CADRec: Reconstructing a CAD Sequence Recursively with Localized Geometric Contexts

HAOXUAN SONG, University of Chinese Academy of Sciences, China
BINGCHEN YANG, Nanyang Technological University, Singapore
JUN XIAO, University of Chinese Academy of Sciences, China
HAIYONG JIANG, University of Chinese Academy of Sciences, China

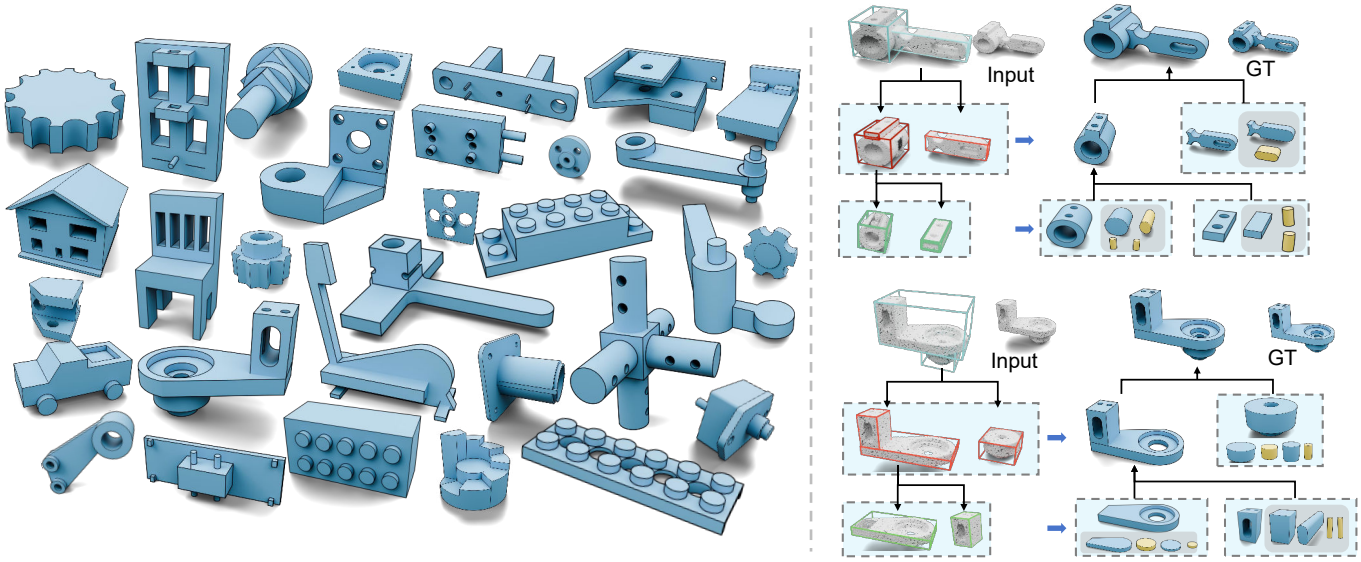

Fig. 1. Reconstruction results for the proposed CADRec. Left: Diverse CAD reconstruction results. Right: The recursive reconstruction process for CAD shapes. We show the recursive bounding box prediction along with the part hierarchy constructed using reconstructed CAD scripts.

We present CADRec, a recursive CAD sequence reconstruction framework based on a hierarchically defined CAD DSL. This method addresses two limitations of previous methods. First, the flat, lengthy script representation of LLM-based reconstruction methods hinders generalization and does not take into account the reuse of code snippets. Second, existing one-shot and iterative geometric context encodings lack the structural clues necessary to accurately localize the geometric context for subsequent sequence reconstruction. CADRec recursively decomposes the input shape into spatially localized parts and reconstructs each part with a local CADQuery program based on a hierarchical CAD DSL. At each node of the hierarchy, the model decides whether to further split the current part into child parts or stop and synthesize its leaf-level CAD script. The generated local programs are then mapped back to the global coordinate frame and composed into the final CAD model. This hierarchical formulation turns global CAD reconstruction into a sequence of simpler local part prediction problems, reducing geometric complexity while preserving the structural organization of the design. To support this process, we encode localized point-cloud context and use an MLLM-based decoder to predict decomposition decisions and CAD programs. We further train the model with geometry-aware objectives that encourage consistent part decomposition and spatially grounded predictions. Experiments on standard CAD reconstruction benchmarks demonstrate that CADRec improves both geometric fidelity and program validity over prior sequence- and code-based baselines. Notably, the robustness of our approach is particularly evident in cross-dataset scenarios: when trained on the limited DeepCAD dataset and tested on the unseen Fusion360 set, our method drastically reduces mean CD, median CD, and IR by 85.8%, 67.2%, and 92.3%, respectively. The code and data for this paper are available on: https://github.com/shinodashx/CADRec.

CCS Concepts: • **Computing methodologies → Parametric curve and surface models**.

Additional Key Words and Phrases: CAD modeling, Constructive Solid Geometry, MLLM, CAD sequences

This work is partially supported by the National Natural Science Foundation of China (62271467, 62476262, 62206263, 62306297, 62306296), the Beijing Nova Program, and the Beijing Natural Science Foundation (4242053, L242096). Haiyong Jiang is the corresponding author.
Authors' Contact Information: Haoxuan Song, University of Chinese Academy of Sciences, Beijing, China, songhaoxuan24@mails.ucas.ac.cn; Bingchen Yang, Nanyang Technological University, Singapore; Jun Xiao, University of Chinese Academy of Sciences, Beijing, China; Haiyong Jiang, University of Chinese Academy of Sciences, Beijing, China, haiyong.jiang@ucas.ac.cn.

ACM 1557-7368/2026/12-ART212
https://doi.org/10.1145/3842561

ACM Reference Format:
Haoxuan Song, Bingchen Yang, Jun Xiao, and Haiyong Jiang. 2026. CADRec: Reconstructing a CAD Sequence Recursively with Localized Geometric Contexts. *ACM Trans. Graph.* 45, 6, Article 212 (December 2026), 16 pages. https://doi.org/10.1145/3842561

1 Introduction

While a point cloud accurately captures an object's surface geometry, it lacks the underlying logic of how the object was constructed, parameterized, or intended to be edited. This distinction is critical in engineering and manufacturing workflows, where downstream design reuse necessitates a Computer-Aided Design (CAD) model with explicit construction operations and user-controllable parameters. Consequently, recovering editable CAD sequences from scans remains a fundamental challenge for geometric modeling and reverse engineering.

Previous approaches primarily reconstruct CAD models as sketch-and-extrude primitives [Li et al. 2023a; Ren et al. 2022; Sharma et al. 2018] or as linear sequences of discrete, learned tokens [Wang et al. 2026; Wu et al. 2021; Xu et al. 2023, 2022].

Recently, the remarkable success of Large Language Models (LLMs) in code synthesis has catalyzed a paradigm shift toward script-based CAD representations [Dai et al. 2026; Dupont et al. 2024; Guan et al. 2026; Kolodiazhnyi et al. 2025; Li et al. 2026; Ma et al. 2023; Rukhovich et al. 2025; Wang et al. 2025a; Xu et al. 2024b; You et al. 2025]. By utilizing Domain-Specific Languages (DSLs) such as CADQuery or Blender scripts [Blender Online Community 2024; Contributors 2023], these methods bridge the gap between geometric perception and programmatic generation. However, despite achieving state-of-the-art results, these code-script-based approaches are inherently constrained by two fundamental limitations: the structural opacity of the flat, sequential solid organization and a one-shot point-cloud feature encoding without considering the recursive modeling process.

In this work, we advance code script-based modeling by introducing a novel hierarchical part decomposition of CAD shapes, as illustrated in Fig. 2. Departing from the conventional solid sequence generation, we formalize the modeling process as a top-down recursive decomposition, where the partition at each level is governed by a dedicated block of localized scripts. This representation is motivated by the duality between the recursive bottom-up modeling process of human designers and the top-down structural analysis of CAD shapes. The proposed hierarchical DSL offers two distinct advantages. First, it facilitates a divide-and-conquer reconstruction, decomposing complex global operations into a suite of manageable code snippets that enhance the modular reuse of intermediate parts. Second, this structured organization enables localized context encoding, effectively isolating sub-part features within the global geometry.

Parallel to the sequence representation, a critical line of investigation focuses on how geometric contexts are encoded to guide the reconstruction process. Existing CAD reconstruction methods either extract holistic contexts all at once [Karadeniz et al. 2026; Liu et al. 2026a], or iteratively update contexts based on previously predicted boxes, planes, or reconstruction states [Ding et al. 2026; Kabisov et al. 2026; Khan et al. 2024; Yang et al. 2025]. While these

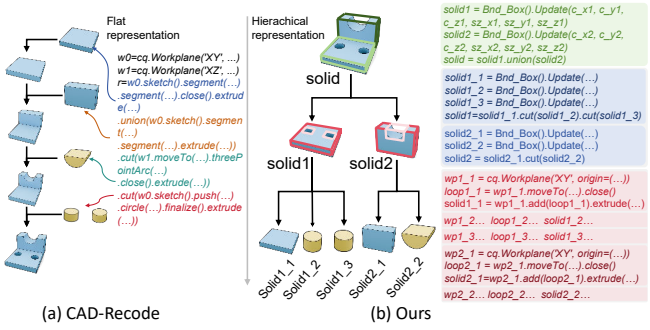

Fig. 2. Illustration of a flat sequence-based representation and our hierarchical part decomposition-based CAD DSL. (**a**) CAD-Recode-style CADQuery scripts [Rukhovich et al. 2025] represent the modeling process as a list of solids along with the corresponding Boolean operation, resulting in a non-structured, lengthy sequence of cascaded API calls. (**b**) Our representation organizes the same script as a hierarchical decomposition of solid parts. The root part $\mathcal{P}_1^1$ is recursively decomposed into child parts $\{\mathcal{P}_{i,k}^{d+1}\}_{k=1}^{K_i}$, where each child is associated with a Boolean command $O_{i,k}^{d+1}$ and a bounding box $B_{i,k}^{d+1}$. The bounding boxes localize part geometry and define the spatial regions for further decomposition until leaf parts are represented by elementary CADQuery code blocks C_i^d.

strategies improve geometry awareness, the precise localization of a specific extrusion cylinder remains an elusive and ill-posed problem without modeling the structural relations between different modeling steps. This work advocates a third recursive geometric encoding strategy grounded in our part hierarchy. By exploiting the property that each sub-part partitions its parent, the context most relevant to a sub-part is bounded by its spatial extent and its parent derivation. This formulation transforms the global reconstruction task into a sequence of localized reconstruction processes. In this framework, each component is treated as an independent CAD reconstruction problem, where the hierarchical reuse of sub-parts significantly enhances the model's generalization across diverse and complex geometries.

Based on the aforementioned dual motivations, we propose CADRec, a hierarchical point-to-CAD reconstruction framework designed to address the lack of a hierarchical organization and inadequate geometric context encoding. CADRec leverages an MLLM-based decoder to recursively decompose the input point cloud as parts. For each decomposed part, the method processes the localized point cloud within the current part's bounding box to determine the next action: either splitting the part into a list of bounding boxes corresponding to its child parts or deriving the localized CAD script for a single-step solid. This process is executed recursively and yields a reconstruction tree with single-step solids as leaf nodes and split parts as non-leaf nodes. Upon completion, the locally synthesized scripts are mapped back to the global coordinate frame and assembled via a bottom-up integration process to produce the final, editable CAD model. Reconstruction results are shown in Fig. 1.

Furthermore, we introduce two specialized objective functions to ensure reconstruction fidelity and structural accuracy. First, a geometric contrastive loss is employed to robustify the encoding

process against shape variations and noise encountered during the recursive cropping of localized sub-parts. Second, we formulate a spatially-grounded localization loss, which utilizes bounding-box and mask segmentation supervision to refine the precision of part-level partition and alignment. Together, these mechanisms enable CADRec to maintain high geometric consistency while navigating the complexities of hierarchical part-based modeling.

Extensive experiments demonstrate that our method outperforms state-of-the-art approaches across all metrics, including CD, IoU, and IR, on both the DeepCAD and Fusion 360 datasets. Moreover, our method achieves drastic error reductions over competing approaches (85.8% in mean CD, 67.2% in median CD, and 92.3% in IR), when trained on the small-scale DeepCAD training set and tested on the unseen Fusion360 dataset.

2 Related Works

This section reviews existing literature on 3D CAD reconstruction, focusing specifically on reconstructing command sequences and on geometry context encoding. Furthermore, we discuss related works in fine-grained visual perception leveraging MLLMs.

