CADBENCH: A MULTIMODAL BENCHMARK FOR AI-ASSISTED CAD PROGRAM GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recovering CAD programs from images or 3D observations is central to AI-assisted design, but progress is difficult to measure because existing evaluations are fragmented across datasets, modalities, and metrics. We introduce CADBENCH, a unified benchmark for multimodal CAD program generation. CADBENCH contains 18,000 evaluation samples spanning six benchmark families derived from DeepCAD, Fusion 360, ABC, MCB, and Objaverse; five input modalities including clean meshes, noisy meshes, single-view renders, physically-based renders, and multi-view renders; and metrics for geometric fidelity and code quality, including executability, compactness, and editability. STEP-based families are stratified by B-rep face count and all families are diversity-sampled to support controlled analysis across complexity and object variation. We benchmark eleven CAD-specialized and general-purpose vision-language systems, generating more than 1.4 million CAD programs. CADBENCH exposes notable variation in model performance across input modalities, geometric complexity, and evaluation criteria. Three recurring failure modes emerge: reconstruction quality degrades with geometric complexity, CAD-specialized models can be brittle under modality shift, and model rankings change across metrics. Together, these results position CADBENCH as a diagnostic testbed for measuring progress in CAD program generation and multimodal CAD understanding.

1 INTRODUCTION

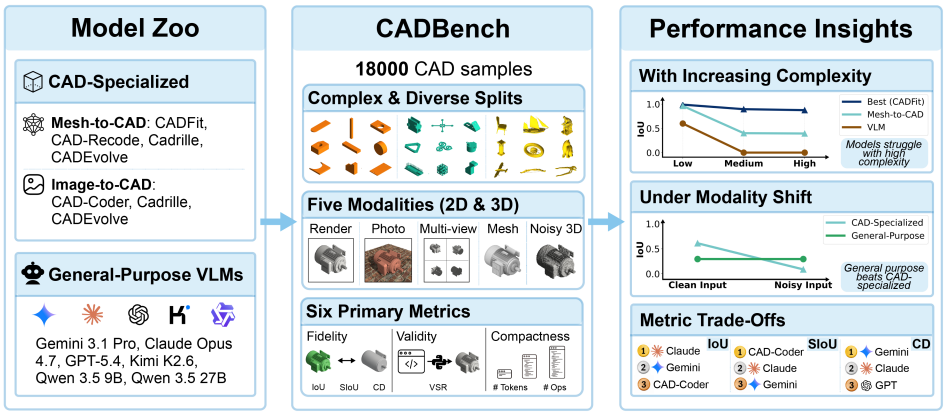


Figure 1: **Overview of CADBENCH.** CADBENCH evaluates eleven CAD-specialized and general-purpose models (left) on 18,000 CAD samples across six families, complexity- and diversity-sampled splits, five input modalities, and geometric fidelity and code-quality metrics (center). It diagnoses three axes (right): performance versus geometric complexity, robustness to modality shift, and metric-dependent model rankings.

Computer-aided design (CAD) models encode solid geometry in structured, parametric programs, enabling precise control over design form in engineering workflows. However, creating such models remains time-consuming and expertise-intensive: prior studies estimate that even experienced

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.

engineers may require on the order of weeks to construct detailed CAD models for real-world components such as aerospace parts (Roy et al., 2001). This cost has motivated significant interest in AI systems that generate CAD programs directly from user inputs (Zhang et al., 2026). Unlike approaches that reconstruct only final 3D geometry, such as meshes, voxels, or point clouds, CAD program generation aims to produce outputs that can be modified, re-parameterized, and integrated into design and manufacturing pipelines. Recent work has explored CAD generation from diverse inputs, including text descriptions (Du et al., 2024; Khan et al., 2024a; Xie & Ju, 2025), 2D images or drawings (Elistratov et al., 2026; Doris et al., 2026; Yu et al., 2025; Li et al., 2025), and 3D observations such as meshes or point clouds (Kolodiazhnyi et al., 2025; Rukhovich et al., 2025; Alam & Ahmed, 2025; Xu et al., 2024; Karadeniz et al., 2025; Khan et al., 2024b). These approaches span general-purpose multimodal models (Makatura et al., 2023; Picard et al., 2025), fine-tuned language models (Du et al., 2024; Doris et al., 2026), and CAD-specific architectures (Alam & Ahmed, 2025; Para et al., 2021).

Benchmarks have played a critical role in the development of AI systems: shared evaluations such as ARC (Chollet, 2019) and MMLU-Pro (Wang et al., 2024) for abstract reasoning and language understanding, SWE-bench (Jimenez et al., 2024) for software engineering, and SpatialBench (Xu et al., 2025) for spatial cognition have exposed model failure modes and guided iterative improvement. Despite rapid progress in AI-driven CAD generation, however, evaluation has not kept pace. Existing CAD generation methods are typically assessed on narrow datasets or test splits, such as DeepCAD (Wu et al., 2021), Fusion 360 Gallery (Willis et al., 2021), and MCB (Kim et al., 2020). Methods are often assessed on different input modalities and metrics, making direct comparison between approaches difficult. Current benchmarks also rarely control for complexity and diversity, often overestimating the performance of CAD systems on relatively simple or uniform examples. They frequently rely on clean, CAD-derived inputs, such as idealized renders or exact geometry, failing to quantify how well models perform in scenarios more reflective of downstream CAD reconstruction tasks. Finally, existing evaluations are often limited to one or a few notions of accuracy, even though CAD program quality depends on multiple aspects of geometric fidelity, program executability, and program compactness. As a result, evaluations provide limited diagnostic insight into three questions: how performance changes with increasing geometric complexity, how robust models are to modality shifts, and how model rankings and failure modes vary across CAD-specific evaluation metrics.

We introduce CADBENCH (Figure 1), a unified benchmark for CAD program generation comprising 18,000 samples across six benchmark families, five 2D and 3D input modalities, and a suite of geometric fidelity and code-quality metrics. CADBENCH addresses the gaps in current CAD evaluation through three core components: complexity- and diversity-sampled splits for controlled analysis across geometric difficulty and benchmark families; clean and noisy input modalities for measuring robustness to downstream reconstruction settings; and a metric suite that jointly captures volumetric fidelity, surface alignment, local geometric error, executability, and program compactness, complemented by measures of program editability. Using CADBENCH, we evaluate eleven CAD-specialized and general-purpose models across more than 1.4 million generations, revealing how current systems degrade with increased geometric complexity, respond to modality shifts, and vary across evaluation metrics. Together, these design choices enable a diagnostic evaluation of CAD generation systems. Our contributions are:

- We introduce CADBENCH, a multimodal benchmark for CAD program generation comprising **18,000 samples** across six benchmark families, five input modalities, and metrics that measure geometric fidelity and code quality.
- We construct complexity- and diversity-aware evaluation splits that enable controlled analysis of model performance across geometric complexity and benchmark families.
- We incorporate clean and noisy 2D and 3D input modalities, including rendered images, physically-based renders, multi-view images, clean meshes, and noisy meshes, enabling evaluation of model robustness to modality shift.
- We define a multi-dimensional metric suite spanning volumetric IoU, surface IoU, Chamfer distance, valid shape rate, token count, operation count, and program-editability measures, capturing complementary aspects of CAD generation performance.
- We conduct a large-scale empirical study of eleven CAD-specialized and general-purpose models across more than **1.4 million generations**, identifying key failure modes including degradation with geometric complexity, sensitivity to modality shift, and metric-dependent differences in model performance.

2 Related Work

Prior work has explored related domains such as text-to-CAD generation, including BlenderLLM (Du et al., 2024), Text2CAD (Khan et al., 2024a), and CADPrompt (Alrashedy et al., 2025). While these methods generate CAD models from natural language descriptions, engineering design is inherently geometric and visual, and many downstream workflows begin from 2D or 3D observations. Moreover, text-to-CAD is often underspecified, whereas image-to-CAD and mesh-to-CAD reconstruction can be evaluated against objective geometric ground truth. In this work, we therefore focus on settings in which models reconstruct executable CAD programs from image or mesh inputs.

Table 1: Comparison of CADBENCH against existing, commonly used benchmarks for assessing CAD generation performance. Notably, CADBENCH has higher complexity (as measured by average face count) and diversity (as measured by pairwise cosine similarity); see Appendix B for details on these metrics. Many existing benchmarks are test splits of existing datasets, so the number of samples corresponds to the test set size. For existing datasets, we report the 2D modalities, 3D modalities, and reconstruction metrics provided in the original dataset; works that extend these datasets to additional modalities are discussed in Section 2.

Benchmark	Samples	Complexity Splits	Avg. Face Complexity ↑	Diversity Sampling	Avg. Similarity ↓	2D Modalities	3D Modalities	Reconstruction Metrics
CADBENCH (Ours)	18,000	✓	118*	✓	0.417	SV render MV render PBR	STEP Mesh Noisy Mesh	IoU, SIoU CD, VSR Token Count, Op. Count
DeepCAD (Wu et al., 2021)	8,052	✗	13	✗	0.533	None	STEP Mesh	ACC_{cmd}, ACC_{param}, CD, IR
Fusion 360 (Willis et al., 2021)	1,725	✗	16	✗	0.456	None	PC	IoU, Exact Reconst., Concise.
MCB (Kim et al., 2020)	11,700	✗	N/A*	✗	0.503	None	Mesh	None
CC3D (Cherenkova et al., 2020)	5,000	✗	N/A†	✗	N/A†	None	STEP 3D Scan	Chamfer Distance (CD)
Omni-CAD (Xu et al., 2024)	∼27K	✗	26	✗	0.483	Multi-view Images	STEP** Point Cloud	CD, F-score, Normal Consistency, Segment Error (SegE), Dangling Edge Length (DangEL), Self-Intersection Ratio (SIR), Flux Enclosure Error (FluxEE)

Notes. * Face counts are reported only for STEP-based splits and datasets, since face count is not a meaningful measure of complexity for mesh-based data. † CC3D is listed as related benchmark context but is not included in CADBENCH aggregate metrics because access requires a separate institutional license agreement.

Complexity and Diversity of Existing CAD Benchmarks. In the absence of standardized benchmarks for modality-to-CAD program generation, prior work has typically evaluated models on dataset-specific test splits from existing CAD datasets, including DeepCAD (Wu et al., 2021), Fusion 360 Gallery (Willis et al., 2021), MCB (Kim et al., 2020), and CC3D (Cherenkova et al., 2020). These datasets have enabled substantial progress, particularly when paired CAD construction sequences are available for training. Table 1 summarizes commonly used CAD benchmarks and reports dataset-level measures of geometric complexity and visual diversity.

DeepCAD (Wu et al., 2021), Fusion 360 Gallery (Willis et al., 2021), and Omni-CAD (Xu et al., 2024) datasets provide human-generated CAD samples, but CAD operations are restricted to sketch-and-extrude only, limiting both the complexity of samples and types of operations represented in evaluation. MCB (Kim et al., 2020) contains realistic mechanical components, but evaluation on this dataset remains constrained to a fixed set of 68 object categories. Other more operation-rich and geometrically diverse CAD datasets, such as CC3D (Cherenkova et al., 2020) and ABC (Koch et al., 2019), have often been overlooked for benchmarking, perhaps since ground truth design histories are not directly available. CADBENCH's primary benchmark should be independently reproducible from released artifacts without third-party institutional data-use negotiations, which is why we exclude CC3D. Details are provided in Appendix A.2.

This reveals a clear gap: existing benchmarks are typically tied to a single dataset source, constraining evaluation to particular CAD operation types, complexity ranges, or object categories. This is misaligned with downstream needs, where CAD generation models must operate across diverse and complex generation problems. Moreover, existing benchmarks do not curate splits explicitly for complexity and diversity, limiting their diagnostic value for identifying model failure modes.

Modalities in Existing CAD Benchmarks. To measure image-to-CAD reconstruction, several works have augmented the DeepCAD, Fusion 360, and MCB datasets with image modalities. For example,

Alam & Ahmed (2025); Wang et al. (2025) use single-view grayscale renders of CAD solids, while Kolodiazhnyi et al. (2025); Elistratov et al. (2026) generate multi-view renders tailored for specific model training paradigms. Evaluation on realistic imagery is less common. For instance, Doris et al. (2026) evaluate generalizability using a small set of five real photographs, while Li et al. (2025) train and evaluate on synthetic RGB renderings with textures and backgrounds, and additionally test on real photographs of physical objects to assess generalization to real-world imagery.

For 3D-to-CAD reconstruction, a number of studies, including Rukhovich et al. (2025); Karadeniz et al. (2025); Khan et al. (2024b); Uy et al. (2022); Dupont et al. (2024), use point clouds derived from ground-truth CAD solids from datasets such as DeepCAD and Fusion 360. However, evaluation on real or noisy data remains uncommon. Cherenkova et al. (2020) provide a dataset of over 50,000 models virtually scanned to simulate realistic sensor noise, and Yu et al. (2025) use real scans of 3D-printed parts to investigate performance on real-world inputs. Furthermore, Omni-CAD (Xu et al., 2024) introduces a large-scale multimodal dataset of over 400,000 samples, integrating text, multi-view images, and point clouds, while also evaluating model adaptability to degraded inputs such as noisy or cropped point clouds.

Overall, current 2D- and 3D-to-CAD evaluations rely heavily on idealized inputs, typically CAD-derived renders, point clouds, or meshes. Comprehensive evaluation on modalities characteristic of downstream real-world deployment, such as photographs or 3D scans, remains limited.

Metrics for CAD Benchmarking. In most existing works on generative models for CAD, as well as the benchmarks used to test them, performance metrics focus mostly on geometric similarity—namely, Chamfer Distance (CD) and Intersection over Union (IoU) (Alam & Ahmed, 2025; Wu et al., 2021; Kolodiazhnyi et al., 2025; Elistratov et al., 2026; Rukhovich et al., 2025; Doris et al., 2026). Moreover, these geometric comparison metrics may not fully encompass how well a target geometry is being reconstructed, as such macro-scale metrics simply lack the fidelity to capture fine details (e.g., fillets on edges). Most notably, the approach used to measure these metrics is inconsistent across publications. Studies use varying numbers of points sampled on the geometry to measure CD, or employ different alignment strategies before measurement. For example, some align bounding box corners (Yu et al., 2025), some align the centroids of bounding boxes (Kolodiazhnyi et al., 2025; Elistratov et al., 2026; Rukhovich et al., 2025), and more recently, others use continuous Procrustes analysis to align (Doris et al., 2026). Together, these limitations motivate a unified evaluation protocol with complementary CAD-specific metrics and consistent measurement procedures.

3 METHODS

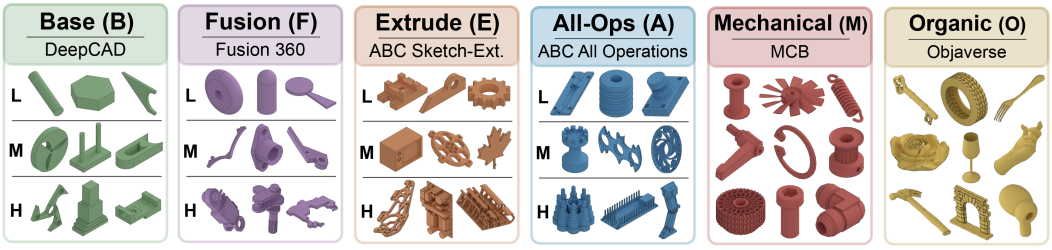


Figure 2: **Overview of CADBENCH families and complexity splits.** CADBENCH is organized into six benchmark families: CAD-Base (B), CAD-Fusion (F), CAD-Extrude (E), CAD-All-Ops (A), CAD-Mechanical (M), and CAD-Organic (O).

Benchmark Construction. Benchmark construction proceeds by selecting source datasets, stratifying and sampling shapes to control complexity and diversity, and generating standardized input modalities.

Benchmark Families. CADBENCH is constructed to evaluate CAD program generation across both CAD reconstruction settings and more challenging out-of-distribution geometry. We therefore curate six benchmark families from existing CAD and 3D object datasets, spanning sketch-and-extrude CAD programs, richer CAD operation histories, mechanical components, and everyday object-like geometry (Figure 2). Preprocessing steps for each family are detailed in Appendix A.3. These steps establish the initial corpus, after which a multi-stage pipeline for complexity stratification and diversity-aware sampling is applied to finalize the benchmark splits.

Complexity Stratification and Diversity-Aware Sampling. We partition the STEP-based benchmark families (B, F, E, A) into Low (L), Medium (M), and High (H) complexity splits using B-rep face count as a proxy for geometric complexity. Face count provides a consistent measure across STEP-based datasets and reflects the number of surfaces needed to define a part; prior work identifies it as one of the geometry-based measures most strongly correlated with expert-perceived CAD modeling complexity (Contero et al., 2023). While initial splits are established by partitioning the logarithmic range of face counts into three equal intervals, we iteratively adjust the thresholds to account for right-skewed distributions, ensuring each bin contains at least 1,000 samples. This stratification is omitted for datasets with only ground-truth meshes (M, O), where face count primarily reflects tessellation density rather than underlying geometric complexity.

To promote geometric diversity, we employ a sampling pipeline using embeddings extracted via the procedure in Appendix A.4. The sampler is designed to prevent visually near-identical geometries from dominating the benchmark. We focus on geometric rather than programmatic diversity, since distinct construction programs can yield nearly identical shapes, and operation sequences are unavailable for some CADBENCH splits. For families with complexity stratification (B, F, E, A), we perform k-means clustering ($k = 1{,}000$) within each complexity tier, totaling 3,000 samples per family. For the remaining families (M, O), we apply k-means ($k = 3{,}000$) to the entire pool. After computing the cluster centroids, we select the nearest-neighbor sample to each centroid to serve as the representative sample. Ultimately, this dual approach ensures that the benchmark is both stratified by geometric complexity and more evenly distributed across the embedding space. Complexity thresholds and diversity quantification for all families are provided in Appendix B (Table 7).

Generating Mesh and Image Input Modalities. To evaluate model capabilities across both controlled and shifted input conditions, we construct five input modalities for each sample in CADBENCH. For image-based evaluation, we generate: (i) *single-view grayscale renders*, which provide one standardized isometric view; (ii) *multi-view grayscale renders*, which combine four isometric views into a single image; and (iii) *physically-based renders* (PBR), which introduce variation in lighting, material, perspective, and background. For 3D evaluation, we create: (iv) *clean meshes* derived from the ground-truth geometry; and (v) *noisy meshes*, obtained by perturbing the clean mesh geometry. Further details regarding modality generation are provided in Appendix A.5.

Evaluation Metrics. We evaluate model-generated CAD programs along four axes: *geometric fidelity*, *executability*, *program compactness*, and *program editability*.

Geometric Fidelity. We report three complementary measures of geometric reconstruction quality. **Volumetric IoU (IoU)** measures overlap between voxelized occupancies of the predicted shape and ground-truth shape, capturing global volumetric agreement. **Chamfer Distance (CD)** is computed between point samples from the predicted and target surfaces, capturing local surface error. **Surface IoU (SIoU)** measures bidirectional surface coverage under a distance threshold. Together, these metrics distinguish volumetric overlap, local geometric deviation, and thresholded surface alignment. We report these metrics after applying the alignment procedure described in Appendix A.6. For benchmark splits, we report median IoU and median SIoU with failed executions counted as zero and median CD with failures penalized by the length of the normalized bounding box diagonal.

Executability. We measure whether a generated CAD program can be executed to produce a valid solid. We report **Valid Shape Rate (VSR)**, the fraction of generated programs that execute successfully and yield a valid shape. Invalid predictions include programs with syntax errors, failed CAD operations, or outputs that cannot be converted into valid solid geometry for evaluation.

Program Compactness. To characterize the degree to which a model reconstructs geometry using concise syntax, we report two measures: **Token Count**, the number of generated tokens, and **Operation Count**, the number of CAD operations in the generated program.

