cadrille: Multi-modal CAD Reconstruction with Online Reinforcement Learning

Maksim Kolodiazhnyi[12*] kolodiazhnyi@airi.net
Denis Tarasov[13*] tarasov@airi.net
Dmitrii Zhemchuzhnikov[12] zhemchuzhnikov@airi.net
Alexander Nikulin[12] nikulin@airi.net
Ilya Zisman[1] zisman@airi.net
Anna Vorontsova
Anton Konushin[12] konushin@airi.net
Vladislav Kurenkov[14] kurenkov@airi.net
Danila Rukhovich

[1]AIRI Institute; [2]Lomonosov Moscow State University; [3]ETH Zurich; [4]Innopolis University

Abstract

Computer-Aided Design (CAD) plays a central role in engineering and manufacturing, making it possible to create precise and editable 3D models. Using a variety of sensor or user-provided data as inputs for CAD reconstruction can democratize access to design applications. However, existing methods typically focus on a single input modality, such as point clouds, images, or text, which limits their generalizability and robustness. Leveraging recent advances in vision-language models (VLM), we propose a multi-modal CAD reconstruction model that simultaneously processes all three input modalities. Inspired by large language model (LLM) training paradigms, we adopt a two-stage pipeline: supervised fine-tuning (SFT) on large-scale procedurally generated data, followed by reinforcement learning (RL) fine-tuning using online feedback, obtained programatically. Furthermore, we are the first to explore RL fine-tuning of LLMs for CAD tasks demonstrating that online RL algorithms such as Group Relative Preference Optimization (GRPO) outperform offline alternatives. In the DeepCAD benchmark, our SFT model outperforms existing single-modal approaches in all three input modalities simultaneously. More importantly, after RL fine-tuning, cadrille sets new state-of-the-art on three challenging datasets, including a real-world one.

1 Introduction

Computer-Aided Design (CAD) is the core of modern engineering and manufacturing, providing the tools to create detailed and modifiable 3D models [4]. Creating CAD models manually requires skills, time, and effort. To simplify this process, CAD reconstruction aims at generating CAD models directly from scanned objects, making the process faster, cheaper, and more accessible overall [36].

Typically, CAD models are created with a sequence of 2D sketches and 3D operations [49, 50]. This representation allows CAD models to be easily edited, making it prevalent in popular CAD tools like SolidWorks and AutoCAD and in CAD generation research. Most existing CAD generation methods define CAD sequences using special command tokens [17, 50]. However, state-of-the-art results are

[*]Equal contribution

Preprint. Under review.

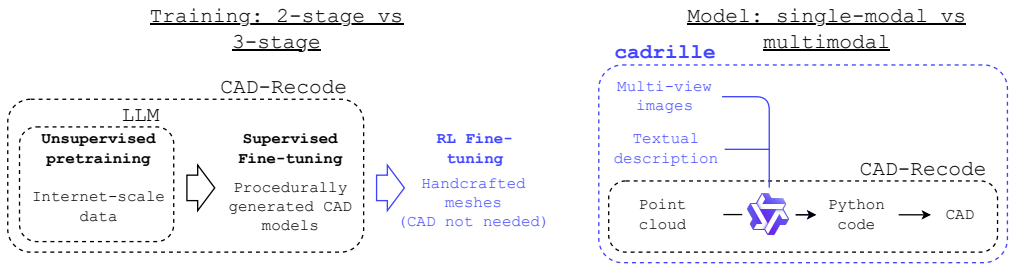

Figure 1: Compared to CAD-Recode, the only existing method of CAD reconstruction as LLM-based code generation, `cadrille` has two key novelties. First, it goes beyond the standard training scheme and pioneers LLM RL fine-tuning for CAD reconstruction (left). Moreover, besides point clouds only accepted by single-modal CAD-Recode, `cadrille` extends to images and textual descriptions, making it the first multi-modal approach delivering state-of-art results (right).

obtained via mapping CAD sequences to casual Python code [36]. Following the same paradigm, we generate CAD models as executable Python scripts.

The most well-studied input modality in CAD reconstruction is naturally a point cloud [36]. However, point clouds can only be obtained when a physical 3D object is available, while scanning usually requires special equipment, making the process complicated for non-experts. Images capture finer details and can be sourced using customer low-end devices (e.g., smartphone cameras), hence relaxing the hardware requirements [5, 6]. In the meantime, textual descriptions can enrich the object representation with semantic context [18]. Using various input modalities, such as multi-view images, or natural language descriptions, would make design assistance applications simple even for non-experienced users. However, existing approaches typically focus on a single modality at a time, limiting their robustness and generality.

The task of Python code generation given various inputs naturally refers to the large Vision-Language models (VLM), which demonstrate strong reasoning capabilities across diverse modalities [38, 45, 55]. In this work, we harness the power of VLMs to build a multimodal CAD reconstruction model that simultaneously processes point clouds, multi-view images, and texts, and generates Python code.

Existing CAD reconstruction methods face generalization issues due to how they are trained. Specifically, handcrafted CAD datasets are small and limited in diversity, while models trained with procedurally generated data struggle to transfer to the real-world domain [17, 50]. Inspired by the standard LLM training pipelines [38], we introduce a multi-stage training paradigm in the context of CAD reconstruction. We use voluminous procedurally generated data for supervised training, while valuable but scarcer handcrafted data is reserved for RL fine-tuning. This scheme eliminates the need for large-scale handcrafted data and allows the model to first generalize across the CAD domain and then specialize using preference-based objectives. We test Direct Preference Optimization (DPO) [33], which was employed for the CAD reconstruction in [5]. Moreover, since online RL techniques dominate in tasks where feedback can be programatically computed, we also try the online Group Relative Preference Optimization (GRPO) technique. Our experiments demonstrate that online RL is more effective for CAD reconstruction, which aligns with the results obtained in other domains [38].

Our experiments show that `cadrille` outperforms existing modality-specific baselines in accuracy. Moreover, RL fine-tuning ensures validity of generated Python code, which posed a challenge for prior works. As a result, the proposed approach demonstrates unattainable robustness and sets a new state-of-the-art on several CAD datasets, including a real-world CC3D [29]. Essentially, this opens up new possibilities for generalization in open-world scenarios.

In summary, our contributions are as follows:

- We propose `cadrille`, a CAD reconstruction model that operates jointly on point cloud, images, and text modalities;
- We are the first to explore RL fine-tuning of LLM models in the context of CAD reconstruction. Specifically, we propose a novel paradigm to split procedurally generated and handcrafted data between supervised fine-tuning (SFT) and RL fine-tuning stages;

- With a single model, we simultaneously achieve state-of-the-art reconstruction quality for all input modalities on the DeepCAD dataset. RL fine-tuning pushes the quality even further in three benchmarks, including real-world CC3D.

2 Related Work

CAD Generation Existing CAD generation methods can be classified into three categories based on CAD model representations: constructive solid geometry (CSG) [7, 9, 11, 16, 31, 34, 39, 42, 57, 58], boundary representation (B-rep) [12, 15, 19, 21, 22, 24, 25, 40, 41, 44, 47, 53] and CAD sequence [3, 6, 5, 8, 17, 18, 20, 27, 30, 35, 36, 46, 50, 54, 52, 59, 61]. In the CSG [10] paradigm, CAD is represented as a CSG tree constructed using boolean operations (union, subtraction, difference) of geometric primitives (e.g., cubes, cylinders, or spheres). This approach fails to express intricate shapes and is generally not well-aligned with how engineers and designers actually build CAD models. B-rep [1] is a graph that describes connections between faces, edges, and vertices of a 3D model. Creating a B-Rep requires enforcing topological consistency on edges, which introduces additional complexity to the generation procedure and complicates editing of generated models.

CAD Sequence Reconstruction Unlike general CAD generation, which may prioritize plausibility, diversity, or creativity in design, CAD reconstruction aims at faithfulness to the given inputs, requiring the output model to match the original shape.