2.1 Command Sequence-based CAD Reconstruction

CAD command sequences serve as the de facto standard in industrial design, capturing step-by-step solid modeling and Boolean operations.

A core paradigm in this field is sketch-and-extrude reconstruction [Li et al. 2023a; Ma et al. 2025; Uy et al. 2022], which defines complex solids through a series of sketch planes and extrusion parameters. Based on this paradigm, Point2Cyl [Uy et al. 2022] formulates reverse engineering as extruded surface detection and extrusion parameter estimation. However, the discrete nature of CAD sequences presents a significant hurdle, making their reconstruction highly challenging. To mitigate these difficulties, some recent approaches [Li et al. 2023a; Ren et al. 2022; Yu et al. 2022] attempt to simplify the learning process by predicting a fixed number of extrusions guided by unsupervised reconstruction losses. While this avoids the need for complex sequential labels, the lack of step-wise modeling supervision often results in a fundamental disconnect from design logic. Consequently, these methods typically produce an unorganized collection of extrusions that accurately approximate the geometry but lack semantically meaningful structures or valid design intent. Meanwhile, another cateogry of methods [Liu et al. 2024a,b] recover boundary representation shapes from point clouds without the step-by-step modeling procedures.

Another prominent line of work directly predicts a sequence of CAD modeling operations. The seminal work, DeepCAD [Wu et al. 2021], employs a Transformer-based autoencoder to generate sequences of sketch-and-extrude commands. However, its architecture is constrained by a fixed-length output, which severely limits its capacity to represent structurally complex command sequences found in real-world designs. To overcome this limitation, subsequent research has moved toward autoregressive generation coupled with specialized tokenization schemes [Wang et al. 2026; Xu et al. 2023, 2022]. While these methods effectively handle variable-length sequences, they introduce new challenges. Specifically, they typically require two decoupled training stages for tokenization learning and token prediction, respectively. More critically, the discrete tokenization of continuous CAD geometry inherently leads to quantization errors, potentially sacrificing fine-grained parameter precision and compromising the fidelity of local geometric details.

Recent advances in LLMs [Anthropic 2025; Chen et al. 2021] have catalyzed a transition towards code-based CAD sequence reconstruction, leveraging script languages such as CADQuery [Contributors 2023] and Blender [Blender Online Community 2024]. Representative work, including CAD-Recode [Rukhovich et al. 2025] and Mesh-Coder [Dai et al. 2026], has achieved promising results by tuning point cloud encoders and LLMs on specialized CAD datasets. This code-centric scheme has proven highly versatile, extending beyond geometry-to-code tasks to multi-modal reconstruction from textual and visual inputs [Li et al. 2026; Wang et al. 2025a; Xu et al. 2024b; You et al. 2025]. To further refine reconstruction quality, subsequent research has explored diverse strategies, including contrastive learning [Ma et al. 2023], sketch-loop-extrusion structures [Dupont et al. 2024], and advanced training paradigms such as masked token modeling [Ma et al. 2024] or reinforcement learning [Guan et al. 2026; Kolodiazhnyi et al. 2025]. Despite these enhancements, these methods often overlook explicit structural contexts during reconstruction, leading to suboptimal encoding of local geometric details. To address this, this work leverages structural bounding boxes and local contexts as informative prompts, ensuring a more geometrically conforming and detail-accurate reconstruction.

2.2 Geometry Context Encoding for 3D Reconstruction

To enhance the reconstructed geometry quality, recent research explores diverse context encoding strategies for raw scan inputs. One strategy is to extract the holistic contexts all at once prior to CAD sequence reconstruction. Such holistic contexts can be constructed via step-aware latent features [Liu et al. 2026a], triplane-based feature encoding [Liu et al. 2026a], and cross-sectional sketch context encoding [Karadeniz et al. 2026]. However, these static holistic contexts fail to account for the evolving information generated during sequence prediction. Some other works alleviate this problem via progressive context encoding that is dynamically updated based on previously predicted CAD sequences. For instance, CAD-SIGNet [Khan et al. 2024] and PS-CAD [Yang et al. 2025] alternate between sketch and extrusion predictions, conditioned on localized point clouds cropped via previously inferred bounding boxes [Khan et al. 2024] or detected planes [Yang et al. 2025]. Similarly, CADReasoner [Kabisov et al. 2026] refines predictions based on geometric discrepancies between the input and execution of reconstructed programs, while ReACT [Ding et al. 2026] formulates the reconstruction as a Markov decision process, utilizing the preceding reconstruction steps as geometry-aware feedback.

2.3 Visual Context Localization in MLLMs

To enable fine-grained visual perception, recent works in MLLMs progressively encode local visual features [Guo et al. 2024; Shi et al. 2025; Wang et al. 2025b] rather than relying on a one-time global feature encoding [Dai et al. 2023; Li et al. 2023b; Liu et al. 2023; Zhu et al. 2023].

These methods usually learn to localize fine-grained visual evidence during training by image slicing [Guo et al. 2024], recursive partitioning [Wang et al. 2025b], and prompt-guided region cropping [Khayatkhoei et al. 2025; Shi et al. 2025; Zhong et al. 2026]. Some works further learn region-aware visual grounding during training through explicit region prompting with RoI features [Zhang et al. 2025], coordinate tokens [You et al. 2024], or visual marks [Yang et al. 2023]. Recent approaches take one step further by investigating inference-time visual search or zooming [Shen et al. 2025; Shi et al. 2026a,b; Wu and Xie 2024; Yu et al. 2025], where high-resolution local patches, selected crops, region features, and re-encoded visual tokens are fed back to MLLMs as fine-grained context.

This scheme also extends to 3D MLLMs-based tasks [Dai et al. 2026; Fang et al. 2025; Qi et al. 2024; Tang et al. 2024], where localized contexts, such as part-level masks, patch features, bounding boxes, or semantic part programs [Ahmed et al. 2025; Hadgi et al. 2026; Wang et al. 2025c], are explored. Our method transfers this principle from multimodal perception to CAD sequence reconstruction. We progressively predict a list of bounding boxes, with each bounding box denoting a potential solid shape. These bounding boxes not only facilitate more localized geometry context encoding for solid-wise CAD sequence reconstruction but also act as a structural decomposition of complex CAD shapes. In this way, CAD shapes with much longer sequences can be factorized and properly reconstructed.

3 Methodology

Our goal is to reconstruct CAD command sequences from a point cloud input. The input $X = \{x_i \in \mathbb{R}^3\}_{i=1}^{N}$ is a list of N 3D points. The output is formatted as a CADQuery script that can be executed to reconstruct the input geometry. To tackle this problem, we propose a recursive reconstruction pipeline that explicitly leverages the hierarchical geometric structure of the input shape, as illustrated in Fig. 3. Our framework consists of three key designs, including a hierarchical CAD DSL, localized context encoding, and the MLLM-based CAD DSL prediction. We first detail the proposed hierarchical CAD DSL that formats the CAD command sequence as a recursively decomposed part hierarchy (see Sec. 3.1). Then, Sec. 3.2 introduces the network architecture with two components: a geometry encoder that captures localized part contexts and an MLLM-based decoder that jointly supports recursive geometric decomposition and CAD DSL prediction. Finally, we describe the training objective (see Sec. 3.3) and data construction pipeline (see Sec. 3.4).

3.1 Hierarchically structured CAD DSL

This section describes how to represent a CAD shape as a structured hierarchy along with a corresponding CADQuery script.

Motivation analysis. Existing CAD scripts are typically organized as a sequence of modeling operations to construct a final shape [Rukhovich et al. 2025]. As illustrated in Fig. 2(a), even a simple five-step construction can result in an excessively long, single-line command due to cascaded API calls. This linear structure suffers from poor readability and fails to capture the spatial relationships between individual solids. In practice, designers reason about geometry through two complementary perspectives. During design, complex shapes are built bottom-up by recursively grouping solids through Boolean operations into a hierarchy of shape parts to facilitate modeling reuse. Conversely, when analyzing shapes, these complex geometries are decomposed top-down into their constituent part hierarchies. Motivated by this observation, we propose a hierarchical CAD DSL that recursively decomposes the input shape into intermediate parts in a top-down manner until reaching the elementary parts (i.e., an extrusion cylinder in our case) as shown in Fig. 2(b). Each elementary part is represented by a CADQuery code, and the overall shape is reconstructed in a bottom-up manner by progressively merging part codes.

Hierarchical CAD DSL. The part hierarchy starts from the root, denoted as $\mathcal{P}_1^1$, representing the entire CAD shape $\mathcal{S}$ with a leaf elementary part denoting a single-step solid $E_* \in \mathcal{E}$, where $\mathcal{E}$ is the complete set of elementary parts. A non-leaf part $\mathcal{P}_i^d$ at depth d encapsulates a subset of elementary parts, denoted by the index set $\mathcal{I}(\mathcal{P}_i^d) \subseteq \mathcal{E}$. Part $\mathcal{P}_i^d$ can be decomposed into a sequence of child parts $\{\mathcal{P}_{i,k}^{d+1}\}_{k=1}^{K_i}$, such that the child parts are mutually exclusive with respect to their elementary part constituents:

$$\mathcal{I}(\mathcal{P}_{i,k}^{d+1}) \bigcap \mathcal{I}(\mathcal{P}_{i,k'}^{d+1}) = \varnothing, \text{ if } k \neq k'; \quad \bigcup_{k=1}^{K_i} \mathcal{I}(\mathcal{P}_{i,k}^{d+1}) = \mathcal{I}(\mathcal{P}_i^d), \tag{1}$$

where K_i denotes the number of child parts for $\mathcal{P}_i^d$.

Based on the hierarchical representation, we can define a hierarchical CAD DSL as:

$$\mathcal{S} \rightarrow B_1^1 \mathcal{P}_1^1; \tag{2}$$

$$\mathcal{P}_i^d \rightarrow \texttt{SPLIT}, \mathcal{P}_{i,\text{split}}^d \mid \texttt{STOP}, \mathcal{P}_{i,\text{stop}}^d; \tag{3}$$

$$\mathcal{P}_{i,\text{split}}^d \rightarrow \{O_{i,1}^{d+1} B_{i,1}^{d+1} \mathcal{P}_{i,1}^{d+1}, \cdots, O_{i,K_i}^{d+1} B_{i,K_i}^{d+1} \mathcal{P}_{i,K_i}^{d+1}\}; \tag{4}$$

$$\mathcal{P}_{i,\text{stop}}^d \rightarrow \{C_i^d\}; \tag{5}$$

where $\{\texttt{STOP}, \texttt{SPLIT}\}$ denotes two deriving operations with `STOP` deriving a part $\mathcal{P}_i^d$ as a CADQuery code C_i^d and `SPLIT` decomposing a part $\mathcal{P}_i^d$ into child parts $\{\mathcal{P}_{i,k}^{d+1}\}_{k=1}^{K_i}$. Each child part $\mathcal{P}_{i,k}^{d+1}$ is associated with a Boolean command $O_{i,k}^{d+1}$ and an axis-aligned bounding box $B_{i,k}^{d+1}$. The Boolean command $O_{i,k}^{d+1}$ can be `Union` or `Difference`. The axis-aligned bounding box provides spatial localization and the relative transformation between the coordinate frame of a child part and its parent. To ensure a consistent output format across both `SPLIT` and `STOP` operations, the bounding box $B_{i,k}^{d+1}$ and Boolean $O_{i,k}^{d+1}$ are implemented with CADQuery APIs. Fig. 2 (b) illustrates an example shape with a three-level CAD DSL hierarchy, where different colored code blocks correspond to different part levels. Note that these code blocks are not directly executable and need further post-processing to form the final CAD shape (see Sec. 3.2). The construction of the part hierarchy for general CAD scripts is elaborated in Sec. 3.4.

