CADFS: A Big CAD Program Dataset and Framework for Computer-Aided Design with Large Language Models

Supplementary Material

In Sec. 6, we present additional ablation studies on key design choices in our framework, including (1) the choice of the base VLM used for fine-tuning, (2) the contribution of individual components of our FeatureScript-based representation, (3) the effect of combining the design history representation with appropriate textual annotations, (4) the effect of providing the model with explicit design dimensions during generation, and (5) the impact of input image resolution on reconstruction quality for models involving the newly introduced operations. In Sec. 7, we report evaluation with additional metrics, on additional data, and an additional comparison with a mesh-based approach. In Sec. 8, we discuss failure cases of our model. In Sec. 9, we provide implementation details. In Sec. 10, we discuss our choice of modeling operations. In Sec. 11, we provide further details of our annotation procedure. In Sec. 12, we describe the evaluation details.

6. Additional ablation studies

6.1. Choice of base model

In our main experiments, we use the Qwen2-VL-2B model [32]. Here, we compare it with the larger Qwen3-8B variant [44] on text-conditioned generation using the DeepCAD test subset. As shown in Tab. 4, the 2B model performs on par with the 8B model, while requiring roughly half the training time. These results indicate that Qwen2-VL-2B achieves a strong balance between computational efficiency and generation accuracy.

Table 4. Comparison of LLMs of different size trained for text-conditioned CAD generation based on our framework. The **best** result is shown in bold text.

Base model	CD↓	ECD↓	NC↑	MMD↓	COV↑	JSD↓	IR↓
Qwen2-VL-2B	**0.06**	8.5	**98.9**	8.10	84.9	**0.59**	10
Qwen3-8B	**0.06**	**7.8**	**98.9**	**8.04**	**85.0**	0.60	**8**

6.2. FeatureScript representation

The Onshape platform does not directly provide clean, executable FeatureScript code. We reconstruct high-quality FeatureScript programs from Onshape's internal representation using our data acquisition pipeline. Table 5 reports an ablation over progressively refined variants of this representation. In these experiments, we train and test Qwen3-8B for text-conditioned generation on the DeepCAD subsets.

We begin with minimally processed executable code extracted from the internal Onshape representation paired with abstract Text2CAD [16] annotations (a). This baseline yields low generation accuracy, indicating that the raw Onshape representation is insufficient for precise generation.

As the first refinement, we replace the arbitrary identifiers of geometric entities (edges, faces, bodies) with compact deterministic ones (*e.g.*, "F0", "E0", "E1" in Fig. 2) (b). This improves geometric accuracy (Chamfer Distance) and diversity (COV) by 22% and 13%, respectively.

Next, we train a model on expert Text2CAD annotations (c), which substantially improves performance across all metrics. However, this also doubles the Invalidity Ratio. This suggests that the detailed expert descriptions from Text2CAD are misaligned with the entangled internal Onshape representation of CAD models.

To address this, we replace implicit definitions of sketch elements with explicit ones (d), *e.g.*, representing line segments by their endpoints rather than by an origin point and direction. This significantly reduces the Invalidity Ratio from 24% to 10% while maintaining accuracy and diversity.

We further disentangle the representation by simplifying modeling operations (e), yielding an additional 21% reduction in Chamfer Distance.

With the FeatureScript code now concise and interpretable, we generate our own textual annotations tailored to this representation and describing the CAD models more precisely (f). This key alignment between representation and annotations produces a 60% reduction in Chamfer Distance and improvements across other metrics as well.

Finally, we standardize numerical precision to two decimal places (g). While this has only a minor impact on performance, it reduces code length and improves consistency.

6.3. Different combinations of representations and annotations

Table 6 shows an additional evaluation of the contributions of our FeatureScript representation and our new textual annotations. We compare models trained with our standardized FeatureScript representation paired with different annotations: (a) T2C short annotations, (b) T2C expert annotations, and (e) ours. We also compare models trained with our annotations paired with different design history representations: (c) Python code, (d) the minimally processed "raw" FeatureScript, and (e) the standardized FeatureScript. In these experiments, we train and test Qwen2-VL-2B on the DeepCAD subsets.

Table 5. Ablation study of our FeatureScript-based representation. We compare text-conditioned models trained using progressively refined variants of this representation. The **best** and second best results are shown in bold text and underlined respectively.

	Annotations	Stable identifiers	Explicit paramerization	Simplified operations	Precision normalization	Median CD↓	Median ECD↓	Median NC↑	Mean CD↓	Mean ECD↓	MMD↓	COV↑	JSD↓	IR↓
(a)	T2C short					124.87	318.1	49.3	190.8	452.1	13.26	58.6	3.44	14
(b)	T2C short	✓				97.82	272.8	52.4	164.2	399.9	12.40	66.2	1.74	12
(c)	T2C expert	✓				0.18	23.0	96.9	23.2	88.9	10.06	77.8	0.79	24
(d)	T2C expert	✓	✓			0.19	22.5	96.9	21.3	81.1	9.29	81.9	0.66	10
(e)	T2C expert	✓	✓	✓		0.15	19.8	97.7	20.4	78.2	9.08	82.2	0.64	9
(f)	Ours	✓	✓	✓		**0.06**	7.9	**98.9**	7.7	46.8	**7.89**	**85.8**	**0.59**	8
(g)	Ours	✓	✓	✓	✓	**0.06**	**7.8**	**98.9**	**7.2**	**45.0**	8.04	85.0	0.60	**8**

Table 6. Comparison of different combinations of representations and annotations.

	Representation	Annotations	CD↓	COV↑	IR↓
(a)	Standardized FeatureScript	T2C short	84.87	24.0	5
(b)	Standardized FeatureScript	T2C expert	0.40	79.7	15
(c)	Python code	Ours	1.93	48.5	4
(d)	Raw FeatureScript	Ours	0.05	30.6	56
(e)	Standardized FeatureScript	Ours	0.06	84.9	10

Only pairing the standardized FeatureScript with our annotations (e) simultaneously achieves low geometric error (CD), high diversity (COV), and a low invalid generation rate (IR). Pairing FeatureScript with the Text2CAD annotations (a,b), or our annotations with Python (c) leads to significantly higher geometric error. This is because these annotations describe the design process in substantially different representations. The Text2CAD annotations and Python scripts are derived from the DeepCAD command sequence, while our annotations are derived from FeatureScript code. For example, in DeepCAD, sketches are created in a normalized coordinate frame and then scaled and translated, whereas in FeatureScript everything is modeled directly in a global coordinate frame.