Point clouds are the most well-studied input modality in CAD reconstruction. The seminal work on point cloud-based CAD reconstruction by Wu et al. [50] proposed encoding CAD sketch-and-extrude sequences as special tokens. Beyond that, DeepCAD, a large-scale dataset of 180k hand-crafted CAD models, was presented. Subsequent works [8, 17, 52] adopted the same CAD representation and trained on the same DeepCAD dataset. More recently, CAD-Recode [36] introduced a paradigm shift by representing CAD models as Python code, providing greater expressiveness and flexibility, and released a new training dataset of approximately 1 million procedurally generated CAD samples.

Only few recent works [5, 6, 18, 48] have explored CAD reconstruction from alternative input modalities, such as single- or multi-view images and natural language descriptions. These approaches extend the DeepCAD dataset by rendering synthetic views or generating textual captions for existing CAD models. Among them, CADCrafter [5] stands out for its unified framework that handles both single- and multi-view inputs, whether rendered or real. In the domain of text-to-CAD, the most notable results come from Text2CAD [18], which uses a vision-language model (VLM) to generate detailed captions for DeepCAD shapes and then trains a model to predict the corresponding sketch-and-extrude sequences from these descriptions.

Generally, state-of-the-art CAD reconstruction approaches are tailored to process specific input modalities with distinct architectures, while multimodal CAD reconstruction remains underexplored. Recent CAD-GPT [46] reconstructs a CAD model given a single image and textual description, yet falls behind CADCrafter [5] and Text2CAD [18] by a large margin (up to two orders of magnitude!). We claim `cadrille` to be the first competitive multimodal CAD reconstruction approach, which can jointly handle three input modalities (point cloud, images, and text) within a unified framework.

RL for CAD and LLM The training of modern LLMs [45, 55] is performed in several stages. The process starts with unsupervised pretraining on internet-scale data to gain broad structural knowledge of the textual domain. Then, supervised fine-tuning (SFT) is applied to train the model for specific tasks. Alignment, also known as preference optimization, helps to strengthen model's focus on the best answers, which can be sourced from human feedback, or automatically by using elaborate RL approaches [32, 33, 38].

Reinforcement learning has also been explored in CAD-related tasks [5, 39, 56, 60]. For example, RLCAD [56] proposes an RL-based policy that interacts with a CAD engine to generate sketch-and-revolve sequences, receiving feedback based on the quality of reconstruction. Chao et. al. [60] applies Deep Q-Networks to recover parametric CAD models from 2D orthographic drawings in the form of loop-paths. CADCrafter [5] implements a code checker to serve as an implicit reward model and fine-tunes using offline Direct Preference Optimization (DPO) [33].

However, none of these works applies reinforcement learning to fine-tune large language models (LLMs) for CAD reconstruction. In this work, we are the first to explore RL fine-tuning of LLMs in

this domain. We show that online RL methods like GRPO [38] are particularly effective, eliminating the need for human feedback.

3 CAD Sequence Reconstruction

Problem Formulation The task of CAD reconstruction implies recovering a CAD model given a multimodal input q, which can be a 3D point cloud, a set of images, or a textual description. We represent CAD models as Python scripts [36] that, when executed, generate a parametric Boundary Representation (B-Rep) of a 3D shape. Respectively, given an input q, we search for a trainable policy π_θ, s. t. $\pi_\theta(q)$ produces a token sequence τ, which is essentially a text of a Python program generating a CAD model.

Multimodal Data For training a model, we derive all input modalities from ground-truth CAD models (Fig. 2). Below, we describe how each modality is constructed according to the established data generation protocols [5, 18, 50].

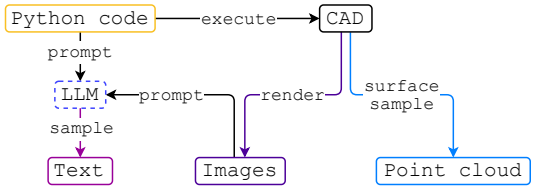

Figure 2: Overview of multimodal data generation pipeline producing textual descriptions, multi-view images and point clouds.

Given a CAD model as a parametric 3D shape (B-Rep), we sample points directly from the parametric surfaces of the model. Modern CAD engines provide built-in routines for surface sampling, making it simple and straightforward.

To generate images, the B-Rep is first tessellated, i.e., converted into a triangle mesh that approximates the surface geometry. Then, this mesh can be rendered from multiple viewpoints to produce multi-view image inputs.

Generating textual data is notably more challenging. Since our goal is accurate geometry reconstruction rather than generating a semantically relevant sample, inputs should provide detailed and comprehensive geometric information. Consequently, loose textual descriptions are generally insufficient. The necessary level of granularity is investigated in Text2CAD [18], where LLMs and VLMs are combined in a multi-stage sophisticated pipeline that generates textual descriptions from both the CAD sequence and rendered images.

4 Proposed Method

4.1 `cadrille` Architecture

The architecture of our `cadrille` is depicted in Fig. 3. The model accepts inputs in the form of a point cloud, a set of images, or a text prompt, and outputs a Python code, that when executed, produces a CAD model. `cadrille` is build on top of a VLM that provides native support of text and image inputs and is already capable of generating Python code. Textual input is passed through the original embedding layer, and images are processed with an original visual encoder. The point cloud

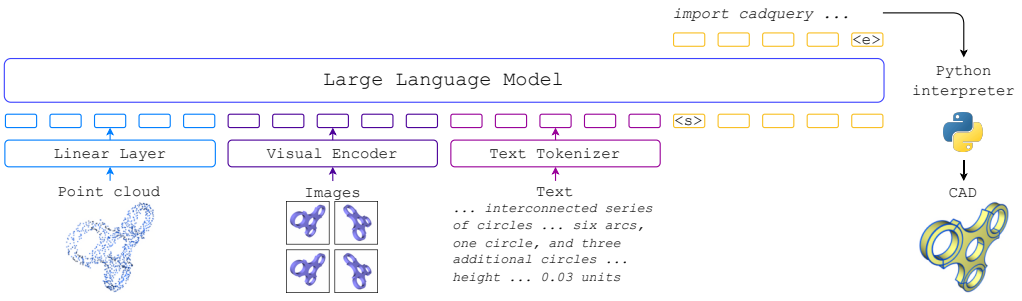

Figure 3: Overview of `cadrille`. It can handle three input modalities within a unified framework. Point clouds are processed with a trainable projection layer, while images and texts are passed to a VLM directly. The output of the model is an executable Python script for CAD generation.

processing logic is the same as in CAD-Recode [36]. Specifically, we use a single projection layer to embed 3D points, sample points from the surface via furthest point sampling, and do not use normals.

4.2 Supervised Fine-tuning

As shown on Fig. 1, `cadrille` benefits from three stages of training. First, we use VLM which is pre-trained on the internet-scale data in the unsupervised manner. After this stage, VLM is able to process textual and visual inputs and generate Python code, but lacks mechanisms to handle point clouds. In this work, we do not perform any unsupervised VLM training, but enjoy the capabilities of an already trained model.

The second stage is supervised fine-tuning for a specific task. During SFT, a model develops the ability to process point clouds and learns a policy π_θ to map multimodal inputs q to Python codes τ, making SFT an essential part of `cadrille` pipeline. We construct a training dataset $\mathcal{D}$ of samples $(q,\ \tau)$, where q is a multimodal input. The training procedure aims to minimize cross-entropy between ground truth and predicted Python code tokens:

$$\mathbb{E}_{(q,\tau)\sim\mathcal{D}}\left[\log \pi_\theta(\tau \mid q)\right]$$

4.3 Limitations of SFT

Two-stage training has already been adopted in CAD-Recode [36], which employs supervised fine-tuning (SFT) to adapt a pretrained language model for point cloud-based CAD reconstruction. However, this strategy reveals its limitations in a cross-domain scenario: CC3D IoU is as low as 60% and the invalidity ratio (IR) is to 10%, which means that every tenth prediction fails to produce a valid output (Tab. 3, row 2). To mitigate this issue, CAD-Recode uses a test-time sampling technique. For each input query, 10 candidate Python programs are generated, and the candidate with the highest IoU is selected. After that, IoU increases to 74%, while IR drops below 0.5%. However, this improvement comes at the cost of a 10x increase in inference time. Can similar gains be achieved without sacrificing test-time efficiency?