3.2 Localized Context-Guided CAD DSL Reconstruction

This section describes an MLLM-based network for reconstructing hierarchical CAD DSLs. The input is the point cloud X and the output is the CAD DSL sequences defined in Eq. 2-5. The DSL sequence is reconstructed recursively from the top level to the leaf

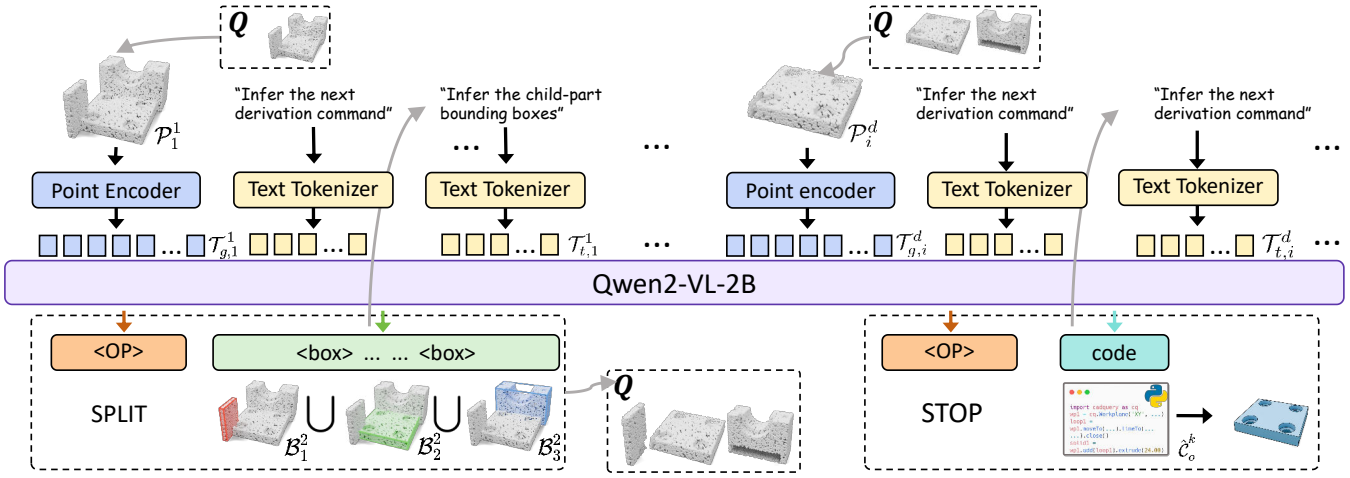

Fig. 3. Overview of the recursive CAD DSL reconstruction pipeline. Given the input point cloud $X = \{x_i \in \mathbb{R}^3\}_{i=1}^N$, the queue Q is initialized by enqueuing the whole shape $\mathcal{S}$ as the root active part $\mathcal{P}_1^1$. At each recursive step, an active part $\mathcal{P}_i^d$ is dequeued from Q, and the localized point cloud X_i^d is cropped from X using its bounding box B_i^d for $\mathcal{P}_i^d$. The localized geometry X_i^d is encoded by the point cloud encoder into geometric tokens $\mathcal{T}_{g,i}^d$, while the text instruction is tokenized into the instruction embedding $\mathcal{T}_{t,i}^d$. Conditioned on $\mathcal{T}_{g,i}^d$ and $\mathcal{T}_{t,i}^d$, the LLM-based decoder f_θ first predicts a derivation operation a_i^d using an `OPERATION` (OP.) instruction. If a_i^d = `SPLIT`, the decoder predicts a set of tokens $\hat{Y}_i^d$ as child parts $\{\mathcal{P}_{i,k}^{d+1}\}_{k=1}^{K_i}$ with their Boolean commands $O_{i,k}^{d+1}$ and bounding boxes $B_{i,k}^{d+1}$, and the point clouds inside each $B_{i,k}^{d+1}$ are inserted into Q for subsequent recursive prediction. If a_i^d = `STOP`, the recursion terminates for $\mathcal{P}_i^d$ and the model synthesizes its CADQuery code C_i^d. After all active parts in Q are resolved, the predicted leaf-level CADQuery scripts are transformed back from local normalized coordinates to the global frame and merged according to their Boolean operations to produce the final executable CADQuery script.

level. For each part $\mathcal{P}_i^d$, we first encode localized point clouds within its bounding box B_i^d, and the network decides whether to further split the part $\mathcal{P}_{i,\text{split}}^d$ or to predict the step-wise solid modeling code $\mathcal{P}_{i,\text{stop}}^d$. Finally, we perform a post-processing step that composes the predicted code blocks bottom-up into a target CADQuery script.

Localized Point Cloud Encoding. Given the bounding box B_i^d of part $\mathcal{P}_i^d$ and the input point cloud X, we first crop the points within B_i^d and then normalize the cropped point cloud to $[-1, 1]^3$, obtaining the localized point cloud X_i^d for geometric context encoding. We encode the point cloud feature with the pretrained Utonia encoder [Zhang et al. 2026]. The point-wise Utonia features are fused with a sine-cosine Fourier positional embedding of the normalized point coordinates. To keep computational efficiency, we sample 512 points from X_i^d and use their features as geometric embeddings. A two-layer MLP maps the embeddings to the MLLM hidden space ($D = 1536$), producing features $\mathcal{T}_{g,i}^d \in \mathbb{R}^{512\times D}$ for CAD DSL decoding.

Context-guided CAD DSL Prediction. Given a part entity $\mathcal{P}_i^d$ and its deriving operation $a_i^d \in \{\texttt{SPLIT}, \texttt{STOP}\}$, we employ Qwen2-VL-2B [Wang et al. 2024], denoted as f_θ, to perform part-wise CAD DSL sequence prediction. To support different types of deriving operations, we construct operation-dependent textual instructions: `OPERATION` ("Infer the next derivation command"), `SPLIT` ("Infer child-part bounding boxes"), and `STOP` ("Synthesize CADQuery code") with the `OPERATION` denoting choosing the next deriving operation. The instruction is tokenized using the Qwen2-VL tokenizer as the embedding $\mathcal{T}_{t,i}^d$.

The model f_θ takes as input the localized geometric context $\mathcal{T}_{g,i}^d$, the instruction embedding $\mathcal{T}_{t,i}^d$, and the historical MLLM context $\mathcal{T}_{m,i}^d$, and predicts the CAD DSL tokens as:

$$\hat{Y}_i^d = f_\theta\left(\mathcal{T}_{g,i}^d, \mathcal{T}_{t,i}^d, \mathcal{T}_{m,i}^d\right), \tag{6}$$

where the MLLM context $\mathcal{T}_{m,i}^d$ for the first step prediction is NULL. The predicted tokens $\hat{Y}_i^d$ are detokenized into the CAD DSL sequence for the part $\mathcal{P}_i^d$, corresponding to the part-level CADQuery code block shown in Fig. 2(b). For a non-leaf part, the target sequence is

$$\left\langle \textbf{SPLIT}, (O_{i,1}^{d+1}, B_{i,1}^{d+1}), \ldots, (O_{i,K_i}^{d+1}, B_{i,K_i}^{d+1}), \textbf{END\_SPLIT} \right\rangle.$$

`END_SPLIT` terminates the child list, after which the decoded children are inserted into Q. <os> and <oe> delimit operation tokens O_*^d, while <bs> and <be> delimit bounding-box tokens B_*^d. A leaf target sequence is $\langle \texttt{STOP}, C_i^d \rangle$.

To support hierarchical CAD DSL prediction for the entire shape, we maintain a queue Q of active parts, as illustrated in Fig. 3. The queue is initialized with the whole shape $\mathcal{S}$ as $\mathcal{P}_1^1$. At each step, an active part $\mathcal{P}_i^d$ is dequeued for prediction, and in the case of a_i^d = `SPLIT`, the predicted sub-parts are inserted back into Q. This process continues until the queue is emptied.

CADQuery script post-processing. Since the part-wise CAD DSL sequence is predicted under the normalized geometric context, an additional coordinate frame transformation is required across

parts at different hierarchy levels. We maintain an accumulated transformation for each part that maps its local coordinate frame to the global root coordinate frame. Specifically, the part $\mathcal{P}_i^d$ stores a scale $s_i^d \in \mathbb{R}^1$ and translation $\mathbf{t}_i^d \in \mathbb{R}^3$, initialized at the root as $s_1^1 = 1 \in \mathbb{R}^1$ and $\mathbf{t}_1^1 = \mathbf{0} \in \mathbb{R}^3$. For a subpart $\mathcal{P}_{i,j}^{d+1}$ of part $\mathcal{P}_i^d$, the scale and the translation can be derived as:

$$s_{i,j}^{d+1} = s_i^d \alpha, \quad \mathbf{t}_{i,j}^{d+1} = s_i^d \boldsymbol{\beta} + \mathbf{t}_i^d, \tag{7}$$

where the scaling α is determined by the maximal bounding box size value of $\mathcal{B}_{i,j}^{d+1}$ and the translation $\boldsymbol{\beta}$ is determined by the relative positions of bounding boxes centroids of $\mathcal{B}_{i,j}^{d+1}$ and $\mathcal{B}_i^d$. With the accumulated transformation in Eq. 7, a local point $\mathbf{p}_{\text{local}} \in X_i^d$ within part $\mathcal{P}_i^d$ can be mapped to the root coordinate system via

$$\mathbf{p}_{\text{root}} = s_i^d \cdot \mathbf{p}_{\text{local}} + \mathbf{t}_i^d. \tag{8}$$

We apply this transformation to convert each leaf-part code into the root coordinate frame. Given the fixed sketch–extrude grammar, this conversion is performed by transforming the API parameters: the 3D `origin` of each `Workplane` is scaled and translated; the 2D sketch coordinates, as well as the extrusion length in `extrude`, are scaled accordingly; and Boolean operations remain unchanged.

3.3 Training Objectives

We train the CAD DSL reconstruction model using three complementary objectives: a sequence-based token prediction loss and two geometry-aware losses.

Token Prediction Loss. For the prediction of CAD DSL tokens $\hat{Y}_i^d$, we employ the standard token-wise negative log-likelihood loss $\mathcal{L}_{ce}$ in a teacher-forcing manner.

Contrastive Geometry Regularization Loss. We find the localized point clouds X_i^d of the part $\mathcal{P}_i^d$ are often incomplete or noisy due to bounding-box cropping and recursive decomposition (Fig. 4). As a result, different observations of the same part may exhibit substantial geometric variations. We assume that, despite an incomplete geometric context, the remaining geometry should preserve the intrinsic structure of the underlying CAD part. Therefore, the feature representations extracted from different partial observations of the same part should remain consistent. We thereby introduce a contrastive geometric regularization based on the InfoNCE loss [Oord et al. 2018]. Specifically, for the point clouds X_i^d, we treat the augmentation of X_i^d, together with the complete geometry obtained from the ground-truth CADquery code execution of $\mathcal{P}_i^d$ as positive samples, while point clouds of other parts are negative samples. We generate each augmentation by randomly rescaling the crop bounding box by a factor in $[0.95, 1.05]$ and masking 0%–20% of its points. We apply mean pooling over the part geometry embeddings $\mathcal{T}_{g,i}^d$ to obtain a compact part-level representation $r_i^d = \text{MeanPool}(\mathcal{T}_{g,i}^d)$ followed by L_2 normalization. Based on these pooled geometry embeddings, we define the contrastive objective as,

$$\mathcal{L}_{ctr} = -\mathbb{E}_{r_i^d} \log \frac{\exp\left((r_i^d)^\top r^+/\tau\right)}{\exp\left((r_i^d)^\top r^+/\tau\right) + \sum_{r^- \in \mathcal{N}(r_i^d)} \exp\left((r_i^d)^\top r^-/\tau\right)}, \tag{9}$$

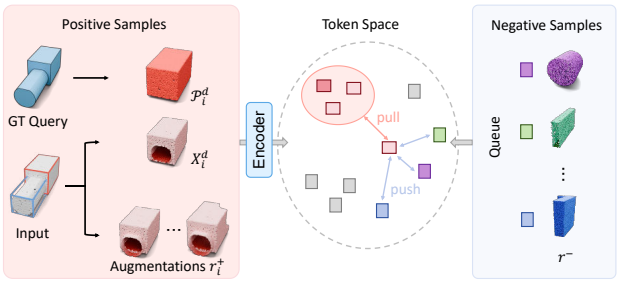

Fig. 4. For each bbox-split point cloud, we treat its cropped or perturbed variants, together with the corresponding ground-truth point cloud as positive samples, shown as red point clouds. The remaining point clouds in the queue are used as negative samples.

where r^+ denotes the embedding from positive samples of the part $\mathcal{P}_i^d$, $\mathcal{N}(r_i^d)$ denotes the set of embeddings from negative samples, r^- denotes an element of $\mathcal{N}(r_i^d)$, and τ is the temperature parameter.

We apply Eq. 9 on all hierarchical parts, including both elementary parts and intermediate non-leaf parts.