Training on raw FeatureScript, even with our annotations (d), results in a high fraction of invalid outputs (IR) due to the high degree of entanglement in the raw FeatureScript.

6.4. Scale-aware multi-view reconstruction of CAD

Generating CAD design histories as code naturally enables models to operate directly in the physical units used by engineers. Prior code-based frameworks, however, inherit normalized coordinate systems from token-sequence representations, preventing generation at real-world scales. In contrast, our CAD programs preserve each design's original dimensions, allowing models to learn and generate geometry at true scale. Our text annotations follow the same principle and specify all measurements in native units.

In this context, reconstruction from multi-view images presents an additional challenge: absolute scale cannot be reliably inferred from visual input alone. To mitigate this, we provide the VLM with the bounding-box dimensions and position of each design as part of the textual prompt (*e.g.*, *"Generate a CAD model using FeatureScript framework. Bounds from (-114.66, -69.35, -31.78) to (68.33, 76.26, 50.8), center = (-23.17, 3.45, 9.51), scale = 91.5"*).

Table 7. Comparison of multi-view CAD reconstruction models trained with and without additional information about the bounding box dimensions of the design. The **best** result is shown in bold text.

Bounding box	CD↓	ECD↓	NC↑	MMD↓	COV↑	JSD↓	IR↓
Not specified	0.58	23.9	94.3	9.90	79.5	0.64	14
Specified	**0.14**	**17.8**	**97.9**	**8.90**	**82.9**	**0.60**	**9**

Table 7 compares models trained with and without this scale information on the DeepCAD test set. Supplying the bounding-box parameters significantly improves geometric accuracy: for example, improves Chamfer Distance by 76% and reduces the Invalidity Ratio by 36%.

6.5. Image resolution

We additionally examine the effect of input image resolution on CAD reconstruction quality. Table 8 compares a model trained with a per-view resolution of 256×256 against one trained with 128×128. We evaluate both on the DeepCAD test set and on our new test set containing the full set of modeling operations.

On the DeepCAD test set, which consists primarily of simple sketch-and-extrude geometry, the gains from higher resolution are modest. In contrast, on our test set featuring more complex geometric structures, the benefits are substantially larger. Overall, doubling the input resolution yields roughly a $2\times$ improvement in Chamfer Distance and a 23% improvement in Edge Chamfer Distance. These results further highlight that models trained on richer, multi-operation CAD data are better positioned to leverage higher-fidelity visual input, reflecting the greater geometric diversity and complexity of our new data.

Table 8. Comparison of multi-view CAD reconstruction models trained with different per-view image resoluton. The **best** result is shown in bold text.

Per-view resolution	DeepCAD test set (sketch, extrude)							Our test set (15 operations)						
	CD↓	ECD↓	NC↑	MMD↓	COV↑	JSD↓	IR↓	CD↓	ECD↓	NC↑	MMD↓	COV↑	JSD↓	IR↓
128×128	0.12	13.7	98.2	8.66	83.5	**0.63**	**11**	0.62	20.5	94.5	10.67	77.3	1.02	**26**
256×256	**0.11**	**11.5**	**98.4**	**8.60**	**83.9**	**0.63**	12	**0.35**	**15.8**	**96.1**	**10.40**	**77.8**	**0.98**	29

Table 9. Additional comparison of our framework with a Python code-based Cadrille in text-conditioned generation (left) and multi-view reconstruction (center and right) on topology validity metrics. The **best** result is shown in bold text.

Model	Text input, DeepCAD test set				Img input, DeepCAD test set				Img input, Our test set (15 operations)			
	SegE↓	DangEL↓	SIR↓	FluxEE↓	SegE↓	DangEL↓	SIR↓	FluxEE↓	SegE↓	DangEL↓	SIR↓	FluxEE↓
Cadrille	0.38	0.3	0.51	**0.21**	0.36	0.3	**0.25**	0.45	3.50	**0.8**	**0.51**	0.92
Ours	**0.16**	**0.1**	**0.48**	**0.21**	**0.27**	**0.1**	0.83	**0.35**	**1.35**	1.2	1.10	**0.80**

7. Additional comparisons

7.1. Additional metrics

Table 9 reports additional topology validity metrics from [40, Sec. 6.1.4]: Segment Error (SegE), Dangling Edge Length (DangEL), Self-Intersection Ratio (SIR), and Flux Enclosure Error (FluxEE). Our model achieves high topological validity.

7.2. Additional test data

In Tab. 10, we compare our model with Cadrille in image-based reconstruction on the CADParser dataset [48], which features 5 operations: sketch, extrude, revolve, fillet, and chamfer. The dataset includes 40k designs obtained from an initial set of 6.8k base designs via augmentation. We test the models on a subset of 6.8k designs originating from different base designs.

The results are consistent with those on the DeepCAD and CADFS test sets. Our model achieves significantly higher accuracy (CD, ECD) than Cadrille, with comparable diversity (COV, JSD).

7.3. Comparison with mesh-based generation

To show that CAD model generation requires specialized methods, we compare our framework against the polygonal mesh generation method TRELLIS [38] on the image-conditioned generation task. In this comparison, TRELLIS takes an isometric CAD image as input and generates a mesh, followed by postprocessing.

Table 11 and Fig. 6 show the quantitative and qualitative results, respectively. The quantitative results show that our framework outperforms TRELLIS by a large margin. The visual results show that our CAD-specific approach generates precise geometry, while the mesh-based method produces non-watertight meshes, over-smoothed edges, disconnected geometric components, and noisy surfaces.

Table 10. Additional comparison of our framework with a Python code-based Cadrille in multi-view reconstruction on the CADParser dataset (5 operations). The **best** result is shown in bold text.

Model	CD↓	ECD↓	NC↑	MMD↓	COV↑	JSD↓	IR↓
Cadrille	0.23	20.9	95.5	**4.49**	**50.9**	**0.36**	**11**
Ours	**0.13**	**12.7**	**97.4**	4.94	48.1	0.37	19

8. Failure cases

Figure 7 shows examples of failure cases for the model trained on our data. In both text- and image-conditioned modes, the model occasionally omits refinement operations such as fillets (a) or applies them to incorrect geometric entities (b). It also occasionally produces inaccurate reconstructions overall (c,d). In image-conditioned mode, the model often produces incorrect text (e), which we attribute to limitations of the Qwen-VL visual encoder. It also frequently resorts to lower-level operations, *e.g.*, multiple sketch-extrudes instead of a pattern. This is likely due to the prevalence of simple operations in Onshape designs.

