CAD-Refiner: A Unified Framework for CAD Generation and Iterative Editing

Supplementary Material

7. Overview

Due to the space limitations of the main paper, we have provided additional results and discussion in this appendix. Specifically, we first provide a comprehensive introduction to the proposed MMCAD dataset in **Section 8**. To further understand CAD-Refiner, more detailed introductions are provided in **Section 9**. Finally, **Section 10** elaborates on the experimental setup and implementation details, while **Section 11** presents the experimental results. Note that qualitative results and prompt templates are presented **at the end of the paper**. In summary, the appendix includes the following key components:

8. MMCAD Dataset Construction Detail

The dataset utilized is derived from DeepCAD [36], a widely adopted benchmark for CAD generation tasks, comprising approximately 178K parametric CAD models with Sketch-Extrusion history, which are represented as a series of sequences consisting of commands c and parameters p as shown in Table 8. A loop is the base unit of the Sketch, beginning with $< SOL >$ followed by a series of curve commands c, which consist of the three most commonly used: $Line(L)$, $Arc(A)$, and $Circle(R)$. The parameters of these commands specify the curve's 2D location in the sketch plane's local frame of reference, whose own position and orientation in 3D will be described shortly in the associated extrusion command. Additionally, extrusion is used to describe how a 2D plane is transformed into a 3D body and how to merge the newly extruded 3D body with the previously created shape by boolean operations (e.g., creating a new body, joining, cutting, or intersecting with the existing body). For details about the definition of these CAD sequences, please refer to [36]. Building upon this collection of approximately 178K parametric CAD models, we construct MMCAD, a new dataset designed to support both CAD generation and editing tasks. MMCAD comprises 80K samples for generation and 20K samples for editing.

Table 8. The specification of CAD sequence.

Commands	Parameters
$< SOL >$	Ø
Line (L)	x, y :line end-point
Arc (A)	x, y : arc end-point α : sweep angle f : counter-clockwise flag
Circle (R)	x, y : center r : radius
Extrude (E)	θ, ϕ, γ:sketch plane orientation p_x, p_y, p_z : sketch plane origin s : scale of associated sketch profile e_1, e_2: extrude distances towards both sides b: boolean type, u: extrude type
$< EOS >$	Ø

Data Pre-processing. We analyze the sequence length distribution of CAD models in DeepCAD, ranging from 4 to 60 across 57 lengths. The cubic-shape category dominates the dataset with approximately 45K instances. These simple and repetitive samples can cause overfitting and limit generalization. To mitigate this issue, we exclude cubic models and those with sequence lengths of ≤ 10, retaining approximately 88K CAD models with more complex geometries.

Image and Text Description Generation. All CAD models are imported from STEP files and rendered using PythonOCC (a Python binding for OpenCASCADE) from a fixed camera pose with a 30° field of view at a resolution of 1024×768 pixels. Shaded rendering is performed using a directional light source positioned at (0, 0.5, −1), with an intensity of 50 and white color; all camera and lighting parameters remain constant across renders. Text descriptions are generated based on the rendered images: initial captions are produced by Qwen-VL-Max and evaluated by GLM-4v-Flash using a structured scoring template. High-scoring captions are retained, while low-scoring ones are manually refined to improve fluency and semantic accuracy.

Instruction Generation. For the generation task, all

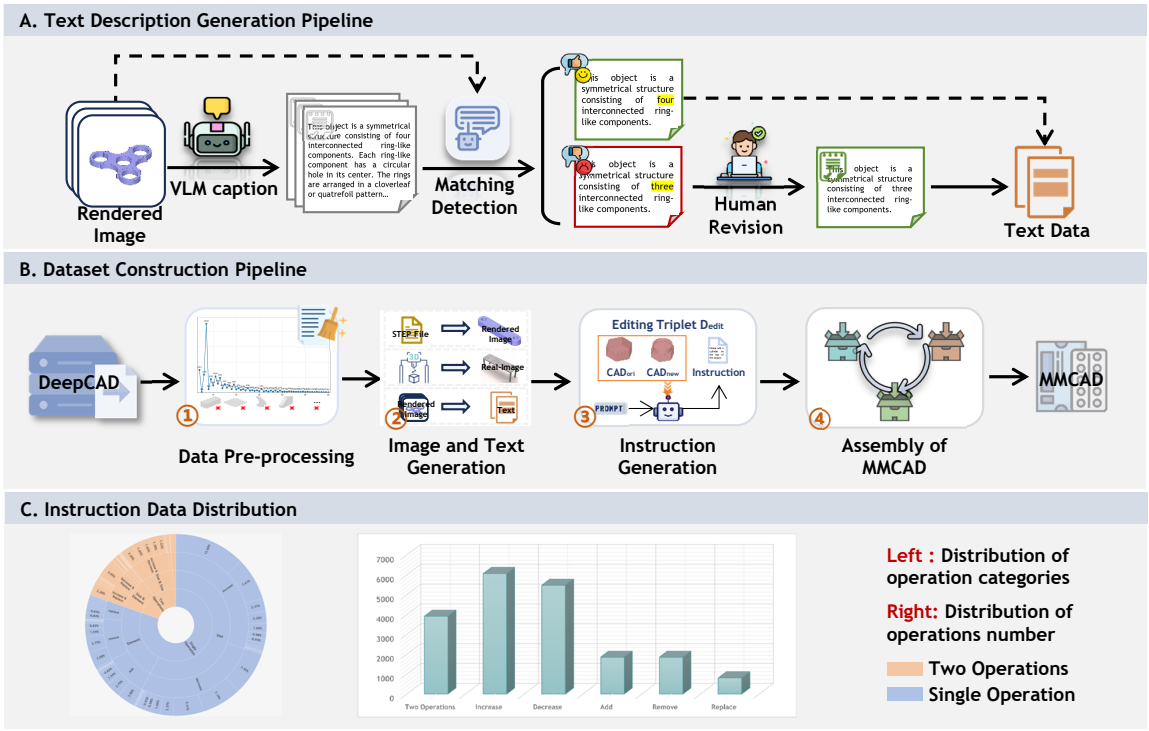


Figure 10. The overview of Dataset Construction. Part (A) illustrates the process of generating text descriptions. Part (B) describes the construction of the MMCAD dataset. Part (C) presents the data distribution within the editing dataset D_{edit}.

textual instructions are standardized as: "Please generate the corresponding CAD object based on the given input." For the editing task, 10,000 CAD models are randomly selected as originals (CAD_{ori}). Following CAD-Editor, geometric variants (CAD_{new}) are generated, and corresponding rendered images and textual descriptions are produced using the same rendering and captioning pipeline. By swapping CAD_{ori} and CAD_{new} we obtain 10,000 editing pairs. Based on the rendered images and textual differences in each pair, Qwen-VL-Max is prompted to generate appropriate editing instructions. Editing operations are categorized into two types: component-level ("add", "delete", "replace"), primarily derived from textual cues, and component-size ("increase," "decrease"), mainly inferred from visual changes. Some instructions involve composite edits that combine both types.

Finally, all modalities are integrated into a unified dataset, which comprises CAD sequences, rendered images, textual descriptions, and editing instructions.