Program Editability. We report three reference-free measures of whether generated programs are practical to modify. **Parameterization Ratio** is the fraction of numeric arguments at CadQuery geometry calls that are supplied through named variables or expressions rather than bare literals. **DoF Compression** measures how few independent parameters control the N numeric arguments, penalizing programs that merely wrap each literal in its own variable. **Edit Robustness** is the fraction of programs that are executable and still produce a watertight solid after every numeric degree of freedom is independently perturbed by $\pm 10\%$.

Aggregate benchmark scores are computed as weighted averages of split-level metrics, with weights proportional to the number of samples in each split. Further information on the definitions and implementation of all metrics is provided in Appendices A.6 and C.5.

Models Evaluated. We focus on models that generate executable CAD programs from image or mesh inputs, as such outputs can be integrated directly into downstream engineering workflows. Consequently, we categorize our evaluation and analysis based on the input modality: i) mesh-conditioned and ii) image-conditioned models.

Mesh-Conditioned Models. We define mesh-conditioned methods as models capable of receiving a mesh as input, often internally converting it into another representation, such as a point cloud or rendered views, in order to generate a CAD program or feature tree. For mesh-conditioned CAD program generation, we evaluate CADFit (Nehme et al., 2026), CADEvolve (Elistratov et al., 2026), CAD-Recode (Rukhovich et al., 2025), and Cadrille (Kolodiazhnyi et al., 2025). We evaluate these methods because they are among the state of the art, demonstrating top-tier reconstruction performance (mean/median IoU ≥ 0.8) on established benchmarks such as DeepCAD and Fusion 360 Reconstruction test sets. Each mesh-conditioned model is evaluated on both clean and noisy mesh modalities using the method's default inference settings. With the exception of CADFit, which is deterministic, each mesh-conditioned method is evaluated over three independent runs, and we report the mean and standard deviation across runs.

Although Cadrille and CADEvolve can also be run from image inputs—and we evaluate both methods on CADBENCH's single-view renders—we primarily report them as mesh-conditioned methods because they achieve their strongest CADBENCH performance when evaluated from mesh-derived inputs. Performance degrades substantially when image inputs are not tailored to each model's native rendering format; this finding is further discussed in Section 4.

Image-Conditioned Models. Image-conditioned models take one or more images as input and generate CAD programs. We further divide these into (1) general-purpose vision-language models (VLMs) and (2) CAD-specific image-to-CAD models.

General-Purpose VLMs. We evaluate a set of frontier proprietary VLMs, including Claude Opus 4.7 (Anthropic, 2026), Gemini 3.1 Pro (Google DeepMind, 2026), and GPT-5.4 (OpenAI, 2026). These models are selected as representative state-of-the-art VLMs from Anthropic, Google, and OpenAI, respectively, based on strong performance on general multimodal reasoning benchmarks (e.g., MMMU-Pro (Yue et al., 2025)). We include Kimi K2.6 (Kimi Team, 2025), an open-weight model, due to strong performance on the same benchmark. We also include open-weight models Qwen3.5 27B and Qwen3.5 9B (Qwen Team, 2026), as the Qwen family has been widely adapted for multimodal and code generation tasks. For all off-the-shelf VLMs, we use recommended inference settings; full details are provided in Appendix A.7. For each of the three image modalities, each model is evaluated over three independent runs, and we report mean and standard deviation.

CAD-Specific Models. For CAD-specialized, image-conditioned generation, we report CAD-Coder (Doris et al., 2026) as the primary image-conditioned CAD-specific model. We also evaluate Cadrille and CADEvolve on image inputs for completeness, but primarily categorize them as mesh-conditioned methods as explained above. Although each model was originally trained on a specific image format, we do not tailor inputs to these model-specific formats at test time. Instead, we evaluate all models using the unified set of renderings described in Section 3: single-view, multi-view, and physically-based renders. CAD-Coder is deterministic and run once per modality.

4 RESULTS AND ANALYSIS

CADBENCH serves both as an aggregate leaderboard and as a diagnostic benchmark for analyzing CAD generation performance. We first summarize overall model performance under clean input conditions. We then use CADBENCH's comprehensive splits, modalities, and metrics to examine how performance changes with geometric complexity, how models respond to input modality shift, and whether different evaluation metrics reveal distinct failure modes.

How do current CAD methods perform under clean inputs? Table 2 summarizes overall model performance on CADBENCH under clean input conditions: clean meshes for mesh-to-CAD models and single-view grayscale renders for image-to-CAD models. Detailed per-split results are provided

in Appendix C.1.1. The results reveal substantial variation in reconstruction quality and executability across models. Among mesh-conditioned models, CADFit achieves the strongest reconstruction performance, with the highest IoU, highest SIoU, highest VSR, and lowest CD, though this accuracy comes with substantially longer inference time (Appendix C.1.2). Among image-conditioned models, performance varies across both models and metrics: Gemini 3.1 Pro achieves the highest aggregate IoU and lowest Chamfer distance, while CAD-Coder achieves the highest SIoU. CAD-Coder also achieves the highest valid shape rate among image-conditioned methods, suggesting that CAD-specific training improves executability even when geometric fidelity remains limited. In contrast, the Qwen models achieve near-zero IoU largely because they often fail to produce valid executable shapes, indicating that general code fluency does not necessarily translate into reliable CAD syntax. Notably, the open-weight Kimi K2.6 model outperforms the closed-source GPT-5.4 model on all three geometric fidelity metrics.

Table 2: **Aggregate CADBENCH performance under idealized input modalities.** Scores aggregate performance across all benchmark splits. Mesh- and image-conditioned results are not directly comparable, as the two input types carry different amounts of geometric information (Section 4). Best values within each modality are bolded (except compactness).

Model	IoU ↑	SIoU ↑	CD ↓	VSR ↑	Token Count ↓	Op Count ↓	Param. Ratio ↑	DoF Compression ↑	Edit Robustness ↑
Mesh-to-CAD									
CADFit	**0.855**	**0.768**	**0.040**	**0.965**	553	42	0.000	0.000	0.370
CAD-Recode	0.661	0.601	0.066	0.895	236	25	0.000	0.000	0.614
CADEvolve	0.708	0.610	0.067	0.950	284	22	0.000	0.000	**0.811**
Cadrille	0.672	0.603	0.063	0.913	220	23	0.000	0.000	0.640
Image-to-CAD									
Claude Opus 4.7	0.437	0.319	0.125	0.860	198	15	0.726	0.193	0.920
Gemini 3.1 Pro	**0.449**	0.297	**0.120**	0.804	200	13	0.774	0.069	0.864
GPT-5.4	0.174	0.113	1.821	0.566	236	16	0.830	0.204	0.877
Kimi K2.6	0.254	0.165	0.729	0.657	248	19	**0.842**	0.289	0.898
Qwen3.5 27B	0.008	0.004	3.290	0.345	160	13	0.788	0.282	0.948
Qwen3.5 9B	0.000	0.000	3.464	0.198	170	15	0.689	**0.314**	**0.966**
CAD-Coder	0.345	**0.344**	0.171	**0.928**	364	43	0.000	0.000	0.721

How much of the mesh–image gap reflects input information? The cross-modal differences observed above should be interpreted in light of the unequal information provided by the input modalities. Clean meshes expose the complete 3D surface, whereas single-view images provide only partial projections and introduce depth and occlusion ambiguities. Mesh-to-CAD and image-to-CAD therefore represent distinct practical settings rather than equivalent tasks, making it difficult to attribute their performance differences solely to model capability. Complete disentanglement of input information from architecture is further complicated because models are typically co-designed with their input modality. To examine the contribution of the input condition, we conduct a same-checkpoint ablation with Cadrille, a multimodal model that supports both point-cloud and image-conditioned inference. Using the same checkpoint, replacing a point cloud sampled from the input mesh with Cadrille's native mesh-derived image input reduces IoU from 0.672 to 0.559 (ΔIoU = -0.113). This controlled comparison provides evidence that input modality contributes to the observed performance differences, although the result is specific to Cadrille. At the same time, model-side differences remain evident: under standardized single-view testing, IoU varies from 0.000 to 0.449 for VLMs. Together, these results indicate that cross-modal performance differences reflect both the information available in the input and the capabilities of the underlying model.

Does CAD reconstruction get harder as geometry becomes more complex? To better understand how geometric complexity affects reconstruction fidelity, we analyze benchmark families containing only sketch-and-extrude operations (B, E). Figure 3 shows that IoU generally decreases as median face count increases for both mesh-conditioned and image-conditioned methods. This suggests that face count captures a meaningful axis of reconstruction difficulty: models that perform well on low-face-count parts often degrade sharply on higher-face-count objects. CADFit is the main exception, remaining comparatively robust across face-count levels.

However, face count alone does not fully characterize reconstruction difficulty. While IoU generally decreases with increasing face count in Figure 3, this trend reverses for the highest face-count Extrude split. Across most evaluated methods, IoU is slightly higher on the high-complexity split (E_H) than on the medium-complexity split (E_M). One possible explanation is that, beyond a certain

threshold, increases in face count are increasingly driven by repeated fine-scale features or patterns rather than more complex global geometry, and we provide supporting evidence for this hypothesis in Appendix C.2.1. We also examine the relationship between face count and reconstruction performance across all CAD operations in Appendix C.2.2. In this broader setting, the relationship between face count and IoU is even less monotonic, further suggesting that face count is a useful but incomplete proxy for reconstruction difficulty. A small ablation further suggests that diversity-aware sampling increases benchmark difficulty: within the same face-count range, diversity-sampled splits produce lower median aligned IoU than randomly sampled splits, with the largest relative drop on Extrude-Medium (Appendix C.2.4).

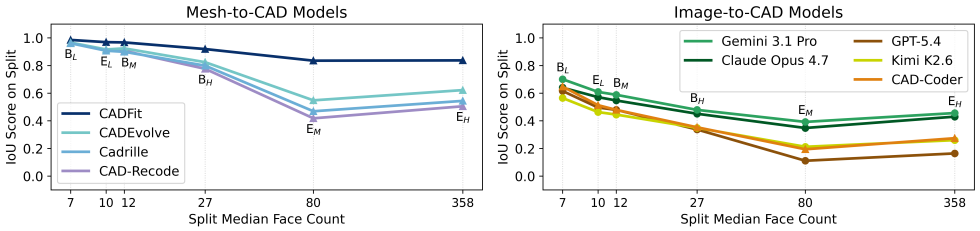


Figure 3: **Model performance as a function of geometric complexity.** Median IoU on each split of the Base (B) and Extrude (E) families, which contain only sketch-and-extrude operations, plotted against the split's median face count (log scale); labels mark each split (e.g., B_L = Base-Low). Left: mesh-to-CAD models; right: image-to-CAD models.

How fragile are CAD generators to modality shift? Figure 4 compares model performance across clean and shifted input modalities. Mesh-conditioned CAD-specialized models are sensitive to mesh noise: CADFit, CADEvolve, Cadrille, and CAD-Recode lose 0.047, 0.585, 0.103, and 0.163 in aggregate IoU, respectively, when moving from clean to noisy mesh inputs (Appendix C.3). The largest degradation occurs for CADEvolve, which drops from 0.708 IoU on clean meshes to 0.123 IoU on noisy meshes (ΔIoU = -0.585). This may be partly explained by a compound modality-shift effect: CADEvolve first renders the input mesh into multi-view images before generating CAD code, so mesh perturbations can propagate through the rendering stage and alter the visual observations passed to the model, potentially amplifying the effect of geometric noise.

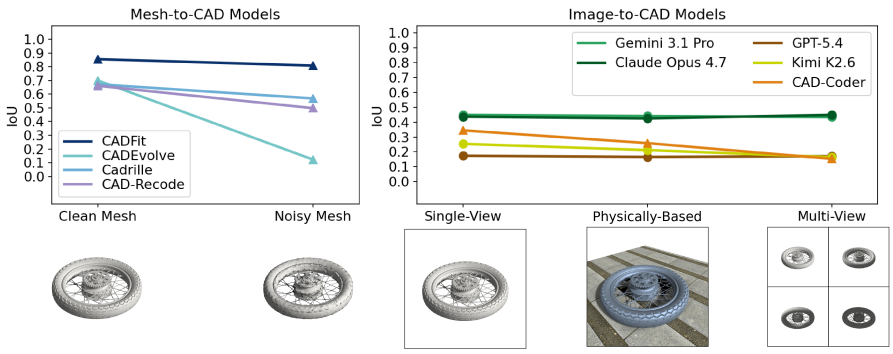


Figure 4: **Robustness to input modality shift.** Aggregate IoU for mesh-to-CAD models on clean and noisy meshes (left) and for image-to-CAD models on single-view grayscale, physically-based, and multi-view renders (right).

Image-conditioned CAD-specialized models also degrade under input shifts. CAD-Coder drops modestly on physically-based renders (ΔIoU = -0.086), but more substantially on multi-view inputs (ΔIoU = -0.192) relative to single-view grayscale renders. Because CAD-Coder was trained on single-view images, this suggests that view composition is more disruptive than rendering style alone. We observe a similar sensitivity for Cadrille and CADEvolve when they are evaluated from image inputs: although both methods can accept images, performance is poor on CADBENCH's standardized single-view renders, likely because these inputs differ from the image formats used during training. In contrast, general-purpose VLMs show comparatively small changes in aggregate IoU across image modalities, indicating stronger robustness to changes in lighting, material, and view layout (Appendix C.3).

These results reveal a trade-off in current models: CAD-specialized models achieve the strongest performance under clean or familiar inputs, but can degrade sharply under realistic modality shifts, while general-purpose VLMs are more robust to input variation but remain substantially less accurate overall. This points to the need for further work on CAD generation methods that combine geometric accuracy with robustness to downstream input conditions, characteristic of real-world applications.

Are multiple geometric fidelity metrics necessary? Aggregate rankings in Table 2 depend on the choice of geometric fidelity metric. For IoU, Gemini 3.1 Pro scores first, Claude Opus 4.7 scores second, and CAD-Coder scores third. For SIoU, CAD-Coder scores first, Claude Opus 4.7 scores second, and Gemini 3.1 Pro scores third. For CD, Gemini 3.1 Pro scores first, Claude Opus 4.7 scores second, and CAD-Coder scores third. Although the models' differences in scores are modest and all image-conditioned models remain far below the strongest mesh-conditioned methods, the change in ranking suggests that IoU, CD, and SIoU capture distinct aspects of reconstruction quality rather than providing interchangeable measurements.

To quantify metric relationships, we compute sample-level Spearman rank correlations between IoU, SIoU, and CD across all syntactically valid model-generated CAD programs (Appendix C.4.1). The metrics exhibit expected directional relationships but are not redundant: IoU and SIoU are weakly to moderately correlated ($\rho = 0.33$), while CD is negatively correlated with IoU ($\rho = -0.75$) and SIoU ($\rho = -0.37$). SIoU's modest correlations suggest that it captures reconstruction quality beyond volumetric overlap or point-based geometric distance, which motivates reporting all three metrics. Appendix C.4.2 illustrates their complementary diagnostic value: high IoU can mask missing fine surface details, while high SIoU can occur when surfaces are near the target but the enclosed volume is incorrect. These distinctions matter in CAD, where teeth, holes, grooves, and interfaces can be critical to part function.

How editable are the generated programs? We also examine whether generated programs are practical to modify. In Table 2, CAD-specialized models hard-code geometric values (Parameterization Ratio of 0), whereas general-purpose VLMs expose most dimensions as named parameters (0.689–0.842), although DoF Compression remains at most 0.314, indicating limited reuse of those parameters. Edit Robustness varies independently of parameterization: CADEvolve remains robust under 10% perturbations (0.811) despite exposing no parameters, whereas CADFit is the least robust (0.370). Full definitions and results at both perturbation levels are given in Appendix C.5.

5 Conclusion

We introduced CadBench, a unified multimodal benchmark for CAD program generation spanning six benchmark families, complexity- and diversity-sampled splits, clean and noisy input modalities, and geometric fidelity and code-quality metrics. Beyond an overall leaderboard, these design choices enable targeted analysis of where current CAD generators succeed and fail. Complexity-stratified splits reveal that most models degrade substantially as geometric difficulty increases. Clean and noisy modality variants assess robustness to input shift, showing that CAD-specialized models often perform strongly on clean inputs but can degrade sharply under noisier or less familiar modalities, while general-purpose VLMs are more robust but remain less accurate overall. Multiple geometric fidelity metrics reveal metric-dependent model rankings, demonstrating that IoU, CD, and SIoU capture complementary aspects of reconstruction quality rather than interchangeable measurements. Overall, CadBench provides a foundation for measuring progress in AI-assisted CAD reconstruction and highlights the need for methods combining geometric accuracy, executable outputs, compact construction procedures, and robustness to realistic inputs.

Limitations. CadBench requires executable CAD programs that produce valid solids, but not recovery of the original feature tree, constraints, construction order, parameterization, or design intent, since only some source families provide human-authored histories. It therefore evaluates solid reconstruction, geometric fidelity, program compactness, and parameter-level editability, while operation-level and semantic equivalence, constraint recovery, and feature-tree correspondence remain future benchmark goals. Our VLM evaluation measures standardized direct generation using fixed prompts; prompt tuning, tool use, self-correction, and multi-turn refinement may improve performance. Finally, STEP-based complexity splits use B-rep face count, a consistent but incomplete proxy: repeated patterns such as hole arrays can dominate high-complexity splits, biasing them toward one form of CAD complexity.

AI USE STATEMENT

Generative AI tools assisted with code implementation, figure preparation, and manuscript editing. The authors independently developed the research design, analysis, and conclusions, verified all AI-assisted work, and take responsibility for the final content. Models evaluated on CADBENCH are objects of study rather than research aids.

ETHICS STATEMENT

CADBENCH is intended to support progress toward more reliable AI-assisted CAD tools by providing standardized evaluation of geometric fidelity, executability, and program compactness across different modalities and benchmark families. CAD-generating AI tools could reduce the time and expertise required to create engineering CAD models and improve access to CAD workflows. However, AI CAD generation systems could also be misused to generate unsafe or poorly validated designs, or could lead users to over-trust generated CAD programs in safety-critical engineering contexts. CADBENCH is an evaluation benchmark rather than a deployed design system, and benchmark performance should not be interpreted as certification that generated CAD models are suitable for manufacturing or real-world use. We recommend that any generated CAD models be reviewed by qualified engineers and validated with domain-appropriate simulation, analysis, and safety checks before fabrication or deployment. CADBENCH is built from publicly available CAD datasets that are used and redistributed under their original licenses (Appendix A.2).

REPRODUCIBILITY STATEMENT

The benchmark data are available as an anonymized Hugging Face dataset,[1] and the licenses of the source datasets are listed in Appendix A.2. The benchmark construction procedure, data preprocessing steps, modality generation pipeline, and evaluation metrics are described in Appendix A.3, while model inference settings and baseline evaluation parameters are provided in Appendix A.7.

REFERENCES

Md Ferdous Alam and Faez Ahmed. GenCAD: Image-conditioned computer-aided design generation with transformer-based contrastive representation and diffusion priors. *Transactions on Machine Learning Research*, 2025.

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. Generating CAD code with vision-language models for 3D designs. In *International Conference on Learning Representations*, 2025.

Anthropic. Claude Opus 4.7 system card. https://www.anthropic.com/claude-opus-4-7-system-card, 2026.

Mohammadmehdi Ataei, Farzaneh Askari, Kamal Rahimi Malekshan, and Pradeep Kumar Jayaraman. Zero-to-CAD: Agentic synthesis of interpretable CAD programs at million-scale without real data. *arXiv preprint arXiv:2604.24479*, 2026.

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)*, pp. 2741–2745. IEEE, 2020.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Manuel Contero, David Pérez-López, Pedro Company, and Jorge D Camba. A quantitative analysis of parametric CAD model complexity and its relationship to perceived modeling complexity. *Advanced Engineering Informatics*, 56:101970, 2023.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13142–13153, 2023.

[1] https://huggingface.co/datasets/anonymous-cadbench-2026/CADBench

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. Ieee, 2009.

Anna C Doris, Ferdous Alam, Amin Heyrani Nobari, and Faez Ahmed. CAD-Coder: An open-source vision-language model for computer-aided design code generation. *Journal of Mechanical Design*, 148(7):071702, 2026.