9. Implementation details

We train our model in two stages. First, we fine-tune the model on ~170k designs containing only sketch and extrude operations, corresponding to the DeepCAD dataset [36]. This stage establishes core geometric reasoning. We then fine-tune the model on ~405k designs from our full dataset that remain after excluding the test splits and scripts longer than 8192 tokens. This enables generalization to all 15 modeling operations. In both stages, the model is conditioned either on textual descriptions or a 2×2 grid of multi-view images, following Cadrille [19]. In the first stage, we use images at a resolution of 128×128. In the second stage, we increase the image resolution to 256×256 to improve geometric reasoning on complex structures, while keeping all other hyperparameters unchanged. At each stage, we train the model for 3 epochs with a batch size of 128, us-

Table 11. Comparison of our CAD-specific method and a mesh-based method of reconstruction from multi-view images. The **best** result is shown in bold text.

Method	DeepCAD test set (sketch, extrude)							Our test set (15 operations)						
	CD↓	ECD↓	NC↑	MMD↓	COV↑	JSD↓	IR↓	CD↓	ECD↓	NC↑	MMD↓	COV↑	JSD↓	IR↓
Trellis	36.04	168.0	67.2	14.67	62.6	2.82	**0**	32.06	173.9	67.5	16.58	63.2	2.36	**3**
Ours	**0.11**	**11.5**	**98.4**	**8.60**	**83.9**	**0.63**	12	**0.35**	**15.8**	**96.1**	**10.40**	**77.8**	**0.98**	29

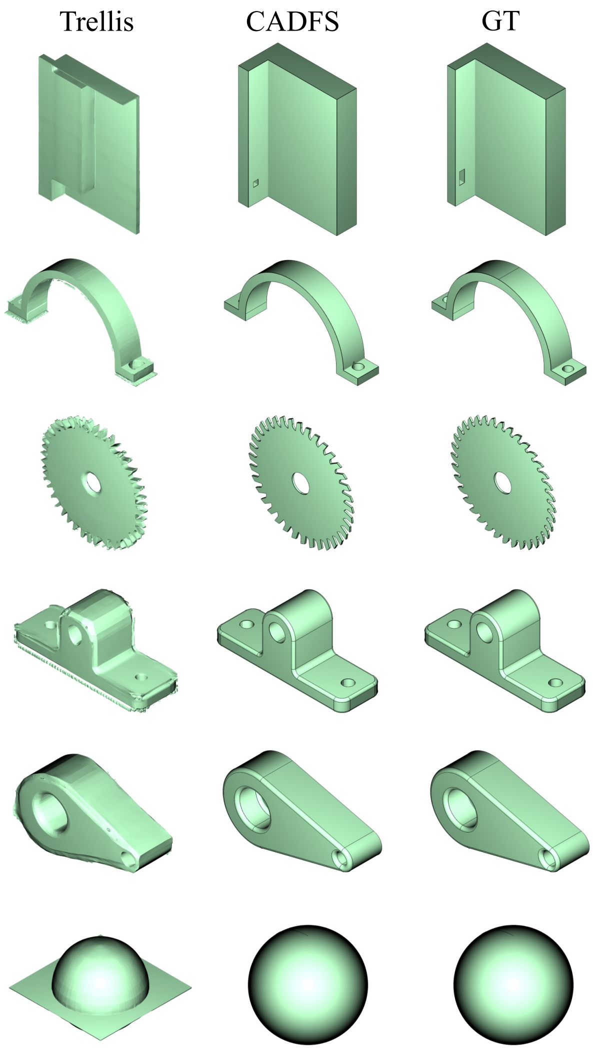

Figure 6. Qualitative comparison of our CAD-specific method and a mesh-based method of reconstruction from multi-view images.

ing the Adam-W optimizer [24] with an initial learning rate of 2e-4, a linear warmup ratio of 0.05, and a cosine decay schedule. Training on 8 NVIDIA A100 GPUs with DeepSpeed [30], FlashAttention-2 [10], and Liger-Kernel [15] optimizations takes 30 and 76 hours for the first and second stages, respectively, using 24 GB of VRAM per GPU.

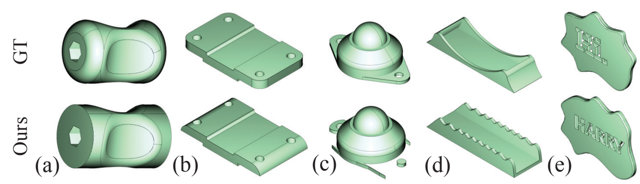

Figure 7. Examples of failure cases for the model trained on our data for text (a-c) and image input (d,e).

10. Choice of modeling operations

Sketch-based construction of primary solids. *Sketch* defines 2D profiles composed of lines, circles, arcs, ellipses, elliptical arcs, Bezier curves, splines, and text. This expands beyond prior datasets limited to lines, circles, and circular arcs. Sketches serve as the foundation for most solids. *Extrude* creates 3D solids by extending sketch profiles linearly, commonly used for prismatic parts and structural components. Unlike prior datasets, the FeatureScript representation enables extruding separate parts of a sketch. *Revolve* sweeps a sketch profile around an axis to create rotationally symmetric solids (*e.g.*, shafts, housings, knobs). *Sweep* moves a profile along a spatial curve to form tubing, wire guides, and ergonomic handles. *Loft* interpolates smoothly between multiple profiles to create aerodynamic or freeform transitions. *Construction plane* defines reference planes used to position sketches, splits, and mirror operations. These operations create the core massing geometry of a part.

Refinement and edge treatment. *Fillet* rounds sharp edges to reduce stress concentrations, improve manufacturability, and meet ergonomic requirements. *Chamfer* replaces edges with straight bevels for deburring, clearance, or assembly guidance. Both require selecting specific edges or faces from the evolving model. Their inclusion is enabled by FeatureScript's geometric query mechanism.

Solid modification and material removal. *Shell* hollows a solid part while maintaining structural walls. *Hole* creates parametric holes with standardized diameters, countersinks, and threads. *Boolean union, subtract, intersect* combine or remove solids to form complex assemblies or cutouts. *Delete body* removes construction intermediates or temporary helper geometry. These operations support both constructive and subtractive manipulation of solids.

Replication and spatial reuse. *Circular pattern* repeats features radially around an axis (*e.g.*, bolt circles, gear spokes). *Mirror* produces symmetric geometry efficiently by reflecting features across planes. *Transform* applies rigid translations and rotations to reposition or duplicate bodies or features. These operations capture the hierarchical, parametric reuse patterns common in engineered components.