9. Details of CAD-Refiner

9.1. Chain-of-Thought Guidance of CAD Insighter

To enable stable and precise inference of the topological structure graph from free-form user inputs, we design a Chain-of-Thought (CoT) guidance and present the corresponding CoT prompt used by CAD Insighter, as shown in Figures 19 and 20. A one-shot example is further provided in Figure 21 to guide the model in constructing a hierarchical structure tree based on the input conditions. This tree is then utilized to generate a topological structure graph, which serves as a structural prior for CAD model generation. Specifically, the Chain-of-Thought guidance comprises the following steps: (1) Prompt Recognition. Identifies the modality of the input prompt and confirms the text description; (2) Instruction Identification. Classifies the user instruction based on predefined rules; (3) Text Redescription. Generates a refined description of the target CAD model by integrating the user instruction and text description from step (1); (4) Hierarchical Structure Tree Construction. Guides the VLM to construct a hierarchical structure tree for the target CAD model based on the text description from step (3); (5) Pre-order Traversal. Computing inter-node connections directly from the hierarchical structure tree may result in disorganized or incomplete relationships. We design structured instructions for systematic node traversal via a pre-order strategy, yielding an intuitive topological representation and enabling organized extraction of relational dependencies. (6) Adjacency Relation Extraction. Evaluates the adjacency relationships between parent and child nodes based on the traversal results, forming a structured adjacency triplet representation; (7) Structured Output Generation. Systematically organizes the final output according to a predefined structured format.

9.2. CAD Hierarchical Tree Construction

Prior works ([36, 38, 39]) have defined multi-level hierarchical structures within CAD sequence data. Such a hierarchical organization naturally lends itself to a tree-based representation. Specifically, the hierarchical structure of a CAD model, from top to bottom, is typically regarded as: "CAD model", "Sketch-Extrusion" (SE), "Sketch" "Extrude", "Face", "Loop", and "Curve". A CAD model is generally composed of multiple "Sketch-Extrude" operations. Each sketch can be decomposed into one or more 2D planar regions (faces), each bounded by one or more Loops. A Loop is either formed by a single closed curve (e.g., a circle) or a connected sequence of curves (e.g., Line–Line–Line). In this work, we consider three foundational elements of Curves: Line, Arc, and Circle. Based on this, we design CAD Insighter, which is used to construct the hierarchical tree representation of a CAD model from its input, enabling the extraction of the topological structure of the CAD model. To further illustrate the process by which CAD Insighter constructs the hierarchical structure tree and extracts topological information, we present two representative examples in Figure 11.

9.3. Graph Embedding

As previously described, CAD Insighter extracts the fundamental geometric elements and topological structures from user input for each CAD model. To construct a valid input representation for graph networks, we first define the node and edge sets and then standardize the data to map them into feature representations. Specifically, node names (e.g., "Sketch") are mapped to their corresponding digital representations (e.g., "21") based on the Topology Corpus. Meanwhile, the edge set is further processed to construct an adjacency matrix, where the entries denote the weights of the corresponding edges. To standardize the input dimensions, the node features and edge weights are passed through embedding layers, which transform them into vector representations.

9.4. The Encoder of CAD-Refiner

As discussed in the main text, the multimodal encoder consists of a text encoder, a graph encoder, an instruction encoder, and a CAD sequence encoder. Specifically, DistilBERT is employed to encode the text prompt into text feature $e_T \in \mathbb{R}^{N_T \times d_T}$. And instruction encoder maps a scalar instruction type, extracted from the CAD Insighter output, into a high-dimensional embedding space using a learnable non-linear layer. Additionally, GraphGPS is utilized to encode each graph representation into $G^e \in R^{Dim}$. We set N_T to 152, d_T and Dim to 768. Furthermore, the CAD sequence encoder encodes the CAD sequence, which consists of commands C and parameters P, to feature $e_s \in \mathbb{R}^{N_s \times d_E}$:

$$e_s\left(i\right) = e_i^{(C)} + e_i^{(P)} + e_i^{pos}, \tag{3}$$

where $e_i^{(C)} = W_C \delta_i^c$, $e_i^{(P)} = W_P^2 f(W_P^1 \delta_i^p)$, $e_i^{pos} = W_{pos}\delta_i$. Here, $e_i^{(C)}$ represents the command type C, $e_i^{(P)}$ corresponds to the embedding of command parameters P, and e_i^{pos} serves as a positional encoding to record the index of the command S_i within the complete CAD construction sequence. Specifically, $W_C \in \mathbb{R}^{d_E \times 6}$, $W_P^1 \in \mathbb{R}^{d_E \times 257}$, $W_P^2 \in \mathbb{R}^{d_E \times 16 d_E}$, $W_{pos} \in \mathbb{R}^{N_s \times d_E}$ are learnable matrix. Additionally, $\delta_i^c \in \mathbb{R}^6$, $\delta_i^p \in \mathbb{R}^{257 \times 16}$ and $\delta_i \in \mathbb{R}^{N_s}$ are one-hot vectors. The vector $\delta_i^c \in \mathbb{R}^6$ indicates the command type C among six possible command types. Each of the 16 parameters for the command is quantized into an 8-bit integer, and each bit is converted into a one-hot vector of dimension $2^8 + 1 = 257$. These one-hot vectors are then stacked to form the matrix $\delta_i^p \in \mathbb{R}^{257 \times 16}$. The vector $\delta_i \in \mathbb{R}^{N_s}$ is filled with 1 at index i and 0 elsewhere. Furthermore, the function $f\left(\cdot\right)$ flattens the input matrix to a vector. Here, we set the dimension d_E to 256.

9.5. The Decoder of CAD-Refiner

The CAD sequence decoder in CAD-Refiner integrates multi-modal encoded features, including text e_t, CAD e_s, instruction e_{ins}, and graph e_{graph} features, through a sophisticated attention-based mechanism. Initially, the CAD embedding e_s undergoes self-attention processing to capture intra-CAD dependencies. This step is followed by adaptive layer normalization (AdaLayerNorm) to stabilize feature representation. Subsequently, text and graph features are integrated via layer-wise cross-attention. The cross-attention output $A \in R^{n \times d_v}$ is computed as:

$$A = Softmax(\frac{Q_{CAD} {K_X}^T}{\sqrt{d_k}}) V_X, \tag{4}$$

where $Q_{CAD} = e_s W_Q$, $K_X = e_X W_K$, $V_X = e_X W_V$. Here, e_X is either text or graph features, and $W_Q \in R^{d \times d_q}$, $W_K \in R^{d \times d_k}$, $W_V \in R^{d \times d_v}$ are learned projection matrices. This mechanism enables the model to focus on relevant external information effectively. Instruction features e_{ins} are recurrently injected to guide the refinement process. Following the cross-attention step, a dropout layer, layer normalization, and a feed-forward network are applied sequentially to the output A. This series of operations enhances the robustness and expressiveness of the feature integration. The transformer decoder block, comprising the aforementioned components, is repeated N times, facilitating deeper integration and refinement of the condition prompt and CAD tokens. Finally, the features output by the decoder are passed through a linear layer to predict the sequences $\hat{S} = \left[\hat{S_1}, \cdots, \hat{S_{N_s}}\right]$, which consist of commands

Table 9. Token costs during the text construction stage.