Yuhao Du, Shunian Chen, Wenbo Zan, Peizhao Li, Mingxuan Wang, Dingjie Song, Bo Li, Yan Hu, and Benyou Wang. BlenderLLM: Training large language models for computer-aided design with self-improvement. *arXiv preprint arXiv:2412.14203*, 2024.

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 *European Conference on Computer Vision*, pp. 19–36. Springer, 2024.

Maksim Elistratov, Marina Barannikov, Gregory Ivanov, Valentin Khrulkov, Anton Konushin, Andrey Kuznetsov, and Dmitrii Zhemchuzhnikov. CADEvolve: Creating realistic CAD via program evolution. *arXiv preprint arXiv:2602.16317*, 2026.

Google DeepMind. Gemini 3.1 Pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, 2026.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Ahmet Serdar Karadeniz, Dimitrios Mallis, Danila Rukhovich, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. MiCADangelo: Fine-grained reconstruction of constrained CAD models from 3D scans. *Advances in Neural Information Processing Systems*, 38, 2025.

Mohammad S Khan, Sankalp Sinha, Talha U Sheikh, Didier Stricker, Sk A Ali, and Muhammad Z Afzal. Text2CAD: Generating sequential CAD designs from beginner-to-expert level text prompts. *Advances in Neural Information Processing Systems*, 37:7552–7579, 2024a.

Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-SIGNet: CAD language inference from point clouds using layer-wise sketch instance guided attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4713–4722, 2024b.

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 *European Conference on Computer Vision*, pp. 175–191. Springer, 2020.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9601–9611, 2019.

Maksim Kolodiazhnyi, Denis Tarasov, Dmitrii Zhemchuzhnikov, Alexander Nikulin, Ilya Zisman, Anna Vorontsova, Anton Konushin, Vladislav Kurenkov, and Danila Rukhovich. cadrille: Multi-modal CAD reconstruction with reinforcement learning. *arXiv preprint arXiv:2505.22914*, 2025.

Joseph G Lambourne, Karl DD Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. BRepNet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12773–12782, 2021.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21448–21457, 2025.

Liane Makatura, Michael Foshey, Bohan Wang, Felix HähnLein, Pingchuan Ma, Bolei Deng, Megan Tjandrasuwita, Andrew Spielberg, Crystal Elaine Owens, Peter Yichen Chen, et al. How can large language models help humans in design and manufacturing? *arXiv preprint arXiv:2307.14377*, 2023.

Ghadi Nehme, Eamon Whalen, and Faez Ahmed. CADFit: Precise mesh-to-CAD program generation with hybrid optimization. *arXiv preprint arXiv:2605.01171*, 2026.

OpenAI. GPT-5.4 Thinking system card. https://openai.com/index/gpt-5-4-thinking-system-card/, 2026.

Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. SketchGen: Generating constrained CAD sketches. *Advances in Neural Information Processing Systems*, 34:5077–5088, 2021.

Cyril Picard, Kristen M Edwards, Anna C Doris, Brandon Man, Giorgio Giannone, Md Ferdous Alam, and Faez Ahmed. From concept to manufacturing: Evaluating vision-language models for engineering design. *Artificial Intelligence Review*, 58(9):288, 2025.

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

Rajkumar Roy, Sara Kelvesjo, Sara Forsberg, and Chris Rush. Quantitative and qualitative cost estimating for engineering design. *Journal of Engineering Design*, 12(2):147–162, 2001.

Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-Recode: Reverse engineering CAD code from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9801–9811, 2025.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. Point2Cyl: Reverse engineering 3D objects from point clouds to extrusion cylinders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11850–11860, 2022.

Siyu Wang, Cailian Chen, Xinyi Le, Qimin Xu, Lei Xu, Yanzhou Zhang, and Jie Yang. CAD-GPT: Synthesising CAD construction sequence with spatial reasoning-enhanced multimodal LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7880–7888, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 Gallery: A dataset and environment for programmatic CAD construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4): 1–24, 2021.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6772–6782, 2021.

Haoyang Xie and Feng Ju. Text-to-CadQuery: A new paradigm for CAD generation with scalable large model capabilities. *arXiv preprint arXiv:2505.06507*, 2025.

Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. CAD-MLLM: Unifying multimodality-conditioned CAD generation with MLLM. *arXiv preprint arXiv:2411.04954*, 2024.

Peiran Xu, Sudong Wang, Yao Zhu, Jianing Li, Gege Qi, and Yunjian Zhang. SpatialBench: Benchmarking multimodal large language models for spatial cognition. *arXiv preprint arXiv:2511.21471*, 2025.

Nomi Yu, Md Ferdous Alam, A John Hart, and Faez Ahmed. GenCAD-3D: CAD program generation using multimodal latent space alignment and synthetic dataset balancing. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 89220, pp. V03AT03A015. American Society of Mechanical Engineers, 2025.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. MMMU-Pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15134–15186, 2025.

Licheng Zhang, Bach Le, Naveed Akhtar, Siew-Kei Lam, and Duc Ngo. Large language models for computer-aided design: A survey. *ACM Computing Surveys*, 58(9):1–39, 2026.

APPENDIX

In the appendix, we provide more thorough details regarding the benchmark construction process, evaluation metrics, and analysis of CADBENCH.

CONTENTS

A Additional Method Details **15**
- A.1 More Details about CADBENCH Families . . . 15
- A.2 Source Datasets and Licenses . . . 17
- A.3 More Details about Benchmark Construction . . . 18
- A.4 Extracting Feature Vectors for Objects using DINOv3 . . . 19
- A.5 Generating Mesh and Image Inputs for Model Evaluation . . . 20
- A.6 More Details on Metrics . . . 22
- A.7 Model Inference Settings . . . 24
- A.8 How Do Best-Achievable and Standardized Performance Differ? . . . 24
- A.9 How Does CADBench Mitigate Potential Train–Test Overlap? . . . 25

B Expanded Comparison of CADBench with Other Benchmarks **25**

C Additional Results **26**
- C.1 Complete Benchmark Results Given Ideal Inputs . . . 26
  - C.1.1 Per-Split Results . . . 26
  - C.1.2 Inference Speed Comparisons . . . 29
- C.2 Understanding the Effects of Complexity and Diversity Sampling . . . 29
  - C.2.1 Why does IoU reconstruction performance improve on the highest face-count Extrude family split? . . . 29
  - C.2.2 Does face count measure difficulty when operation vocabulary is expanded? . . . 29
  - C.2.3 How Robust Is Face Count as a Complexity Proxy? . . . 30
  - C.2.4 Does DINOv3 diversity sampling matter? — Diversity-selected versus random evaluation . . . 31
- C.3 Performance Deltas Across Input Modalities . . . 31
  - C.3.1 How Realistic Are CADBench's Synthetic Modalities? . . . 32
  - C.3.2 Is CAD-Specialized Model Fragility Driven by Architecture or Training-Domain Mismatch? . . . 33
- C.4 Are the Metrics Diagnostic? . . . 34
  - C.4.1 Are IoU, Chamfer distance, and SIoU redundant? — Metric correlation analysis . . . 34
  - C.4.2 When is one geometric fidelity metric not enough? — High-IoU/low-SIoU and high-SIoU/low-IoU examples . . . 35
  - C.4.3 Can Geometric Fidelity Metrics Be Unified into a Single Ranking? . . . 35
- C.5 How Editable Are the Generated CAD Programs? . . . 36
- C.6 How Do General-Purpose VLMs Perform Under Iterative Refinement? . . . 38
- C.7 Qualitative Results . . . 39

A Additional Method Details

A.1 More Details about CadBench Families

In this section, we provide more details about individual CadBench families, including data sources, operation sets, operation counts, face-count distributions, and overlap/Jaccard statistics between families. Table 3 provides the details about the source of the parts for different CadBench families, with the corresponding evaluation focus.

Table 3: Overview of CadBench families.

Subcategory	Source	Core Evaluative Focus
CAD-Base (B)	DeepCAD (Wu et al., 2021)	Establishes a standard reference using benchmarks to ensure continuity with prior research.
CAD-Fusion (F)	Fusion 360 (Willis et al., 2021; Lambourne et al., 2021)	
CAD-Extrude (E)	ABC (Koch et al., 2019)	Escalates difficulty via intricate sketch-extrude geometry (E) and models utilizing an expanded operation set (A).
CAD-All-Ops (A)	ABC (Koch et al., 2019)	
CAD-Mechanical (M)	MCB (Kim et al., 2020)	Bridges the gap to practical application by presenting engineering components (M) and real-world objects (O).
CAD-Organic (O)	Objaverse (Deitke et al., 2023)	

Table 4 provides information about face count and operation count for different CadBench families, and Figure 5 shows their full distributions for each split. For operation counts, we report two measures. **Top Ops.** counts each high-level CAD operation as a single step, e.g., sketch, extrude, revolve, fillet, chamfer, loft, sweep, and thread operations. **Expanded Ops.** provides a finer-grained count by expanding sketches into their underlying sketch primitives and, when available, expanding feature selections such as extruded profiles, filleted edges, and chamfered edges. Thus, Top Ops. captures the length of the CAD construction sequence, while Expanded Ops. better reflects the amount of lower-level geometric specification used to define the model. Note that operation and face-count metadata are unavailable for M and O because these families are mesh-only/freeform sources rather than CAD program sources.

Table 4: Family-level structural metadata for CadBench.

Fam.	Split	Operation Set	GT Type	Face Count				Top Ops.				Expanded Ops.			
				Mean	Med.	Min.	Max.	Mean	Med.	Min.	Max.	Mean	Med.	Min.	Max.
B	Low	Sketch, Extrude	STEP	6.55	7	3	9	3.27	3	2	21	5.70	5	2	34
	Medium			12.93	12	10	18	4.74	4	2	19	11.62	11	4	38
	High			30.23	27	19	110	9.54	9	2	28	24.52	22	8	54
F	Low	Sketch, Extrude, Revolve, Fillet, Chamfer	STEP	5.08	5	1	7	1.50	1	1	8	1.64	1	1	23
	Medium			16.27	13	8	55	2.77	2	1	20	3.59	2	1	73
	High			93.32	79	57	421	6.21	4	1	59	9.59	5	1	486
E	Low	Sketch, Extrude	STEP	11.26	10	3	54	3.77	3	2	22	14.70	10	2	120
	Medium			91.43	80	57	179	7.25	4	2	73	134.08	110	2	2592
	High			687.23	358	181	19851	5.68	4	2	117	869.65	430	2	11635
A	Low	Sketch, Extrude, Revolve, Fillet, Chamfer, Loft, Sweep, Threads	STEP	13.80	12	2	36	5.71	5	1	43	17.88	14	2	216
	Medium			59.59	51	37	137	10.55	9	1	78	70.78	60	1	444
	High			383.70	206	138	12292	13.27	8	1	361	337.39	202	2	6252
M	—	N/A	OBJ	—	—	—	—	—	—	—	—	—	—	—	—
O	—	N/A	GLB	—	—	—	—	—	—	—	—	—	—	—	—

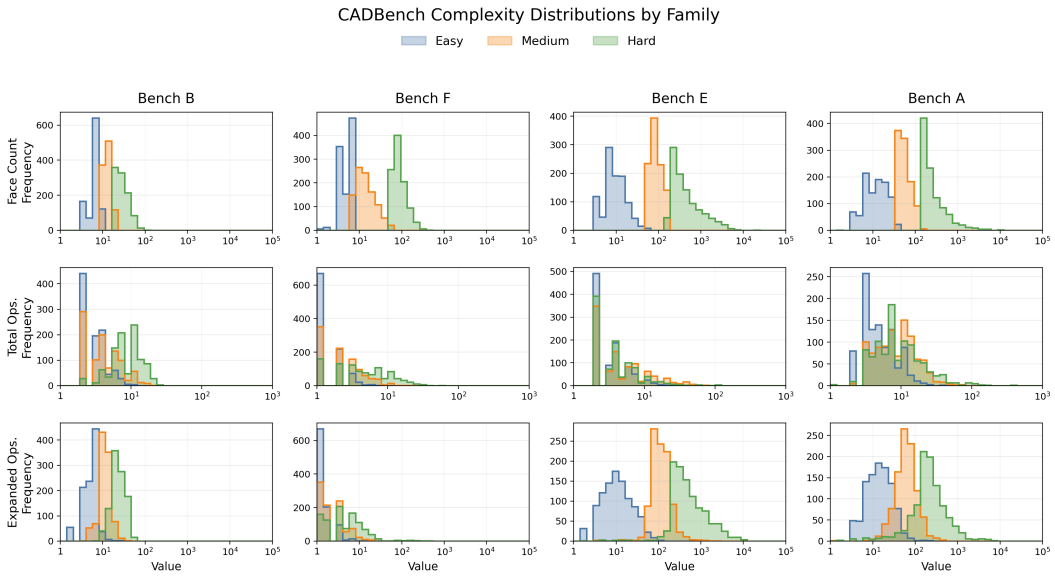


Figure 5: **Complexity distributions of the STEP-based families.** Rows show face count, top-level operation count (Top Ops., labeled "Total Ops." in the plots), and expanded operation count (Expanded Ops., which additionally counts sketch primitives and fillet/chamfer selections) for CAD-Base (B), CAD-Fusion (F), CAD-Extrude (E), and CAD-All-Ops (A); colors denote the Low, Medium, and High complexity splits (labeled Easy, Medium, and Hard), and x-axes are logarithmic. Because splits are defined by face count, the face-count distributions separate cleanly, whereas operation counts, particularly top-level counts, overlap substantially across splits.

Tables 5 and 6 show overlap statistics between families, and Figure 7 visualizes each family in the DINOv3 embedding space. Count-based entries are Jaccard similarities over empirical count distributions. DINO entries are symmetric k-nearest-neighbor embedding overlap with $k = 50$. Higher values indicate greater overlap. In Table 5, all complexity splits within a family are treated as one aggregate, whereas in Table 6, each complexity split is treated as a separate category.

Table 5: Broad family-level overlap statistics

Metric	Family	B	F	E	A	M	O
Face Count	B	1.000	0.382	0.208	0.255	—	—
	F	0.382	1.000	0.440	0.325	—	—
	E	0.208	0.440	1.000	0.383	—	—
	A	0.255	0.325	0.383	1.000	—	—
Top Ops.	B	1.000	0.411	0.660	0.493	—	—
	F	0.411	1.000	0.367	0.288	—	—
	E	0.660	0.367	1.000	0.354	—	—
	A	0.493	0.288	0.354	1.000	—	—
Expanded Ops.	B	1.000	0.193	0.205	0.258	—	—
	F	0.193	1.000	0.137	0.108	—	—
	E	0.205	0.137	1.000	0.455	—	—
	A	0.258	0.108	0.455	1.000	—	—
DINO	B	1.000	0.383	0.379	0.323	0.144	0.200
	F	0.383	1.000	0.380	0.398	0.226	0.297
	E	0.379	0.380	1.000	0.393	0.162	0.245
	A	0.323	0.398	0.393	1.000	0.198	0.276
	M	0.144	0.226	0.162	0.198	1.000	0.165
	O	0.200	0.297	0.245	0.276	0.165	1.000

Table 6: Split-category overlap statistics.

Metric	Cat.	B/low	B/med.	B/high	F/low	F/med.	F/high	E/low	E/med.	E/high	A/low	A/med.	A/high	M	O
Face Count	B/low	1.000	0.000	0.000	0.460	0.132	0.000	0.332	0.000	0.000	0.238	0.000	0.000	—	—
	B/med.	0.000	1.000	0.000	0.000	0.332	0.000	0.229	0.000	0.000	0.210	0.000	0.000	—	—
	B/high	0.000	0.000	1.000	0.000	0.152	0.017	0.069	0.017	0.000	0.154	0.123	0.000	—	—
	F/low	0.460	0.000	0.000	1.000	0.000	0.000	0.215	0.000	0.000	0.150	0.000	0.000	—	—
	F/med.	0.132	0.332	0.152	0.000	1.000	0.000	0.474	0.000	0.000	0.491	0.025	0.000	—	—
	F/high	0.000	0.000	0.017	0.000	0.000	1.000	0.000	0.654	0.022	0.000	0.238	0.064	—	—
	E/low	0.332	0.229	0.069	0.215	0.474	0.000	1.000	0.000	0.000	0.635	0.009	0.000	—	—
	E/med.	0.000	0.000	0.017	0.000	0.000	0.654	0.000	1.000	0.000	0.000	0.243	0.064	—	—
	E/high	0.000	0.000	0.000	0.000	0.000	0.022	0.000	0.000	1.000	0.000	0.000	0.217	—	—
	A/low	0.238	0.210	0.154	0.150	0.491	0.000	0.635	0.000	0.000	1.000	0.000	0.000	—	—
	A/med.	0.000	0.000	0.123	0.000	0.025	0.238	0.009	0.243	0.000	0.000	1.000	0.000	—	—
	A/high	0.000	0.000	0.000	0.000	0.000	0.064	0.000	0.064	0.217	0.000	0.000	1.000	—	—
Top Ops.	B/low	1.000	0.582	0.134	0.198	0.435	0.323	0.745	0.528	0.665	0.379	0.195	0.180	—	—
	B/med.	0.582	1.000	0.311	0.198	0.418	0.490	0.661	0.634	0.720	0.500	0.373	0.336	—	—
	B/high	0.134	0.311	1.000	0.040	0.138	0.359	0.199	0.318	0.261	0.345	0.491	0.446	—	—
	F/low	0.198	0.198	0.040	1.000	0.517	0.252	0.198	0.190	0.197	0.106	0.062	0.066	—	—
	F/med.	0.435	0.418	0.138	0.517	1.000	0.505	0.372	0.351	0.367	0.338	0.212	0.198	—	—
	F/high	0.323	0.490	0.359	0.252	0.505	1.000	0.352	0.440	0.399	0.543	0.511	0.488	—	—
	E/low	0.745	0.661	0.199	0.198	0.372	0.352	1.000	0.647	0.781	0.358	0.250	0.233	—	—
	E/med.	0.528	0.634	0.318	0.190	0.351	0.440	0.647	1.000	0.762	0.421	0.384	0.382	—	—
	E/high	0.665	0.720	0.261	0.197	0.367	0.399	0.781	0.762	1.000	0.400	0.310	0.294	—	—
	A/low	0.379	0.500	0.345	0.106	0.338	0.543	0.358	0.421	0.400	1.000	0.488	0.461	—	—
	A/med.	0.195	0.373	0.491	0.062	0.212	0.511	0.250	0.384	0.310	0.488	1.000	0.729	—	—
	A/high	0.180	0.336	0.446	0.066	0.198	0.488	0.233	0.382	0.294	0.461	0.729	1.000	—	—
Expanded Ops.	B/low	1.000	0.095	0.014	0.098	0.267	0.378	0.280	0.006	0.006	0.169	0.008	0.015	—	—
	B/med.	0.095	1.000	0.152	0.015	0.085	0.215	0.348	0.007	0.005	0.375	0.031	0.020	—	—
	B/high	0.014	0.152	1.000	0.005	0.028	0.088	0.255	0.015	0.004	0.365	0.158	0.029	—	—
	F/low	0.098	0.015	0.005	1.000	0.516	0.260	0.070	0.005	0.004	0.035	0.005	0.011	—	—
	F/med.	0.267	0.085	0.028	0.516	1.000	0.509	0.186	0.009	0.006	0.131	0.022	0.020	—	—
	F/high	0.378	0.215	0.088	0.260	0.509	1.000	0.411	0.020	0.010	0.307	0.041	0.034	—	—
	E/low	0.280	0.348	0.255	0.070	0.186	0.411	1.000	0.026	0.007	0.638	0.119	0.047	—	—
	E/med.	0.006	0.007	0.015	0.005	0.009	0.020	0.026	1.000	0.079	0.027	0.322	0.257	—	—
	E/high	0.006	0.005	0.004	0.004	0.006	0.010	0.007	0.079	1.000	0.009	0.028	0.184	—	—
	A/low	0.169	0.375	0.365	0.035	0.131	0.307	0.638	0.027	0.009	1.000	0.138	0.043	—	—
	A/med.	0.008	0.031	0.158	0.005	0.022	0.041	0.119	0.322	0.028	0.138	1.000	0.149	—	—
	A/high	0.015	0.020	0.029	0.011	0.020	0.034	0.047	0.257	0.184	0.043	0.149	1.000	—	—
DINO	B/low	1.000	0.414	0.335	0.411	0.366	0.212	0.450	0.219	0.208	0.386	0.194	0.163	0.192	0.177
	B/med.	0.414	1.000	0.472	0.298	0.412	0.287	0.483	0.296	0.238	0.388	0.285	0.210	0.176	0.233
	B/high	0.335	0.472	1.000	0.226	0.380	0.318	0.445	0.326	0.246	0.359	0.324	0.244	0.154	0.245
	F/low	0.411	0.298	0.226	1.000	0.373	0.206	0.359	0.183	0.177	0.367	0.168	0.137	0.233	0.218
	F/med.	0.366	0.412	0.380	0.373	1.000	0.367	0.431	0.296	0.225	0.452	0.346	0.236	0.269	0.303
	F/high	0.212	0.287	0.318	0.206	0.367	1.000	0.308	0.410	0.325	0.331	0.427	0.387	0.241	0.347
	E/low	0.450	0.483	0.445	0.359	0.431	0.308	1.000	0.328	0.272	0.428	0.310	0.241	0.216	0.247
	E/med.	0.219	0.296	0.326	0.183	0.296	0.410	0.328	1.000	0.422	0.273	0.371	0.416	0.194	0.303
	E/high	0.208	0.238	0.246	0.177	0.225	0.325	0.272	0.422	1.000	0.211	0.255	0.380	0.161	0.220
	A/low	0.386	0.388	0.359	0.367	0.452	0.331	0.428	0.273	0.211	1.000	0.375	0.254	0.276	0.272
	A/med.	0.194	0.285	0.324	0.168	0.346	0.427	0.310	0.371	0.255	0.375	1.000	0.381	0.207	0.309
	A/high	0.163	0.210	0.244	0.137	0.236	0.387	0.241	0.416	0.380	0.254	0.381	1.000	0.187	0.259
	M	0.192	0.176	0.154	0.233	0.269	0.241	0.216	0.194	0.161	0.276	0.207	0.187	1.000	0.165
	O	0.177	0.233	0.245	0.218	0.303	0.347	0.247	0.303	0.220	0.272	0.309	0.259	0.165	1.000

A.2 Source Datasets and Licenses

CadBench comprises data from the following sources, each governed by its own licensing terms:

- **DeepCAD:** Licensed under the MIT License.
- **Fusion 360 Gallery:** The dataset and supporting codebase are publicly available on GitHub[2] under a license permitting non-commercial research, mirroring the terms of the ImageNet (Deng et al., 2009) license.
- **ABC:** Licensed under the MIT License.