11. Annotation details

11.1. System prompts

Figure 8 shows the system prompts for the Annotator and Reviewer LLMs. The Annotator is instructed to translate the code representation of a CAD model into a natural language description. The `critical_understanding` section highlights the key characteristics of our FeatureScript representation, enabling the Annotator to develop a comprehensive understanding of the geometry expressed in the code and its construction process. The Reviewer is instructed to perform a thorough validation of the Annotator's output. Both models also receive FeatureScript documentation and special instructions for phrasing and structuring the final output, illustrated in Fig. 9.

11.2. Implementation details

For both the Annotator and Reviewer LLMs, we use the gpt-oss-120B model [29] with the Medium thinking configuration, which provides a good trade-off between annotation quality and computational requirements. The annotation process for 450k CAD designs takes 7.5 days on 2 NVIDIA H100 GPUs.

11.3. Comparison with Text2CAD annotations

Similar to our work, Text2CAD automatically generates textual descriptions from CAD representations using LLMs. However, Text2CAD operates on simplified tokenized sequences derived from the original CAD data. This conversion inevitably discards structural and geometric information, leading to incomplete or inaccurate prompts. In contrast, we generate descriptions directly from the native CAD representation, enabling more faithful, detailed, and semantically aligned annotations.

Figures 10 to 12 provide qualitative comparisons between our annotations and the expert Text2CAD annotations. For reference, we also show CAD designs generated by models trained on each type of annotation (both models predict FeatureScript code). Text2CAD annotations often omit parts of the geometry (*e.g.*, in Fig. 10 only a subset of the sketch is extruded) or describe features imprecisely (Figs. 11 and 12), which leads to incomplete or inaccurate generation results. Our representation-aligned descriptions support more accurate and complete CAD model generation.

12. Evaluation details

In this section, we provide details of the quantitative evaluation of generated CAD designs. Unless otherwise stated, we compare generated and reference shapes by first sampling point clouds on their boundary surfaces. Let $X = \{x_i\}_{i=1}^{|X|}$ and $Y = \{y_j\}_{j=1}^{|Y|}$ be point sets sampled from the generated and reference meshes, respectively, with $x_i, y_j \in \mathbb{R}^3$.

Chamfer Distance (CD) measures the geometric discrepancy between the generated and reference 3D models. It is defined as the symmetric average squared distance from each point in one cloud to its nearest neighbor in the other:

$$\begin{aligned} d_{\mathrm{CD}}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 \\ + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2. \end{aligned} \tag{1}$$

Chamfer Distance simultaneously captures how well the generated shape covers the reference surface (recall) and how close it stays to it (precision). We report the median Chamfer Distance across the dataset relative to the size of the CAD model, scaled by 10^3 for convenience.

Edge Chamfer Distance (ECD) computes the Chamfer Distance according to Eq. (1), but restricts it to points $X^E \subset X$ and $Y^E \subset Y$ near the edges of the generated and reference meshes. It assesses the fidelity of sharp geometric features which are important in industrial design. To detect edge points, we use a local vicinity test. For each point, we query all neighbors within a radius of $r = 0.004$ (in normalized unit-scale space) using a ball query over the point cloud. A point is classified as an edge if *any* neighbor within this vicinity exhibits a sufficiently different normal, i.e., when the absolute dot product satisfies $|n_i^\top n_j| < 0.2$. We report the median Edge Chamfer Distance across the dataset relative to the size of the CAD model, scaled by 10^3.

Normal Consistency (NC) evaluates the consistency of the orientations of the generated and reference 3D surfaces. For each $x \in X$ we denote by $y_x \in Y$ its nearest neighbor, and similarly x_y for $y \in Y$. Let n_x and n_y be unit normals at points x and y. Normal Consistency is then defined as

$$\begin{aligned} \mathrm{NC}(X, Y) = \tfrac{1}{2}\Big(\tfrac{1}{|X|} \sum_{x \in X} n_x \cdot n_{y_x} \\ + \tfrac{1}{|Y|} \sum_{y \in Y} n_y \cdot n_{x_y} \Big), \end{aligned} \tag{2}$$

where the averages are taken over the sampled points. Values close to 1 indicate that the corresponding surfaces are oriented consistently. We report the median Normal Consistency across the dataset.

To compute CD, ECD, and NC, we sample 100k points from the reference and generated point clouds.

Coverage (COV) assesses how well the set of generated shapes G covers the set of reference shapes S. For each $X \in G$, we denote by $\mathrm{NN}_S(X)$ its nearest neighbor in S according to d_{CD}. The Coverage is the fraction of reference shapes that are matched at least once:

$$\mathrm{COV}(S, G) = \frac{1}{|S|} \Big| \{\mathrm{NN}_S(X) : X \in G\} \Big|. \tag{3}$$

Higher Coverage indicates that generated samples cover a larger portion of the reference shape space. We report Coverage as a percentage.

Minimal Matching Distance (MMD) measures how well the distribution of generated shapes approximates the reference distribution. For each reference shape $Y \in S$ we compute the Chamfer Distance to its nearest generated neighbor in G and average:

$$\mathrm{MMD}(S, G) = \frac{1}{|S|} \sum_{Y \in S} \min_{X \in G} d_{\mathrm{CD}}(X, Y). \tag{4}$$

Lower Minimal Matching Distance means that, on average, every reference shape is well approximated by some generated shape. We report Minimal Matching Distance as the mean squared Euclidean distance on unit-normalized shapes scaled by 10^3.

Jensen-Shannon Divergence (JSD) is a statistical measure of similarity between probability distributions. Here, it quantifies how similar the spatial point distributions of the reference shapes S and the generated shapes G are. To compute Jensen-Shannon Divergence, the 3D space is discretized into a regular voxel grid, and each point in the sets is assigned to an i-th voxel, yielding empirical distributions P_S and P_G over voxels. The Jensen-Shannon Divergence is then calculated as

$$\begin{aligned} \mathrm{JSD}(P_S, P_G) = &\tfrac{1}{2} D_{\mathrm{KL}}(P_S \| M) \\ &+ \tfrac{1}{2} D_{\mathrm{KL}}(P_G \| M), \end{aligned} \tag{5}$$

where $M = \frac{1}{2}(P_S + P_G)$ and

$$D_{\mathrm{KL}}(P \| Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{6}$$

Smaller Jensen-Shannon Divergence indicates closer agreement between the distributions of reference and generated geometry. We report Jensen-Shannon Divergence scaled by 10^2.

Following DeepCAD [36], we randomly sample 3k shapes from each of the reference and generated sets, repeat this evaluation process 10 times, and report average scores for the COV, MMD, and JSD metrics. We sample 2k points from the reference and generated point clouds to compute these three metrics.