To maintain fast and simple inference, we shift our focus to improving the training process. Training solely on procedurally generated CAD data might limit performance in real-world applications. Nevertheless, training on handcrafted models also presents challenges, e.g., prior work [36] shows that SFT directly on the DeepCAD dataset harms performance, leading to a 10% drop in IoU.

Our experiments confirm that simply mixing procedurally generated and handcrafted data for training fails to improve results and can even degrade performance (Tab. 3, row 4). We attribute this to inconsistency in CAD sequences across datasets: for instance, DeepCAD models are constructed using commands like extruded cuts and symmetric extrusions, which are not present in the generation procedure of the CAD-Recode dataset.

To address this limitation, we introduce a novel third stage in the training pipeline, namely, reinforcement learning fine-tuning on handcrafted data not annotated with CAD sequences. This approach resolves inconsistency issues while still allowing the model to adapt to real-world domain.

4.4 RL Fine-tuning

We formulate RL fine-tuning as follows. Given a dataset of inputs (either images or point clouds) $\mathcal{D} = \{q_i\}_{i=1}^N$, and reward function $R(\tau)$, we learn LLM policy $\pi_\theta(\tau \mid q)$ that generates a Python code τ for an input q, s.t. it maximizes the expected reward $\mathbb{E}_{q_i\sim D,\tau_i\sim\pi_\theta(\cdot|q_i)}\left[R(\tau_i)\right]$.

Note that at this stage annotated pairs of (q, τ) are not needed for supervision, since Python codes τ are being sampled from the trained SFT model. In fact, **CAD sequences are not needed for RL fine-tuning**, and the data requirements can be relaxed to 3D meshes instead. This is especially beneficial from the practical perspective, as RL fine-tuning can be performed using generally more accessible mesh datasets, which opens new possibilities for training models accommodated to artifacts present in real-world data.

The reward function $R(\tau)$ is a combination of terms that address precision and robustness:

$$R(\tau) = r_{\text{IoU}}(\tau) + r_{\text{invalid}}(\tau),$$

where r_{IoU} is an IoU between the CAD model produced by τ and ground truth 3D mesh, additionally multiplied by a factor of 10 to enforce precise reconstruction. r_{invalid} penalizes invalid predictions: it is set to -10 for invalid τ and 0 otherwise.

Empirically, we found that hard example mining leads to a faster convergence of RL fine-tuning. Consequently, we only use examples q where the reward $R(\tau)$ averaged over three samples produced by the SFT model is less than 7.5.

DPO Direct Preference Optimization (DPO) [33] learns from pairwise preference data, approximating an implicit reward via a reparameterized Bradley-Terry model.

We construct the training dataset by sampling $K = 5$ Python codes τ for each input q from the SFT model π_{θ_r} at temperature $T = 1.0$. At each training step for the given sample, we randomly select two outputs. The output with a larger reward $R(\tau)$ is considered to be a preferred prediction τ_w, and another is non-preferred τ_l. The optimization objective is formulated as:

$$\mathbb{E}_{(q,\tau_w,\tau_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_{\theta_t}(\tau_w \mid q)}{\pi_{\theta_r}(\tau_w \mid q)} - \beta\log\frac{\pi_{\theta_t}(\tau_l \mid q)}{\pi_{\theta_r}(\tau_l \mid q)}\right)\right]$$

DPO training starts with π_{θ_r} and proceeds for 10 epochs. After that, the SFT model is replaced with the latest π_{θ_t}, and trained for another 10 epochs. In this way, the model gradually diverges from the original SFT model. In our experiments, we found it to be beneficial for performance.

However, DPO performance is upper-bounded by the quality of the best generated sample for a given example. This limitation cannot be overcome without generating additional samples, so we adapt an online RL approach that can benefit from newly generated samples.

Dr. CPPO We combine two recent modifications of the popular GRPO [38] method: Dr. GRPO [26] which eliminates the need for a reference model and modifies the objective, and CPPO [23] which uses samples with the strongest signal. The hybrid approach ensures both computational efficiency and accuracy; hereinafter, it is referred to as Dr. CPPO.

G sequences $\{\tau_g\}_{g=1}^{G}$ are sampled from the current policy $\pi_{\theta_{\text{old}}}(\tau \mid q)$ for a given input q with temperature $T = 1.0$. For each output g, the advantage A_g is estimated as $A_g = r_g - \text{mean}(\{r_i\}_{i=1}^{G})$. N samples with the highest $|A_g|$ are used to form a batch $\mathcal{B}$ and perform policy update by maximizing PPO [37] objective:

$$\mathbb{E}_{\{\tau_g\}\sim\mathcal{B}}\left[\min\left(\frac{\pi_{\theta_t}(\tau_g \mid q)}{\pi_{\theta_{\text{old}}}(\tau_g \mid q)}A_g,\ \text{clip}\left(\frac{\pi_{\theta_t}(\tau_g \mid q)}{\pi_{\theta_{\text{old}}}(\tau_g \mid q)},\ 1-\epsilon,\ 1+\epsilon\right)A_g\right)\right]$$

5 Experiments

Datasets DeepCAD [50] (denoted as D in Tables) serves as our primary benchmark for supervised training. We adopt the Text2CAD version of DeepCAD, which enriches it with textual descriptions. The training set comprises approximately 160k samples, while 8046 are left for testing.

For SFT, we also use the procedurally generated CAD-Recode [36] dataset (denoted as R). It is an order of magnitude larger than DeepCAD, consisting of approximately 1 million CAD programs written in CadQuery [2], a parametric Python-based CAD language.

Fusion360 [49] (denoted as F) is a small CAD reconstruction benchmark with complex and realistic CAD models. In the standard experimental protocol, only the test subset (1725 samples) is used, as the lack of Python CAD sequences makes it unsuitable for conventional supervised training. Still, we can use its training set (6900 samples) for our annotation-free RL fine-tuning.

To show the versatility and applicability of our approach, in addition to handcrafted and procedurally generated meshes, we report metrics on the CC3D dataset [29]. It features 2973 input point clouds sampled from real scans of CAD models with noisy values, missing parts, and smoothed edges.

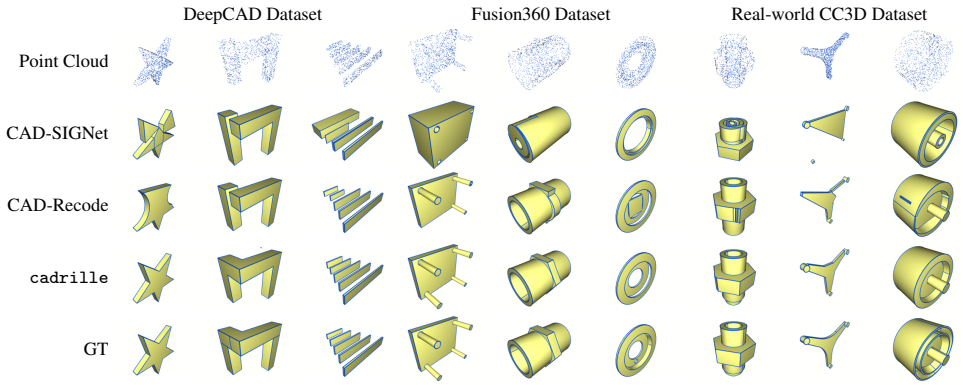

Figure 4: CAD models reconstructed from point clouds from the DeepCAD, Fusion360, and CC3D datasets.