Model	Input	Output	Total
Text generation	1383	136	1519
Text scoring	1567	119	1686

Table 10. Token costs during the graph generation stage.

Model	Input	Output	Total
Text Input	3478	915	4393
Image Input	4436	922	5358
Multimodal Input	4594	919	5513

Table 11. The ablation study on CAD Editing.

Model	Acc_c ↑	Acc_p ↑	VR ↑	MMD ↓	JSD ↓
Decoder(text)	87.27	79.34	72.48	3.37	10.32
Decoder(graph)	87.34	80.86	68.99	3.36	9.69
w/o CoT	85.81	76.79	60.63	3.04	8.35
w/o Checker	87.91	81.58	68.39	3.42	10.22
CAD-Refiner	88.91	82.55	70.69	3.30	9.42

Table 12. More completion result for both image and text.

Model	Acc_c ↑	Acc_p ↑	VR ↑	MMD ↓	JSD ↓
90% CAD	99.35	96.11	87.40	2.15	4.44
80% CAD	98.69	92.83	79.74	2.16	4.29
70% CAD	97.92	89.49	74.37	2.28	4.47
60% CAD	97.05	86.01	69.14	2.38	4.87
50% CAD	95.90	82.27	65.07	2.43	5.50
40% CAD	94.54	78.54	61.34	2.59	5.84
30% CAD	92.87	74.53	58.60	2.63	7.07
20% CAD	90.66	70.16	55.96	2.90	9.70
10% CAD	88.07	65.25	54.36	2.94	11.92

$\hat{C}$ and parameters $\hat{P}$ for each token in the sequences. In this work, we set the number of decoder layers to $N = 4$.

10. More Details of Experiments

10.1. Implementation Details

Details for CAD-Refiner. For the generation task, we randomly select 50K samples from an initial pool of 80K and split them into training and test sets in a 6:4 ratio. For the editing task, we apply the same split to the collected 20K edit pairs. To evaluate the generalization capability of our method under real-world conditions, we reserve all 2K collected real-scenario samples as a held-out test set. To avoid potential advantages arising from mixed training, we train two separate models independently on the generation and editing datasets, respectively, ensuring a fair comparison. Experiments are conducted on an NVIDIA RTX 4090 GPU, with a batch size of 64 and a total of 200 training epochs. The Adam optimizer is employed with a fixed learning rate of 0.001. A warm-up phase consisting of 20 training epochs is implemented, during which the model is first trained on the full set of CAD sequences to acquire stable representations of the data distribution and underlying semantic structure.

Table 13. More completion result for text input.

Model	Acc_c ↑	Acc_p ↑	VR ↑	MMD ↓	JSD ↓
90% CAD	99.43	94.86	87.57	2.05	3.88
80% CAD	98.78	91.66	69.15	2.16	3.85
70% CAD	98.07	88.27	79.90	2.36	4.61
60% CAD	97.07	84.87	74.37	2.36	4.68
50% CAD	95.97	81.16	46.46	2.47	4.73
40% CAD	94.49	77.39	60.91	2.44	5.93
30% CAD	92.71	73.35	41.80	2.64	6.92
20% CAD	90.39	68.92	55.04	2.81	9.12
10% CAD	87.58	64.07	38.70	2.89	11.20

Table 14. More completion result for image input.

Model	Acc_c ↑	Acc_p ↑	VR ↑	MMD ↓	JSD ↓
90% CAD	99.48	95.30	88.17	2.09	4.32
80% CAD	98.68	91.12	78.78	2.30	4.57
70% CAD	97.93	87.71	73.26	2.25	4.37
60% CAD	96.97	84.19	68.44	2.36	4.46
50% CAD	95.70	80.41	64.15	2.44	5.09
40% CAD	94.50	77.30	61.36	2.53	5.47
30% CAD	92.80	73.63	58.07	2.63	7.01
20% CAD	90.36	69.04	55.17	2.79	8.88
10% CAD	87.57	64.09	53.20	3.01	12.39

Details for LLM-based method. To evaluate the effectiveness of our method, Qwen2.57B and LLaMA3.1-8B are selected as baseline models, both of which use a maximum sequence length of 384. For efficient fine-tuning, we adopt Low-Rank Adaptation (LoRA) [9] with hyperparameters $r = 8$, $\alpha = 32$ and $dropout = 0.1$. During training, we set both the batch size and gradient accumulation steps to 4, with a learning rate of 1×10^{-4}. The model is trained for a total of 3 epochs, and checkpoints are saved every 100 training steps. Similar to our work, all experiments are conducted on the same GPU to ensure a fair comparison.

10.2. CAD representation for LLM-based method

The CAD sequence, composed of commands and parameters, has emerged as a widely adopted representation to

capture the design history of CAD models. As shown in Table 8, this representation format comprises a diverse set of commands and a large number of associated parameters. It is typically encoded as an $N \times 17$ dimensional matrix, where N denotes the number of sequential operations. However, due to its complexity and structural limitations, the current representation does not align with the prompt paradigm required for fine-tuning large language models.

To this end, we further transform the current data representation into a textual sequence format. Specifically, we first translate the numeric commands into semantically meaningful terms according to the following mapping rules: {0: Line, 1: Arc, 2: Circle, 3: EOS, 4: SOL, 5: Extrude}. For the parameter portion, we retain only the non-zero elements to compress the two-dimensional matrix into a one-dimensional vector. Based on the above rules, we construct prompt data suitable for fine-tuning large language models in the form of instruction-response pairs.

11. More Experiment Results

11.1. Token Cost

In this paper, we leverage the VLMs through API integration to facilitate dataset construction and generate graph-structured representations. This prompt engineering technique does not require fine-tuning of VLMs, thus incurring negligible computational resource overhead. Since API calls for certain LLMs incur costs based on token usage, we further report the token consumption incurred during dataset construction and graph structure building.

During the dataset construction phase, we primarily utilize VLMs to generate textual descriptions of CAD objects and to evaluate the correctness of these descriptions by scoring them. The token consumption is summarized in Table 9. Meanwhile, we provide the token consumption in the graph generation stage in Table 10. The experimental results demonstrate that image-based input incurs higher token consumption compared to text-based input. It should be noted that the token cost becomes negligible when employing certain freely available models.

11.2. More Ablation Study

Ablation Study on CAD Editing Task In the main text, we present ablation study results on conditional guided CAD generation. To further validate the effectiveness of our proposed contributions, we present ablation results on the editing task in Table 11. The experimental results show that our method achieves the best performance when incorporating both CoT and the CAD Checker, demonstrating their effectiveness.

11.3. More CAD Completion Results

We present more challenging completion results in Tables 12 to 14, where the proportion of provided CAD content ranges from 10% to 90%. Specifically, "60% CAD" indicates that 60% of the input tokens are real CAD tokens, while the remaining 40% are masked. As shown in the tables, performance improves consistently as more CAD content is provided, under both text and/or image settings. Notably, our method maintains strong performance across all conditions, preserving the overall quality of the completed models. Particularly on parameter prediction, a more challenging target, the accuracy exceeds 64% in all settings, demonstrating the effectiveness of our approach. These results show that CAD-Refiner not only supports generation and editing tasks within a unified framework but also generalizes well to practical CAD completion scenarios.