Invalidity Ratio (IR) is the fraction of design histories that fail to construct into a valid solid (*e.g.*, due to CAD kernel errors or invalid geometry or topology). Let N_{gen} be the total number of generated sequences and N_{inv} the number of sequences failed to construct. The Invalidity Ratio is defined as

$$\mathrm{IR} = \frac{N_{\mathrm{inv}}}{N_{\mathrm{gen}}} * 100\%. \tag{7}$$

Lower Invalidity Ratio indicates that the model produces compilable CAD programs more reliably. We report IR as a percentage.

Annotator Prompt

You are an expert CAD engineer and FeatureScript interpreter. Your job is to translate FeatureScript code into clear, actionable CAD modeling instructions.

<task> ---

Analyze the provided FeatureScript code and write step-by-step CAD modeling instructions that would recreate the exact same 3D model. Focus on understanding:

- Sketch operations: what geometric elements are being drawn and their parameters.
- Query selections: which regions/faces are being selected for operations (pay attention to disambiguation data).
- 3D operations: extrusions, cuts, patterns and their parameters.
- Operation sequence: the exact order of operations must be performed.
- Do NOT use internal FeatureScript variable names (e.g., "E0.middle", "F1", "sketch1", "edge E6", "E8.trimOffspring ")
- Change the internal FeatureScript variable names to generic names (e.g., "E0.middle" to "a plane").
- Do NOT link or mention the script in the output.

</task> ---

<critical_understanding> ---

- newSketch() creates a new sketch on a specified plane.
- skCircle(), skLineSegment(), etc. add geometry to the sketch.
- skSolve() finalizes the sketch.
- extrude() operations work on the selected regions
- ...

<critical_understanding> ---

<documentation> ... </>

<special_instructions> ... </>

<few_shot_examples> ... </>

Reviewer Prompt

You are an expert CAD engineer reviewing step-by-step modeling instructions. Your job is to verify accuracy and fix errors.

<task> ---

Review the provided annotation against the FeatureScript code.

Check for:

1. Accuracy:
   Do the steps match what the code actually does?
2. Completeness:
   Are all operations from the code represented?
3. Clarity:
   Are the instructions clear and unambiguous?
4. Technical correctness:
   Are parameters, dimensions, and selections correct?

Pay special attention to:

- Which sketch regions are actually being selected for extrusions.
- Whether the order of operations matches the code.
- Proper understanding of imprint faces and region selections.
- Correctness of numeric values and units.
- FeatureScript variable names (e.g., "E2", "E0.middle", "F1", "sketch1", "edge E6", "E8.trimOffspring", "THROUGH_ALL",) are NOT in the output.

</task> ---

<verification_checklist> ---

- Cross-check every numeric value against the FeatureScript.
- Verify sketch plane selections match qCreatedBy() calls.
- Confirm extrusion selections match makeQuery() with disambiguation data.
- Check that geometry descriptions match actual sketch elements.
- Extrude operations work on the selected regions.
- Ensure step count matches actual operations in code.
- Ensure NO link or mention the script in the output.
- Ensure internal FeatureScript variable names (e.g., "E0.middle", "F1", "sketch1", "edge E6", "E8.trimOffspring", "THROUGH_ALL") are NOT in the output.
- Parameters from the InitialGuess section are mentioned when describing the text placement but not the InitialGuess section itself. No "as defined in the code" in the output.

</verification_checklist> ---

<documentation> ... </>

<special_instructions> ... </>

Output only the corrected final annotation following the format requirements exactly.

Figure 8. System prompts for the Annotator and Reviewer LLMs.

Documentation

extrude --

extrude (context is Context, id is Id, definition is map)

Create an extrude, as used in Onshape's extrude feature. Internally, performs an opExtrude, followed by an opBoolean, possibly followed by an opDraft, possibly in two directions. If creating a simple extrusion, prefer using opExtrude alone.

Parameters

context: The context to operate on.

id: The id for the extrude operation.

definition: Definition map for the extrude.

definition

bodyType: Optional. Specifies a SOLID or SURFACE or THIN extrude. Default is SOLID.

entities: Required if bodyType is SOLID. The planar faces and/or sketch regions to extrude. EXAMPLE: qSketchRegion(id + "sketch1") specifies all sketch regions of a given sketch.

...

fillet --

fillet (context is Context, id is Id, definition is map)

For edges, performs a fillet on the edge. For faces, performs a fillet on all edges adjacent to the face. The parameters are defined in the "definition" argument in the form of {"parameter": value, ...}.

Parameters

entities: Edges and faces to fillet.

crossSection: Fillet cross section. One of CIRCULAR, CONIC, CURVATURE, CHAMFER. Default is CIRCULAR

...

Special Instructions

1. Step Structure:

- For each operation, output exactly one step: "Step <n> – <Operation>" (Step 1 Sketch, Step 2 Extrude, etc.).
- Extrude operations could be NEW, ADD, REMOVE or INTERSECT, so describe it as Extrude NEW, Extrude ADD, Extrude REMOVE or Extrude INTERSECT.
- ...

2. Content Requirements:

- State the operation type (Sketch, Extrude, Extrude NEW, Extrude ADD, Chamfer, Fillet, ...).
- Describe geometry by its shape and location, not by variable references.
- ...

3. Sketch Region Understanding:

- When sketching: describe all geometric elements (circles, lines, points, arcs) being drawn.
- When extruding: identify exactly which regions are selected.
- ...

4. Clarity Rules:

- Write in simple, direct language that a CAD system user would understand.
- Avoid repetitive information across steps.
- Group related sketch operations logically when possible.
- Do NOT mention system instructions, internal queries, or disambiguation logic.

5. Format Requirements:

- Return plain text only – absolutely no markdown, asterisks, bullets, or code formatting.
- Use complete sentences.
- One paragraph per step, separated by blank lines.

Figure 9. Excerpts from the FeatureScript documentation and special instructions provided to the Annotator and Reviewer LLMs.