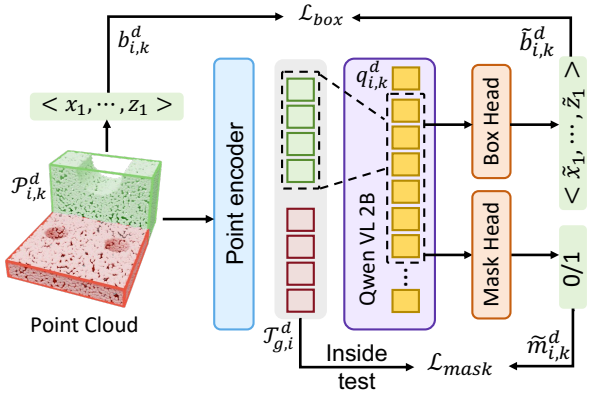

Fig. 5. The two-branch grounding head. The box prediction head takes the pooling features $q_{i,k}^d$ of box span tokens for bounding box prediction and outputs the bounding box. The mask prediction head takes as input $q_{i,k}^d$ and point-wise token for the prediction of a binary mask denoting inside and outside of a bounding box, respectively.

The Grounding Loss.

During decomposing a part $\mathcal{P}_{i,\text{split}}^{d-1}$ into the child parts $\{\mathcal{P}_{i,k}^d\}_{k=1}^{K_i}$, the token-level loss on generated bounding box $\{B_{i,k}^d\}$ does not explicitly enforce the geometric-level alignment of the predicted boxes. To address this, we incorporate an auxiliary box grounding loss applied to the output token embedding representing the bounding box, referred to as box span tokens. For the box span tokens of part $\mathcal{P}_{i,k}^d$, we pool them as $q_{i,k}^d$ and predict an auxiliary box $\tilde{B}_{i,k}^d$ and a part segmentation mask $\tilde{m}_{i,k}^d$:

$$(\tilde{B}_{i,k}^d, \tilde{m}_{i,k}^d) = \left(\phi_{\text{box}}(q_{i,k}^d), \phi_{\text{mask}}(\mathcal{T}_{g,i}^d)\right), \tag{10}$$

where $\phi_{\text{box}}(\cdot)$ and $\phi_{\text{mask}}(\cdot)$ are grounding heads implemented as two 2-layer MLPs. Specifically, $\phi_{\text{box}}(\cdot)$ regresses the normalized box

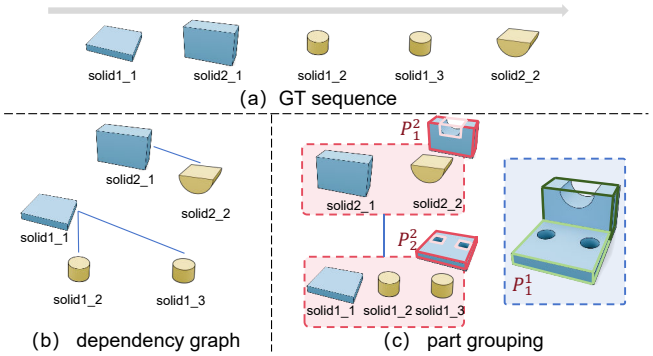

Fig. 6. Rule-based construction of hierarchical CAD DSL data. (**a**) The original ground-truth CAD sequence is represented as a list of modeling steps. (**b**) We first resolve order-sensitive `Difference` operations by grouping solids with cutting dependencies into subtrees with blue line connections. (**c**) We then merge the single-step solids and extracted subtrees according to their overlapping relationships, producing the final hierarchy.

coordinates from $q_{i,k}^d$ and ϕ_{mask} predicts a segmentation mask for each point cloud token within $B_{i,k}^d$ with points belonging to the part marked as one. The target box $B_{i,k}^d$ is parsed from the ordered ground-truth DSL, and the target mask $m_{i,k}^d$ is obtained by testing whether each geometry token lies inside $B_{i,k}^d$. The grounding loss between the target ones and the predicted ones can be formulated as:

$$\mathcal{L}_{bg} = \sum_k \left[\lambda_{box} \text{SmoothL1}\left(\tilde{B}_{i,k}^d, B_{i,k}^d\right) + \lambda_{mask} \mathcal{L}_{mask}\left(\tilde{m}_{i,k}^d, m_{i,k}^d\right) \right], \tag{11}$$

where $\mathcal{L}_{mask}$ combines binary cross-entropy and Dice loss. This objective encourages each generated bbox span to be spatially aligned with its local geometric support, leading to more consistent recursive decomposition.

The final objective can be defined as follows:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_{ctr}\mathcal{L}_{ctr} + \lambda_{bg}\mathcal{L}_{bg}, \tag{12}$$

where λ_* are balancing terms for different losses.

3.4 Rule-based Data Construction

To construct the training data, we propose a rule-based strategy that transforms CAD command sequences into hierarchical CAD DSLs. We treat each sketch–extrude step as a leaf node and convert it into the corresponding CADQuery target. This transformation is grounded in three geometric observations. First, unioned solids are commutative, meaning their order can be permuted without altering the final geometry. Second, a difference operation is order-sensitive, as it cannot be reordered with preceding solids if they have spatial overlaps. Third, overlapping solids within a union operation are more likely to constitute a reusable part.

Algorithm 1 summarizes the rule-based construction of hierarchical CAD DSL targets. Let a CAD command sequence be represented as a set of modeling steps $\{(E_t, O_t)\}_{t=1}^T$, where E_t denotes a single-step solid and O_t its corresponding Boolean operator.

Our algorithm first resolves dependencies introduced by difference operations. We construct a dependency graph where an edge is established between a solid $E_{t'}$ involving a difference operation and any preceding unioned solid E_t $(t < t')$ if their spatial domains intersect. By identifying connected components within this graph, we extract sets of solids with *cutting* relationships, denoted as $\mathcal{D}$, while the remaining independent solids are grouped into $\mathcal{U}$. Each subset in $\mathcal{D}$ is encapsulated as a part and subsequently unioned with single-step solids in $\mathcal{U}$ as $\hat{\mathcal{U}}$ to form the final shape. See Fig. 6 (b) for an example.

For the remaining unioned solids $\hat{\mathcal{U}}$, we implement a part-grouping strategy based on spatial proximity to further refine the hierarchy. The unioned solids are represented as an overlap graph, where nodes correspond to individual solids and edges denote geometric intersections, weighted by their respective intersection ratios. Initially, the entire set $\hat{\mathcal{U}}$ is encapsulated as a root part node $\mathcal{P}_1^1$. Each connected component identified within $\mathcal{G}$, including isolated nodes without intersections, is designated as a child part of $\mathcal{P}_1^1$. To ensure manageable complexity, we enforce a maximum cardinality constraint of $K = 6$. For any candidate part exceeding this threshold, we construct its Maximum Spanning Tree (MST) and apply a recursive bi-partitioning strategy by removing the edge with the minimum weight. This decomposition continues until each resulting subtree contains no more than K nodes. See Fig. 6 (c) for an example. Consequently, this process generates a full hierarchical decomposition of the CAD shape, where each internal node represents a `SPLIT` command and each leaf node denotes a `STOP` command for a single-step solid associated with its generative CADQuery code.

4 Experiments

4.1 Experimental Setups

Datasets. We mainly report results by training methods on the DeepCAD [Wu et al. 2021] training set. For testing, the DeepCAD test set is used for in-domain evaluation, while the Fusion360 [Willis et al. 2021] dataset is used to evaluate out-of-distribution (OOD) generalizability. The DeepCAD training set contains approximately 160K samples, and the DeepCAD and Fusion360 test sets contain 8,046 and 1,725 samples, respectively. We further report results obtained by training on the substantially larger CAD-Recode [Rukhovich et al. 2025] dataset, which contains around 1M training samples. This setting is included to enable fair comparisons with prior MLLM-based methods trained on the CAD-Recode dataset.

Baselines. We compare our method with seven representative methods focusing on reconstructing editable geometric solids or executable CAD programs from point clouds, including DeepCAD [Wu et al. 2021], Point2Cyl [Uy et al. 2022], HNC-CAD [Xu et al. 2023], PartCAD [Liu et al. 2026b], PS-CAD [Yang et al. 2025], Cadrille [Kolodiazhnyi et al. 2025], and CAD-Recode [Rukhovich et al. 2025]. Point2Cyl [Uy et al. 2022] segments potential extrusion cylinders and locates the cross-section (base) and side surface (barrel) for each segmented instance. Afterward, Point2Cyl fits the extrusion cylinder based on the located base and barrel. For HNC-CAD [Xu et al. 2023], we inject point-cloud features to adapt it to our point-cloud input setting. We compare with these methods by running the available codes and pre-trained weights with the same test splits for a fair

ALGORITHM 1: Rule-based construction of hierarchical CAD DSLs

Input : CAD modeling steps $\mathcal{A} = \{(E_t, O_t)\}_{t=1}^{T}$ and the maximum number of subparts K
Output: Hierarchical CAD DSL target $\mathcal{Y}$

```
// Convert modeling steps to single-step solids
foreach t = 1, ..., T do
    v_t ← ExecuteCADCmd(E_t)
end
V ← {v_t}_{t=1}^{T}
G_D ← (V, ∅)
// Resolve dependencies among step-wise solids
foreach t' = 1, ..., T and O_t' = Difference do
    foreach t = 1, ..., t' − 1 do
        if E_t' ∩ E_t ≠ ∅ then
            AddEdge(G_D, v_t, v_t')
        end
    end
end
B ← ConnectedComponents(G_D)
// Build the overlap graph
G ← (B, ∅)
foreach i = 1, ..., |B| do
    foreach j = i + 1, ..., |B| do
        if b_i ∩ b_j ≠ ∅ then
            w_ij ← IntersectionRatio(b_i, b_j)
            AddEdge(G, b_i, b_j, w_ij)
        end
    end
end
// Build the hierarchy
T ← MaximumSpanningTree(G)
H ← BuildTree(T, K)
Y ← ConvertToDSL(H)
Function BuildTree(T, K):
    if CountNodes(T) ≤ K then
        return BuildNode(T)
    end
    (T_l, T_r) ← SplitAtWeakestEdge(T)
    return BuildNode (BuildTree(T_l, K), BuildTree(T_r, K))
end
```

comparison. For PartCAD [Liu et al. 2026b] without open-source code and checkpoints, we report the results in the original paper.

Training Details. Our framework adopts Qwen2-VL-2B-Instruct [Wang et al. 2024] as the backbone language model for CAD DSL prediction, together with Utonia as the point-cloud encoder. For each input shape, we uniformly sample 8,192 points as input. Training is performed with a batch size of 4 and a gradient accumulation step of 8 for a maximum of 120,000 iterations. We use an initial learning rate of 2×10^{-4} with a cosine decay schedule after a 1,000-step warmup. For the geometry-invariant contrastive objective, we set the loss weight to $\lambda_{\text{con}} = 0.05$, the temperature to $\tau = 0.07$. For bbox grounding, we use a total weight of $\lambda_{\text{bg}} = 0.1$. The grounding loss combines a Smooth L1 box regression term with weight $\lambda_{\text{box}} = 1.0$ and a point-mask term with weight $\lambda_{\text{mask}} = 0.25$, where the point-mask term consists of BCE and Dice losses.

Metrics. Following prior CAD reconstruction works [Kolodiazhnyi et al. 2025; Yang et al. 2025], we evaluate geometry quality and sequence fidelity. For geometric reconstruction, we report Chamfer Distance (CD), Hausdorff Distance (HD), Edge Chamfer Distance (ECD), Normal Consistency (NC), intersection-over-union (IoU), and Invalid Rate (IR). CD and HD are computed from points uniformly sampled on the predicted and ground-truth CAD surfaces, measuring the average and worst-case bidirectional surface discrepancy, respectively. ECD is computed on sampled CAD edge points and reflects the accuracy of sharp boundaries and structural edges. Note that the values of CD, HD, and ECD metrics are rescaled by 1000 in our reported results. NC measures the cosine similarity between corresponding surface normals, capturing local surface-orientation consistency. IoU is computed on occupied voxels of the normalized prediction and ground-truth into a unit bounding box. IR measures the percentage of test samples for which no valid executable CAD reconstruction can be obtained. In addition to Mean CD, we also report Median CD to reduce sensitivity to a small number of difficult failures.

For sequence fidelity, we report the step-wise negative log-likelihood (NLL) of the CAD reconstruction process. Each step corresponds to the CAD DSL sequence of an elementary part (i.e., a single-step solid). Given the input CAD point cloud and all preceding ground-truth steps, we evaluate the predicted logits for the next step.