[2] https://github.com/AutodeskAILab/Fusion360GalleryDataset

- **Objaverse:** The dataset is collectively licensed under ODC-By v1.0, while individual objects are distributed as Creative Commons assets under various specific licenses.[3]
- **MCB:** Licensed under the MIT License.

Access-controlled datasets: CC3D is an important and widely used scan-to-CAD benchmark, and we include it in our discussion of related datasets and prior evaluation settings. However, we do not include CC3D in the primary CADBENCH evaluation suite or released aggregate benchmark because access to the dataset is governed by a separate institutional data-use agreement that must be requested and executed by prospective users. In contrast, CADBENCH is designed around artifacts that can be directly inspected, reproduced, and rerun by reviewers and future users without requiring additional third-party licensing negotiations. For this reason, CADBENCH does not report quantitative results on CC3D. This exclusion reflects a reproducibility and release-design consideration rather than any judgment about the scientific value or importance of CC3D as a benchmark.

A.3 MORE DETAILS ABOUT BENCHMARK CONSTRUCTION

CAD-Base (B). This family consists of CAD models derived from the DeepCAD dataset. We used the DeepCAD test split introduced in the GenCAD paper (Alam & Ahmed, 2025), consisting of 7,629 samples. We first filtered the dataset to de-duplicate and to filter to only samples containing single bodies. This exclusion of multi-body designs and assemblies is motivated by the nature of the feature tree reconstruction task, which is fundamentally intended for individual parts. In standard engineering workflows, assemblies are not designed as a monolithic entity within a single feature tree. Instead, each component is designed independently with its own feature history. To align with this design intent, we filtered out models consisting of multiple bodies, leaving a total of 6,210 samples. We then applied complexity-based stratification and diversity-aware sampling as described in Section 3.

CAD-Fusion (F). This family consists of CAD models derived from the Fusion 360 Reconstruction and Segmentation datasets. Since the Fusion 360 Reconstruction test set contains only 1,725 samples, we augmented it with models from the Fusion 360 Segmentation Dataset. Following the CAD-Base filtering criteria, we excluded all multi-body designs, leaving a total of 37,184 CAD models. We then applied complexity-based stratification and diversity-aware sampling as described in Section 3.

CAD-Extrude (E) and CAD-All-Ops (A). These families consist of CAD models derived from the ABC dataset (Koch et al., 2019). We began by preprocessing the ABC dataset—originally comprising one million samples—using a pipeline that sought to identify and exclude objects with multiple bodies and stray surfaces. We further targeted objects containing specific geometric entities within the STEP file (such as `TRIANGULATED_FACE`) to filter out unsuitable samples, like pseudo-STEP models, as these function merely as mesh-to-STEP wrappers and lack true geometric primitives. Utilizing the original FeatureScript (`ofs`) files, we further filtered out models containing text-based (`BTMSketchTextEntity`) or image-based (`BTMSketchImageEntity`) sketches. The remaining models were then divided into two distinct partitions based on their operation history: one partition ($\approx$ 126K samples) consisting of models defined by sketch (`newSketch`) and extrude (`extrude`) operations, and the other partition ($\approx$ 62K samples) comprising models that incorporate more complex features such as `fillet`, `revolve`, `chamfer`, `loft`, `sweep`, `externalThreads`, and `internalThreads`. Finally, complexity-based stratification and diversity-aware sampling were applied to create CAD-Extrude and CAD-All-Ops, respectively.

CAD-Mechanical (M). This family consists of samples derived from the MCB dataset. We use the test split from the MCB Dataset, consisting of 11,716 samples. We then ran a watertight check and repair, filtered to single-body samples, removed duplicates, and eliminated samples for which no images were generated; this left 7,106 samples. We then applied diversity-aware sampling as described in Section 3 to obtain a subset of 3,000 samples. The original MCB dataset is split into 68 classes of mechanical objects. Through our diversity sampling procedure, 67/68 classes were represented in the 3,000-sample benchmark. The three most represented classes are 1) conventional rivets, 2) keys and keyways, splines, and 3) screws and bolts with cylindrical heads.

[3]Detailed license information is available at https://huggingface.co/datasets/allenai/objaverse.

CAD-Organic (O). This family consists of samples derived from the Objaverse dataset. While Objaverse is a large-scale collection of general 3D models, we select specific categories deemed most "CAD-suitable" based on their geometric structure and functional design:

- Furniture-Home
- Science-Technology
- Architecture
- Cars-Vehicles
- Electronics-Gadgets

From an initial pool of approximately 800,000 models, filtering for these categories yielded a final subset of 226,865 samples. To ensure geometric validity of the meshes used in our dataset, we applied a mesh repair pipeline to convert non-watertight meshes into watertight manifolds whenever possible. A mesh is considered watertight when it forms a closed 2-manifold surface with no holes or boundary edges. Watertight meshes are essential for many downstream tasks such as volume computation, physical simulation, geometry learning models, and surface reconstruction. However, many meshes in large-scale datasets such as Objaverse contain geometric defects, including holes in the surface, duplicate vertices or faces, non-manifold edges, unreferenced vertices, and degenerate triangles. To address these issues, we implemented a repair pipeline using PyMeshLab, which provides robust mesh-processing filters. We subsequently de-duplicated and filtered for single-body samples, which left 20,148 samples. We then applied the mesh-based dataset procedure described above to select a benchmark subset of 3,000 samples.

A.4 Extracting Feature Vectors for Objects using DINOv3

To obtain a representative feature vector for each object across our datasets, we employ a vision-based encoding pipeline inspired by the procedure utilized in Ataei et al. (2026) (Figure 6). Each object is rendered into eight grayscale isometric views using `pythonOCC` and processed through a DINOv3 (Siméoni et al., 2025) backbone (`dinov3-vitb16-pretrain-lvd1689m`) to extract 768-dimensional latent embeddings. These vectors are then averaged to produce a single representative feature vector per object. This image-based procedure is highly effective across various 3D file formats, including STEP, STL, GLB, and OBJ.

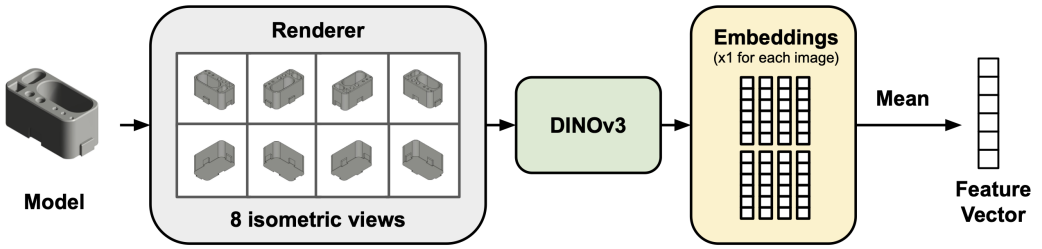


Figure 6: **Feature extraction with DINOv3.** Each object is rendered from eight isometric viewpoints, each view is encoded by DINOv3, and the eight image embeddings are averaged into a single feature vector per object. These vectors are used for diversity-aware sampling and for the diversity and overlap statistics of each family.

Figure 7 shows a t-SNE projection of these feature vectors for all CadBench samples, highlighting each family in turn.

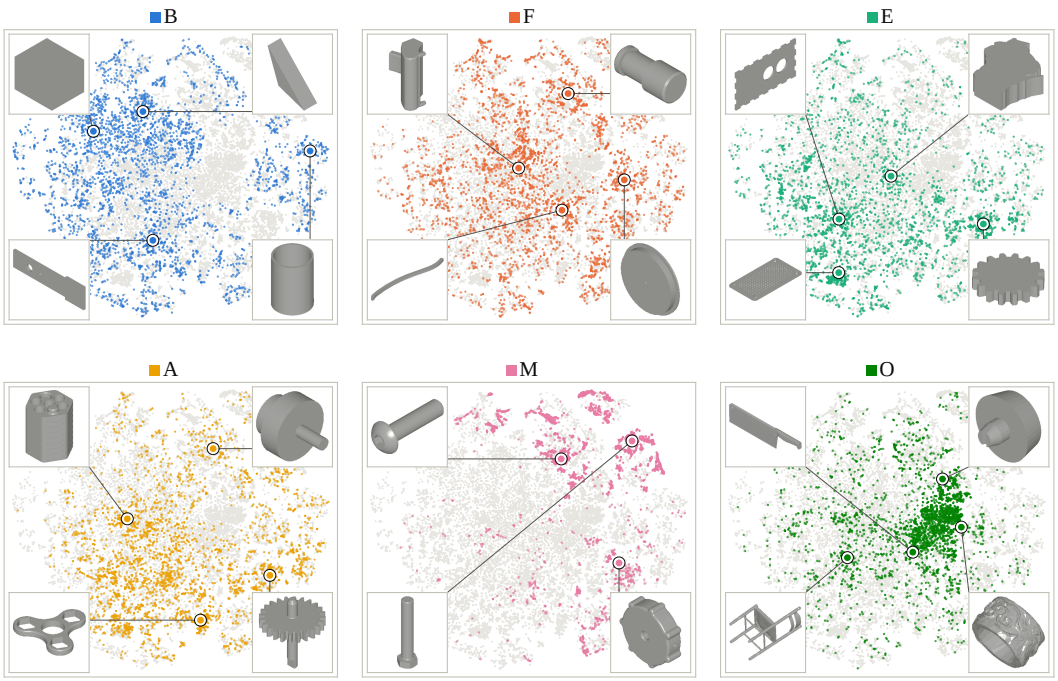


Figure 7: **t-SNE projection of DINOv3 embeddings of CADBENCH samples.** Each panel highlights one benchmark family in color against all other families (gray), with representative parts linked to their positions in the embedding. The families occupy overlapping but distinct regions of the embedding space.

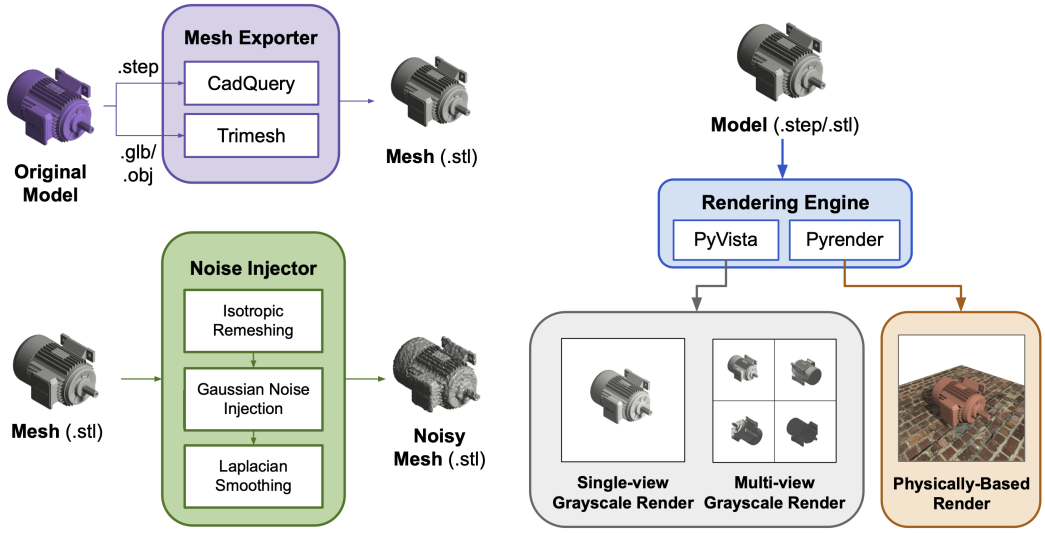
A.5 GENERATING MESH AND IMAGE INPUTS FOR MODEL EVALUATION

We describe in greater detail the methodology used to generate the mesh and image inputs for benchmarking and evaluating various models; Figure 8 summarizes both pipelines.

Mesh Generation. For the B, F, E, and A families, we utilize `CadQuery` to export STEP models to STL format using default tessellation settings. For M and O, we import source OBJ or GLB files and perform STL conversion via the `trimesh` library. In all cases, meshes are centered at the origin and normalized to fit within a $[-1, 1]^3$ bounding box. To generate noisy variants, models undergo isotropic remeshing via `PyMeshLab`, followed by the injection of Gaussian noise ($\sigma = 0.005$) into the vertices. We subsequently apply Laplacian smoothing to suppress sharp artifacts prior to final export.

Grayscale Rendering. High-resolution (1200×1200) single-view grayscale images are generated by rendering STL models in an isometric orientation using `PyVista`. To emphasize structural definition, we apply a neutral matte finish (`#BFBEBA`). Realistic depth and soft grounded shadows are achieved through a softbox lighting technique, employing 50 jittered light sources to produce a professional, CAD-like aesthetic. For multi-view grayscale renders, the STL models are rendered from four distinct isometric orientations using this identical setup, and the resulting renders are stitched together into a single composite image.

Physically-Based Rendering (PBR). The physically-based renders are produced using `Pyrender`. We utilize source STEP files for B, F, E, and A, and mesh files for M and O. Material properties for each component are randomly assigned: RGB components are sampled from $U(0, 1)$, while metallicity and roughness are sampled from $U(0.05, 0.95)$. Ground planes are assigned random textures from a curated library of realistic materials (e.g., asphalt, grass, brick, and concrete). Illumination is provided by a three-point lighting configuration (key, fill, and back lights). The camera pose is configured to render an isometric view under perspective projection with a 60° Field of View



(a) Mesh inputs (b) Image inputs

Figure 8: **Pipelines for generating CADBENCH's input modalities.** (a) Clean meshes are exported from STEP sources with CadQuery or from GLB/OBJ sources with trimesh; noisy meshes are then produced by isotropic remeshing, Gaussian noise injection, and Laplacian smoothing. (b) Single-view and four-view grayscale renders are produced with PyVista, and physically-based renders with randomized materials and environments are produced with Pyrender.

(FOV). Additionally, the position of the camera is adjusted based on the diameter of the bounding sphere of each model to ensure that the object is not cropped.

We show samples from various CADBENCH families, along with their corresponding mesh and image modalities, in Figure 9.

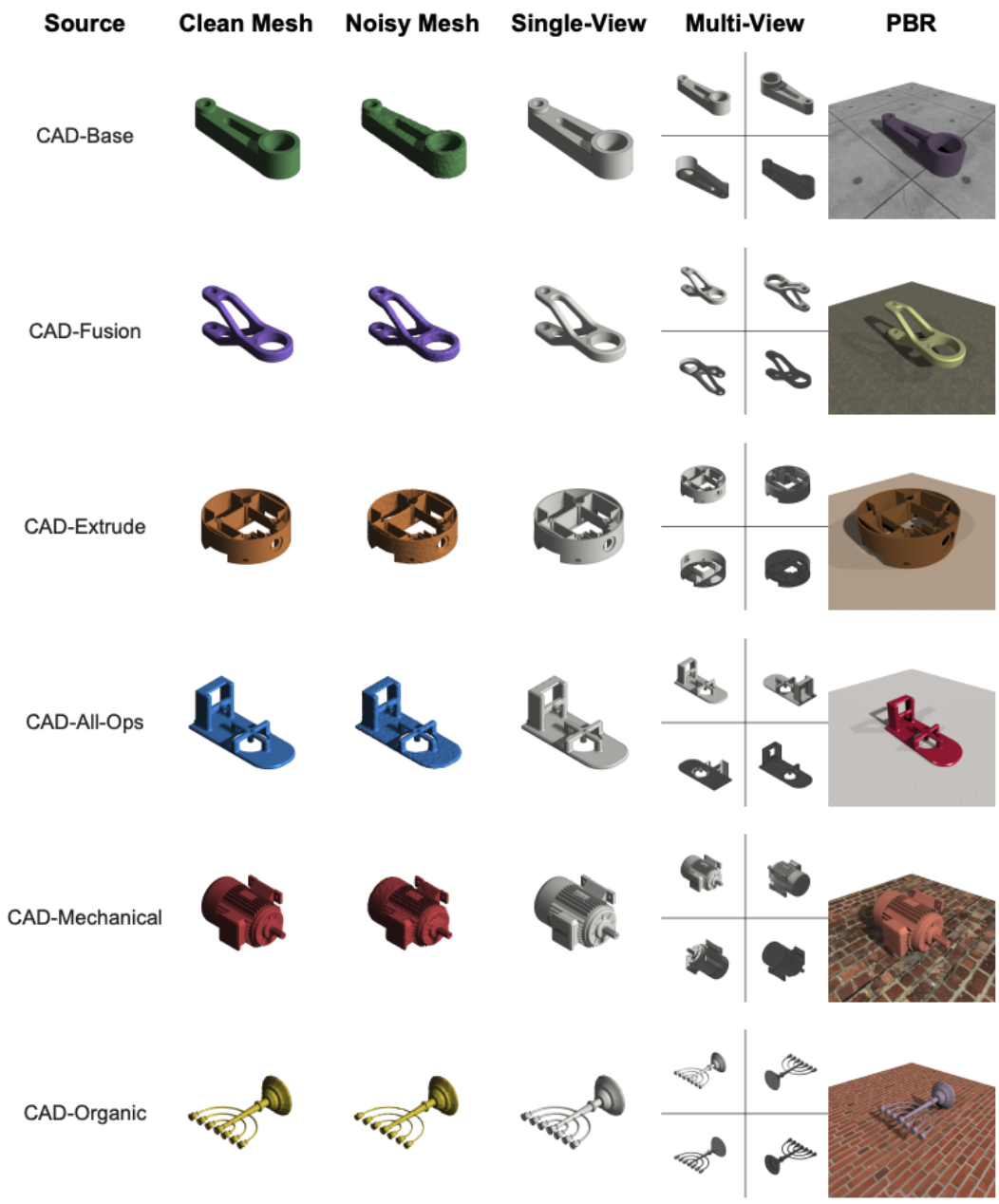


Figure 9: **Input modalities for each CADBENCH family.** Samples from each family in CADBENCH illustrating the available data modalities. From left to right: original clean meshes, their corresponding noisy counterparts, single-view grayscale renders, multi-view grayscale renders, and physically-based renders (PBR) featuring randomized material properties and environmental textures.