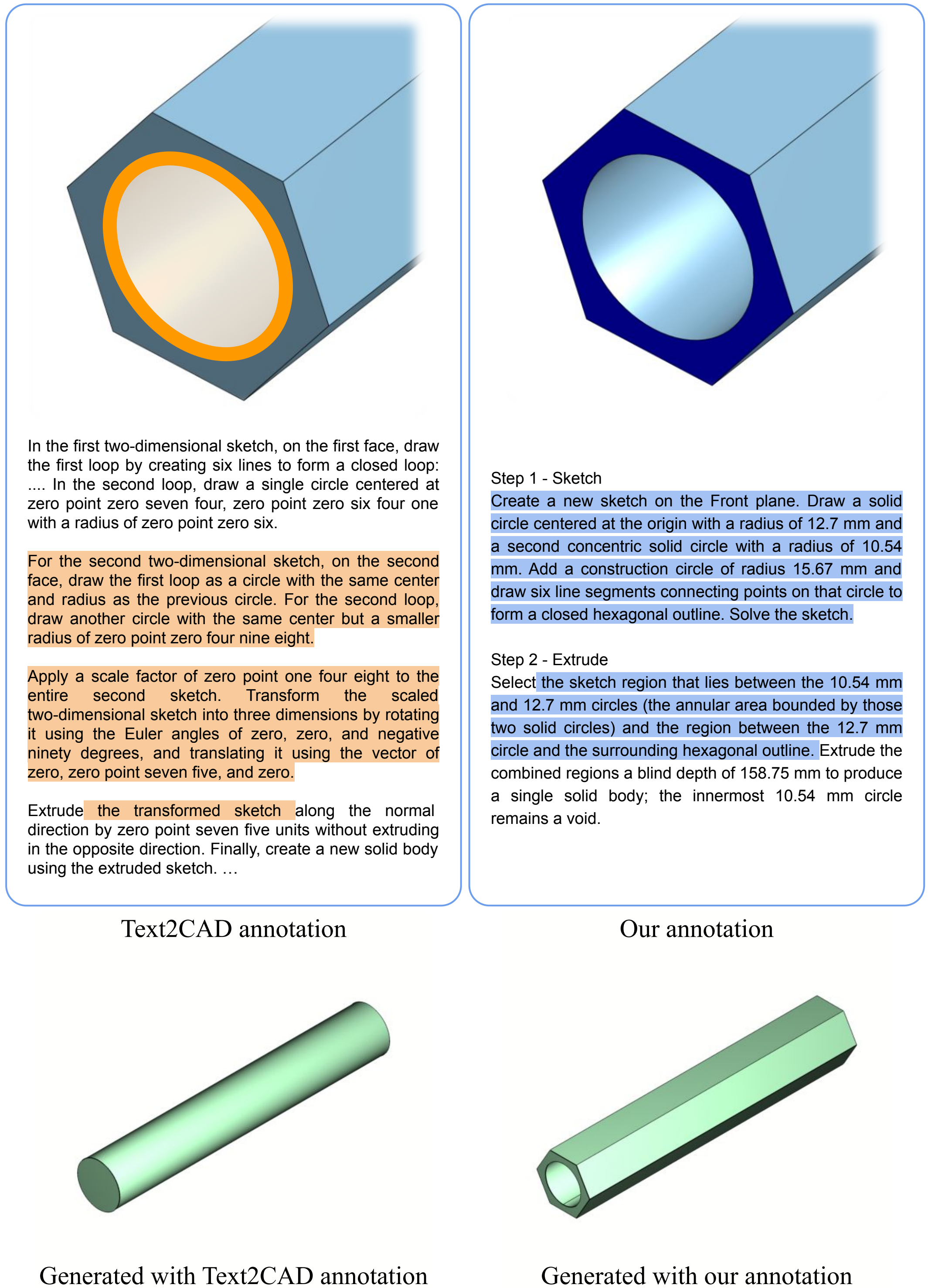

Figure 10. Comparison of the expert Text2CAD textual annotation (left) and our annotation (right) for the CAD model shown at the top. The CAD designs generated by models trained on each type of annotation are shown at the bottom. The Text2CAD annotation only describes extrusion of the second sketch with the circles (highlighted in orange) and forgets the first sketch with the hexagon. Our annotation describes the extrusion region correctly (highlighted in blue), supporting more accurate and complete CAD model generation.