Metrics Following CAD-Recode [36], we evaluate the quality of the predicted CAD models using three metrics: Chamfer Distance (CD), Intersection over Union (IoU), and Invalidity Ratio (IR). Since invalid CAD models introduce a notable bias into mean estimates, we report both mean and a more robust median CD, both computed using 8192 points. CD values are multiplied by 10^3. The divergence between ground truth and reconstructed meshes is measured using IoU (in %). The IR indicates the percentage of generated sequences that do not produce a valid CAD model.

5.1 Supervised Fine-Tuning

Results on DeepCAD In Tab. 1, we compare `cadrille` with single-modality CAD reconstruction methods on DeepCAD. Here, we specify input modalities with subscripts: *p* stands for point clouds, *i* denotes images, and *t* is for texts. `cadrille` trained jointly on point clouds, multi-view images, and texts from the DeepCAD training set (D_{pit}) outperforms the modality-specific baselines. Noticeably, IR is reduced almost twice for point clouds (from 1.1 to 0.4) and 7x for images (3.6 to 0.5).

Training using the large-scale procedurally generated CAD-Recode dataset (R) consistently improves accuracy over training on the DeepCAD dataset. Since we also use a Qwen LLM model as in CAD-Recode (Qwen2-VL-2B against Qwen2-1.5B), comparable quality of point cloud-based reconstruction is expected. When training `cadrille` on point clouds and images (R_{pi}), it maintains the same accuracy on point clouds but additionally extends to images. After training with point clouds, images, and texts ($S_{pi} + D_i$), `cadrille` generalizes across modalities without loss of quality on each modality. For fair comparison, we do not apply any RL techniques in this series of experiments, and mix up training datasets trivially for SFT.

Results on Fusion360 and CC3D Both Fusion360 and CC3D datasets do not provide annotations in a compatible format, and are only used for testing in the standard evaluation protocol [17]. Accordingly, testing on these datasets is performed in a zero-shot scenario, which allows assessing the generalization ability of CAD reconstruction approaches. Furthermore, since CC3D contains real scans of objects, this experiment emulates real-world application.

We report quality of image-based and point cloud-based CAD reconstruction in Tab. 2 and 3, respectively. CADCrafter [5] is the only method performing CAD reconstruction based on multi-view images. However, the authors of CADCrafter only report metrics on the DeepCAD dataset, and benchmarking it on other datasets is problematic since the code has never been released. To establish a baseline in image-based CAD reconstruction, we combine two off-the-shelf state-of-the-art methods, namely, multi-view reconstruction method LRM [13, 51] and CAD-Recode. LRM takes multi-view images as inputs and produces a mesh, which is turned into a point cloud via surface sampling, and this point cloud is then passed to CAD-Recode to create a CAD model. As can be seen in Tab. 2, `cadrille` trained on CAD-Recode outperforms both baselines.

In Tab. 3, we compare `cadrille` against the state-of-the-art approaches originally trained on the DeepCAD (CAD-SIGNet [17]) and CAD-Recode [36] datasets. As could be expected, `cadrille` is on par with CAD-Recode, while delivering substantially better quality w.r.t. CAD-SIGNet.

Method	Train Data	Point Cloud			Multi-view Images			Text		
		CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓
PointNet→DeepCAD [50]	D_p	9.64	46.7	7.1						
Point-BERT→HNC-CAD [52]	D_p	8.64	65.3	5.6						
MultiCAD [28]	D_p	8.09		11.5						
TransCAD [8]	D_p	4.51	65.5	1.1						
PrismCAD [20]	D_p	4.28	72.1	16.2						
Point2Cyl [43]	D_p	4.27	73.8	3.9						
CAD-Diffuser [27]	D_p	3.02	74.3	1.5						
CAD-SIGNet [17]	D_p	0.29	77.3	5.0						
DINOv2→HNC-CAD [52]	D_i				2.08		10.1			
DINOv2→DeepCAD [50]	D_i				1.13		10.6			
CADCrafter [5]	D_i				0.26		3.6			
BERT→DeepCAD [50]	D_t							32.82		10.0
Text2CAD [18]	D_t							0.37	71.5	**0.9**
`cadrille`	D_{pit}	**0.25**	**79.4**	**0.4**	**0.25**	**78.2**	**0.5**	**0.21**	**81.1**	1.4
CAD-Recode [36]	R_p	0.18	87.1	3.1						
`cadrille`	R_{pi}	0.18	87.1	2.1	0.18	86.1	1.5			
`cadrille`	R_{pi}+D_t	**0.18**	**87.1**	**2.1**	**0.18**	**86.1**	**1.5**	**0.20**	**82.1**	**1.4**

Table 1: Results on DeepCAD test set. The best results are **bold**, the second best are underlined. Our `cadrille` trained jointly on three modalities outperforms all existing modality-specific methods. Here, we report metrics obtained *without* RL fine-tuning or test-time sampling for fair comparison.

Method	RL	Train Data		DeepCAD			Fusion360			CC3D		
		SFT	RL	CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓
LRM→CAD-Recode	✗	R_p	✗	0.53	69.8	14.3	0.62	62.5	18.7	1.19	50.1	20.1
CADCrafter [5]	DPO	D_i	D_i	0.26		3.6						
`cadrille`	✗	R_{pi}	✗	0.18	86.1	1.5	0.20	77.6	3.2	0.81	56.1	7.7
`cadrille`	✗	R_{pi}+D_{pi}	✗	0.19	85.6	0.6	0.23	75.2	2.6	1.17	53.1	6.0
`cadrille`	DPO	R_{pi}	D_i^-+F_i^-	0.18	86.9	1.8	0.20	78.5	1.7	0.89	56.0	3.9
`cadrille`	Dr. CPPO	R_{pi}	D_i^-+F_i^-	**0.17**	**92.2**	**0.0**	**0.17**	**84.6**	**0.0**	**0.57**	**65.0**	**0.1**

Table 2: Results of CAD reconstruction from multi-view images. With RL fine-tuning, `cadrille` achieves best results across three benchmarks.

5.2 Reinforcement Learning

RL on single modality boosts other modalities In Tab. 2, we report accuracy of image-based CAD reconstruction on three benchmarks. Respectively, we fine-tune `cadrille` with images from DeepCAD and Fusion360 datasets. It is worth noticing that while Fusion360 cannot be used for direct supervised training, it can still contribute to RL fine-tuning where CAD sequences are not required, so we can benefit from adding it to the mixture. We denote DeepCAD and Fusion360 datasets without CAD sequence annotations as D^- and F^-.

Surprisingly, RL fine-tuning on images appears to be beneficial for other modalities: as reported in Tab. 3, the model tuned on D_i^-+F_i^- (row 6) delivers state-of-the-art quality of CAD reconstruction from point clouds as well.

Method	RL	Train Data		DeepCAD			Fusion360			Real-world CC3D		
		SFT	RL	CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓
CAD-SIGNet [17]	✗	D_p	✗	0.29	77.3	5.0	0.70	58.4	9.3	4.42	39.1	15.5
CAD-Recode [36]	✗	R_p	✗	0.18	87.1	3.1	0.19	79.1	5.0	0.54	60.5	9.8
`cadrille`	✗	R_{pi}	✗	0.18	87.1	2.1	0.19	79.8	2.8	0.54	61.8	5.9
`cadrille`	✗	R_{pi}+D_{pi}	✗	0.19	86.6	0.9	0.22	76.5	2.0	0.79	58.7	4.1
`cadrille`	DPO	R_{pi}	D_i^-+F_i^-	0.18	88.1	0.7	0.19	80.9	1.3	0.54	61.3	2.6
`cadrille`	Dr. CPPO	R_{pi}	D_i^-+F_i^-	**0.17**	**90.2**	**0.0**	**0.17**	**85.0**	**0.2**	**0.47**	**67.9**	**0.2**

Table 3: Results of CAD reconstruction from point clouds. `cadrille` performs on par with CAD-Recode [36] when trained on the CAD-Recode dataset (R). With RL, `cadrille` establishes state-of-the-art on DeepCAD, Fusion360 and real-world CC3D.