A.6 MORE DETAILS ON METRICS

Let $\hat{S}$ denote the predicted geometry obtained from the generated CAD program, and let S denote the ground-truth geometry. We define the geometric fidelity metrics as follows.

Volumetric IoU. Let $V(\cdot)$ denote voxelized occupancy. The intersection-over-union is:

$$\text{IoU} = \frac{|V(\hat{S} \cap S)|}{|V(\hat{S}) \cup V(S)|}$$

Chamfer Distance (CD). Let $\hat{P}, P \subset \mathbb{R}^3$ be point samples from $\hat{S}$ and S. The symmetric Chamfer distance is:

$$\mathrm{CD}(\hat{P}, P) = \frac{1}{|\hat{P}|} \sum_{x \in \hat{P}} \min_{y \in P} \|x - y\|_2^2 + \frac{1}{|P|} \sum_{y \in P} \min_{x \in \hat{P}} \|y - x\|_2^2$$

Surface IoU (SIoU). Let $\hat{P}$ and P be N points (we use $N = 10{,}000$) sampled uniformly (with a fixed seed) from the predicted surface $\hat{S}$ and the ground-truth surface S. Let $d(x, S) = \min_{s \in S} \|x - s\|_2$ be the distance from a point x to the continuous surface S. We measure surface alignment as symmetric point-to-surface coverage, using a distance threshold $\tau = 0.02$, which is 1% of the bounding box side length. Then, SIoU can be calculated as

$$\mathrm{SIoU} = \frac{1}{2} \left(\frac{1}{|\hat{P}|} \sum_{x \in \hat{P}} \mathbb{I}\big[d(x, S) \le \tau\big] + \frac{1}{|P|} \sum_{y \in P} \mathbb{I}\big[d(y, \hat{S}) \le \tau\big] \right)$$

Geometry Preprocessing and Alignment. Before the metrics are measured, we perform an alignment based on the continuous Procrustes analysis solution of aligning two solids (Doris et al., 2026). In this way, we do not simply align bounding boxes like many have before (Yu et al., 2025; Alam & Ahmed, 2025; Kolodiazhnyi et al., 2025; Elistratov et al., 2026; Rukhovich et al., 2025), but rather perform a mathematically sound affine transformation that includes aligning principal axes. This has been noted as an important step, especially in frontier models where the direction models choose to build geometry along can vary widely, as shown by Doris et al. (2026). Overall, the transformation applied to the mesh of the generated code, Ω_g, with respect to the ground truth solid Ω_t can be described as:

$$\hat{\Omega}_g = \{ \frac{\mathbf{R}^\star (\mathbf{x} - \bar{\mathbf{x}}_\mathbf{g}) + \bar{\mathbf{x}}_t}{\sqrt{\frac{\mathrm{tr}(\mathbf{I_g})}{2 \times \mathrm{Vol}(\Omega_g)}}} \times \sqrt{\frac{\mathrm{tr}(\mathbf{I_t})}{2 \times \mathrm{Vol}(\Omega_t)}} \mid \mathbf{x} \in \Omega_g \},$$

where $\bar{\mathbf{x}}$ refers to the centroid, $\mathbf{I}$ refers to the matrix of inertia, and Vol refers to the volume of solids. Moreover, $\mathbf{R}^\star$ refers to the optimal rotation aligning the principal axes of the two solids determined exhaustively by measuring IoU for each of the four proper rotations (elements of $SO(3)$) among the eight sign combinations of the principal axes and picking the transformation with the highest IoU. Note that, besides this, all ground-truth meshes are centered and scaled so that the longest side of their bounding box has length 2, i.e., they fit within $[-1, 1]^3$, consistent with the mesh normalization described above; the SIoU threshold τ and the CD failure penalty ($2\sqrt{3}$, the bounding-box diagonal) are defined at this scale. For generated CAD outputs whose resulting geometries are non-watertight, we consistently apply alpha wrapping before performing optimal alignment.

Executability. A predicted program may fail due to invalid operations or inconsistent constraints.

Invalid Ratio (IR). Let $\mathcal{D}$ be the evaluation set and $\mathbb{I}[\cdot]$ an indicator function:

$$\mathrm{IR} = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbb{I}\left[\mathcal{E}(f_\theta(x)) \text{ fails}\right].$$

We also report **Valid Shape Rate (VSR)**:

$$\mathrm{VSR} = 1 - \mathrm{IR}.$$

Program compactness. Because all outputs are standardized as CadQuery scripts, Token Count and Operation Count are comparable across models within CADBENCH, but are not directly comparable across DSLs without DSL-specific parsing and tokenization rules. We chose CadQuery because most evaluated public models already generate it and its Python syntax suits current VLMs.

Token Count. We count script tokens using `tokenize.generate_tokens(...)`.

Operation Count. We count the CAD operations in the generated program by parsing the script's Python AST and counting calls matching a predefined CadQuery-operation vocabulary.

A.7 Model Inference Settings

Qwen3.5 9B In running Qwen3.5 9B, we follow the recommended inference settings for instruct (non-thinking) mode on general tasks[4]. Specifically, we disable thinking and use temperature 0.7, top-p 0.8, and top-k 20, with a presence penalty of 1.5 and repetition penalty of 1.0. We additionally set `min_p` $= 0.0$ and a maximum generation length of 4096 tokens.

Qwen3.5 27B For Qwen3.5 27B, we use the recommended inference settings for instruct (non-thinking) mode on general tasks[5]. Specifically, we disable thinking and use temperature 0.7, top-p 0.8, and top-k 20, with a presence penalty of 1.5 and repetition penalty of 1.0. We additionally set `min_p` $= 0.0$ and a maximum generation length of 4096 tokens. These fixed decoding parameters are used across all evaluated splits and input modalities.

Kimi K2.6 For Kimi K2.6, we use `kimi-k2.6` through our batch inference script with the default parameters of the API as described in https://platform.moonshot.ai/ with chain of thought (thinking) disabled. Each input includes the rendered CAD image and a fixed CAD-specific prompt requiring executable CadQuery code, all necessary imports, valid solid geometry, parametric dimensions where appropriate, and a final variable named `result`. The model is instructed to return only Python code, with no explanation.

Gemini 3.1 Pro For Gemini 3.1 Pro, we use `gemini-3.1-pro-preview` through our batch inference script with a maximum generation length of 4096 tokens. Each input includes the rendered CAD image and a fixed CAD-specific prompt requiring executable CadQuery code, all necessary imports, valid solid geometry, parametric dimensions where appropriate, and a final variable named `result`. The model is instructed to return only Python code, with no explanation.

GPT-5.4 In running GPT-5.4, we use the default `gpt-5.4` model through the OpenAI Batch API with a 24-hour completion window. Each example includes the rendered CAD image at high image detail and a fixed CAD-specific prompt requiring executable CadQuery code, all necessary imports, valid solid geometry, and a final variable named `result`. The model is instructed to return only Python code, with no explanation. We do not explicitly set the temperature, so the API default is used.

Claude Opus 4.7 For Claude Opus 4.7, we use `claude-opus-4-7` with a maximum generation length of 4096 tokens. Each query includes the rendered CAD image and the same CAD-specific prompt used for the other proprietary vision-language models, requiring executable CadQuery code, all necessary imports, valid solid geometry, and a final variable named `result`. The model is instructed to return only Python code, with no explanation.

A.8 How Do Best-Achievable and Standardized Performance Differ?

Our main leaderboard reports each CAD-specialized method under its strongest evaluated setting rather than enforcing a single shared input protocol. CADFit, CAD-Recode, Cadrille, and CADEvolve use clean meshes with their default preprocessing pipelines, while CAD-Coder uses CadBench single-view renders, which closely match the single-view modality used during its training. This setting is intended to characterize the performance achievable by each method under its preferred evaluated input configuration.

Separately, we evaluate Cadrille and CADEvolve using CadBench's standardized single-view renders in Table 17. This experiment serves as a robustness stress test by fixing the input modality and rendering pipeline across methods, rather than allowing inputs to be tailored to each model. It therefore characterizes performance under a common evaluation condition that may be more representative of settings where model-specific preprocessing or preferred input modalities are unavailable. We distinguish these two settings throughout the paper: the main results measure performance under each method's strongest evaluated configuration, while the standardized single-view results measure robustness under a shared input protocol.

[4] https://huggingface.co/Qwen/Qwen3.5-9B

[5] https://huggingface.co/Qwen/Qwen3.5-27B

A.9 How Does CADBench Mitigate Potential Train–Test Overlap?

Potential train–test overlap is an inherent concern when evaluating pretrained CAD and vision-language models, particularly because complete training-data provenance is not available for every evaluated model. CADBench mitigates this risk at the benchmark level by using source-dataset test splits wherever available, thereby separating evaluation samples from the corresponding training data used to construct the benchmark families.

The benchmark composition further limits direct supervised overlap for several families. CAD-Mechanical (M) and CAD-Organic (O) provide only final meshes, while CAD-Extrude (E) and CAD-All-Ops (A) do not provide standardized, directly parseable feature trees or CadQuery programs. Consequently, these families cannot be used straightforwardly as supervised CAD-program reconstruction targets in their provided form. In addition, performance across these benchmark families remains generally limited, providing no indication of systematic memorization of their underlying geometries or programs.

Nevertheless, dataset-level split isolation cannot rule out prior exposure through external pretraining data. Because complete training provenance is unavailable for every evaluated model, we cannot determine whether individual benchmark geometries were encountered during pretraining or instruction tuning. We therefore do not claim that CADBench is completely free of train–test contamination; instead, we use source-level test splits where available and explicitly acknowledge residual uncertainty arising from the opaque training provenance of pretrained models.

B Expanded Comparison of CADBench with Other Benchmarks

We evaluate CADBench relative to established benchmarks, such as DeepCAD, Fusion 360 Reconstruction, MCB, and Omni-CAD, across three primary axes: complexity, similarity, and geometric canonicality. We quantify complexity via B-rep face count. Similarity of objects within a benchmark is assessed using the pairwise cosine similarities between the feature vectors of all objects in the set (see Appendix A.4 for the procedure to get the feature vectors). To further characterize geometric canonicality, we measure the similarity of benchmark objects to a unit cube using two metrics: (i) the volumetric Intersection over Union (IoU) and (ii) the cosine similarity between the feature vector of the object and that of a unit cube.

Table 7: Comparison of structural complexity, diversity, and canonicality across benchmarks. **Complexity** is measured by B-rep face count (↑). **Similarity** measures dataset redundancy (↓ indicates higher diversity). **Geometric Canonicality** assesses proximity to a unit cube (↓ indicates models further from simple primitives).

Bench.	Split	Complexity ↑				Similarity ↓		Geometric Canonicality ↓			
								IoU-based		DINO-based	
		Mean	Med.	Min.	Max.	Mean	Med.	Mean	Med.	Mean	Med.
B	Low	6.55	7	3	9	0.514	0.499	0.210	0.143	0.489	0.449
	Medium	12.93	12	10	18	0.512	0.509	0.177	0.104	0.490	0.459
	High	30.23	27	19	110	0.506	0.500	0.180	0.129	0.477	0.438
F	Low	5.08	5	1	7	0.451	0.435	0.187	0.110	0.395	0.364
	Medium	16.27	13	8	55	0.434	0.423	0.153	0.093	0.363	0.327
	High	93.32	79	57	421	0.375	0.360	0.146	0.086	0.297	0.276
E	Low	11.26	10	3	54	0.470	0.457	0.164	0.095	0.461	0.426
	Medium	91.43	80	57	179	0.358	0.342	0.090	0.044	0.334	0.308
	High	687.23	358	181	19851	0.373	0.341	0.088	0.042	0.370	0.327
A	Low	13.81	12	2	36	0.454	0.445	0.198	0.141	0.371	0.337
	Medium	59.59	51	37	137	0.396	0.384	0.137	0.080	0.301	0.280
	High	383.71	206	138	12292	0.346	0.323	0.131	0.074	0.288	0.255
M	—	—	—	—	—	0.464	0.448	0.177	0.115	0.351	0.353
O	—	—	—	—	—	0.306	0.288	0.196	0.118	0.294	0.262
Aggregate‡	—	**117.62**	—	1	19851	**0.417**	—	0.166	—	0.365	—
DeepCAD	—	12.94	9	3	110	0.533	0.511	0.229	0.151	0.561	0.517
Fusion 360	—	16.25	10	3	388	0.456	0.440	0.165	0.101	0.409	0.363
MCB	—	—	—	—	—	0.503	0.491	**0.156**	0.090	**0.301**	0.296
Omni-CAD	—	25.97	11	2	5448	0.483	0.464	0.189	0.104	0.503	0.450

‡ Values represent a weighted average across all subsets. ↑ indicates increasing complexity; ↓ indicates lower similarity (corresponding to higher diversity or lower canonicality).

C Additional Results

C.1 Complete Benchmark Results Given Ideal Inputs

C.1.1 Per-Split Results

Complete benchmark results given idealized inputs (single-view grayscale renders and clean meshes) are reported in Table 8 (geometric fidelity metrics), Table 9 (executability metrics), and Table 10 (program quality metrics). Results for the other input modalities are summarized in Appendix C.3.

Table 8: **Geometric fidelity metrics** for models tested on ideal inputs. CD counts failed executions as the normalized bounding-box diagonal ($2\sqrt{3}$), as in Table 2.

		Low Face Count				Medium Face Count				High Face Count				Other		
Metric	Model	B	F	E	A	B	F	E	A	B	F	E	A	M	O	**Aggregate**
IoU	CADFit	0.985	0.980	0.969	0.924	0.967	0.955	0.835	0.790	0.919	0.869	0.837	0.789	0.903	0.618	**0.855**
	CAD-Recode	0.963 ±0.001	0.919 ±0.002	0.916 ±0.004	0.837 ±0.011	0.906 ±0.007	0.817 ±0.006	0.418 ±0.010	0.482 ±0.017	0.776 ±0.018	0.497 ±0.009	0.505 ±0.016	0.390 ±0.016	0.764 ±0.007	0.392 ±0.009	**0.661**
	CADEvolve	0.970 ±0.000	0.918 ±0.002	0.931 ±0.003	0.857 ±0.003	0.924 ±0.002	0.829 ±0.001	0.549 ±0.002	0.541 ±0.012	0.824 ±0.004	0.605 ±0.003	0.623 ±0.006	0.481 ±0.019	0.793 ±0.006	0.439 ±0.003	**0.708**
	Cadrille	0.961 ±0.001	0.911 ±0.002	0.907 ±0.003	0.837 ±0.008	0.899 ±0.003	0.800 ±0.004	0.469 ±0.006	0.501 ±0.011	0.799 ±0.001	0.566 ±0.033	0.544 ±0.002	0.421 ±0.020	0.742 ±0.066	0.418 ±0.004	**0.672**
	Claude Opus	0.640 ±0.003	0.577 ±0.010	0.571 ±0.003	0.517 ±0.005	0.547 ±0.001	0.431 ±0.008	0.348 ±0.007	0.325 ±0.003	0.451 ±0.003	0.354 ±0.009	0.430 ±0.010	0.338 ±0.006	0.512 ±0.002	0.267 ±0.005	**0.437**
	Gemini	0.701 ±0.006	0.631 ±0.006	0.609 ±0.008	0.545 ±0.003	0.588 ±0.005	0.482 ±0.007	0.392 ±0.010	0.312 ±0.014	0.479 ±0.004	0.372 ±0.004	0.456 ±0.004	0.338 ±0.009	0.512 ±0.009	0.216 ±0.012	**0.449**
	GPT-5.4	0.616 ±0.007	0.479 ±0.005	0.496 ±0.027	0.199 ±0.021	0.479 ±0.016	0.246 ±0.003	0.111 ±0.035	0.000 ±0.000	0.337 ±0.004	0.000 ±0.000	0.164 ±0.059	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	**0.174**
	Kimi K2.6	0.565 ±0.001	0.442 ±0.011	0.463 ±0.007	0.349 ±0.021	0.445 ±0.004	0.281 ±0.010	0.212 ±0.014	0.122 ±0.011	0.349 ±0.002	0.140 ±0.015	0.260 ±0.016	0.176 ±0.021	0.255 ±0.027	0.000 ±0.000	**0.254**
	Qwen3.5 27B	0.142 ±0.027	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	**0.008**
	Qwen3.5 9B	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	**0.000**
	CAD-Coder	0.648	0.539	0.513	0.407	0.479	0.388	0.194	0.216	0.353	0.216	0.274	0.199	0.418	0.180	**0.345**
CD	CADFit	0.027	0.026	0.027	0.035	0.028	0.028	0.033	0.041	0.033	0.035	0.032	0.042	0.035	0.074	**0.040**
	CAD-Recode	0.028 ±0.000	0.030 ±0.000	0.030 ±0.000	0.044 ±0.000	0.033 ±0.000	0.039 ±0.001	0.075 ±0.001	0.086 ±0.003	0.046 ±0.000	0.078 ±0.002	0.056 ±0.001	0.090 ±0.002	0.053 ±0.001	0.129 ±0.002	**0.066**
	CADEvolve	0.031 ±0.000	0.034 ±0.000	0.033 ±0.001	0.045 ±0.001	0.034 ±0.000	0.041 ±0.000	0.064 ±0.001	0.080 ±0.000	0.044 ±0.000	0.070 ±0.001	0.051 ±0.001	0.088 ±0.001	0.055 ±0.000	0.140 ±0.002	**0.067**
	Cadrille	0.029 ±0.000	0.031 ±0.000	0.031 ±0.000	0.042 ±0.001	0.034 ±0.000	0.041 ±0.000	0.067 ±0.001	0.082 ±0.004	0.045 ±0.000	0.068 ±0.003	0.050 ±0.000	0.084 ±0.001	0.056 ±0.002	0.118 ±0.000	**0.063**
	Claude Opus	0.082 ±0.001	0.092 ±0.001	0.083 ±0.001	0.118 ±0.005	0.101 ±0.001	0.122 ±0.003	0.092 ±0.001	0.141 ±0.002	0.131 ±0.001	0.130 ±0.002	0.069 ±0.001	0.113 ±0.002	0.116 ±0.000	0.189 ±0.002	**0.125**
	Gemini	0.065 ±0.001	0.073 ±0.001	0.075 ±0.001	0.107 ±0.004	0.088 ±0.001	0.111 ±0.001	0.084 ±0.000	0.146 ±0.004	0.119 ±0.002	0.126 ±0.002	0.069 ±0.001	0.116 ±0.001	0.115 ±0.000	0.211 ±0.002	**0.120**
	GPT-5.4	0.100 ±0.001	0.125 ±0.003	0.123 ±0.003	0.253 ±0.005	0.138 ±0.003	0.213 ±0.003	0.220 ±0.003	3.464 ±0.004	0.189 ±0.003	3.464 ±0.005	0.197 ±0.003	3.464 ±0.001	3.464 ±0.002	3.464 ±0.003	**1.821**
	Kimi K2.6	0.105 ±0.002	0.133 ±0.002	0.125 ±0.004	0.204 ±0.004	0.142 ±0.004	0.190 ±0.004	0.152 ±0.002	0.282 ±0.001	0.184 ±0.005	0.241 ±0.004	0.092 ±0.002	0.193 ±0.003	0.212 ±0.002	3.464 ±0.002	**0.729**
	Qwen3.5 27B	0.321 ±0.007	3.464 ±0.006	3.464 ±0.003	3.464 ±0.009	3.464 ±0.004	3.464 ±0.006	3.464 ±0.009	3.464 ±0.009	3.464 ±0.002	3.464 ±0.014	3.464 ±0.005	3.464 ±0.012	3.464 ±0.003	3.464 ±0.003	**3.290**
	Qwen3.5 9B	3.464 ±0.006	3.464 ±0.016	3.464 ±0.004	3.464 ±0.009	3.464 ±0.013	3.464 ±0.010	3.464 ±0.027	3.464 ±0.009	3.464 ±0.009	3.464 ±0.007	3.464 ±0.006	3.464 ±0.009	3.464 ±0.006	3.464 ±0.005	**3.464**
	CAD-Coder	0.089	0.106	0.113	0.167	0.135	0.157	0.172	0.203	0.173	0.198	0.136	0.194	0.146	0.260	**0.171**
SIoU	CADFit	1.000	0.999	0.990	0.840	0.990	0.974	0.824	0.614	0.889	0.751	0.853	0.626	0.713	0.443	**0.768**
	CAD-Recode	0.985 ±0.001	0.850 ±0.004	0.901 ±0.011	0.637 ±0.011	0.853 ±0.008	0.656 ±0.011	0.535 ±0.005	0.472 ±0.008	0.635 ±0.021	0.482 ±0.002	0.593 ±0.004	0.469 ±0.003	0.528 ±0.004	0.387 ±0.003	**0.601**
	CADEvolve	0.988 ±0.001	0.831 ±0.015	0.922 ±0.005	0.639 ±0.005	0.882 ±0.009	0.670 ±0.002	0.583 ±0.003	0.479 ±0.001	0.660 ±0.003	0.525 ±0.001	0.629 ±0.007	0.487 ±0.006	0.522 ±0.007	0.375 ±0.002	**0.610**
	Cadrille	0.976 ±0.002	0.837 ±0.003	0.866 ±0.007	0.642 ±0.017	0.832 ±0.011	0.643 ±0.013	0.559 ±0.008	0.485 ±0.001	0.656 ±0.005	0.516 ±0.008	0.626 ±0.005	0.490 ±0.004	0.518 ±0.035	0.392 ±0.004	**0.603**
	Claude Opus	0.298 ±0.004	0.274 ±0.008	0.330 ±0.005	0.302 ±0.010	0.348 ±0.010	0.312 ±0.004	0.401 ±0.001	0.346 ±0.004	0.343 ±0.004	0.334 ±0.004	0.344 ±0.007	0.349 ±0.011	0.266 ±0.006	0.320 ±0.003	**0.319**
	Gemini	0.311 ±0.006	0.274 ±0.004	0.328 ±0.000	0.284 ±0.014	0.335 ±0.001	0.297 ±0.010	0.386 ±0.009	0.310 ±0.009	0.335 ±0.003	0.300 ±0.005	0.336 ±0.010	0.307 ±0.007	0.221 ±0.012	0.294 ±0.002	**0.297**
	GPT-5.4	0.302 ±0.013	0.190 ±0.016	0.283 ±0.027	0.123 ±0.013	0.311 ±0.008	0.231 ±0.019	0.190 ±0.078	0.000 ±0.000	0.326 ±0.004	0.000 ±0.000	0.073 ±0.042	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	**0.113**
	Kimi K2.6	0.227 ±0.012	0.122 ±0.010	0.241 ±0.008	0.174 ±0.027	0.286 ±0.006	0.227 ±0.013	0.309 ±0.006	0.203 ±0.021	0.317 ±0.002	0.194 ±0.007	0.200 ±0.008	0.258 ±0.016	0.070 ±0.006	0.000 ±0.000	**0.165**
	Qwen3.5 27B	0.066 ±0.007	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	**0.004**
	Qwen3.5 9B	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	**0.000**
	CAD-Coder	0.336	0.324	0.361	0.331	0.386	0.368	0.400	0.374	0.353	0.360	0.420	0.361	0.294	0.313	**0.344**