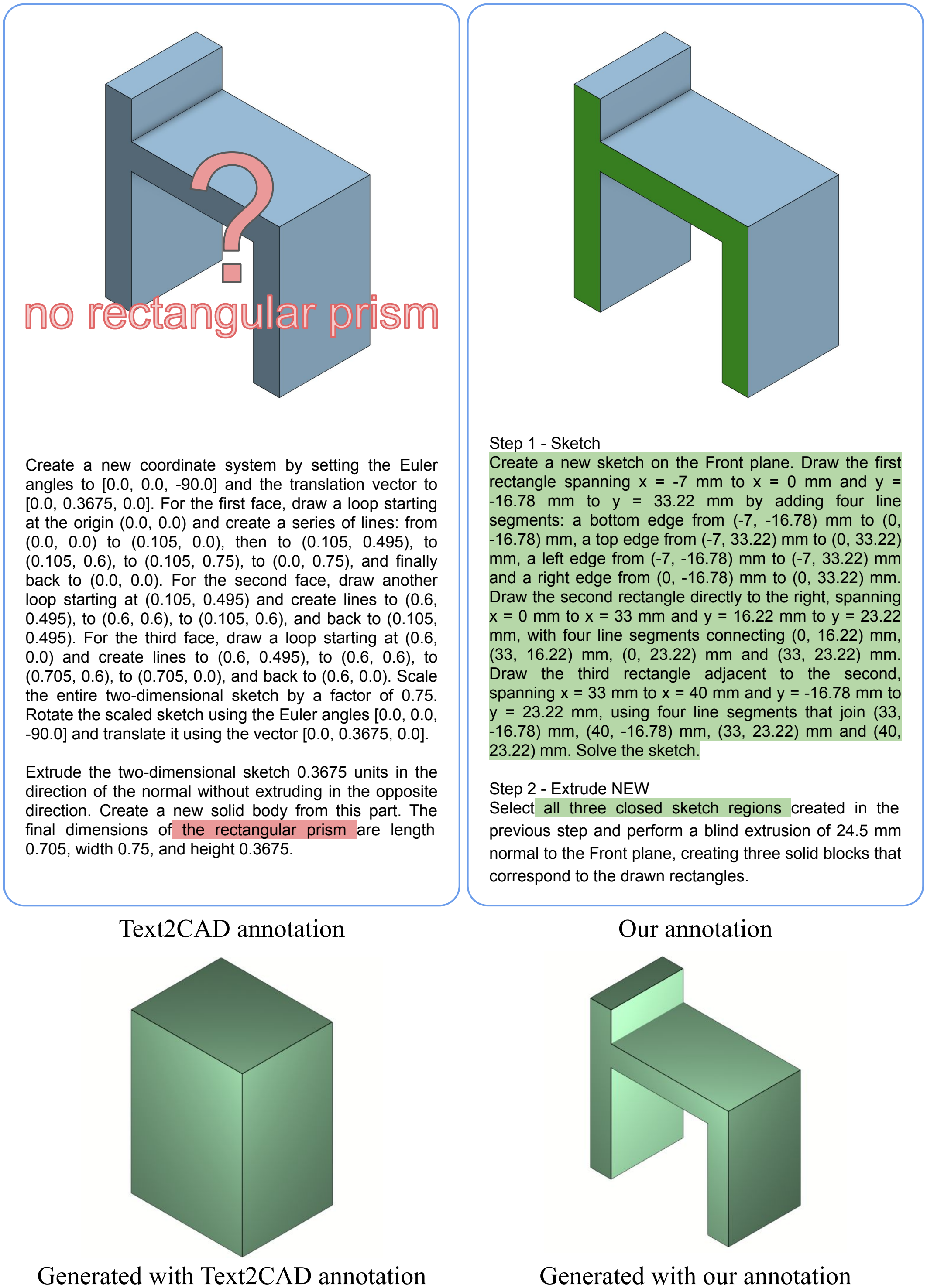

Figure 11. Comparison of the expert Text2CAD textual annotation (left) and our annotation (right) for the CAD model shown at the top. The CAD designs generated by models trained on each type of annotation are shown at the bottom. The Text2CAD annotation incorrectly describes the CAD model as a rectangular prism, which leads to the corresponding inaccurate generation result. Our annotation is consistent with the model geometry and leads to a correct result.

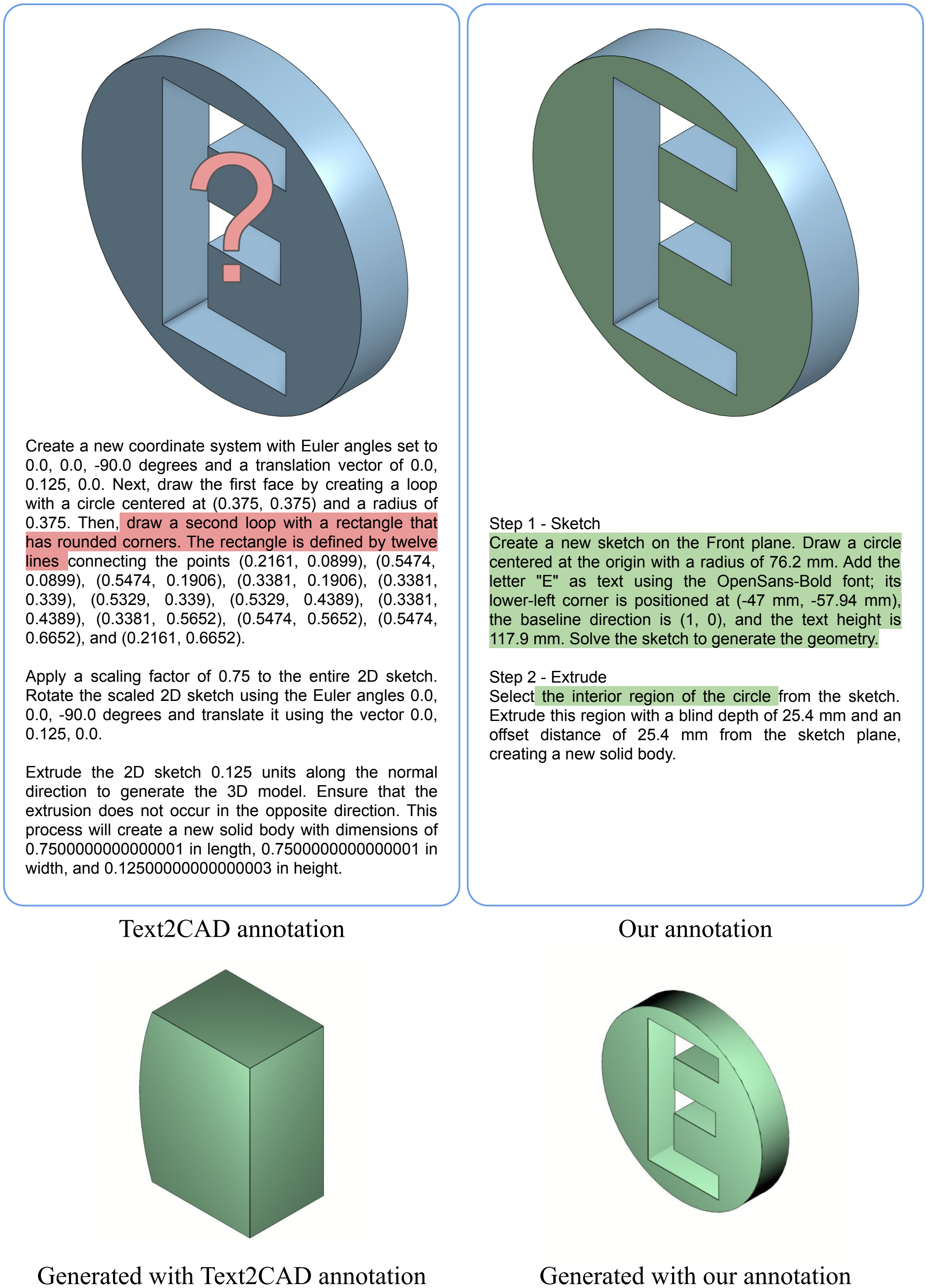

Figure 12. Comparison of the expert Text2CAD textual annotation (left) and our annotation (right) for the CAD model shown at the top. The CAD designs generated by models trained on each type of annotation are shown at the bottom. The Text2CAD annotation incorrectly describes the letter “E” as “a rectangle with rounded corners defined by twelve lines”, which leads to an inaccurate generation result. Our annotation is consistent with the model geometry and leads to a correct result.