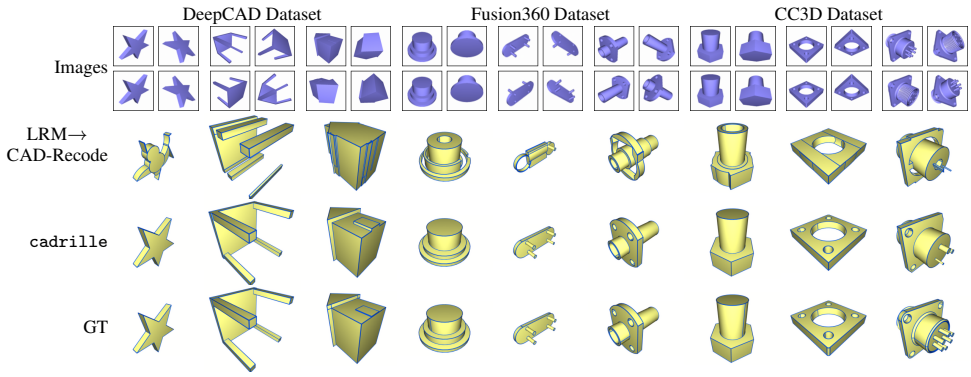

Figure 5: CAD models reconstructed from multi-view images on the DeepCAD, Fusion360, and CC3D datasets.

RL improves metrics in cross-dataset scenario RL fine-tuning with DeepCAD and Fusion360 boosts accuracy on the test splits of the respective datasets. Yet, the performance gain is not limited to the domains seen by a model during SFT and RL fine-tuning. In image-based CAD reconstruction, CD is reduced from 0.81 to 0.57, while IR dropped dramatically from 7.7 to 0.1 (rows 3 and 6, respectively). When testing on point clouds, RL also improves all scores on CC3D, making IR less than 0.2%, which is negligible.

Online RL outperforms offline RL Fine-tuning `cadrille` using offline DPO reduces IR twice in most cases, while accuracy scores are not affected (rows 3 and 5 in both Tables). In the meantime, Dr. CPPO beats SFT in terms of all metrics, adding 3-9% to IoU scores and bringing IR under 0.2% in all benchmarks (row 6). The observed improvement of CAD reconstruction accuracy aligns well with the experimental results obtained in other tasks where feedback can be programmatically computed [38].

RL fine-tuning beats SFT on a mixture A common assumption is that mixing datasets improves generalization by increasing data diversity and volume. However, our experiments show that SFT with a plain mixture of CAD-Recode and DeepCAD datasets (R_{pi}+D_{pi}, row 4) does not lead to performance gains, and can even degrade results w.r.t. SFT with R_{pi} (row 3). We attribute this effect to the domain gap between datasets, specifically, some CAD operations present in DeepCAD (e.g., symmetric extrusion, extruded cut) are lacking from CAD-Recode.

Qualitative results CAD models obtained with RL fine-tuning are depicted in Fig. 4 (from point clouds) and Fig. 5 (from multi-view images). Compared to predecessors, `cadrille` produces more geometrically plausible reconstructions and better restores fine details.

6 Limitations

Despite strong performance, our method has several limitations. It relies on either relatively small and limited hand-crafted benchmarks (DeepCAD), or large-scale procedurally generated datasets (CAD-Recode), which may not reflect the complexity of real-world CAD data. Due to the unavailability of CC3D's training split in the public domain, training or annotation-free fine-tuning on real data remains underexplored. Additionally, the text modality is underutilized, since natural language descriptions are scarce, and VLM-generated captions from Text2CAD often lack realism and might impose risks of data leakage due to being overly descriptive and detailed. Moreover, while `cadrille` can process all three modalities, it treats them independently and lacks mechanisms to compensate for low-quality or missing inputs. Besides, we only use images for RL fine-tuning. Tuning on point clouds appeared to be unstable, which we attribute to 1) point clouds being a non-native input modality for an LLM and 2) suboptimal point cloud processing strategies inherited from CAD-Recode.

7 Conclusion

We introduced `cadrille`, a multimodal CAD reconstruction model that is capable of processing point clouds, multi-view images, and text inputs within a unified VLM-based framework. By adopting a two-stage training paradigm, namely, supervised fine-tuning on synthetic data followed by reinforcement learning fine-tuning with programmatic feedback, we improved both reconstruction quality and validity ratio. Our empirical study demonstrated that online RL approaches are especially beneficial in the CAD reconstruction scenario. The proposed method achieves new state-of-the-art results in multiple benchmarks, including a real-world dataset, highlighting its robustness, generalizability, and potential for further use in applications.

A Qualitative Results

Real-world single-image experiment Since our `cadrille` is trained only on synthetic renders, sim-to-real transfer can be eligibly questioned. To address the potential concerns about the applicability of our approach, apart from validating on real scans from the CC3D dataset, we also experiment with CAD reconstruction from a single real-world image.

The pipeline consists of three steps. First, an input image is processed using the recent image-to-mesh InstantMesh [51] method, that produces a mesh. Second, we follow the same protocol as in other experiments to convert this mesh to a point cloud 6 or four multi-view images 7. We claim that the obtained results look promising, and this practical pipeline opens new opportunities of in-the-wild CAD reconstruction.

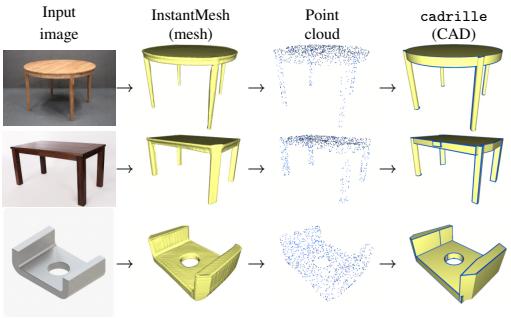

Figure 6: Results of CAD reconstruction from a *single* real-world image. `cadrille` takes point clouds sampled from mesh reconstructed by InstantMesh as input.

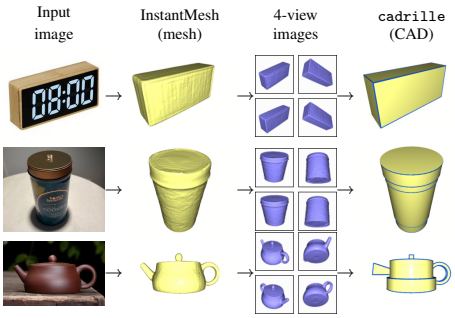

Figure 7: Results of CAD reconstruction from a *single* real-world image. `cadrille` takes multi-view images rendered from mesh reconstructed by InstantMesh as input.

Text-based CAD reconstruction Fig. 8 shows results of text-based CAD reconstruction on the DeepCAD dataset. Long textual descriptions still cannot define even simple 3D shapes comprehensively and unambiguously, making CAD reconstruction from textual inputs the most challenging. Both Text2CAD [18] and `cadrille` struggle to recover correct geometry, yet our approach yields more accurate predictions, which is also reflected in the quantitative metrics.

Failure cases are depicted in Fig. 9. `cadrille` always predicts a geometrically relevant shape, but still might miss details, especially for objects with complex and granular surfaces.

B Quantitative Results

Data for RL fine-tuning As described in Sec. 4.4 in the main paper, not all available data is used for RL fine-tuning, but only the most hard examples are considered. By varying the threshold, we can adjust the difficulty level and the ratio of data selected for fine-tuning. Finding the proper balance is crucial for RL fine-tuning, since it has a notably larger time- and memory footprint compared to SFT, and is hardly feasible without hard example mining.