Table 9: **VSR** for models tested on ideal inputs.

Metric	Model	Low Face Count				Medium Face Count				High Face Count				Other		Aggregate
		B	F	E	A	B	F	E	A	B	F	E	A	M	O	
VSR	CADFit	0.934	0.934	0.934	0.936	0.999	0.998	0.991	0.997	0.996	0.997	0.931	0.979	0.977	0.935	**0.965**
	CAD-Recode	0.965 ±0.009	0.949 ±0.007	0.949 ±0.011	0.926 ±0.014	0.949 ±0.019	0.936 ±0.010	0.858 ±0.008	0.897 ±0.008	0.899 ±0.014	0.825 ±0.005	0.810 ±0.014	0.837 ±0.027	0.918 ±0.001	0.855 ±0.009	**0.895**
	CADEvolve	0.935 ±0.001	0.932 ±0.000	0.932 ±0.002	0.917 ±0.003	0.997 ±0.001	0.982 ±0.004	0.985 ±0.001	0.927 ±0.003	0.991 ±0.002	0.951 ±0.003	0.971 ±0.004	0.909 ±0.007	0.947 ±0.003	0.944 ±0.003	**0.950**
	Cadrille	0.960 ±0.001	0.954 ±0.001	0.942 ±0.003	0.937 ±0.003	0.935 ±0.004	0.925 ±0.008	0.893 ±0.002	0.910 ±0.005	0.941 ±0.004	0.899 ±0.017	0.898 ±0.008	0.890 ±0.012	0.898 ±0.079	0.887 ±0.005	**0.913**
	Claude Opus	0.941 ±0.001	0.903 ±0.006	0.928 ±0.007	0.870 ±0.019	0.926 ±0.007	0.861 ±0.009	0.866 ±0.009	0.831 ±0.009	0.909 ±0.009	0.825 ±0.006	0.806 ±0.009	0.831 ±0.009	0.865 ±0.009	0.794 ±0.011	**0.860**
	Gemini	0.947 ±0.007	0.909 ±0.003	0.916 ±0.002	0.827 ±0.011	0.928 ±0.009	0.863 ±0.006	0.847 ±0.012	0.711 ±0.014	0.885 ±0.006	0.749 ±0.011	0.801 ±0.013	0.710 ±0.011	0.775 ±0.007	0.682 ±0.006	**0.804**
	GPT-5.4	0.887 ±0.004	0.767 ±0.013	0.795 ±0.051	0.590 ±0.015	0.824 ±0.017	0.671 ±0.006	0.587 ±0.037	0.456 ±0.010	0.779 ±0.013	0.464 ±0.006	0.558 ±0.032	0.418 ±0.017	0.466 ±0.003	0.329 ±0.017	**0.566**
	Kimi K2.6	0.847 ±0.006	0.718 ±0.010	0.801 ±0.007	0.653 ±0.022	0.842 ±0.004	0.699 ±0.012	0.726 ±0.011	0.580 ±0.015	0.820 ±0.002	0.607 ±0.005	0.721 ±0.017	0.633 ±0.019	0.581 ±0.008	0.475 ±0.004	**0.657**
	Qwen3.5 27B	0.575 ±0.008	0.470 ±0.014	0.458 ±0.016	0.351 ±0.008	0.463 ±0.015	0.382 ±0.007	0.237 ±0.011	0.226 ±0.016	0.421 ±0.007	0.222 ±0.002	0.227 ±0.016	0.195 ±0.006	0.363 ±0.002	0.297 ±0.006	**0.345**
	Qwen3.5 9B	0.336 ±0.017	0.243 ±0.009	0.258 ±0.025	0.160 ±0.007	0.298 ±0.010	0.206 ±0.020	0.160 ±0.005	0.124 ±0.004	0.278 ±0.006	0.134 ±0.011	0.168 ±0.012	0.124 ±0.021	0.232 ±0.003	0.127 ±0.003	**0.198**
	CAD-Coder	0.927	0.922	0.926	0.928	0.985	0.981	0.904	0.944	0.976	0.918	0.903	0.883	0.960	0.878	**0.928**

Table 10: **Program quality metrics** for models tested on ideal inputs.

Metric	Model	Low Face Count				Medium Face Count				High Face Count				Other		Aggregate
		B	F	E	A	B	F	E	A	B	F	E	A	M	O	
Token Count	CADFit	153	181	191	259	224	304	805	533	300	779	1041	918	418	1005	**553**
	CAD-Recode	76 ±1	86 ±0	112 ±2	178 ±6	144 ±0	178 ±3	356 ±7	346 ±3	201 ±2	348 ±5	286 ±3	392 ±4	179 ±2	337 ±3	**236**
	CADEvolve	123 ±0	117 ±1	175 ±2	184 ±1	202 ±2	226 ±1	455 ±3	350 ±6	246 ±1	396 ±6	488 ±6	493 ±5	229 ±0	322 ±1	**284**
	Cadrille	76 ±0	85 ±1	114 ±3	174 ±1	141 ±3	174 ±3	319 ±8	328 ±3	189 ±2	317 ±7	258 ±4	359 ±1	156 ±2	319 ±3	**220**
	Claude Opus	58 ±0	56 ±1	86 ±3	103 ±4	105 ±1	133 ±4	332 ±2	291 ±5	165 ±2	298 ±4	281 ±3	340 ±3	145 ±2	291 ±25	**198**
	Gemini	59 ±1	59 ±0	82 ±0	100 ±2	99 ±1	129 ±2	304 ±1	271 ±6	150 ±2	296 ±4	291 ±4	350 ±7	160 ±2	308 ±3	**200**
	GPT-5.4	77 ±1	80 ±2	105 ±2	126 ±3	130 ±2	157 ±2	400 ±10	317 ±3	193 ±1	373 ±6	300 ±4	392 ±11	179 ±1	353 ±2	**236**
	Kimi K2.6	63 ±0	60 ±0	86 ±1	105 ±2	114 ±0	138 ±5	435 ±13	318 ±4	184 ±3	385 ±16	404 ±11	494 ±16	128 ±1	430 ±3	**248**
	Qwen3.5 27B	74 ±1	71 ±3	92 ±1	98 ±5	118 ±0	130 ±4	253 ±6	222 ±8	157 ±3	250 ±8	227 ±3	267 ±11	123 ±2	184 ±3	**160**
	Qwen3.5 9B	81 ±2	76 ±1	119 ±4	115 ±2	137 ±2	151 ±4	246 ±7	237 ±3	159 ±7	229 ±6	230 ±4	274 ±14	135 ±4	202 ±10	**170**
	CAD-Coder	132	125	172	185	202	204	586	444	312	520	598	601	245	580	**364**
Operation Count	CADFit	12	13	15	19	18	23	66	42	24	59	83	73	32	73	**42**
	CAD-Recode	8 ±1	9 ±0	12 ±0	19 ±0	15 ±0	18 ±1	38 ±1	36 ±1	21 ±1	36 ±1	30 ±1	41 ±1	19 ±0	36 ±1	**25**
	CADEvolve	10 ±1	9 ±0	14 ±0	15 ±0	17 ±1	18 ±0	34 ±0	28 ±0	20 ±0	32 ±1	26 ±0	36 ±0	18 ±1	25 ±0	**22**
	Cadrille	8 ±1	9 ±0	12 ±0	18 ±0	15 ±1	18 ±0	34 ±1	35 ±1	20 ±0	33 ±1	27 ±0	38 ±1	17 ±0	34 ±0	**23**
	Claude Opus	6 ±0	6 ±0	7 ±0	11 ±1	9 ±1	11 ±0	20 ±1	22 ±0	13 ±1	22 ±1	14 ±0	22 ±1	14 ±0	21 ±2	**15**
	Gemini	5 ±0	5 ±0	6 ±0	9 ±0	7 ±0	10 ±0	16 ±0	19 ±0	12 ±1	19 ±0	13 ±0	21 ±1	12 ±0	19 ±0	**13**
	GPT-5.4	6 ±0	6 ±1	7 ±0	10 ±1	9 ±0	11 ±1	21 ±1	22 ±1	14 ±0	24 ±1	14 ±0	24 ±1	15 ±1	24 ±0	**16**
	Kimi K2.6	6 ±0	6 ±0	8 ±1	11 ±1	11 ±0	13 ±1	28 ±1	26 ±0	16 ±0	27 ±2	20 ±2	28 ±0	13 ±1	32 ±1	**19**
	Qwen3.5 27B	7 ±0	7 ±0	8 ±1	9 ±1	10 ±0	11 ±0	18 ±1	18 ±1	13 ±1	18 ±2	13 ±1	18 ±1	12 ±1	15 ±0	**13**
	Qwen3.5 9B	7 ±0	7 ±0	11 ±1	11 ±0	14 ±1	15 ±0	19 ±1	21 ±1	15 ±1	20 ±1	16 ±1	21 ±2	14 ±1	18 ±1	**15**
	CAD-Coder	16	13	20	22	24	24	68	52	38	62	69	71	29	67	**43**

C.1.2 Inference Speed Comparisons

In this section, we compare the inference speeds of the evaluated models. Table 11 details the time required to generate CadQuery code for each sample across the different methods. To ensure a fair comparison, all local models are run on a standardized setup featuring an NVIDIA RTX PRO 6000 GPU and an AMD Ryzen Threadripper PRO 7975WX CPU. We evaluate the models across the entire benchmark suite using the authors' original code, without applying any supplementary optimizations. For the large frontier models, we report API response times using single-view image inputs on a random subset of 100 samples. As the results show, CADFit requires significantly more inference time compared to the deep learning-based methods.

Table 11: **Inference time metrics** (in seconds) across evaluated methods. Local models were tested on the complete benchmark suite using standardized hardware. For frontier models, metrics reflect API response times over a random subset of 100 samples.

Metric	CADFit	CAD-Recode	CADEvolve	Cadrille	Claude Opus	Gemini	GPT-5.4	Kimi K2.6	Qwen 27B	Qwen 9B	CAD-Coder
Mean	453.1	2.095	1.491	3.262	6.870	30.388	7.478	41.089	44.566	19.575	81.058
Median	291.8	1.751	1.26	3.070	5.743	21.264	6.006	17.620	42.429	19.067	64.847
Std	540.2	1.460	2.715	1.286	3.779	45.554	5.127	80.087	17.910	4.540	57.201
Minimum	37.3	0.464	0.516	1.212	1.787	4.033	1.699	4.676	21.058	12.251	12.750
Maximum	6817.6	7.754	13.22	8.937	18.177	410.823	23.811	612.8	150.342	46.037	205.186

C.2 Understanding the Effects of Complexity and Diversity Sampling

In this section, we further investigate the impact of including complexity-stratification and diversity-aware sampling in our benchmark splits.

C.2.1 Why does IoU reconstruction performance improve on the highest face-count Extrude family split?

In Figure 3, we observed that median IoU increases for all evaluated models on the Extrude-High split relative to the Extrude-Medium split, despite a substantial increase in median face count (358 vs. 80). We hypothesize that, beyond a certain threshold, increases in face count are increasingly driven by localized surface detail rather than more complex global geometry. To investigate this hypothesis, we compute two measures of global shape compactness: solidity (mesh volume divided by convex hull volume) and bounding box occupancy (mesh volume divided by the volume of the minimum-volume oriented bounding box), see Table 12. Both metrics are higher for Extrude-High than for Extrude-Medium, indicating that these objects tend to occupy a larger fraction of their convex hull and oriented bounding box despite containing many faces. Figure 10 provides representative examples that are qualitatively consistent with this interpretation, with many Extrude-High parts exhibiting repeated fine features (often introduced through pattern operations) on otherwise simple bulk shapes.

Table 12: **Shape statistics for the Extrude-Medium and Extrude-High benchmark splits.** Extrude-High exhibits higher median solidity and bounding box occupancy than Extrude-Medium, indicating that the highest-face-count parts tend to have more compact global geometry.

Metric	Extrude-Medium (E_M)	Extrude-High (E_H)
Solidity	0.553	0.605
Bounding Box Occupancy	0.447	0.508

C.2.2 Does face count measure difficulty when operation vocabulary is expanded?

Figure 11 repeats the analysis of Figure 3 with all STEP-based families (B, F, E, A).

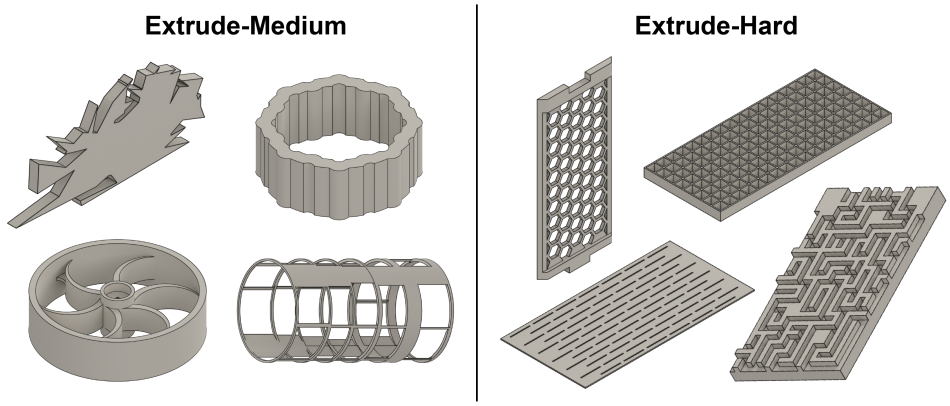


Figure 10: **Representative parts from the Extrude-Medium (left) and Extrude-High (right) splits.** Many Extrude-High parts reach large face counts through repeated patterns, such as hole arrays, grids, and slots, or through localized surface detail, while retaining simple global geometry; Extrude-Medium parts have more varied global shapes. This supports the hypothesis that the highest face-count split is not uniformly harder to reconstruct.

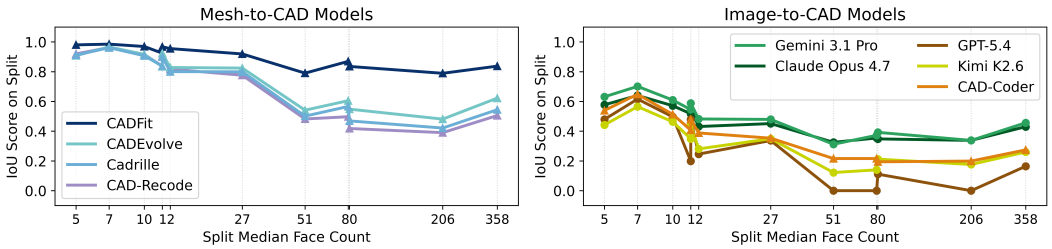


Figure 11: **Model performance versus face count across all STEP-based families.** Model performance as a function of face count when all STEP-derived benchmark families (B, F, E, A) are included. The y-axis shows median IoU score for a split, while the x-axis shows split median face count on a logarithmic scale. In contrast to the controlled sketch-and-extrude analysis in Figure 3, the relationship between face count and reconstruction performance is less monotonic when Fusion and All-Ops splits are included. This suggests that face count is a useful proxy for complexity within controlled operation families (e.g., sketch-and-extrude only), but does not fully capture reconstruction difficulty when operation vocabulary varies more broadly.

C.2.3 How Robust Is Face Count as a Complexity Proxy?

We use face count primarily to define difficulty splits within each benchmark family. This choice is motivated by its availability for every STEP sample, whereas source datasets do not provide consistent program metadata or feature trees that can be mapped to a common DSL. Face count has also been shown to correlate with expert-perceived CAD complexity (Contero et al., 2023). To assess the robustness of this proxy, we recover and report high-level source-operation counts for all benchmark families.

Across the 12,000 STEP-derived samples, pooling outputs from all eleven evaluated models, face count exhibits a stronger association with IoU than source-operation count, with Spearman correlations of $\rho = -0.534$ and $\rho = -0.279$, respectively. A similar pattern holds within CAD-Fusion ($\rho = -0.469$ vs. -0.262) and CAD-All-Ops ($\rho = -0.405$ vs. -0.305). These results indicate that both measures capture aspects of reconstruction difficulty, while face count provides a stronger empirical association with geometric fidelity in our benchmark.