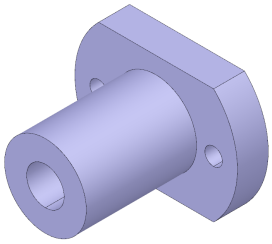
11.4. Qualitative results

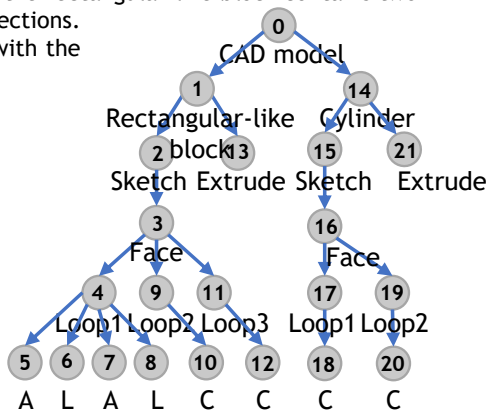
To comprehensively evaluate the performance of our approach, we provide qualitative results across multiple tasks. Specifically, we present the results of the "generate-then-edit" pipeline in Figures 13 to 16. Additionally, we present further visualization results of CAD construction sequence generation based on multi-modal prompts in Figures 17 and 18. The experimental results show that our method achieves precise generation not only for basic simple models but also for complex CAD models, such as chairs and bookcases.

11.5. Examples of Failure Cases

Despite the promising results achieved by CAD-Refiner, it still faces limitations in predicting key "Extrude" parameters such as sketch origin and direction. As shown in Figure 12, some generation results exhibit notable failures. Further analysis identifies two main causes: (1) Sketch position parameters are represented as continuous values without explicit constraints, making them difficult to predict accurately. (2) Complex CAD models often consist of multiple components whose positional parameters should be correlated rather than independent. Ignoring these relationships during prediction leads to positional misalignments. Therefore, in future work, we plan to model the correlations among CAD sequence parameters to enable more precise and controllable generation and editing.

Text Description: This object is a combined structure consisting of a cylinder and a rectangular-like block. The cylinder is positioned horizontally, and the rectangular-like block is attached perpendicularly to one end of the cylinder. The cylindrical contains one circular hole located axially through its center. The rectangular -like block has a semicircular arc at both short sides, which connects with the long sides to form a whole. And the rectangular-like block contains two circular holes, near the arcs located in two separate sections. The cylinder and the rectangular block are integrated, with the block extending outward from the cylinder's end.

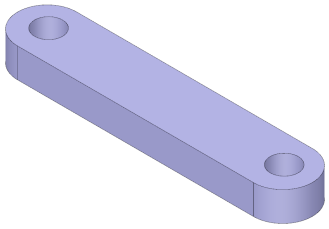


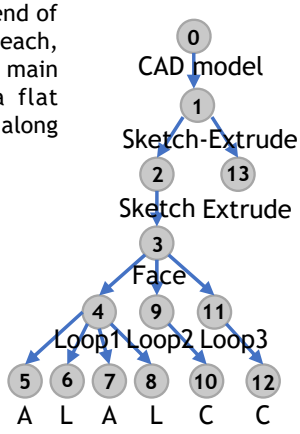


Preorder traversal Result:
['0':'CAD model'; '1':'Rectangular-lik block'; '2':'Sketch'; '3':'Face'; '4':'Loop1'; '5':'Arc'; '6':'Line'; '7':'Arc'; '8':'Line'; '9':'Loop2'; '10':'Cicle'; '11':'Loop3'; '12':'Circle'; '13':'Extrude'; '14':'Clinder'; '15':'Sketch'; '16':'Face'; '17':'Loop1'; '18':'Circle'; '19':'Loop2'; '20':'Circle'; '21':'Extrude']

Adjacency Triplets:
[[0,1,1],[0,14,1],[1,2,1],[1,13,5],[2,3,1],[3,4,1],[3,9,1],[3,11,1],[4,5,3],[4,6,2],[4,7,3],[4,7,3],[9,10,4],[11,12,4],[14,15,1],[14,21,5],[15,16,1],[16,17,1],[16,19,1],[17,18,4],[19,20,4]]

Text Description:
This object is a rectangular bar structure with rounded ends. Each end of the bar has a circular shape. Both circular ends have one hole each, positioned near the center of their respective circular faces. The main body of the bar, which connects the two circular ends, is a flat rectangular section with no holes. The overall shape is symmetrical along its length.





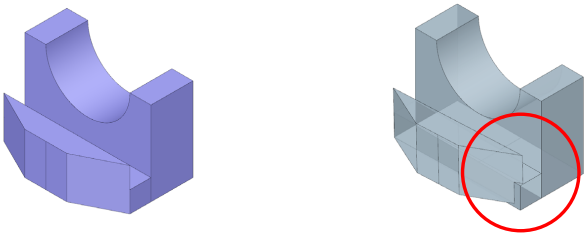
Preorder traversal Result:
['0':'CAD model'; '1':'Sketch-Extrude'; '2':'Sketch'; '3':'Face'; '4':'Loop1'; '5':'Arc'; '6':'Line'; '7':'Arc'; '8':'Line'; '9':'Loop2'; '10':'Cicle'; '11':'Loop3'; '12':'Circle'; '13':'Extrude';]

Adjacency Triplets:
[[0,1,1],[1,2,1],[1,13,5],[2,3,1],[3,4,1],[3,9,1],[3,11,1],[4,5,3],[4,6,2],[4,7,3],[4,7,3],[9,10,4],[11,12,4]]

Figure 11. The example of constructing hierarchical structure tree

This object is a composite structure consisting of three main components: two ring-shaped ends and a central rectangular bar. Left Ring Component: This component is ring-shaped, featuring a circular hole in the center. It is positioned at one end of the rectangular bar. Central Rectangular Bar: This component is a rectangular bar that connects the two ring-shaped ends. It has a smaller circular hole located near its center, off to one side. Right Ring Component: Similar to the left ring component, this part is also ring-shaped with a circular hole in the center. It is positioned at the opposite end of the rectangular bar. The interrelationship between these components is such that the rectangular bar serves as the connecting element, with each ring component attached to either end of the bar. The smaller circular hole in the bar is not centrally located but is instead closer to one of the ring components.

This object is a composite structure consisting of several interconnected components. Main Body: The main body of the object has a U-shaped profile with a curved top surface. This component does not appear to have any holes. Left Extension: Extending from the left side of the main body is a slanted, triangular section that connects to a vertical rectangular section. This part also does not have any holes. Right Extension: On the right side of the main body, there is a rectangular section that extends vertically. Similar to the other parts, this section does not have any holes. Base Section**: Below the main body, there is a base section that includes a flat surface and some angled surfaces. This base section also does not have any holes. In summary, the object is a complex structure with a U-shaped main body, a slanted left extension, a vertical right extension, and a base section. None of the components have any holes.