4.2 Comparison Study

Quantitative comparisons of DeepCAD-trained models. Table 1 reports the main comparison on DeepCAD and Fusion360. CADRec consistently achieves state-of-the-art results across all reported metrics for both the DeepCAD and Fusion360 test sets. On DeepCAD, CADRec reduces the Mean CD to 0.62, representing a 70.5% improvement over the second-best method, PS-CAD (2.10). This suggests that our method consistently enhances reconstruction quality across the entire dataset, rather than being driven by improvements in a few outlier cases. The Median CD decreases from 0.22 to 0.17. Both metrics clearly demonstrate a significant improvement in the quality of the reconstructed geometry. Compared to the second-best IoU metric of Cadrille, CADRec improves the IoU from 79.40 to 92.70 and reduces the IR from 0.90 to 0.20.

The advantage of CADRec becomes more pronounced on the out-of-distribution Fusion360 dataset. We observe that the reconstruction quality of existing methods deteriorates substantially under the domain shift from DeepCAD to Fusion360, with significant increases in both Mean CD (1.95×) and Median CD (2.3×). In particular, CAD-Recode and Cadrille obtain Mean CD values of 6.45 and 6.65, respectively. By contrast, CADRec maintains stable reconstruction quality, reducing Mean CD to 0.81, corresponding to an 87.8% improvement over Cadrille. These results indicate that CADRec captures more transferable geometry and structural priors, enabling robust generalization to structurally diverse CAD models beyond the training distribution. Consistent improvements are also observed in other metrics, where IoU increases from 67.1 to 85.8 and IR decreases from 1.5 to 0.1.

Test-time sampling. Table 2 reports the test-time sampling results [Rukhovich et al. 2025] trained on DeepCAD. In this setting, we execute 10 reconstructions for each input and take the best result to compute the overall metric. CADRec outperforms CAD-Recode and Cadrille in Mean CD, Median CD, and IoU on both DeepCAD and Fusion360, and achieves a lower IR on Fusion360.

Table 1. Quantitative comparison on the DeepCAD and Fusion360 test sets. We report mean Chamfer Distance (Mean CD), median Chamfer Distance (Med. CD), intersection-over-union (IoU), and invalid rate (IR). Lower is better for CD and IR, while higher is better for IoU. Results marked with # are taken directly from the original papers.

Method	Train Set	DeepCAD Test Set				Fusion360 Test Set			
		Mean CD↓	Med. CD↓	IoU↑	IR↓	Mean CD↓	Med. CD↓	IoU↑	IR↓
DeepCAD [Wu et al. 2021]	DeepCAD	42.5	9.64	46.7	7.1	330	89.2	39.9	25.2
Point2Cyl [Uy et al. 2022]	DeepCAD	10.2	4.27	73.8	3.9	–	4.18	67.5	3.2
HNC-CAD [Xu et al. 2023]	DeepCAD	10.91	8.64	65.3	5.6	13.8	36.8	63.5	7.3
PartCAD [Liu et al. 2026b]#	DeepCAD	4.93	0.24	–	0.9	–	1.13	–	1.3
PS-CAD [Yang et al. 2025]	DeepCAD	2.10	0.22	–	0.4	5.70	0.94	–	1.5
CAD-Recode [Rukhovich et al. 2025]	DeepCAD	3.37	0.25	78.1	1.0	6.45	0.72	66.4	1.8
Cadrille [Kolodiazhnyi et al. 2025]	DeepCAD	3.41	0.25	79.4	0.9	6.65	0.58	67.1	1.5
Ours	DeepCAD	**0.62**	**0.17**	**92.7**	**0.2**	**0.81**	**0.19**	**85.8**	**0.1**
CAD-Recode [Rukhovich et al. 2025]	CAD-Recode	0.76	0.18	87.1	3.8	1.18	0.20	81.9	4.4
Cadrille [Kolodiazhnyi et al. 2025]	CAD-Recode	0.66	0.18	87.1	2.1	0.65	0.19	84.4	1.1
Ours	CAD-Recode	**0.46**	**0.16**	**93.3**	**0.4**	**0.48**	**0.17**	**84.7**	**0.3**

Table 2. Quantitative comparison under test-time sampling with 10 reconstruction candidates per input on the DeepCAD and Fusion360 test sets.

Method	Mean CD↓	Med. CD↓	IoU↑	IR↓
DeepCAD				
CAD-Recode	1.696	0.230	85.306	**0.025**
Cadrille	1.657	0.231	85.912	**0.025**
Ours	**0.344**	**0.164**	**93.882**	0.037
Fusion360				
CAD-Recode	2.168	0.320	77.733	0.174
Cadrille	2.012	0.307	78.92	0.168
Ours	**0.565**	**0.178**	**88.091**	**0.058**

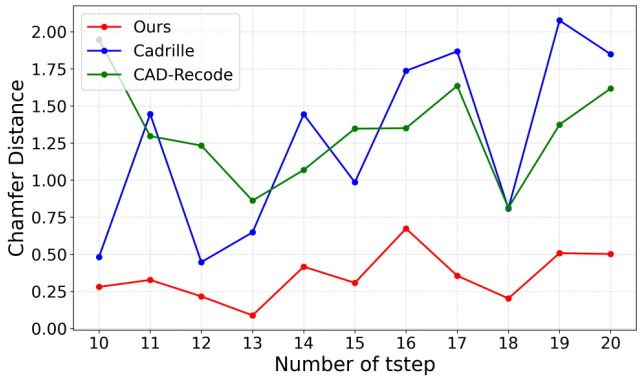

Fig. 7. The plot of Mean CD with respect to the number of modeling steps of CAD shapes, evaluated using CAD-Recode-trained models and tested on the Fusion360 dataset.

Quantitative comparisons of CAD-Recode-trained models. Training on the substantially larger CAD-Recode dataset significantly improves the performance of code-based reconstruction methods [Kolodiazhnyi et al. 2025; Rukhovich et al. 2025]. CADRec remains consistently better than CAD-Recode and Cadrille across both test sets. On DeepCAD, CADRec reduces Mean CD by 30.3% from 0.66 to 0.46 compared with Cadrille and further improves Median CD from 0.18 to 0.16. The IoU increases from 87.10 to 93.30, and IR decreases from 2.10 to 0.40, demonstrating improvement in geometric accuracy, volumetric consistency, and executable validity.

On Fusion360, CADRec reduces Mean CD from 0.65 to 0.48 and Median CD from 0.19 to 0.17, while lowering IR from 1.1 to 0.3 in a large margin. The IoU gain on Fusion360 is smaller, increasing from 84.40 to 84.70, which suggests that CAD-Recode-trained baselines already recover much of the coarse object volume under this setting. However, the larger improvements in CD and IR indicate that CADRec better resolves fine-scale geometric details and produces more reliable executable programs. To assess the performance on CAD shapes with long sequences, we plot the stepwise Mean CD curve for CAD shapes with more than 10 modeling steps in Fig. 7. Our method maintains a lower stepwise Mean CD on CAD models across all steps, indicating better robustness when reconstructing structurally more complex shapes.

We report the NLL comparison in Table 3. Compared with Cadrille, CADRec reduces the NLL from 1.21 to 0.76. This large improvement suggests that the proposed hierarchical representation and geometric guidance substantially improve the accuracy of CAD DSL sequence prediction.

Qualitative comparison. In Figs. 8 and 9, we qualitatively compare CADRec with PS-CAD, CAD-Recode, and Cadrille on the DeepCAD and Fusion360 test sets. We illustrate the strongest versions of the competing methods, including PS-CAD trained on DeepCAD and CAD-Recode/Cadrille trained on the CAD-Recode dataset. Overall, CADRec produces the most accurate overall structures and the best local geometric details. Existing methods generally

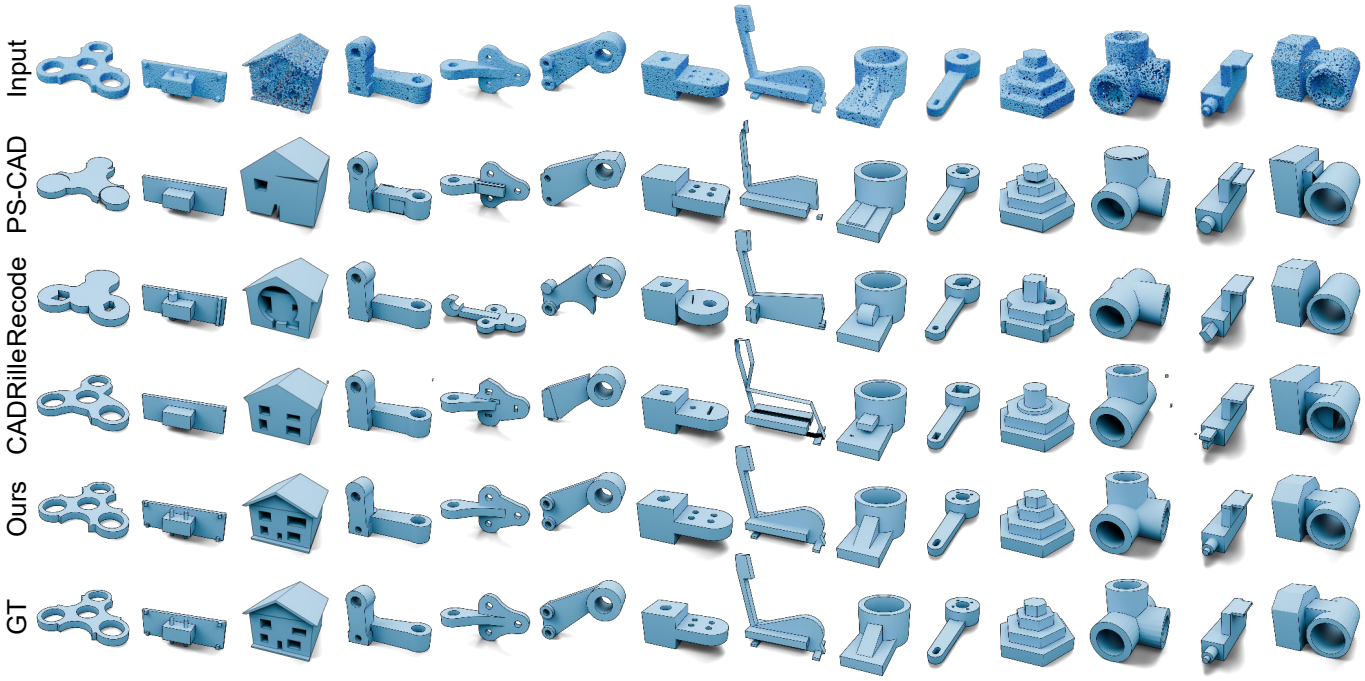

Fig. 8. Qualitative comparison of different methods on DeepCAD. PS-CAD is trained using the DeepCAD training set, while the other methods are trained using CAD-Recode. Please zoom in to see the deviations from ground truth.