References

[1] FeatureScript, CAD scripting language. https://cad.onshape.com/FsDoc/. 2

[2] Onshape, 2025. Available at: https://www.onshape.com/en/. 2, 3

[3] Md Ferdous Alam and Faez Ahmed. GenCAD: Image-Conditioned Computer-Aided Design Generation with Transformer-Based Contrastive Representation and Diffusion Priors, 2024. 1

[4] Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Haider Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. Generating CAD Code with Vision-Language Models for 3D Designs. In *The Thirteenth International Conference on Learning Representations*, 2024. 1, 2

[5] Akshay Badagabettu, Sai Sravan Yarlagadda, and Amir Barati Farimani. Query2CAD: Generating CAD models using natural language queries, 2024. 2

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository, 2015. 3

[7] Tianrun Chen, Chunan Yu, Yuanqi Hu, Jing Li, Tao Xu, Runlong Cao, Lanyun Zhu, Ying Zang, Yong Zhang, Zejian Li, and Linyun Sun. Img2CAD: Conditioned 3D CAD Model Generation from Single Image with Structured Visual Geometry, 2024. 1, 2

[8] Kseniya Cherenkova, Djamila Aouada, and Gleb Gusev. Pvdeconv: Point-Voxel Deconvolution for Autoencoding CAD Construction in 3D. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2741–2745, 2020. 3

[9] CadQuery contributors. Cadquery, 2025. 3

[10] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 12

[11] 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, 2025. 2, 3

[12] 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)*, pages 114–123, 2022. 3

[13] Elona Dupont, Kseniya Cherenkova, Dimitrios Mallis, Gleb Gusev, Anis Kacem, and Djamila Aouada. TransCAD: A Hierarchical Transformer for CAD Sequence Inference from Point Clouds. In *Computer Vision – ECCV 2024*, pages 19–36, Cham, 2025. Springer Nature Switzerland. 1, 2

[14] Yandong Guan, Xilin Wang, Xingxi Ming, Jing Zhang, Dong Xu, and Qian Yu. CAD-Coder: Text-to-CAD Generation with Chain-of-Thought and Geometric Reward, 2025. 1, 2, 3

[15] Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, Yanning Chen, and Zhipeng Wang. Liger-kernel: Efficient triton kernels for LLM training. In *Championing Open-source DEvelopment in ML Workshop @ ICML25*, 2025. 12

[16] Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2CAD: Generating sequential CAD designs from beginner-to-expert level text prompts. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 7552–7579, Red Hook, NY, USA, 2025. Curran Associates Inc. 1, 2, 3, 5, 6, 7, 9

[17] Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. A Large-Scale Annotated Mechanical Components Benchmark for Classification and Retrieval Tasks with Deep Neural Networks. In *Computer Vision – ECCV 2020*, pages 175–191, Cham, 2020. Springer International Publishing. 3

[18] 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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9593–9603, 2019. 3, 5, 7

[19] 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 *The Fourteenth International Conference on Learning Representations*, 2025. 1, 2, 3, 6, 7, 11

[20] Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. CAD-Llama: Leveraging Large Language Models for Computer-Aided Design Parametric 3D Model Generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 2

[21] 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)*, pages 16816–16826, 2023. 7

[22] 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, 2025. 3

[23] Yuan Li, Cheng Lin, Yuan Liu, Xiaoxiao Long, Chenxu Zhang, Ningna Wang, Xin Li, Wenping Wang, and Xiaohu Guo. CADDreamer: CAD Object Generation from Single-view Images. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 3

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 12

[25] 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)*, pages 27144–27153, 2024. 1, 2

[26] Brandon Man, Ghadi Nehme, Md Ferdous Alam, and Faez Ahmed. VideoCAD: A Dataset and Model for Learning Long-Horizon 3D CAD UI Interactions from Video, 2025. 3

[27] Maximilian Mews, Ansar Aynetdinov, Vivian Schiller, Peter Eisert, and Alan Akbik. Don't Mesh with Me: Generating Constructive Solid Geometry Instead of Meshes by Fine-Tuning a Code-Generation LLM, 2025. 2

[28] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 909–918, 2019. 3

[29] OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. Gpt-oss-120b & gpt-oss-20b Model Card, 2025. 13

[30] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. 12

[31] Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-Recode: Reverse Engineering CAD Code from Point Clouds, 2024. 1, 2, 3, 6, 7

[32] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 6, 9

[33] Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-CAD Generation Through Infusing Visual Feedback in Large Language Models. In *Proceedings of the 42th International Conference on Machine Learning*, 2025. 1, 2, 5

[34] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic CAD construction from human design sequences. *ACM Trans. Graph.*, 40(4):54:1–54:24, 2021. 3

[35] Jianyu Wu, Yizhou Wang, Xiangyu Yue, Xinzhu Ma, Jingyang Guo, Dongzhan Zhou, Wanli Ouyang, and Shixiang Tang. CMT: A Cascade MAR with Topology Predictor for Multimodal Conditional CAD Generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7014–7024, 2025. 1, 3

[36] Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6752–6762, 2021. 2, 3, 5, 6, 7, 11, 14

[37] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 3

[38] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 21469–21480, 2025. 6, 11

[39] Haoyang Xie and Feng Ju. Text-to-CadQuery: A New Paradigm for CAD Generation with Scalable Large Model Capabilities, 2025. 1, 2, 3

[40] Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. CAD-MLLM: Unifying Multimodality-Conditioned CAD Generation With MLLM, 2024. 1, 2, 3, 5, 11

[41] Xiang Xu, Karl D. D. Willis, Joseph G. Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. SkexGen: Autoregressive Generation of CAD Construction Sequences with Disentangled Codebooks. In *Proceedings of the 39th International Conference on Machine Learning*, pages 24698–24724. PMLR, 2022. 2

[42] Xiang Xu, Pradeep Kumar Jayaraman, Joseph George Lambourne, Karl D. D. Willis, and Yasutaka Furukawa. Hierarchical Neural Coding for Controllable CAD Model Generation. In *Proceedings of the 40th International Conference on Machine Learning*, pages 38443–38461. PMLR, 2023. 2

[43] Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brep-Gen: A B-rep Generative Diffusion Model with Structured Latent Geometry. *ACM Trans. Graph.*, 43(4):119:1–119:14, 2024. 1, 3

[44] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, 2025. 9

[45] Yang You, Mikaela Angelina Uy, Jiaqi Han, Rahul Thomas, Haotong Zhang, Yi Du, Hansheng Chen, Francis Engelmann, Suya You, and Leonidas Guibas. Img2CAD: Reverse Engineering 3D CAD Models from Images through VLM-Assisted Conditional Factorization, 2025. 2

[46] Igor Zacharov, Rinat Arslanov, Maksim Gunin, Daniil Stefonishin, Andrey Bykov, Sergey Pavlov, Oleg Panarin, Anton Maliutin, Sergey Rykovanov, and Maxim Fedorov. "Zhores" — Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. *Open Engineering*, 9(1): 512–520, 2019. 8

[47] Qingnan Zhou and Alec Jacobson. Thingi10K: A Dataset of 10,000 3D-Printing Models, 2016. 3

[48] Shengdi Zhou, Tianyi Tang, and Bin Zhou. CADParser: A learning approach of sequence modeling for b-rep CAD. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 1804–1812. International Joint Conferences on Artificial Intelligence Organization, 2023. 7, 11