This object is a composite structure consisting of three main components: two ring-shaped ends and a central rectangular bar. Left Ring Component: This component is ring-shaped, featuring a circular hole in the center. It is positioned at one end of the rectangular bar. Central Rectangular Bar: This component is a rectangular bar that connects the two ring-shaped ends. It has a smaller circular hole located near its center, off to one side. Right Ring Component: Similar to the left ring component, this part is also ring-shaped with a circular hole in the center. It is positioned at the opposite end of the rectangular bar. The interrelationship between these components is such that the rectangular bar serves as the connecting element, with each ring component attached to either end of the bar. The smaller circular hole in the bar is not centrally located but is instead closer to one of the ring components.

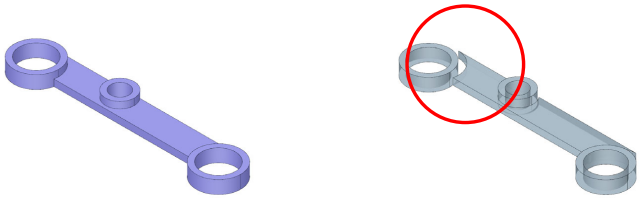
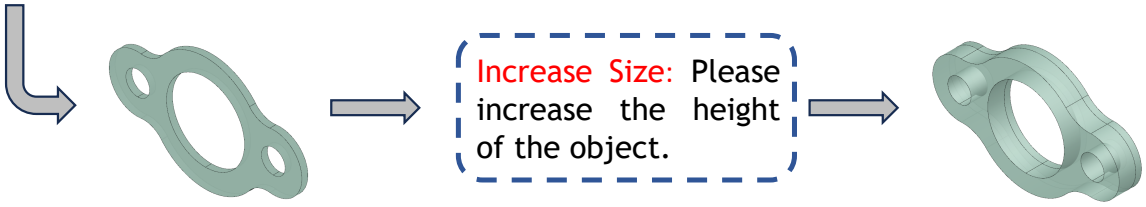
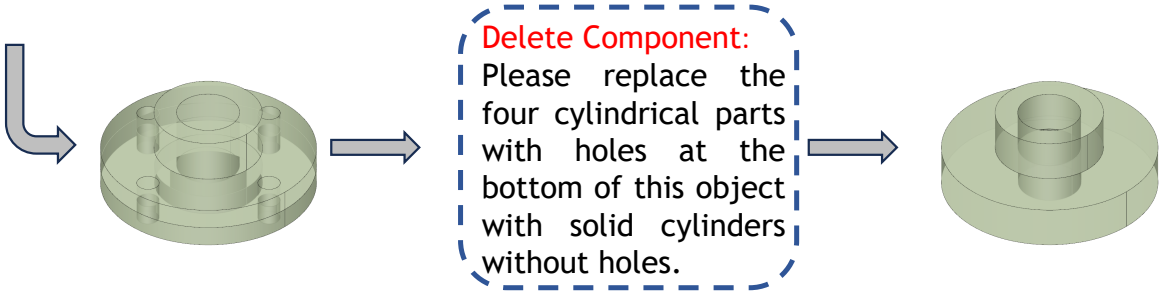


Figure 12. Failure cases of CAD-Refiner. The prediction of location parameters remains a challenging issue.

Generation: This object is an irregular ring-like structure. It consists of a central large oval opening surrounded by a thickened frame. The frame has two protruding sections on opposite sides, each with a circular hole near the edge.



Generation: This object is a composite structure consisting of two main components. Base Component: The base is a flat, circular disc. It has four circular holes positioned symmetrically around its perimeter. And top Component: The top part is a smaller, raised cylindrical section that sits centrally on the base. This cylinder has one central circular hole. The top cylindrical component is centered on the base disc, and both parts are aligned coaxially.



Generation: This object is a rectangular plate structure. It has two circular holes. The first circular hole is located near one of the shorter edges, and the second circular hole is positioned closer to the opposite shorter edge. Both holes are not in the center but are offset towards their respective edges. The plate itself is flat with no additional protrusions or indentations apart from the holes. The overall shape is a simple rectangular form with these two distinct circular openings.

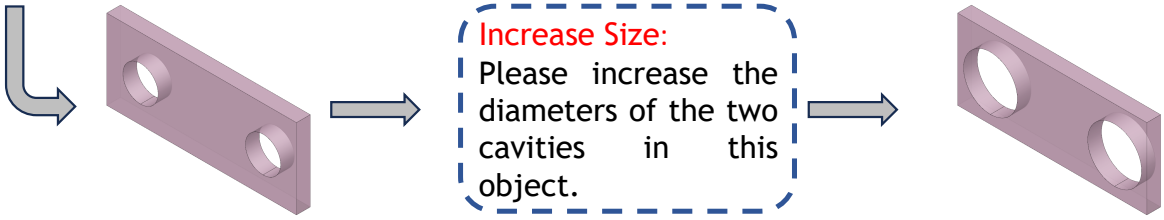
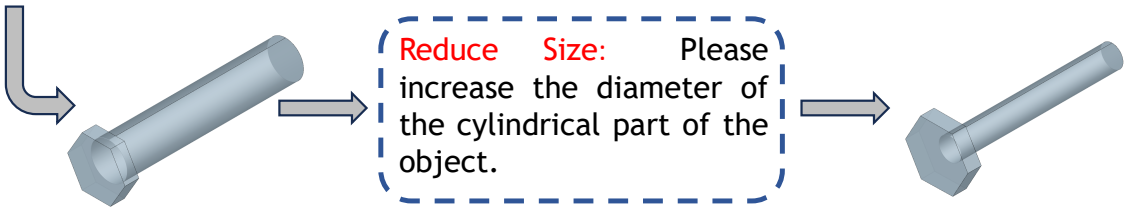
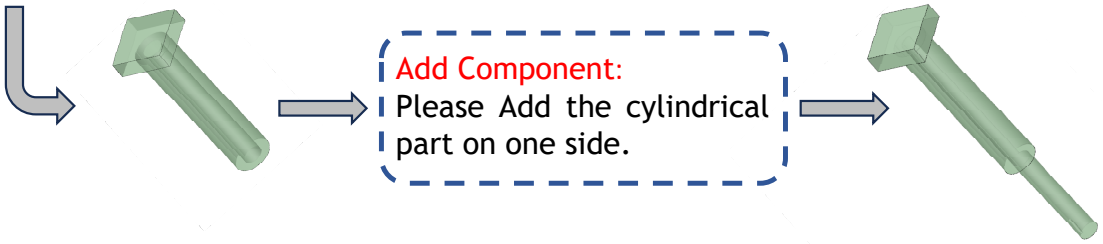


Figure 13. The results of the text-to-CAD generation and editing tasks.

Generation: This object is a combined structure consisting of a cylinder and a hexagonal prism. The cylinder extends from one face of the hexagonal prism. Hexagonal Prism: This component has six flat faces meeting at sharp edges, forming a hexagonal shape. It has no holes. Cylinder: This component is attached to one of the hexagonal faces of the prism. It is a solid cylinder with no holes. The interrelationship between the two components is such that the cylinder is aligned along the axis of one of the hexagonal faces, extending outward from it. Neither the hexagonal prism nor the cylinder contains any holes.



Generation: This is a combined structure consisting of a cylinder and a cuboid. The cylinder is positioned below, and the cuboid is placed on top of the cylinder. Neither the cylinder nor the cuboid has any holes. The cuboid is centered on the top surface of the cylinder, forming a T-shaped profile when viewed from certain angles. Both components are solid with no visible openings or perforations.