However, the correlations remain moderate, and performance does not vary monotonically across all complexity splits. Moreover, absolute face counts are not directly comparable across benchmark families with different geometric distributions and modeling conventions. We therefore use face

count as a within-family stratification proxy rather than as a universal measure of CAD complexity across families. The high-count regime also exhibits a degree of saturation, further motivating this within-family interpretation.

C.2.4 DOES DINOV3 DIVERSITY SAMPLING MATTER? — DIVERSITY-SELECTED VERSUS RANDOM EVALUATION

To assess the effect of diversity-aware sampling on measured model performance, we construct randomly sampled variants of the Base-Low, Base-Medium, Extrude-Low, and Extrude-Medium splits while preserving the same face-count ranges used for complexity stratification. We then evaluate CADEvolve, CAD-Recode, and CAD-Coder on both the diversity-sampled CADBENCH splits and the corresponding random splits (Table 13). Across all four splits, diversity-selected samples yield lower median IoU than randomly sampled examples, suggesting that diversity-aware sampling can expose more challenging evaluation cases within the same nominal complexity range. Only two near-saturated Extrude-Low mesh cases are effectively tied (+0.3%, −0.2%), whereas IoU declines in all 10 others. On Extrude-Medium, it falls by 11.6%, 23.0%, and 21.1% for CADEvolve, CAD-Coder, and CAD-Recode. This larger drop may indicate that random sampling in this range includes more visually or geometrically redundant examples, while DINOv3-based sampling selects a more varied set of geometries. To verify the sampler's intended effect, we compare within-split DINOv3 embedding similarity under random and CADBENCH sampling for the four splits. Near-identical counterparts (similarity above 0.99) are eliminated in three splits and on Base-Low, their prevalence falls from 57.5% to 5.7%. These results confirm that the sampler substantially reduces rendered-shape redundancy.

Table 13: Comparison of model median IoU on Base and Extrude splits with and without diversity-aware sampling. Percent change is computed relative to the randomly sampled split.

Model	**Split**	**Diversity Sampled** (CADBENCH)	**Randomly Sampled**	**% Change**
CADEvolve	Base-Low	0.970 ± 0.000	0.984 ± 0.000	-1.4%
	Base-Medium	0.924 ± 0.002	0.936 ± 0.001	-1.3%
	Extrude-Low	0.931 ± 0.003	0.928 ± 0.000	$+0.3\%$
	Extrude-Medium	0.549 ± 0.002	0.621 ± 0.003	-11.6%
CAD-Coder	Base-Low	0.648	0.747	-13.3%
	Base-Medium	0.479	0.515	-7.0%
	Extrude-Low	0.513	0.575	-10.8%
	Extrude-Medium	0.194	0.252	-23.0%
CAD-Recode	Base-Low	0.963 ± 0.001	0.979 ± 0.001	-1.6%
	Base-Medium	0.906 ± 0.007	0.940 ± 0.001	-3.6%
	Extrude-Low	0.916 ± 0.004	0.918 ± 0.002	-0.2%
	Extrude-Medium	0.418 ± 0.010	0.530 ± 0.014	-21.1%

C.3 PERFORMANCE DELTAS ACROSS INPUT MODALITIES

Tables 14, 15, and 16 report performance deltas across input modalities, providing a more detailed view of model robustness to mesh noise, physically-based rendering, and multi-view inputs, and Table 17 reports the effect of standardized single-view render formats.

Table 14: **Robustness of mesh-to-CAD models to noisy mesh inputs.** We report aggregate IoU and VSR across CADBENCH splits for clean and noisy mesh inputs. Δ denotes the change from clean to noisy inputs, with negative values indicating degradation.

Method	Clean IoU	Noisy IoU	Δ IoU	Clean VSR	Noisy VSR	Δ VSR
CADFit	0.855	0.808	−0.047	0.965	0.998	+0.033
CADEvolve	0.708	0.123	−0.585	0.950	0.971	+0.021
Cadrille	0.672	0.569	−0.103	0.913	0.897	−0.016
CAD-Recode	0.661	0.498	−0.163	0.895	0.852	−0.043

Table 15: **Robustness of image-to-CAD models across varying image modalities.** We report aggregate IoU across CADBENCH splits and modalities: single-view (SV), physically-based (PBR), and multi-view (MV) renders. Deltas are computed relative to IoU on single-view inputs.

Method	SV IoU	PBR IoU	Δ_{PBR}	MV IoU	Δ_{multi}
Gemini 3.1 Pro	0.449	0.441	−0.008	0.434	−0.015
Claude Opus 4.7	0.437	0.425	−0.012	0.449	+0.012
GPT-5.4	0.174	0.166	−0.008	0.171	−0.003
Kimi K2.6	0.254	0.212	−0.042	0.164	−0.090
CAD-Coder	0.345	0.259	−0.086	0.153	−0.192
Qwen3.5 27B	0.008	0.007	−0.001	0.000	−0.008
Qwen3.5 9B	0.000	0.000	+0.000	0.000	+0.000

Table 16: **Executability robustness across image modalities.** We report aggregate valid shape rate (VSR) across CADBENCH for different modalities: single-view (SV), physically-based (PBR), and multi-view (MV). Deltas are computed relative to VSR on single-view inputs.

Method	SV VSR	PBR VSR	Δ_{PBR}	MV VSR	Δ_{multi}
Gemini 3.1 Pro	0.804	0.829	+0.025	0.811	+0.007
Claude Opus 4.7	0.860	0.871	+0.011	0.870	+0.010
GPT-5.4	0.566	0.581	+0.015	0.557	−0.009
Kimi K2.6	0.657	0.666	+0.009	0.609	−0.048
CAD-Coder	0.928	0.930	+0.002	0.883	−0.045
Qwen3.5 27B	0.345	0.361	+0.016	0.290	−0.055
Qwen3.5 9B	0.198	0.173	−0.025	0.132	−0.066

Table 17: **Effect of standardized image inputs on models with mesh- and image-conditioned inference modes.** We compare aggregate IoU and VSR when models are evaluated using their mesh-conditioned pipeline versus CADBENCH's standardized single-view (SV) renders. Δ denotes the change from mesh-conditioned to SV-conditioned inputs, with negative values indicating degradation. For both Cadrille and CADEvolve, switching from mesh-conditioned inputs to our standardized single-view renders causes IoU to drop effectively to zero.

Method	Mesh IoU	SV IoU	Δ IoU	Mesh VSR	SV VSR	Δ VSR
Cadrille	0.672	0.036	−0.636	0.913	0.771	−0.142
CADEvolve	0.708	0.092	−0.616	0.950	0.944	−0.006

C.3.1 How Realistic Are CADBench's Synthetic Modalities?

We assess the realism of CADBENCH's synthetic modalities through matched-instance comparisons with real-world scans and photographs. Specifically, we compare procedurally perturbed meshes against physical scans and physically-based renders (PBRs) against photographs of the corresponding objects. These comparisons provide a direct measure of how closely the synthetic modalities approximate their real-world counterparts, while retaining the scalability of fully synthetic generation.

For noisy meshes, we identify overlapping objects in the GenCAD-3D (Yu et al., 2025) scan dataset that are available as clean meshes, procedurally perturbed noisy meshes, and raw physical scans. We evaluate the surface fidelity of the clean and noisy meshes against their corresponding scans using SIoU and CD, as defined in Section 3; Table 18 reports the results.

The noisy meshes achieve higher SIoU and lower CD than the clean meshes, indicating that the procedural perturbations move the synthetic geometry closer to the corresponding physical scans. In particular, median SIoU improves by 18.3%, while median CD decreases by 7.3%. These results support the use of procedural mesh perturbations as a scalable approximation of real scan artifacts.

For rendered images, we similarly compare PBRs and single-view grayscale renders against real photographs of overlapping objects from CADBENCH CAD-Base provided by Doris et al. (2026).

Table 18: Comparison of clean and noisy meshes against corresponding physical scans.

Comparison	Mean SIoU ↑	Median SIoU ↑	Mean CD ↓	Median CD ↓
Clean Mesh vs. Raw Scan	0.6450	0.6450	0.0410	0.0257
Noisy Mesh vs. Raw Scan	**0.7114**	**0.7637**	**0.0387**	**0.0238**

Table 19: Comparison of synthetic renders with real photographs.

Image Type	KID ↓	Patch KID ↓	VGG Style Sim. ↑	BRISQUE ↓	NIQE ↓
Real Photos	0.0000	0.0000	1.0000	15.80 / 15.46	4.53 / 4.36
PBRs	**0.0412**	**0.0667**	**0.7063**	**29.69 / 29.02**	**7.43 / 7.64**
Single-View Grayscale Renders	0.0757	0.2268	0.4142	91.79 / 91.86	23.68 / 24.28

We measure distributional similarity using whole-image and patch-level Kernel Inception Distance (KID) and VGG Gram/style cosine similarity. We additionally report BRISQUE and NIQE to assess image naturalness without requiring real photographs as references. Table 19 reports the results.

PBRs consistently exhibit greater similarity to real photographs than single-view grayscale renders across the distributional metrics, while also achieving substantially better BRISQUE and NIQE scores. This suggests that PBRs better capture aspects of real imagery, including surface appearance and illumination, than the simpler grayscale rendering pipeline. Importantly, these realistic modalities retain the scalability and reproducibility of synthetic generation, allowing CADBENCH to introduce controlled real-world perturbations without requiring proprietary scanning or imaging hardware. Together, the matched-instance comparisons provide evidence that CADBENCH's noisy meshes and PBRs offer more realistic alternatives to idealized synthetic inputs for evaluating CAD generation under practical conditions.

C.3.2 Is CAD-Specialized Model Fragility Driven by Architecture or Training-Domain Mismatch?

The sensitivity of CAD-specialized models to input modality raises the question of whether this fragility is primarily architectural or instead reflects a mismatch between training and evaluation domains. To investigate this distinction, we conduct a targeted pilot experiment by fine-tuning a CAD-Coder-style model with a Qwen3.5 2B backbone on the DeepCAD training split using physically-based renders (PBR), and evaluate the resulting model on CAD-Base using both PBR and single-view grayscale inputs.

As shown in Table 20, the PBR-trained model achieves an IoU of 0.513 on PBR CAD-Base, compared with 0.409 when evaluated on single-view CAD-Base inputs. Thus, the modality ordering reverses relative to the original CAD-Coder, which was trained using single-view renders. The result indicates that training on a modality closer to the evaluation distribution can substantially affect performance, providing preliminary evidence that the observed modality sensitivity is at least partly attributable to training-domain mismatch rather than solely to model architecture.

At the same time, the pilot does not eliminate modality sensitivity: the PBR-trained model still performs differently across the two input conditions. Moreover, the experiment considers a single architecture, backbone, and training dataset, and therefore cannot fully disentangle architectural effects from training-data effects. Determining whether mixed-modality or noisy-input training can produce models that generalize consistently across modalities would require a broader controlled training study across architectures and datasets, which is beyond the scope of CADBENCH. We therefore interpret this experiment as evidence that training-domain alignment is an important contributor to modality sensitivity, rather than as a complete explanation of CAD-specialized model fragility.

Table 20: Pilot modality-matched training experiment on CAD-Base.

Training	Input	IoU ↑
PBR	Single-view	0.409
PBR	PBR	**0.513**

C.4 Are the Metrics Diagnostic?

C.4.1 Are IoU, Chamfer distance, and SIoU redundant? — Metric correlation analysis

To assess whether our geometric fidelity metrics provide complementary information, we compute sample-level Spearman correlations across approximately 1M syntactically valid model-generated CAD programs and visualize the corresponding pairwise metric relationships (Figures 12 and 13). IoU and SIoU are only moderately correlated, while CD is negatively correlated with both higher-is-better metrics, indicating that these metrics are related but not redundant.

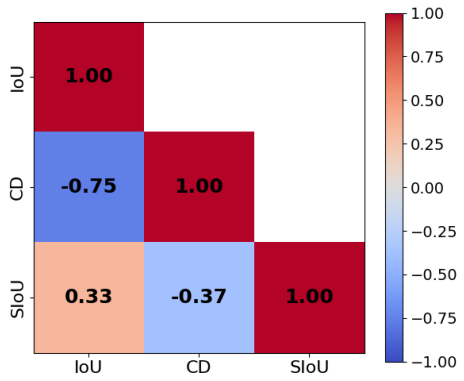


Figure 12: **Sample-level correlations between geometric fidelity metrics.** Spearman rank correlations are computed across approximately 1M syntactically valid model-generated CAD programs. IoU and SIoU are weakly to moderately correlated, while CD is negatively correlated with both higher-is-better metrics.

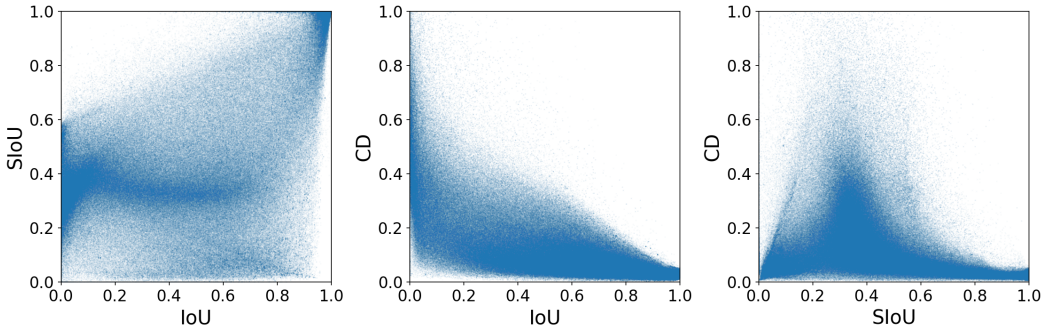


Figure 13: **Pairwise relationships between geometric fidelity metrics.** Each point corresponds to one syntactically valid model-generated CAD program from the full set of approximately 1M predictions. In particular, the broad scatter between IoU and SIoU shows that volumetric overlap and thresholded surface coverage can differ substantially at the sample level.

C.4.2 When is one geometric fidelity metric not enough? — High-IoU/low-SIoU and high-SIoU/low-IoU examples

Figure 14 visualizes two cases where IoU and SIoU disagree. In the high-IoU/low-SIoU case, the generated solid recovers the target's overall volume but misses fine surface structure, leading to poor surface coverage despite strong volumetric overlap. In the low-IoU/high-SIoU case, the generated model places surfaces near the target geometry but fails to recover the correct enclosed volume, for example due to incorrect thickness. These examples illustrate why volumetric and surface-based metrics are complementary rather than interchangeable.

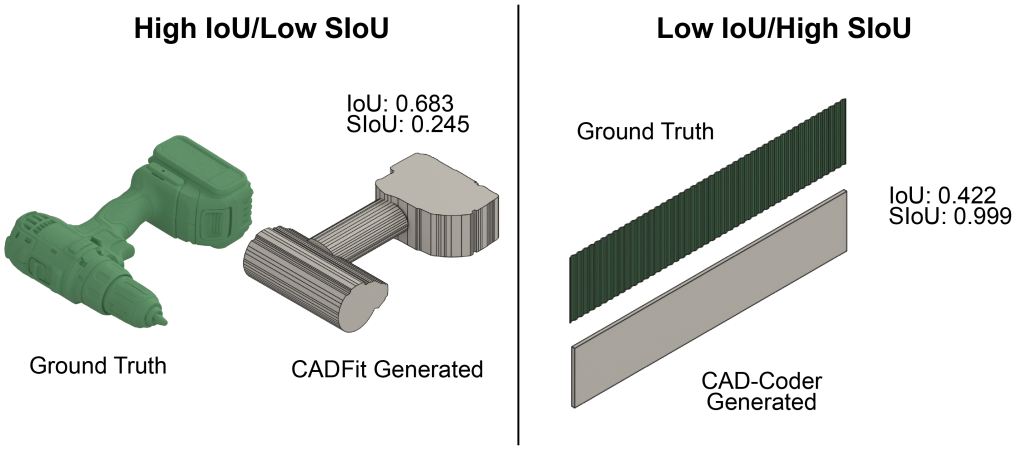


Figure 14: **Examples where IoU and SIoU disagree.** Left: a CADFit reconstruction of a cordless drill recovers the coarse volume (IoU 0.683) but misses most surface detail (SIoU 0.245). Right: a CAD-Coder reconstruction of a finely ridged plate lies within the surface-distance threshold of the target (SIoU 0.999) but does not match its enclosed volume (IoU 0.422). Reporting IoU alone would hide the first failure, and SIoU alone the second.

C.4.3 Can Geometric Fidelity Metrics Be Unified into a Single Ranking?

IoU, SIoU, and CD capture complementary aspects of geometric fidelity and can yield different method orderings when considered individually. To provide a unified comparative view, we perform a statistical ranking-based multi-metric analysis across all 18,000 CADBench samples. For each sample, methods are independently ranked by IoU in descending order, SIoU in descending order, and CD in ascending order. Averaging the per-example ranks for each metric gives an Average IoU Rank, Average SIoU Rank, and Average CD Rank. We then average these three ranks to obtain the **Average Multi-Metric Rank (AMMR)**, which provides a single consolidated measure across the three complementary geometric fidelity metrics.

We further assess whether differences in the resulting rankings are statistically meaningful using a paired non-parametric analysis. Specifically, we apply a Friedman test followed by a Nemenyi post-hoc comparison to the per-example rank values. The resulting critical distance determines whether differences between method rankings are statistically significant. Table 21 reports the analysis separately for mesh-to-CAD methods evaluated with clean meshes and image-to-CAD methods evaluated with single-view grayscale images.

The rank-based analysis reveals a sharper separation among mesh-to-CAD methods than among image-to-CAD methods. CADFit achieves the lowest AMMR at 1.773, with a rank gap of 0.936 to CADEvolve. This exceeds the Mesh-to-CAD critical distance of 0.03496 by a large margin, indicating statistically significant separation between CADFit and the remaining methods. In contrast, the remaining mesh-to-CAD methods have closely spaced AMMR values, with rank gaps of 0.044 and 0.010.

For image-to-CAD, Claude Opus 4.7 achieves the lowest AMMR at 2.829, followed closely by CAD-Coder at 2.846 and Gemini 3.1 Pro at 2.967. With a critical distance of 0.06714, the rank gap

Table 21: Average Multi-Metric Rank (AMMR) across CADBENCH geometric fidelity metrics. Lower is better. Rank Gap denotes the difference from the immediately preceding method in the ranking.

Setting	Method	AMMR	Gap
Mesh-to-CAD	CADFit	**1.773**	–
	CADEvolve	2.709	0.936
	Cadrille	2.753	0.044
	CAD-Recode	2.764	0.010
Image-to-CAD	Claude Opus 4.7	**2.829**	–
	CAD-Coder	2.846	0.017
	Gemini 3.1 Pro	2.967	0.121
	Kimi K2.6	3.890	0.923
	GPT-5.4	4.245	0.355
	Qwen3.5 27B	5.329	1.084
	Qwen3.5 9B	5.893	0.564

(a) Gemini 3.1 Pro

```
import cadquery as cq
length_x = 25.4
width_y = 38.1
height_z = 44.45
result = cq.Workplane("XY").box(
    length_x, width_y, height_z)
```

(b) CADFit

```
import cadquery as cq
import math
plane_1 = cq.Plane(
    origin=(0.0, -0.67, 0.0),
    normal=(0.0, 1.0, 0.0),
    xDir=(1.0, 0.0, 0.0))
sketch_1 = cq.Workplane(plane_1)
loop_1 = sketch_1.moveTo(-1.0, 1.0)
loop_1 = loop_1.lineTo(1.0, 0.99467)
loop_1 = loop_1.lineTo(0.99467, -1.0)
loop_1 = loop_1.lineTo(-1.0, -0.99467)
loop_1 = loop_1.lineTo(-1.0, 0.99733)
loop_1 = loop_1.close()
sketch_1 = sketch_1.add(loop_1)
solid_1 = sketch_1.extrude(1.34)
result = solid_1
```

Figure 15: **Programs generated for the same CAD-Base block.** (a) Gemini 3.1 Pro exposes the block's dimensions as named parameters; (b) CADFit hard-codes the vertex coordinates of a sketch loop. Line breaks inside function calls were added for display.

between Claude Opus 4.7 and CAD-Coder (0.017) does not indicate statistical separation, whereas the subsequent gaps exceed the critical distance. The remaining methods exhibit progressively larger rank differences. Overall, AMMR provides a unified view of performance across complementary geometric fidelity metrics, while the statistical analysis distinguishes closely ranked methods from those with consistently different per-example performance.

