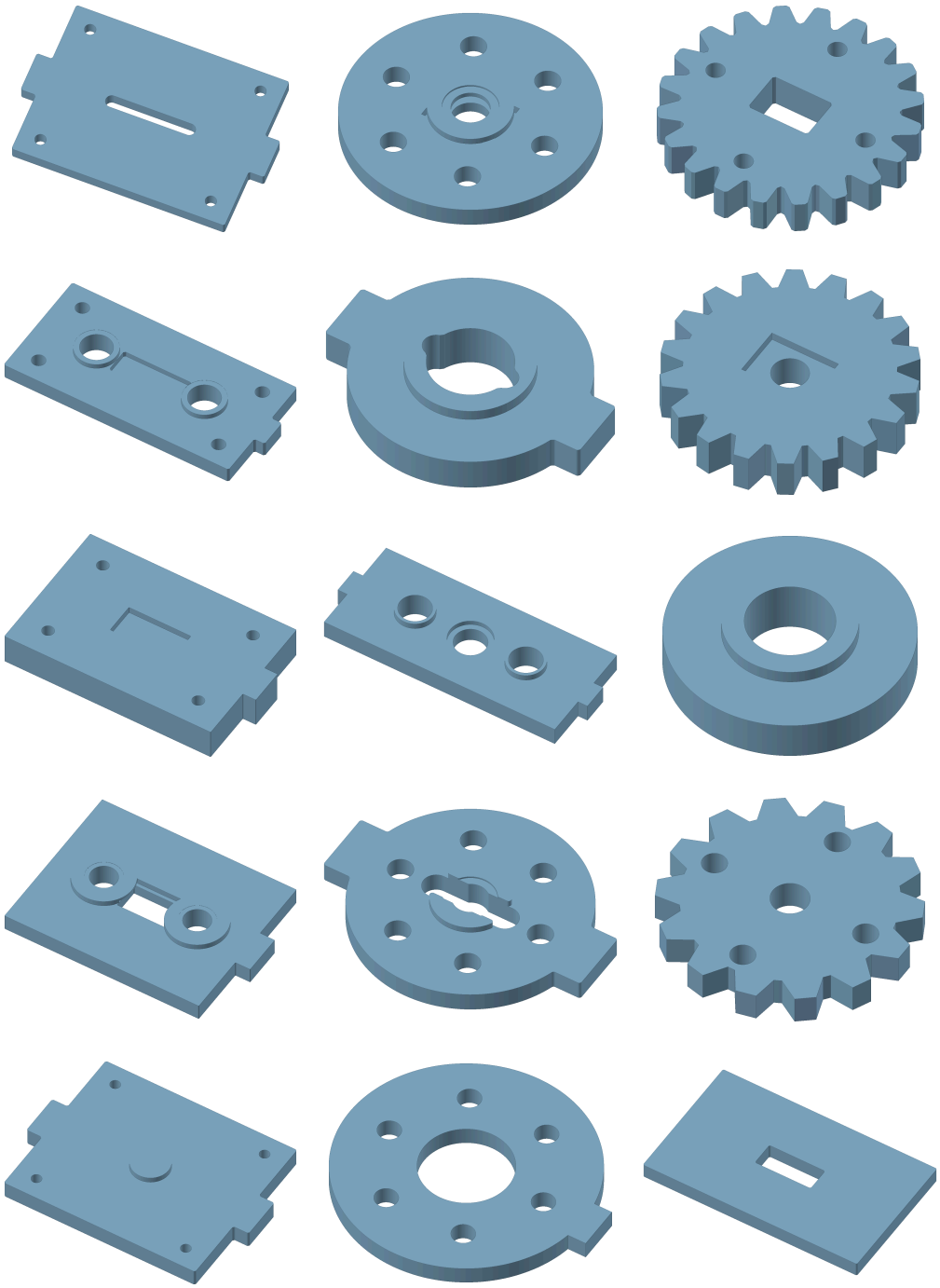

Figure 8: **Source reference shapes from CAD-D2E.** Fifteen examples rendered from ground-truth source programs $\mathcal{C}^0$ illustrate different part profiles and geometric feature combinations. Shapes are independently scaled for visualization.