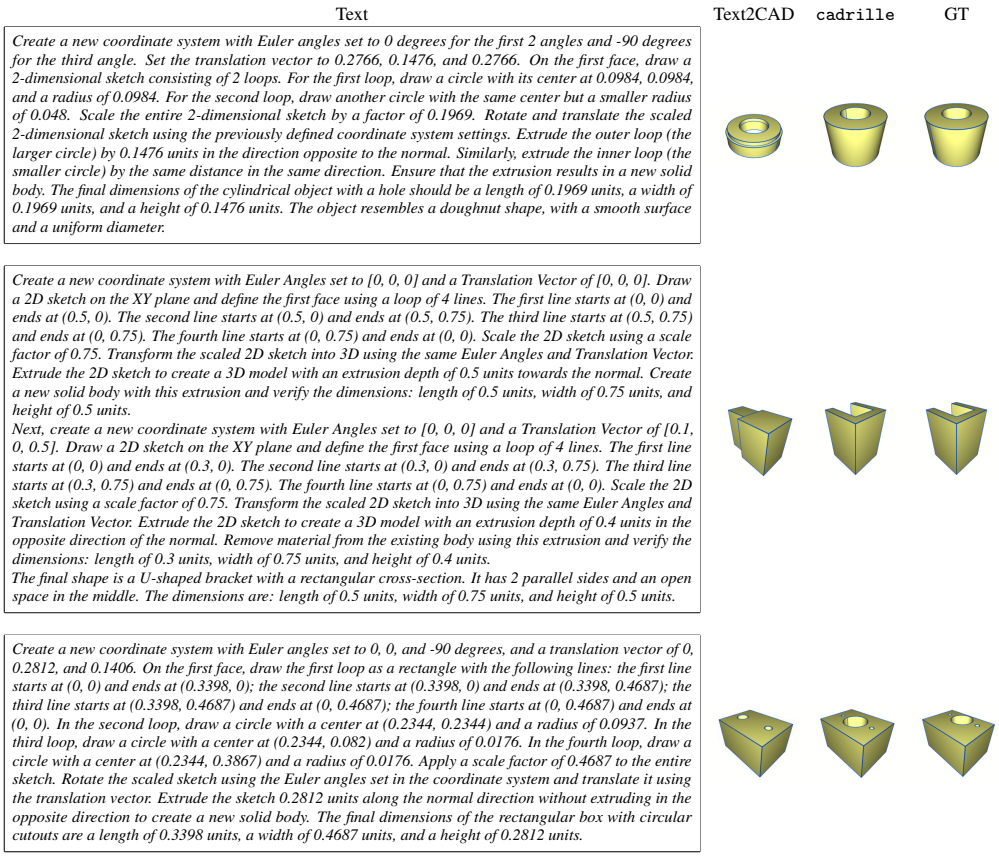

Figure 8: Results of text-based CAD reconstruction on the DeepCAD dataset.

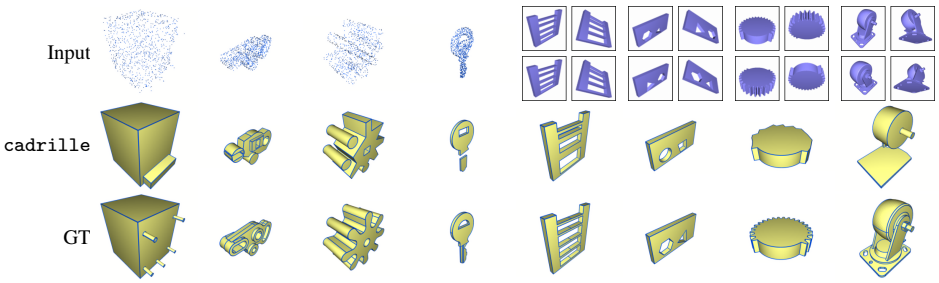

Figure 9: Failure cases of CAD reconstruction from point clouds and multi-view images on DeepCAD, Fusion360, and CC3D datasets.

RL Data		DeepCAD			Fusion360			CC3D		
DeepCAD	Fusion360	CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓	CD↓	IoU↑	IR↓
10k	0.7k	0.18	89.0	1.0	0.18	81.0	2.6	0.74	59.8	6.3
20k	1.4k	0.17	90.4	0.7	0.18	82.3	0.9	0.63	62.4	4.0
30k	2k	0.17	90.9	0.2	0.17	83.5	0.5	0.62	62.3	0.9
40k	2.5k	0.17	91.2	0.1	0.17	83.9	0.2	0.60	63.3	0.7
50k	3k	**0.17**	**92.2**	**0.0**	**0.17**	**84.6**	**0.0**	**0.57**	**65.0**	**0.1**

Table 4: Results of CAD reconstruction from multi-view images with varying amount of data used for RL fine-tuning.

In Tab. 4, we report results obtained when fine-tuned on data of different volume. All metrics improve gradually with an increase of the amount of the training data. Still, all results reported in this table supersede the SFT baseline (Tab. 2 of the main paper).

Mean CD Mean CD is sometimes reported alongside *median* CD. In the main paper, we only provided median CD as the most common CAD reconstruction metric. Nevertheless, the evaluation protocol might be considered incomplete without mean CD. In Tab. 5, we report *mean* CD metric for all scenarios on all three datasets. Our SFT model outperforms all competitors. The gain is especially tangible in text-based CAD reconstruction, where Text2CAD mean CD is reduced 6x from 26.4 to 3.95. With RL fine-tuning (Dr. CPPO) the new state-of-the-art is set in both image-based and point cloud-based CAD reconstruction on all three datasets. The most dramatic improvement is demonstrated on the Fusion360 dataset, where the relative increase exceeds 40% (Tab. 6).

Method	Train Data	RL	DeepCAD			Fusion360		CC3D	
			Points	Images	Text	Points	Images	Points	Images
PointNet→DeepCAD [50]	D_p	✗	42.5			330			
CAD-SIGNet [17]	D_p	✗	6.81			14.5		32.6	
CAD-Recode [36]	R_p	✗	0.83			1.21		3.21	
LRM→CAD-Recode	R_p	✗		3.36			4.33		4.75
BERT→DeepCAD [50]	D_t	✗			97.9				
Text2CAD [18]	D_t	✗			26.4				
`cadrille`	D_{pit}	✗	3.43	3.57	4.24	7.61	8.59	12.2	13.2
`cadrille`	R_{pi}+D_t	✗	**0.76**	**0.81**	**3.95**	**1.10**	**1.13**	**2.32**	**3.50**
`cadrille`	R_{pi}	DPO	0.61	1.35		0.84	1.27	2.32	3.33
`cadrille`	R_{pi}	Dr. CPPO	**0.57**	**0.43**		**0.58**	**0.64**	**1.86**	**2.68**

Table 5: *Mean* CD scores obtained across all benchmarks and available input modalities. RL fine-tuning is performed using D_i^-+F_i^- data.

More baselines on Fusion360 In Tab. 6, we compare `cadrille` to more point cloud-based CAD reconstruction baselines on the Fusion360 dataset. As can be observed, `cadrille` outperforms competitors trained either on DeepCAD or CAD-Recode datasets.

Inference time When target at practical use, efficiency is an issue. In Tab. 7, we compare inference time of our multi-modal `cadrille` against previous best single-modal methods: CAD-Recode [36], Text2CAD [18], and our baseline LRM→CAD-Recode.

Our `cadrille` is built on top of Qwen2-VL-2B, the smallest Qwen2 model with vision capabilities. When inferred on point clouds, it is 20% slower compared to CAD-Recode using Qwen2-1.5B. The image inference takes comparable time to proceed. Processing text prompts from Text2CAD dataset lasts notably longer, so that the inference time almost doubles and reaches 3.9 seconds. Text2CAD uses a smaller and faster BERT-large model, that allows achieving efficiency at cost of accuracy. Compared to Text2CAD, `cadrille` is delivers 6x better mean CD 5 being only 2x slower.