C.5 How Editable Are the Generated CAD Programs?

Executability and geometric fidelity do not directly capture whether generated programs are practical to modify. We therefore evaluate three editability-oriented metrics, whose aggregate scores appear in Table 2: **Parameterization Ratio**, **DoF Compression**, and **Edit Robustness**. Together, these metrics assess whether geometric quantities are exposed as parameters, whether those parameters are reused efficiently, and whether the resulting programs remain valid under numerical edits.

Figure 15 illustrates this distinction for a simple block from CAD-Base (B): Gemini 3.1 Pro produces a compact parameterized representation, whereas CADFit represents the same geometry using a sequence of hard-coded vertex coordinates. The former exposes the principal dimensions as named parameters, whereas the latter hard-codes individual coordinates, making the intended geometric degrees of freedom less explicit and more difficult to modify.

Table 22: Parameterization metrics across CADBENCH models. Higher is better. Param. denotes Parameterization Ratio and DoF denotes DoF Compression.

Model	Param.	DoF	Model	Param.	DoF
CADFit	0.000	0.000	Claude Opus 4.7	0.726	0.193
CAD-Recode	0.000	0.000	Gemini 3.1 Pro	0.774	0.069
CADEvolve	0.000	0.000	GPT-5.4	0.830	0.204
Cadrille	0.000	0.000	Kimi K2.6	**0.842**	0.289
CAD-Coder	0.000	0.000	Qwen3.5 27B	0.788	0.282
			Qwen3.5 9B	0.689	**0.314**

Parameterization Ratio ($\in [0, 1]$) measures the fraction of numeric values at CadQuery geometry call sites that are supplied through named variables or expressions rather than bare literals. For example, `box(10, 20, 30)` scores 0, whereas `box(length, width/2, height)` scores 1. **DoF Compression** measures how efficiently a program's geometry is controlled by a small number of independent parameters. We define it as

$$\text{DoF Compression} = 1 - \frac{\text{DoF}}{N} \in [0, 1], \tag{1}$$

where DoF is the number of bare literals plus the number of independent root parameters that drive the geometry, and N is the number of numeric slots at CadQuery geometry call sites. This complements Parameterization Ratio by penalizing redundant parameterization: wrapping every literal in a separate variable can increase Parameterization Ratio without reducing the number of independent degrees of freedom.

Edit Robustness measures whether a generated program remains executable and produces a watertight solid under parameter perturbations. We perturb every numeric degree of freedom independently by $\pm\delta$, re-execute the program, and report the fraction of programs that produce a watertight solid. We evaluate robustness at $\delta = 5\%$ and $\delta = 10\%$.

Tables 22 and 23 summarize the results across all benchmark families. The parameterization results reveal a clear distinction between exposing numeric quantities and actually compressing the underlying degrees of freedom. All CAD-specialized models have zero Parameterization Ratio and DoF Compression under our definitions, whereas the general-purpose VLMs parameterize a substantial fraction of geometric quantities, with Parameterization Ratios ranging from 0.689 to 0.842. However, DoF Compression remains modest, reaching at most 0.314. Thus, explicitly naming geometric quantities does not necessarily imply that the program captures a compact set of reusable design parameters.

Edit Robustness provides a complementary view of editability. Several general-purpose VLMs remain highly robust under both perturbation levels, with robustness reaching 0.969 at 5% and 0.966 at 10%. CADEvolve also achieves relatively high robustness despite having zero Parameterization Ratio and DoF Compression, demonstrating that numerical robustness is distinct from explicit parameter exposure. Conversely, CADFit exhibits substantially lower robustness, particularly at 10%, despite achieving strong geometric fidelity. Qualitative inspection suggests that dense, vertex-level sketches are more susceptible to self-intersection and rebuild failure under parameter perturbations than simpler primitive-based constructions.

These results show that editability is not captured by a single property. A program can expose few explicit parameters yet remain robust to numerical perturbations, as illustrated by CADEvolve, while a highly parameterized program may still contain many independent degrees of freedom. The three metrics therefore provide complementary measurements of how generated CAD programs can be modified, rather than treating editability as synonymous with either parameterization or geometric robustness alone.

These metrics extend CADBENCH beyond executability and geometric fidelity toward practical aspects of program editability. They do not, however, capture design intent, feature-tree correspondence, sketch constraints, or the semantic meaning of parameters. Evaluating these properties would require construction-history annotations that are not consistently available across the CADBENCH source datasets. Moreover, a given geometry may admit multiple valid and editable construction programs.

Table 23: Edit Robustness across CADBENCH models. Higher is better.

Model	@ 5%	@ 10%
CADFit	0.463	0.370
CAD-Recode	0.705	0.614
CADEvolve	0.879	0.811
Cadrille	0.728	0.640
CAD-Coder	0.773	0.721
Claude Opus 4.7	0.929	0.920
Gemini 3.1 Pro	0.878	0.864
GPT-5.4	0.885	0.877
Kimi K2.6	0.907	0.898
Qwen3.5 27B	0.954	0.948
Qwen3.5 9B	**0.969**	**0.966**

Table 24: Single-shot and agentic refinement performance on the balanced 540-sample subset.

Metric	Single-shot	Agentic
VSR ↑	0.839	**1.000**
IoU ↑	0.414	**0.493**
Executions / sample	0.0	2.4
Renders / sample	0.0	6.9
Wall-clock cost (s)	5.74	30.99 (5.4×)

We therefore view semantic and feature-level editability as complementary directions for future evaluation.

C.6 How Do General-Purpose VLMs Perform Under Iterative Refinement?

Single-shot generation evaluates whether a model can produce a valid CAD program from an input observation, whereas practical CAD workflows often involve iterative execution, inspection, and correction. To assess the extent to which such feedback improves reconstruction, we evaluate Claude Opus 4.7 in a five-execution agentic refinement loop on a 540-sample subset of CADBENCH consisting of 30 samples from each difficulty level in B, F, E, A families and 90 samples each from M and O families.

We use the same model and prompts as in the single-shot setting, but additionally provide tools for executing the generated CadQuery program, receiving execution errors as tracebacks, and rendering the resulting solid from arbitrary viewpoints. The number of renders is unrestricted, and the model determines when to submit its final program.

As shown in Table 24, agentic refinement raises VSR from 0.839 to 1.000, eliminating all execution failures on this subset, while aggregate IoU increases from 0.414 to 0.493. However, this aggregate IoU improvement is largely attributable to repairing initially invalid programs, which receive an IoU of zero in the single-shot evaluation. Newly executable samples account for 95% of the aggregate IoU gain. To isolate geometric refinement from execution repair, we therefore restrict the analysis to samples that execute successfully in both settings (Table 25).

For programs that are executable in both settings, the geometric differences are small and all 95% confidence intervals include zero. Thus, the refinement loop provides no evidence of a statistically significant improvement in geometric reconstruction once execution failures are excluded. The per-iteration results (Table 26) show the same pattern: most of the aggregate improvement occurs during the early iterations as execution errors are corrected.

Visual self-verification can reveal conspicuous defects, such as a failed program or an entirely missing hole, but visual feedback does not reliably expose fine geometric deviations or errors in CAD topology. Consistent with this limitation, the agent performs only 2.4 executions and requests

Table 25: Change in geometric fidelity between agentic and single-shot generation on samples executable in both settings. Confidence intervals are 95%.

Metric	**Δ**	**95% CI**
IoU	+0.005	$[-0.015, +0.026]$
CD	-0.004	$[-0.015, +0.007]$
SIoU	-0.004	$[-0.037, +0.031]$

Table 26: Performance as a function of the maximum execution budget k on the 540-sample subset.

Budget k	1	2	3	4	5
VSR ↑	0.833	0.944	0.989	0.994	1.000
IoU ↑	0.403	0.466	0.486	0.490	0.493

6.9 renders per sample on average before submitting its final program. This additional interaction increases wall-clock inference cost from 5.74 s to 30.99 s, corresponding to a $5.4\times$ increase.

These results indicate that iterative execution and visual feedback primarily improve *executability* rather than geometric precision. Even with multiple execution opportunities, arbitrary-view rendering, and access to execution tracebacks, the agent does not produce a statistically significant geometric improvement on programs that are already executable. The experiment therefore highlights a limitation of current general-purpose VLMs: feedback can help repair invalid CAD code, but does not by itself provide the spatial precision and CAD-specific reasoning required for consistently more accurate geometric reconstruction.

C.7 Qualitative Results

Figures 16 and 17 show reconstructions of CAD-Mechanical (M) parts by mesh-to-CAD and image-to-CAD models, respectively, and Figures 18 and 19 compare all evaluated models on the same samples.

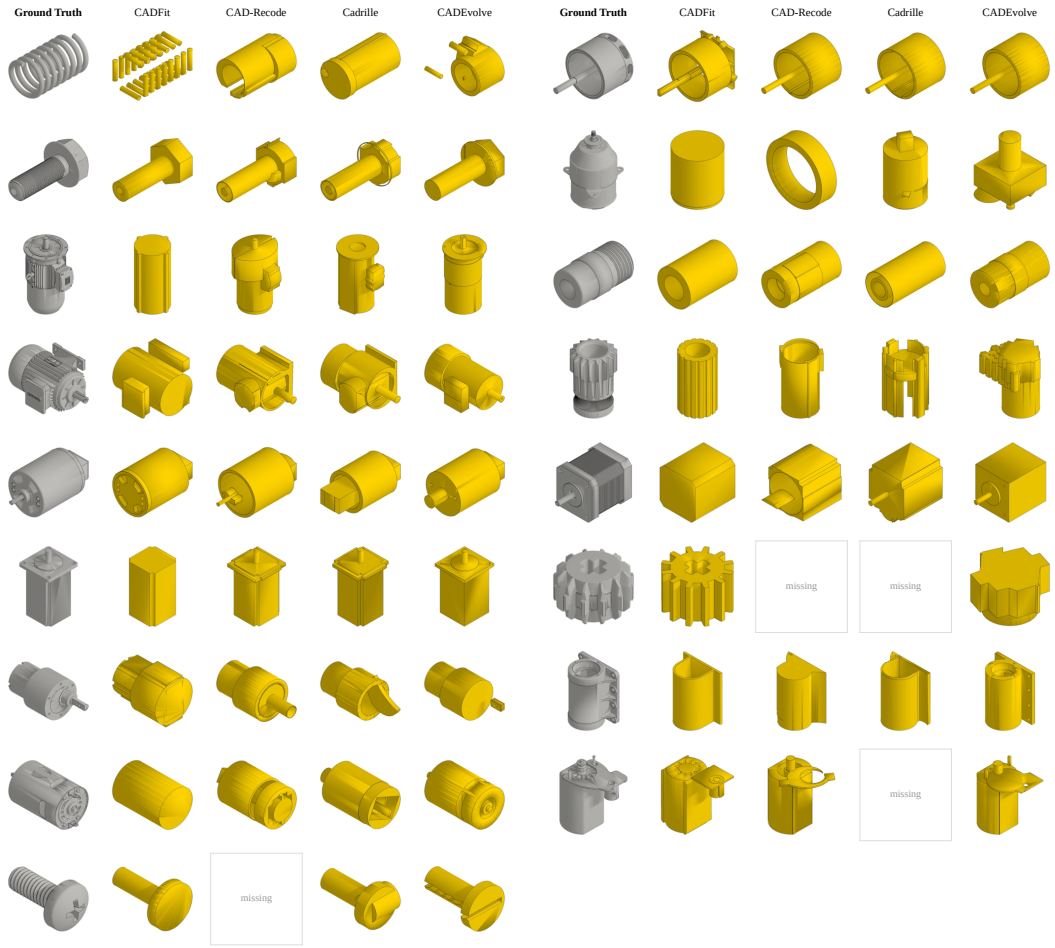


Figure 16: **Qualitative results for mesh-to-CAD models on CAD-Mechanical (M).** Each row shows a ground-truth shape (gray) followed by the reconstructions from CADFit, CAD-Recode, Cadrille, and CADEvolve. "missing" indicates that the model did not produce a valid solid.

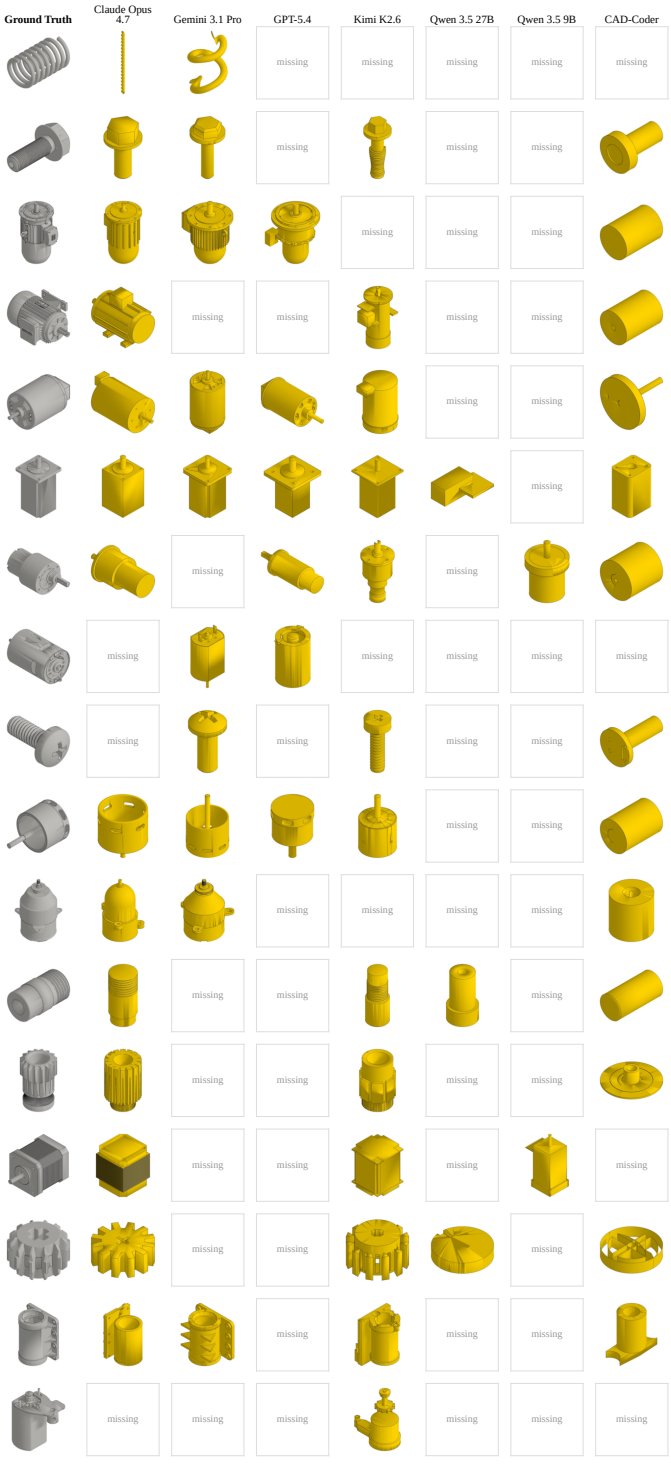


Figure 17: **Qualitative results for image-to-CAD models on CAD-Mechanical (M).** Each row shows a ground-truth shape (gray) followed by the reconstructions from Claude Opus 4.7, Gemini 3.1 Pro, GPT-5.4, Kimi K2.6, Qwen3.5 27B, Qwen3.5 9B, and CAD-Coder. "missing" indicates that the model did not produce a valid solid.

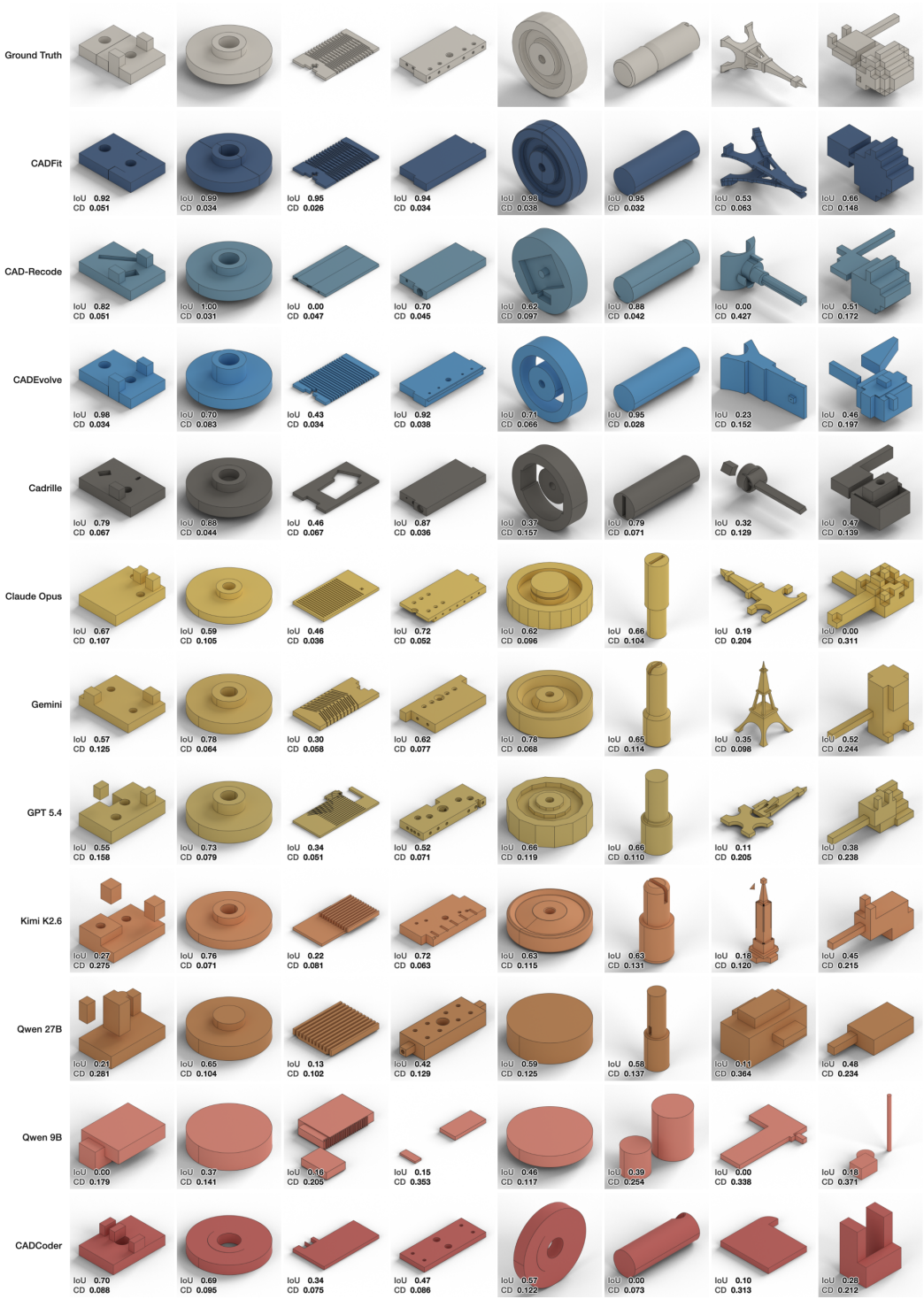


Figure 18: **Qualitative comparison of all evaluated models.** Each column is a benchmark sample; the top row shows the ground truth and each subsequent row shows one model's reconstruction, annotated with its IoU and CD.



Figure 19: **Additional qualitative comparison of all evaluated models across benchmark families.** Each column is a benchmark sample; the top row shows the ground truth and each subsequent row shows one model's reconstruction, annotated with its IoU, SIoU, and CD.