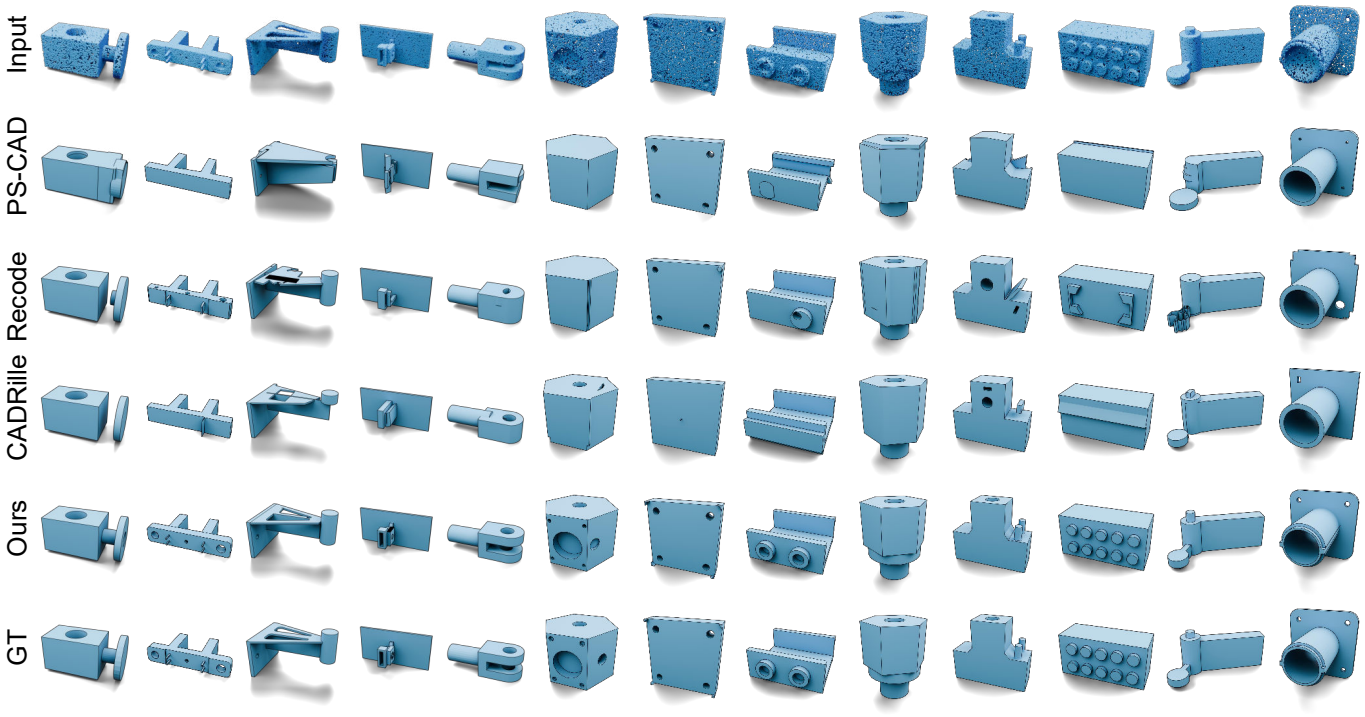

Fig. 9. Qualitative comparison of different methods on Fusion360. PS-CAD is trained using the DeepCAD training set, while the other methods are trained using CAD-Recode. Please zoom in to see the deviations from ground truth.

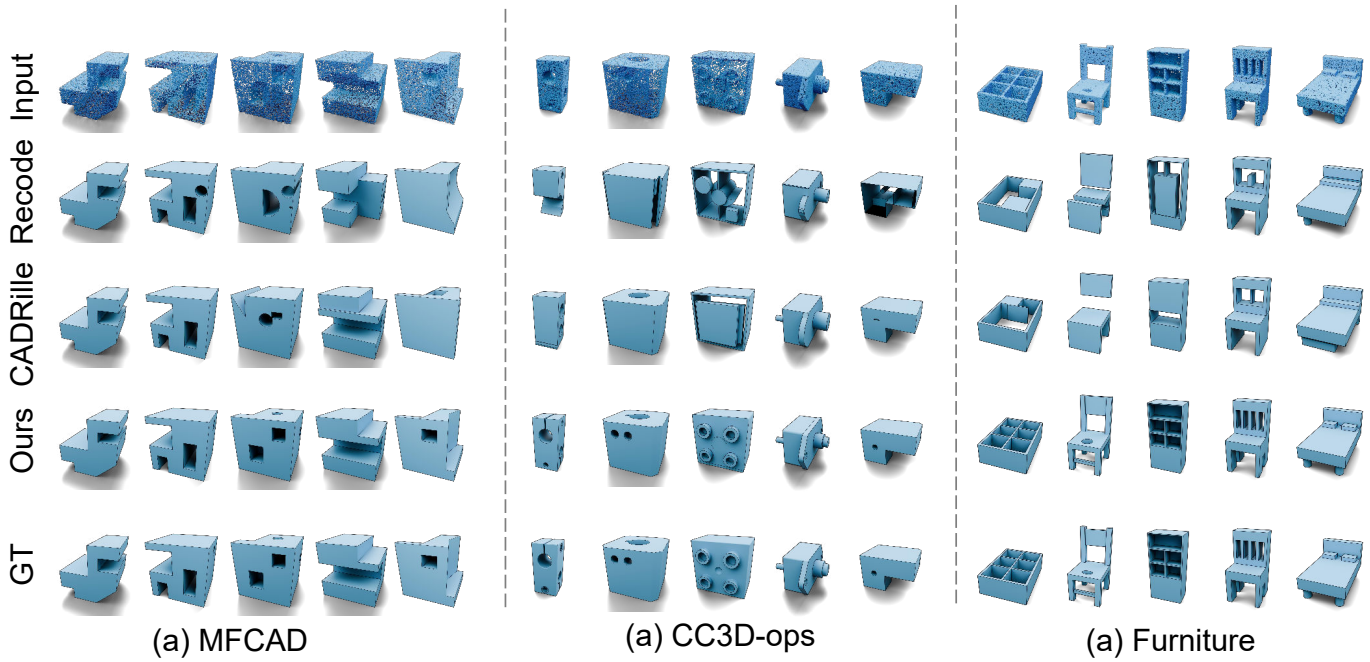

Fig. 10. Qualitative comparison on the in-the-wild datasets (MFCAD, CC3D-ops, and Furniture). We use Cad-Recode-trained models for the comparison.

Table 3. Quantitative comparison of CAD sequence prediction on the DeepCAD test set. We report the negative log-likelihood (NLL) of the predicted CAD sequence. Lower is better.

Method	NLL↓
Cadrille [Kolodiazhnyi et al. 2025]	1.21
Ours	**0.76**

Table 4. Quantitative comparison on the CAD-MLLM dataset. The results for CAD-MLLM are directly taken from the original paper (denoting with #).

Method	Mean CD↓	IoU↑	IR↓
DeepCAD [Wu et al. 2021]	45.1	–	5.7
CAD-MLLM [Xu et al. 2024b][#]	18.5	–	1.3
Cadrille [Kolodiazhnyi et al. 2025]	0.77	84.2	0.6
Ours	**0.59**	**91.4**	**0.3**

recover plausible coarse shapes, but cannot reconstruct complete fine-grained structures. In the second and sixth columns of Fig. 8 and Fig. 9, competing methods miss several small solids. Their robustness further deteriorates as structural complexity increases. In particular, they frequently confuse circular and rectangular holes (Fig. 8 column 7, Fig. 9 column 10), leading to incorrect local topology and mismatched subtractive operations. We also observe that prior methods struggle with repetitive structures and may omit repeated components. This behavior can be observed in the antepenultimate examples of Fig. 9. In addition, Cadrille often produces small spurious disconnected components, as shown in the third, fifth, and antepenultimate examples of Fig. 8. By recursively decomposing the shape into localized reconstruction regions, CADRec reduces the complexity of predicting the entire CAD shape with a single long sequence and grounds the CADQuery code of each elementary part to a precise geometric region in the input point cloud. This contributes to more complete small structures, more accurate hole types, and fewer isolated artifacts.

Qualitative evaluations on in-the-wild dataset. We further evaluate the generalization ability of CADRec on additional in-the-wild datasets, including MFCAD [Guo 2021], CC3D-Ops [Dupont et al. 2022], and the furniture dataset following BrepGen [Xu et al. 2024a]. As shown in Fig. 10, our method can reconstruct editable CAD models from point clouds beyond the DeepCAD and Fusion360 domains. The results indicate that the proposed recursive decomposition and localized CAD reconstruction are not restricted to synthetic data, and can also handle real CAD models and point clouds from other object categories.

Comparison on the CAD-MLLM dataset. While CAD-MLLM [Xu et al. 2024b] released their training dataset, the official source code remains unavailable. To ensure a fair comparison, we retrained the competing methods using the CAD-MLLM dataset and evaluated them. The results are summarized in Table 4. Compared with DeepCAD, CAD-MLLM provides a larger and more diverse source of CAD command sequences, with more detailed and realistic shapes. This setting therefore tests whether the proposed hierarchy can scale to other training corpora, rather than being specialized to the DeepCAD command distribution. Under this protocol, CADRec

Table 5. Quantitative comparison of CAD reconstruction quality on the DeepCAD test set by training methods with the DeepCAD training set.

Method	HD↓	ECD↓	NC↑
PS-CAD [Yang et al. 2025]	8.66	4.65	0.89
CAD-Recode [Rukhovich et al. 2025]	5.28	1.83	0.90
Cadrille [Kolodiazhnyi et al. 2025]	5.01	1.70	0.91
Ours	**4.12**	**1.15**	**0.92**

achieves the best performance among all compared methods. Compared with Cadrille, CADRec reduces Mean CD from 0.77 to 0.59, improves IoU from 84.20 to 91.40, and lowers IR from 0.60 to 0.30.

The result shows that our method is scalable to different CAD-sequence training sources. When trained on the more complex CAD-MLLM data, CADRec maintains strong reconstruction accuracy and reduces Mean CD compared with the DeepCAD-trained setting.

Boundary and normal consistency. Table 5 evaluates the reconstruction quality using HD, ECD, and NC, following the metric protocol of PS-CAD [Yang et al. 2025]. HD reflects large surface deviations, ECD focuses on edge-level structural accuracy, and NC evaluates local surface-orientation consistency. CADRec achieves the best result on all three metrics. Compared with Cadrille, CADRec reduces HD from 5.01 to 4.12, corresponding to a 17.8% improvement. This indicates that the proposed method suppresses severe local deviations rather than only improving the overall surface reconstruction accuracy The improvement is more pronounced on ECD, where CADRec reduces the error from 1.70 to 1.15, a 32.4% improvement over Cadrille and a 75.3% improvement over PS-CAD. Since ECD is computed on CAD edge points, this gain suggests that recursive local reconstruction better preserves sharp boundaries, profile curves, and subtractive feature edges. CADRec also obtains the highest normal consistency (NC), improving from 0.91 to 0.92 over Cadrille. Although the NC margin is smaller, the consistent gain indicates that the reconstructed surfaces have slightly better normal alignment and fewer orientation artifacts. Together, these results show that CADRec improves not only the overall structure, but also the local boundary and surface properties that are critical for editable CAD reconstruction.

4.3 Ablation Study

For all ablation studies, we train the models under the DeepCAD training set and evaluate them on the DeepCAD test set.

Effect of the CAD sequence format. Table 6 investigates the impact of different sequence organizations on performance. We retrain CAD-Recode and Cadrille using two distinct code representations: (1) a flat representation that stacks all CAD commands into a single long sequence (as shown in Fig. 2(a)), and (2) a stepwise representation that organizes the CAD commands as a composition of code blocks, where each code block corresponds to the CAD commands of a single-step solid, following the leaf-part format illustrated in Fig. 2(b). Experimental results demonstrate that stepwise grouping consistently improves performance. For CAD-Recode, the Mean CD decreases from 3.37 to 2.67, while the IoU improves from 78.10% to 83.30%. Similarly, the Mean CD of CADrille drops from 3.41 to 2.32 and its IoU rises from 79.40% to 82.40%. These results indicate that reorganizing CAD command sequences into stepwise code blocks is inherently beneficial, even without modifying the underlying model architecture. Nevertheless, a substantial gap remains compared with CADRec, suggesting that improvements in sequence organization alone are insufficient. By introducing the proposed hierarchical CAD DSL, our method further reduces Mean CD from 1.27 to 0.62 (a 51.2% improvement), increases IoU from 88.0 to 92.7, and lowers IR from 0.6 to 0.2.

Table 6. Ablations on different data representations. $^{\diamond}$ denotes the training with a flat, lengthy script, * denotes the training with stepwise script blocks, and ours in the last row is trained with the proposed hierarchical CAD DSL. The metrics are evaluated on the DeepCAD test set.

Method	Mean CD↓	Med. CD↓	IoU↑	IR↓
CAD-Recode [Rukhovich et al. 2025]$^{\diamond}$	3.37	0.25	78.1	1.0
CAD-Recode*	2.67	0.24	83.3	0.8
Cadrille [Kolodiazhnyi et al. 2025]$^{\diamond}$	3.41	0.25	79.4	0.9
Cadrille*	2.32	0.22	82.4	0.9
Ours*	1.27	0.19	88.0	0.6
Ours (hierarchical DSL)	**0.62**	**0.17**	**92.7**	**0.2**

Table 7. Ablation study of each module on the DeepCAD test set. PE, BM, H, RG, CL, and BG denote point-cloud encoder, bbox multi-task supervision, hierarchical CAD DSL, recursive part-wise context encoding, contrastive learning, and bbox grounding, respectively.