Method	Train	CD↓	IoU↑	IR↓
DeepCAD [50]	D_p	89.2	39.9	25.2
MultiCAD [28]	D_p	42.2		16.5
HNC-CAD [52]	D_p	36.8	63.5	7.3
TransCAD [8]	D_p	33.4	60.2	2.4
CAD-Diffuser [27]	D_p	3.85	63.2	1.7
CAD-SIGNet [17]	D_p	0.70	58.3	9.3
`cadrille`	D_p	**0.66**	**63.7**	**0.6**
CAD-Recode [36]	R_p	0.19	79.1	5.0
`cadrille`	R_p	**0.19**	**79.8**	**2.8**

Table 6: Results of point-based CAD reconstruction on the Fusion360 test set. All reported metrics are obtained using an SFT model without RL.

Method	LLM	Points	Images	Text
CAD-Recode [36]	Qwen2-1.5B	1.8		
LRM→CAD-Recode	Qwen2-1.5B		4.4	
Text2CAD [18]	BERT-336M			1.7
`cadrille`	Qwen2-VL-2B	2.0	2.0	3.9

Table 7: Inference time in seconds measured on the DeepCAD dataset. All methods are benchmarked on the single H100 GPU with a batch size of 1.

Single-image CAD reconstruction To compare against single-image CAD reconstruction methods, namely, CAD-GPT [46] and Img2CAD [6], we conduct an experiment with single-view images on the DeepCAD test set. Our SFT model improves over CADCrafter [5], reducing median CD from 0.72 to 0.21. As could be expected, the results of single-view CAD reconstruction are slightly inferior to the ones obtained from multi-view images (81% vs 86% IoU).

Method	CD↓	IoU↑	IR↓
GPT-4o [14, 46]	62.6		64.4
CAD-GPT [46]	9.77		1.6
DINOv2→HNC-CAD [52]	2.14		11.4
Img2CAD [6]	1.60		28.8
DINOv2→DeepCAD [50]	1.26		12.3
CADCrafter [5]	0.72		8.1
`cadrille`	**0.21**	**81.7**	**1.3**

Table 8: Results of CAD reconstruction from a *single* image on the DeepCAD dataset. All reported metrics are obtained with an SFT model without RL.

Method	CD↓	IoU↑	IR↓
GPT-o4-mini	2.37	60.4	15.9
`cadrille`	**0.17**	**92.2**	**0.0**

Table 9: Results of CAD reconstruction from multi-view images on the DeepCAD dataset.

Zero-shot CAD reconstruction Image-based CAD reconstruction can be performed in zero-shot way using existing VLMs. CAD-GPT [46] sets a weak baseline using GPT-4o, that has an invalidity ratio of 64% Tab. 8

We construct another baseline for CAD reconstruction from multi-view images. Four images rendered with orthogonal viewing directions are given to GPT-o4-mini to produce a Python code of CAD model. We apply iterative closest point (ICP) to align predictions before computing metrics so that correct predictions with wrong orientation are not penalized. As can be seen in Tab. 9, this strategy allows achieving significantly better results compared to CAD-GPT. Still, invalidity ratio is as high as 15% and IoU is 30% lower compared to our `cadrille`.

C Implementation Details

Architecture Our model is built on top of Qwen2-VL [45] and uses its native capabilities of image and text understanding. In all experiments on multi-view image CAD reconstruction, we use four images. Images are rendered with fixed camera positions, and concatenated into 2x2 grid, forming a combined image of size 268 $\times$ 268 px (as shown in Fig. 7 and 9). This combined image is passed through the Qwen vision encoder, that outputs 400 input tokens. The point cloud injecting into LLM is implemented exactly as in CAD-Recode [36]. Specifically, our input consists of 256 unordered 3D points without normals, sampled from the surface using the furthest point sampling method. The points are projected into shared embedding space with a single linear layer.

Training SFT in `cadrille` mostly follows the training procedure of CAD-Recode. The only difference is using batch size of 8 and four gradient accumulation steps due to increase of memory for longer prompts in the multimodal scenario. The SFT model is trained with AdamW optimizer for 120k steps and learning rate of 2e-4 on a single H100 GPU.

RL fine-tuning The RL fine-tuning hyperparameters are listed in Tab. 10 (DPO) and 11 (Dr. CPPO). In each experiment, the model is initialized with the weights of an SFT model trained on point clouds and images, and then fine-tuned only on images. All fine-tuning experiments are performed on 8 H100 GPUs.

Hyperparameter	Value
Optimizer	Adam
Number of epochs	20
Batch size	160
Learning rate	1e-05
KL regularization coef (β)	0.3

Table 10: DPO tuning hyperparameters.

Hyperparameter	Value
Optimizer	Adam
Number of epochs	20
Batch size	128
Learning rate	3e-05
Updates per batch	3
PPO ϵ	0.1
GRPO group size (G)	16
CPPO number of samples (N)	4

Table 11: RL fine-tuning hyperparameters.

D Miscellaneous

Python codes In Fig. 10, we provide Python code with CadQuery library, produced by `cadrille`. When executed, these Python scripts generate CAD models. Evidently, `cadrille` is not limited to basic geometric primitives from the DeepCAD dataset (such as line, arc, circle), but is also capable of producing more advanced shapes from the CAD-Recode dataset (box, rectangle, cylinder).

```
import cadquery as cq
w0=cq.Workplane('XY',origin=(0,0,11))
r=w0.sketch().segment((-100,23),(-43,-23)).segment((-63,-95)).segment((0,-55))
    .segment((62,-96)).segment((42,-23)).segment((100,23)).segment((27,26))
    .segment((0,96)).segment((-26,26)).close().assemble().finalize().extrude(-23)
```

```
import cadquery as cq
w0=cq.Workplane('XY',origin=(0,0,88))
r=w0.sketch().push([(-87,-89)]).rect(20,22).push([(-87,89)]).rect(20,22)
    .push([(88,-89)]).rect(20,22).push([(88,90)]).rect(20,20).finalize().extrude(-176)
  .union(w0.workplane(offset=-15/2).box(200,200,15))
```

```
import cadquery as cq
w0=cq.Workplane('YZ',origin=(-63,0,0))
r=w0.sketch().circle(33).circle(27,mode='s').finalize().extrude(-37)
  .union(w0.sketch().segment((-55,-67),(55,-67)).segment((55,-66)).arc((87,0),
    (55,66)).segment((-9,67)).segment((-55,67)).arc((-87,1),(-55,-67)).assemble()
    .push([(-53,-39)]).circle(12,mode='s').push([(-53,39)]).circle(11,mode='s')
    .push([(53,-43)]).circle(12,mode='s').finalize().extrude(21))
  .union(w0.sketch().circle(34).circle(27,mode='s').finalize().extrude(163))
```

```
import cadquery as cq
w0=cq.Workplane('XY',origin=(0,0,56))
w1=cq.Workplane('XY',origin=(0,0,26))
r=w0.workplane(offset=-149/2).cylinder(149,18.5)
  .union(w0.sketch().circle(100).circle(78,mode='s').finalize().extrude(-111))
  .union(w1.workplane(offset=-52/2).cylinder(52,77))
  .union(w1.workplane(offset=6/2).moveTo(97,0).box(3.5,9.5,6))
```

Figure 10: `cadrille` predictions on DeepCAD, Fusion360, and CC3D datasets. Each row contains predicted CadQuery Python code and its result after execution in Python interpreter.

CC3D scans Following CAD-Recode, we treat experiments with CC3D dataset as an emulation of a real-world experiment. CC3D contains scans of the physical objects paired with ground truth CAD models. As shown in Fig. 11, CC3D scans contain artifacts such as surface noise, smoothed edges, and missing parts. In our experiments, we sample points from the surfaces of these real noisy scans instead of the perfect CAD models, which essentially makes the experimental scenario much more realistic.