Generation: This object is a multi-layer mechanical component with a stepped design, consisting of several parts. The bottom part consists of three rectangles. There is a hole in the middle of each of the two rectangles on the left and right sides, and there is also a hole in the small rectangle near the short edges on both sides. These three rectangles are joined together to form a base, on which there is a rectangular block.

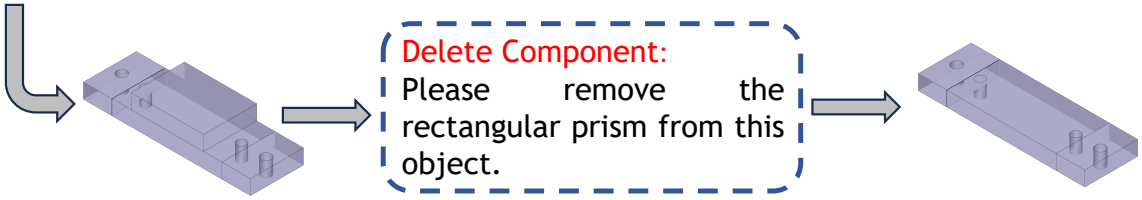


Figure 14. The results of the text-to-CAD generation and editing tasks.

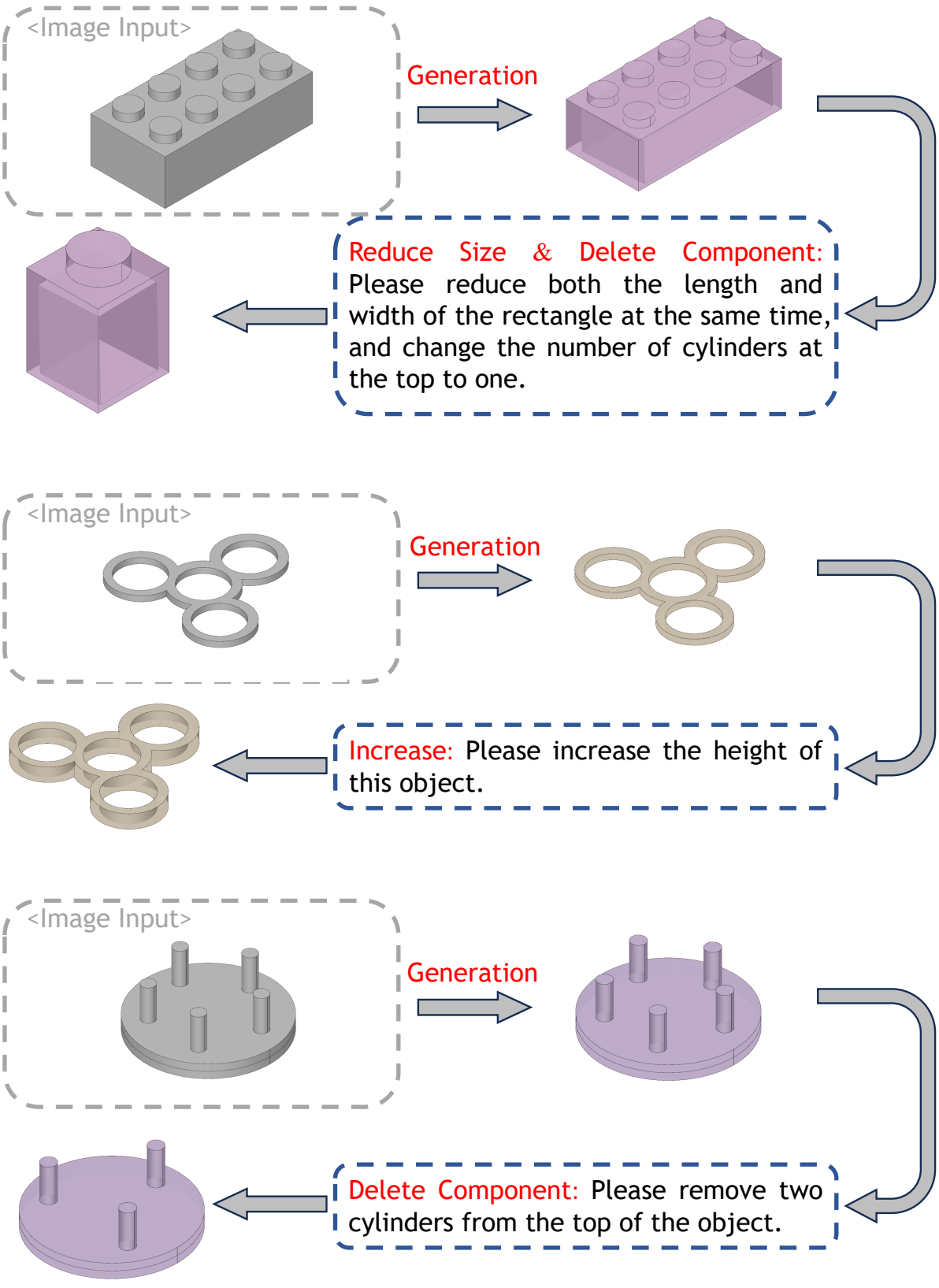


Figure 15. The results of the image-to-CAD generation and editing tasks.