Components						Reconstruction			
PE	BM	H	RG	CL	BG	Mean CD↓	Med. CD↓	IoU↑	IR↓
						2.64	0.24	79.4	0.84
√						1.92	0.21	81.0	0.75
	√					2.06	0.22	84.1	0.83
√	√					1.52	0.20	85.6	0.47
√	√	√				1.23	0.18	87.6	0.76
√	√	√	√			0.79	0.17	92.1	0.27
√	√	√	√	√		0.69	0.18	91.0	0.57
√	√	√	√	√	√	**0.62**	**0.17**	**92.7**	**0.20**

Point encoder and multitask losses. We ablate the primary components of CADRec in Table 7. Replacing the MLP-based positional embedding encoder (row 1) with a Utonia-based point cloud encoder (row 2) leads to a substantial performance gain, reducing Mean CD to 1.92 and lowering IR from 0.84 to 0.75. These results highlight the importance of high-quality geometric feature extraction. Qualitative results in Fig. 11 further show that while the baseline often misses local point-cloud details, our point cloud encoder maintains a more accurate grasp of both local patterns and global structure.

Adding the token prediction loss for bounding boxes of stepwise solids (*BM*) provides a complementary effect, reducing Mean CD to 2.06 and improving IoU from 79.4 to 84.1. The IoU gain suggests that the guidance from stepwise bounding boxes helps the model learn a better spatial structure. Combining the point-cloud encoder and the token prediction loss for stepwise solid bounding boxes

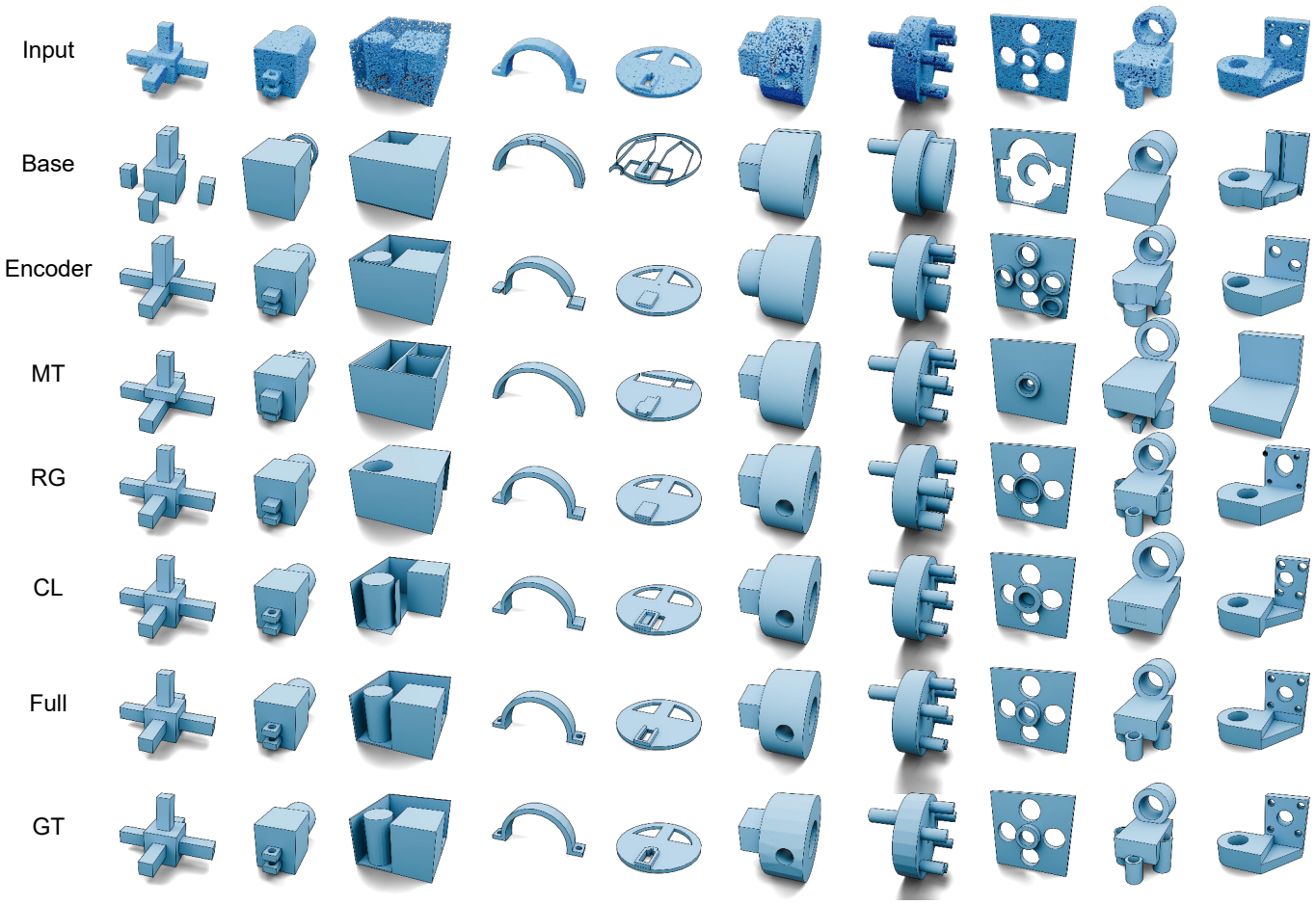

Fig. 11. Qualitative comparison of ablation on DeepCAD. Please zoom in to see the deviations from ground truth.

further reduces Mean CD to 1.52, improves IoU to 85.60, and lowers IR to 0.47, confirming their complementary influences.

Recursive part splitting. By extending the flat solid-wise script to our hierarchical DSL (H), we observe an improvement of 0.29 in Mean CD, 0.02 in Median CD, and 2.0 in IoU. However, the IR drops for 0.29, possibly because of the increase in the CAD script complexity. Adding both the hierarchical DSL (H) and recursive geometric context encoding $\mathcal{T}_{g,i}^{d}$ (RG) within the part bounding box reduces Mean CD from 1.52 to 0.79, improving IoU from 85.60 to 92.10 and lowers IR from 0.47 to 0.27. This result supports the central design of CADRec. Instead of predicting the whole CAD program in one global sequence, recursive generation repeatedly uses predicted boxes to localize geometric contexts, allowing the model to allocate more resolution to complex regions and synthesize shorter terminal CADQuery programs. As shown in the fifth row of Fig. 11, the recursive decomposition enables the model to split the input point cloud into meaningful local regions, reconstruct each part independently, and then assemble them into a coherent CAD model. This leads to a more faithful recovery of the global structure while preserving small geometric details that are often missed by non-recursive variants.

Contrastive learning. Although recursive generation substantially improves reconstruction quality, the method can still confuse the roles of cropped local point clouds and the complete object point cloud. As shown in the third and last columns of the fifth row in Fig. 11, this confusion may lead to missing local details or inconsistent geometry of reconstructed shapes. We enforce a contrastive loss to encode robust contexts for point clouds with diverse artifacts. This design reduces Mean CD from 0.79 to 0.69, further improving the reconstruction quality.

The bbox grounding loss. Even with recursive generation and contrastive learning, the model may still suffer from localization errors in some cases. As shown in the fifth column of the sixth row in Fig. 11, some reconstructed components are shifted from their correct positions. Similarly, the third-from-last column of the fourth row shows that inaccurate splitting can lead to incomplete or misaligned local reconstruction. To further improve spatial perception, we introduce bbox grounding losses. The use of this component improves the model's localization of different parts, as shown in the

Table 8. The impact of the recursive bounding box prediction. GT denotes using the GT bounding box for CAD reconstruction.

	Reconstruction				Box Pred.
	Mean CD↓	Med. CD↓	IoU↑	IR↓	mIoU↑
Pred	0.62	0.17	92.7	0.20	73.2
GT	0.58	0.17	93.4	0.09	N/A

Table 9. Ablation study of the DFS and BFS-based part deriving.

	Mean CD↓	Med. CD↓	IoU↑	IR↓
BFS	0.72	0.17	91.1	0.62
DFS	0.62	0.17	92.7	0.20

third and fifth columns of the penultimate row in Fig. 11, as well as single-step operations, especially for handling holes, as shown in the first and third columns from the right of the penultimate row.

With all components enabled, CADRec achieves the best results on all metrics. Compared with the recursive variant without contrastive learning and bbox grounding, the full model further reduces Mean CD by 21.5% and IR by 25.9%. These gains indicate that bbox grounding improves the model's ability to localize each split region and anchor local program prediction to the correct spatial support.

Effect of predicted boxes. Table 8 evaluates the impact of the bounding box quality. Using predicted boxes, our method already achieves strong reconstruction performance, with Mean CD 0.62, Median CD 0.17, and IoU 92.7; following [Wang et al. 2025c], the predicted boxes achieve a bbox mIoU of 73.2. Ground-truth boxes further improve Mean CD to 0.58 and IoU to 93.4, and reduce IR from 0.20 to 0.09. The small gap between the two settings indicates that the predicted boxes are sufficiently reliable for recursive decomposition despite the inaccurate bounding box prediction.

Generation order. Table 9 compares two generation orders for recursive reconstruction. In the BFS variant, all split boxes are first obtained, and the point cloud inside each box is reconstructed independently. In contrast, DFS generates each step with the accumulated context from previous steps. DFS improves Mean CD from 0.72 to 0.62 and IoU from 91.1 to 92.7, while keeping the same Median CD of 0.17. It also reduces IR from 0.62 to 0.20. These results suggest that the accumulated context in DFS helps maintain generation consistency and reduces invalid CAD programs. Therefore, we adopt DFS as the default generation order in our final model and all the other experiments.

Qualitative results for hierarchical CAD reconstruction. In Fig. 12, we further examine the hierarchical reconstruction along with its recursive bounding box prediction. Our method is able to recover the complex hierarchical structure with the guidance of recursively derived bounding boxes.

4.4 Robustness Evaluation

To further assess the robustness of our approach, we report results on the full DeepCAD test set with different levels of noise. We add Gaussian noise sampled from a zero-mean normal distribution to each 3D point, with the standard deviation σ ranging from 0.01 to 0.10. Table 10 reports the Median CD under different noise levels. The Median CD of our method remains relatively stable from $\sigma = 0.01$ to $\sigma = 0.05$, and increases at $\sigma = 0.10$, showing the robustness of our approach to small-scale noise. Compared with both baselines, CADRec consistently obtains the lowest Median CD at different levels of noise, and its advantage is most pronounced at $\sigma = 0.10$.

Table 10. Robustness evaluation for point cloud inputs of different noise scales. We report Median CD on the full DeepCAD test set.

Method	Noise Std. σ				
	0.01	0.02	0.03	0.05	0.10
CAD-Recode	0.193	0.223	0.254	0.296	0.392
Cadrille	0.181	0.184	0.193	0.227	0.351
Ours	0.163	0.167	0.165	0.168	0.185

Table 11. Runtime performance.

	Params (M)	Inference (s/sample)
Cadrille	**2209.07**	**0.57**
Ours	2355.69	1.26

4.5 Runtime Performance

Tab. 11 reports the runtime statistics extracted from the profiling summaries. We report the total number of parameters (Params, M) and the average inference time per sample (s/sample). Our method requires longer inference time due to its recursive decomposition and reconstruction process. This design prioritizes reconstruction fidelity and fine-grained CAD structure over the reconstruction speed.

4.6 Limitations

Fig. 13 demonstrates two typical failure cases. In the first case, the input point cloud has complex nesting, with multiple sets of cylinders nested together. Our method cannot recover details, as the point cloud for some parts is affected by interference from surrounding parts. In the second case, the input point cloud are multiple cylinders that can be combined, but the current point cloud sampling may identify them as a single cylinder by mistake. Moreover, our framework does not support advanced modeling operations, such as fillets, chamfer, and freeform surfaces and extending both the training data and CADQuery codes to cover these modeling operations is an important direction for future work.

5 Conclusions

We presented CADRec, a hierarchical point-cloud-to-CAD reconstruction framework that recovers editable CAD models through recursive decomposition and local CADQuery script prediction. By organizing single-step solids into a part hierarchy, CADRec separates global structural reasoning from localized geometry reconstruction and assembles the predicted local programs into a

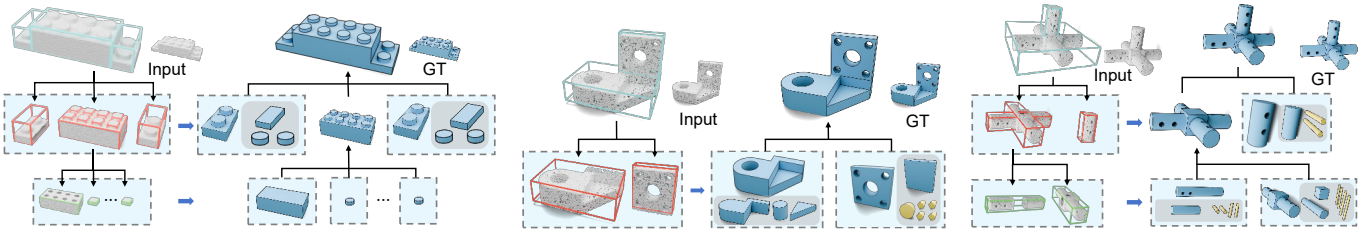

Fig. 12. Qualitative results for hierarchical CAD reconstruction. Each example displays recursive top-down bounding box predictions (left) alongside the corresponding part hierarchy, constructed bottom-up from CAD reconstructions (right).