Besides, CC3D contains generally more complex CAD models, which are constructed using operations beyond simple extrusion, including revolution, chamfer, and fillet.

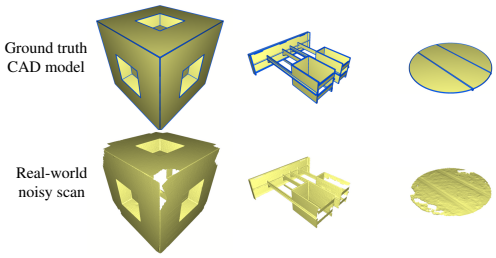

Figure 11: Ground truth CAD models (top row) and noisy scans (bottom row) from the real-world CC3D dataset.

References

[1] Silvia Ansaldi, Leila De Floriani, and Bianca Falcidieno. Geometric modeling of solid objects by using a face adjacency graph representation. *ACM SIGGRAPH Computer Graphics*, 19(3):131–139, 1985.

[2] CadQuery Authors. Cadquery/cadquery: Cadquery 2.4.0, January 2024.

[3] Akshay Badagabettu, Sai Sravan Yarlagadda, and Amir Barati Farimani. Query2cad: Generating cad models using natural language queries. *arXiv preprint arXiv:2406.00144*, 2024.

[4] Antoine Briere-Cote, Louis Rivest, and Roland Maranzana. Comparing 3d cad models: uses, methods, tools and perspectives. *Computer-Aided Design and Applications*, 9(6):771–794, 2012.

[5] Cheng Chen, Jiacheng Wei, Tianrun Chen, Chi Zhang, Xiaofeng Yang, Shangzhan Zhang, Bingchen Yang, Chuan-Sheng Foo, Guosheng Lin, Qixing Huang, et al. Cadcrafter: Generating computer-aided design models from unconstrained images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[6] Tianrun Chen, Chunan Yu, Yuanqi Hu, Jing Li, Tao Xu, Runlong Cao, Lanyun Zhu, Ying Zang, Yong Zhang, Zejian Li, et al. Img2cad: Conditioned 3d cad model generation from single image with structured visual geometry. *arXiv preprint arXiv:2410.03417*, 2024.

[7] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[8] 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*, pages 19–36. Springer, 2024.

[9] Kevin Ellis, Maxwell Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. Write, execute, assess: Program synthesis with a repl. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] James D Foley. *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional, 1996.

[11] Markus Friedrich, Pierre-Alain Fayolle, Thomas Gabor, and Claudia Linnhoff-Popien. Optimizing evolutionary csg tree extraction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1183–1191, 2019.

[12] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

[13] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In *International Conference on Learning Representations*, 2024.

[14] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[15] Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *Transactions on Machine Learning Research*, 2023.

[16] Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. Ucsg-net-unsupervised discovering of constructive solid geometry tree. *Advances in neural information processing systems*, 33:8776–8786, 2020.

[17] 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*, pages 4713–4722, 2024.

[18] Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. *Advances in Neural Information Processing Systems*, 37:7552–7579, 2024.

[19] 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*, pages 12773–12782, 2021.

[20] Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. Reconstructing editable prismatic cad from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[21] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2652–2660, 2019.

[22] 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 International Conference on Computer Vision*, 2025.

[23] Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. *arXiv preprint arXiv:2503.22342*, 2025.

[24] Yilin Liu, Jiale Chen, Shanshan Pan, Daniel Cohen-Or, Hao Zhang, and Hui Huang. Split-and-fit: Learning b-reps via structure-aware voronoi partitioning. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024.

[25] Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. Point2cad: Reverse engineering cad models from 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3763–3772, 2024.

[26] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[27] Weijian Ma, Shuaiqi Chen, Yunzhong Lou, Xueyang Li, and Xiangdong Zhou. Draw step by step: Reconstructing cad construction sequences from point clouds via multimodal diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27154–27163, 2024.

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

[29] Dimitrios Mallis, Ali Sk Aziz, Elona Dupont, Kseniya Cherenkova, Ahmet Serdar Karadeniz, Mohammad Sadil Khan, Anis Kacem, Gleb Gusev, and Djamila Aouada. Sharp challenge 2023: Solving cad history and parameters recovery from point clouds and 3d scans. overview, datasets, metrics, and baselines. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1786–1795, 2023.

[30] Dimitrios Mallis, Ahmet Serdar Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-assistant: Tool-augmented vllms as generic cad task solvers? *arXiv preprint arXiv:2412.13810*, 2024.

[31] Chandrakana Nandi, James R Wilcox, Pavel Panchekha, Taylor Blau, Dan Grossman, and Zachary Tatlock. Functional programming for compiling and decompiling computer-aided design. *Proceedings of the ACM on Programming Languages*, 2(ICFP):1–31, 2018.

[32] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[33] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[34] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, et al. Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12478–12487, 2021.

[35] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. In *European Conference on Computer Vision*, pages 482–498. Springer, 2022.

[36] Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds. *arXiv preprint arXiv:2412.14042*, 2024.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[39] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5523, 2018.

[40] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 261–276. Springer, 2020.

[41] Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. Learning manifold patch-based representations of man-made shapes. In *International Conference on Learning Representations*, 2021.

[42] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Learning to infer and execute 3d shape programs. In *International Conference on Learning Representations*, 2019.

[43] 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*, pages 11850–11860, 2022.

[44] Kehan Wang, Jia Zheng, and Zihan Zhou. Neural face identification in a 2d wireframe projection of a manifold object. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1622–1631, 2022.

[45] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[46] 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, pages 7880–7888, 2025.

[47] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. *Advances in neural information processing systems*, 33:20167–20178, 2020.

[48] Xilin Wang, Jia Zheng, Yuanchao Hu, Hao Zhu, Qian Yu, and Zihan Zhou. From 2d cad drawings to 3d parametric models: A vision-language approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7961–7969, 2025.

[49] 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.

[50] 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*, pages 6772–6782, 2021.

[51] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[52] Xiang Xu, Pradeep Kumar Jayaraman, Joseph G Lambourne, Karl DD Willis, and Yasutaka Furukawa. Hierarchical neural coding for controllable cad model generation. In *International Conference on Machine Learning*, pages 38443–38461, 2023.

[53] Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024.

[54] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In *International Conference on Machine Learning*, pages 24698–24724. PMLR, 2022.

[55] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. corr, abs/2407.10671, 2024. doi: 10.48550. *arXiv preprint ARXIV.2407.10671*, 2024.

[56] Xiaolong Yin, Xingyu Lu, Jiahang Shen, Jingzhe Ni, Hailong Li, Ruofeng Tong, Min Tang, and Peng Du. Rlcad: Reinforcement learning training gym for revolution involved cad command sequence generation. *arXiv preprint arXiv:2503.18549*, 2025.

[57] Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. D2csg: Unsupervised learning of compact csg trees with dual complements and dropouts. *Advances in Neural Information Processing Systems*, 36:22807–22819, 2023.

[58] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. Capri-net: Learning compact cad shapes with adaptive primitive assembly. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11768–11778, 2022.

[59] Yu Yuan, Shizhao Sun, Qi Liu, and Jiang Bian. Cad-editor: A locate-then-infill framework with automated training data synthesis for text-based cad editing. *arXiv preprint arXiv:2502.03997*, 2025.

[60] Chao Zhang, Arnaud Polette, Romain PINQUIÉ, Mirai Iida, Henri De Charnace, and Jean-Philippe Pernot. Reinforcement learning-based parametric cad models reconstruction from 2d orthographic drawings. *Available at SSRN 5174280*, 2025.

[61] Zhanwei Zhang, Shizhao Sun, Wenxiao Wang, Deng Cai, and Jiang Bian. Flexcad: Unified and versatile controllable cad generation with fine-tuned large language models. *International Conference on Learning Representations*, 2025.