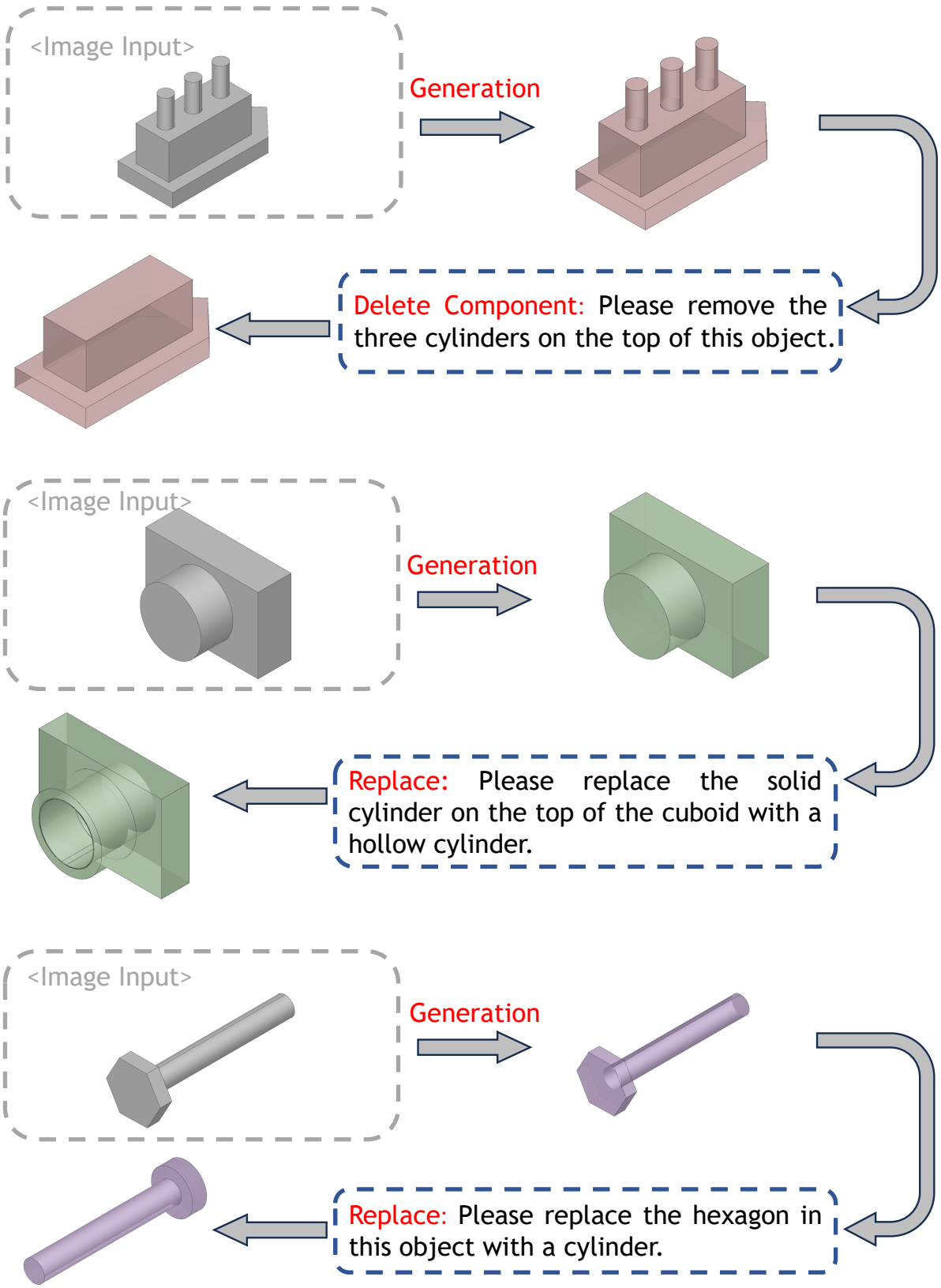


Figure 16. The results of the image-to-CAD generation and editing tasks.

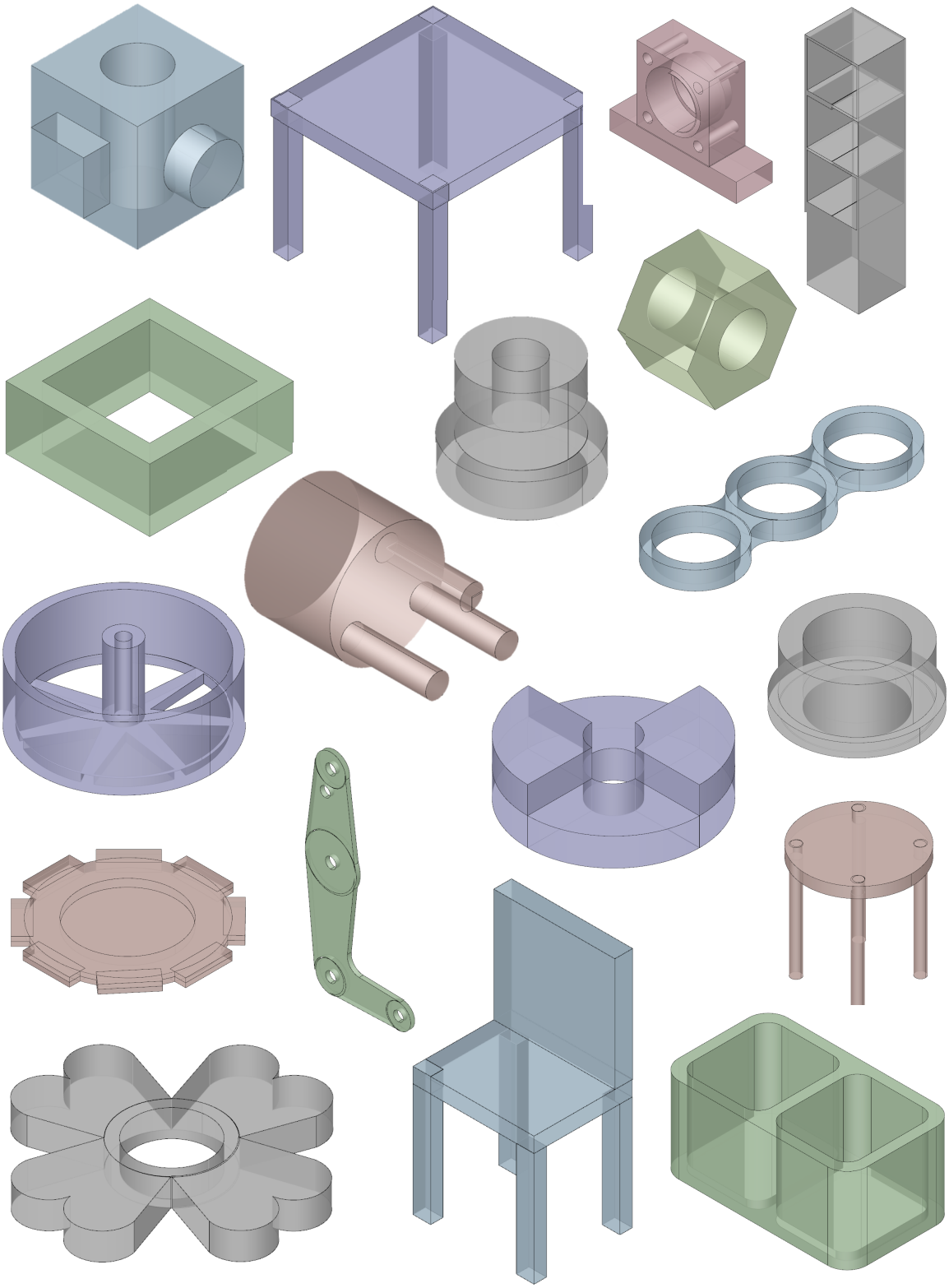


Figure 17. More conditional CAD generation results from CAD-Refiner.