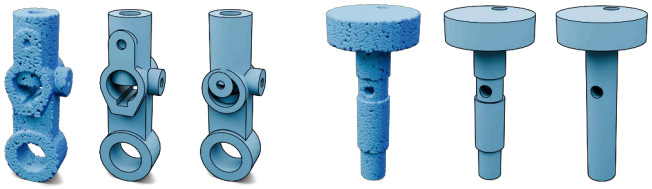

Fig. 13. Failure cases. The method has difficulty in recovering (a) complex nested structures and (b) small geometry details.

complete CAD model. Experiments on DeepCAD and Fusion360 show that this hierarchical formulation improves geometric accuracy, program validity, and geometric regularity over existing methods, and further suggests its potential for data-efficient CAD reconstruction. A current limitation is that our framework relies on a predefined sketch–extrude grammar and bounding-box-based localization; extending it to richer CAD operations and more flexible part decompositions is an important direction for future work.

References

Mahmoud Ahmed, Junjie Fei, Jian Ding, Eslam Mohamed Bakr, and Mohamed Elhoseiny. 2025. Kestrel: 3D Multimodal LLM for Part-Aware Grounded Description. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 8973–8983.

Anthropic. 2025. Claude Code: A command line interface for software engineering. https://code.claude.com/. Accessed: 2026-05-10.

Blender Online Community. 2024. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. http://www.blender.org

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

CadQuery Contributors. 2023. CadQuery: A Python parametric CAD scripting framework based on OCCT. https://github.com/CadQuery/cadquery.

Bingquan Dai, Luo Li, Qihong Tang, Jie Wang, Xinyu Lian, Hao Xu, Minghan Qin, Xudong Xu, Bo Dai, Haoqian Wang, et al. 2026. Meshcoder: Llm-powered structured mesh code generation from point clouds. *Advances in Neural Information Processing Systems* 38 (2026), 8917–8954.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems* 36 (2023), 49250–49267.

Yijie Ding, Yang Liu, Haobo Jiang, and Jianmin Zheng. 2026. ReACT: Reward-informed Autoregressive Decision CAD Transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 3614–3622.

Elona Dupont, Kseniya Cherenkova, Anis Kacem, Sk Aziz Ali, Ilya Arzhannikov, Gleb Gusev, and Djamila Aouada. 2022. Cadops-net: Jointly learning cad operation types and steps from boundary-representations. In *2022 International Conference on 3D Vision (3DV)*. IEEE, 114–123.

Elona Dupont, Kseniya Cherenkova, Dimitrios Mallis, Gleb Gusev, Anis Kacem, and Djamila Aouada. 2024. Transcad: A hierarchical transformer for cad sequence inference from point clouds. In *European Conference on Computer Vision*. Springer, 19–36.

Shuangkang Fang, I Shen, Yufeng Wang, Yi-Hsuan Tsai, Yi Yang, Shuchang Zhou, Wenrui Ding, Takeo Igarashi, Ming-Hsuan Yang, et al. 2025. Meshllm: Empowering large language models to progressively understand and generate 3d mesh. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14061–14072.

Yandong Guan, Xilin Wang, Ximing Xing, Jing Zhang, Dong Xu, and Qian Yu. 2026. Cad-coder: Text-to-cad generation with chain-of-thought and geometric reward. *Advances in Neural Information Processing Systems* 38 (2026), 59765–59789.

Huidong Guo. 2021. MFCAD: A Dataset of 3D CAD Models with Machining Feature Labels. https://github.com/hducg/MFCAD. Accessed: 2026-05-13.

Zonghao Guo, Ruyi Xu, Yuan Yao, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. 2024. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. In *European Conference on Computer Vision*. Springer, 390–406.

Souhail Hadgi, Bingchen Gong, Ramana Sundararaman, Emery Pierson, Lei Li, Peter Wonka, and Maks Ovsjanikov. 2026. PatchAlign3D: Local Feature Alignment for Dense 3D Shape understanding. *arXiv preprint arXiv:2601.02457* (2026).

Soslan Kabisov, Vsevolod Kirichuk, Andrey Volkov, Gennadii Savrasov, Marina Barannikov, Anton Konushin, Andrey Kuznetsov, and Dmitrii Zhemchuzhnikov. 2026. CADReasoner: Iterative Program Editing for CAD Reverse Engineering. *arXiv preprint arXiv:2603.29847* (2026).

Ahmet Karadeniz, Dimitrios Mallis, Danila Rukhovich, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. 2026. MiCADangelo: Fine-Grained Reconstruction of Constrained CAD Models from 3D Scans. *Advances in Neural Information Processing Systems* 38 (2026), 82830–82848.

Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. 2024. 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*. 4713–4722.

Mahyar Khayatkhoei, Prateek Chhikara, Filip Ilievski, et al. 2025. Mllms know where to look: Training-free perception of small visual details with multimodal llms. In *International Conference on Learning Representations*, Vol. 2025. 68194–68213.

Maksim Kolodiazhnyi, Denis Tarasov, Dmitrii Zhemchuzhnikov, Alexander Nikulin, Ilya Zisman, Anna Vorontsova, Anton Konushin, Vladislav Kurenkov, and Danila Rukhovich. 2025. Cadrille: Multi-modal CAD Reconstruction with Reinforcement Learning. In *The Fourteenth International Conference on Learning Representations*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023b. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*. PMLR, 19730–19742.

Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. 2023a. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16816–16826.

Xueyang Li, Jiahao Li, Yu Song, Yunzhong Lou, and Xiangdong Zhou. 2026. SeekCAD: A Self-refined Generative Modeling for 3D Parametric CAD Using Local Inference via DeepSeek. In *The Fourteenth International Conference on Learning Representations*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems* 36 (2023), 34892–34916.

Yilin Liu, Jiale Chen, Shanshan Pan, Daniel Cohen-Or, Hao Zhang, and Hui Huang. 2024a. Split-and-Fit: Learning B-Reps via Structure-Aware Voronoi Partitioning. *ACM Transactions on Graphics* 43, 4 (2024), 108:1–108:13.

Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. 2024b. Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3763–3772.

Yang Liu, Daxuan Ren, Yijie Ding, Jianmin Zheng, and Fang Deng. 2026a. Learning CAD Modeling Sequences via Projection and Part Awareness. *Advances in Neural Information Processing Systems* 38 (2026), 98243–98275.

Yang Liu, Daxuan Ren, Yijie Ding, Jianmin Zheng, and Fang Deng. 2026b. Learning CAD Modeling Sequences via Projection and Part Awareness. *Advances in Neural Information Processing Systems* 38 (2026), 98243–98275.

Weijian Ma, Shuaiqi Chen, Yunzhong Lou, Xueyang Li, and Xiangdong Zhou. 2024. 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*. 27154–27163.

Weijian Ma, Minyang Xu, Xueyang Li, and Xiangdong Zhou. 2023. 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*. 1766–1776.

Xinzhu Ma, Cheng Wang, Chen Tang, Bin Wang, Shixiang Tang, Yuan Meng, Yunhong Wang, and Di Huang. 2025. Point2Primitive: CAD Reconstruction from Point Cloud by Direct Primitive Prediction. arXiv:2505.02043 [cs.CV] https://arxiv.org/abs/2505.02043

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

Zekun Qi, Runpei Dong, Shaochen Zhang, Haoran Geng, Chunrui Han, Zheng Ge, Li Yi, and Kaisheng Ma. 2024. Shapellm: Universal 3d object understanding for embodied interaction. In *European Conference on Computer Vision*. Springer, 214–238.

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

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

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2018. Csgnet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5515–5523.

Haozhan Shen, Kangjia Zhao, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, Mingwei Zhu, and Jianwei Yin. 2025. Zoomeye: Enhancing multimodal llms with human-like zooming capabilities through tree-based image exploration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 6613–6629.

Baifeng Shi, Boyi Li, Han Cai, Yao Lu, Sifei Liu, Marco Pavone, Jan Kautz, Song Han, Trevor Darrell, Pavlo Molchanov, et al. 2025. Scaling vision pre-training to 4k resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9631–9640.

Yuheng Shi, Xiaohuan Pei, Minjing Dong, and Chang Xu. 2026a. Catching the Details: Self-Distilled RoI Predictors for Fine-Grained MLLM Perception. In *The Fourteenth International Conference on Learning Representations*.

Yuheng Shi, Xiaohuan Pei, Linfeng Wen, Minjing Dong, and Chang Xu. 2026b. Q-Zoom: Query-Aware Adaptive Perception for Efficient Multimodal Large Language Models. *arXiv preprint arXiv:2604.06912* (2026).

Yuan Tang, Xu Han, Xianzhi Li, Qiao Yu, Yixue Hao, Long Hu, and Min Chen. 2024. Minigpt-3d: Efficiently aligning 3d point clouds with large language models using 2d priors. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 6617–6626.

Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. 2022. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11850–11860.

Chunshi Wang, Junliang Ye, Yunhan Yang, Yang Li, Zizhuo Lin, Jun Zhu, Zhuo Chen, Yawei Luo, and Chunchao Guo. 2025c. Part-X-MLLM: Part-aware 3D Multimodal Large Language Model. *arXiv preprint arXiv:2511.13647* (2025).

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv preprint arXiv:2409.12191* (2024).

Ruiyu Wang, Shizhao Sun, Weijian Ma, and Jiang Bian. 2026. CAD-Tokenizer: Towards Text-Based CAD Prototyping via Modality-Specific Tokenization. In *The Fourteenth International Conference on Learning Representations*.

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

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. 2025b. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 7907–7915.

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

Penghao Wu and Saining Xie. 2024. V*: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13084–13094.

Rundi Wu, Chang Xiao, and Changxi Zheng. 2021. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6772–6782.

Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. 2024b. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. *arXiv preprint arXiv:2411.04954* (2024).

Xiang Xu, Pradeep Kumar Jayaraman, Joseph G Lambourne, Karl DD Willis, and Yasutaka Furukawa. 2023. Hierarchical Neural Coding for Controllable CAD Model Generation. *arXiv preprint arXiv:2307.00149* (2023).

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

Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. 2022. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. *arXiv preprint arXiv:2207.04632* (2022).

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

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441* (2023).

Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. 2024. Ferret: Refer and ground anything anywhere at any granularity. In *International Conference on Learning Representations*, Vol. 2024. 57153–57180.

Yang You, Mikaela Angelina Uy, Jiaqi Han, Rahul Thomas, Haotong Zhang, Yi Du, Hansheng Chen, Francis Engelmann, Suya You, and Leonidas Guibas. 2025. Img2cad: Reverse engineering 3d cad models from images through vlm-assisted conditional factorization. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*. 1–12.

Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. 2022. CAPRI-Net: Learning Compact CAD Shapes With Adaptive Primitive Assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11768–11778.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. 2025. Auto-Controlled Image Perception in MLLMs via Visual Perception Tokens. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 21822–21831.

Shilong Zhang, Peize Sun, Shoufa Chen, Min Xiao, Wenqi Shao, Wenwei Zhang, Yu Liu, Kai Chen, and Ping Luo. 2025. Gpt4roi: Instruction tuning large language model on region-of-interest. In *European Conference on Computer Vision*. Springer, 52–70.

Yujia Zhang, Xiaoyang Wu, Yunhan Yang, Xianzhe Fan, Han Li, Yuechen Zhang, Zehao Huang, Naiyan Wang, and Hengshuang Zhao. 2026. Utonia: Toward One Encoder for All Point Clouds. *arXiv preprint arXiv:2603.03283* (2026).

Liangyu Zhong, Fabio Philipp Rosenthal, Joachim Sicking, Fabian Hüger, Thorsten Bagdonat, Hanno Gottschalk, and Leo Schwinn. 2026. Focus: Internal mllm representations for efficient fine-grained visual question answering. *Advances in Neural Information Processing Systems* 38 (2026), 96498–96535.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. *arXiv preprint arXiv:2304.10592* (2023).