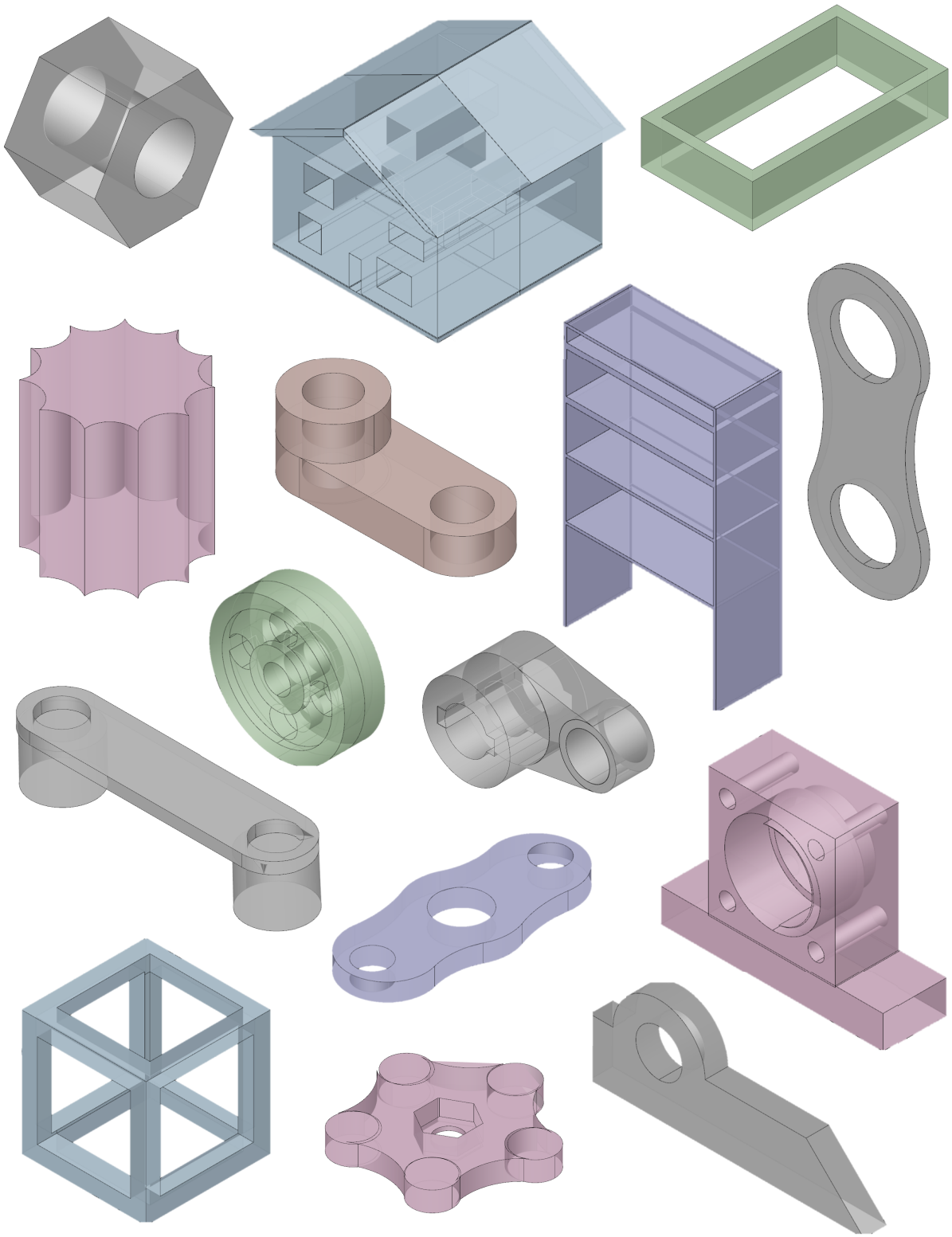


Figure 18. More conditional CAD generation results from CAD-Refiner.

Prompt Template for CAD Insighter (1)

<Background Information>
Computer-Aided Design (CAD) serve as a cornerstone in modern industrial and engineering disciplines, permeating all stages of the design, manufacturing, collaboration, and optimization processes. As an experienced CAD model designer with specialized expertise, you possess comprehensive knowledge of object nomenclature, geometric parametrization, and the strategic implementation of CAD methodologies across multidisciplinary workflows. You will be provided a description of a CAD model, which may consist of a reference image description, a text description, or both, along with an operation instruction from the user regarding this CAD object. Based on these two parts of information, you are required to complete the following three tasks:
</Background Information>

<Task Description and Rules Explanation>
Task 1: Instruction Type Extraction. Based on the user's instruction, identify and extract the instruction type. All available instruction class indices and names must strictly follow this instruction class list: [0: 'Generate CAD Object Based on Description', 1: 'Add CAD Model Component Based on Instruction', 2: 'Remove CAD Model Component Based on Instruction', 3: 'Increase the size of a CAD model component based on the instruction', 4: 'Decrease the size of a CAD model component based on the instruction', 5: 'Replace CAD Model Component Based on Instruction'].
Task 2: Re-Description of Target CAD Model. Using the original CAD model description and the user's operation instruction, generate a revised textual description of the target CAD model that reflects the expected outcome after applying the instruction. **Notes:** Do **not** include any information related to material, color, texture, or visual appearance. Avoid using prohibited terms such as: "blue", "shadow", "transparent", "metal", "plastic", "image", "black", "grey", "CAD model", "abstract", "orange", "purple", "golden", "green". Example bad descriptions: "CAD model", "Metal object".
Task 3: Adjacency Relation Extraction. It is known that a CAD model has a hierarchical structure from top to bottom, composed of multiple Sketch–Extrude pairs (each pair corresponds to one CAD component). Each Sketch–Extrude pair consists of a Sketch and an Extrude part. Each Sketch contains multiple Faces, each Face consists of multiple Loops, and each Loop is formed by multiple Curves. Among them, Curve is the base element of a CAD model, and commonly used types include Line, Arc, and Circle. Based on the revised text description of the target CAD model obtained in Task 2, generate the list of adjacency triplet representing adjacency matrix Adj. The value at row *i* and column *j* in the adjacency matrix Adj represents the adjacency relationship between node *i* and node *j*. The rules for constructing the non-zero element list are as follows: If there is an edge between node *i* and node *j*: Add [i , j ,1] and [j, i, 1]; If there is an edge between *i* and *j*, and *j* is a Line: Add [i , j ,2] and [j, i, 2]; If there is an edge between *i* and *j*, and *j* is an Arc: Add [i , j ,3] and [j, i, 3]; If there is an edge between *i* and *j*, and *j* is a Circle: Add [i , j ,4] and [j, i, 4]; If there is an edge between *i* and *j*, and *j* is an Extrude: Add [i , j ,5] and [j, i, 5].
Notes: When node *j* is an Extrude, make sure to include both[i , j ,5] and [j, i, 5] in the list of non-zero elements.
</Task Description and Rules Explanation>

<Multimodal Input>
Input Reference Image: {image_path}
Input Text Description: {caption}
Input Instruction: {instruction}
</Multimodal Input>

You can derive your answer following these steps:
<Chain-of-Thought Guidance>
Step 1: Prompt Recognition
If the input consists of **[Input Reference Image, Input Text Description, Input Instruction]**:
- Generate a textual description (**Generated Text Description**) from the **Input Reference Image**.
- Revise this generated description in combination with the **Input Text Description** to obtain an updated **Revised Text Description**, and use this **Revised Text Description** to replace the original **Input Text Description**.
If the input consists of **[Input Reference Image, Input Instruction]**:
- Generate a textual description (**Generated Text Description**) from the **Input Reference Image**.
- Use this **Generated Text Description** directly as the **Input Text Description**.
If the input consists of **[Input Text Description, Input Instruction]**:
- No modification is required.
Step 2: Instruction Identification
Based on the **Input Instruction**, determine the instruction class.
All available instruction classes must be selected from the instruction class list.
Step 3: Text Redescription
Using the current **Input Text Description** and the **Input Instruction**, generate a revised textual description of the CAD model that reflects the user's intended outcome after applying the instruction. This output should be referred to as **Text Redescription**.

Figure 19. The prompt templates for CAD Insighter.

Prompt Template for CAD Insighter (2)

Step 4: Construct the Hierarchical Structure Tree of the CAD Model
Based on the **Text Redescription** obtained in Step 3, represent the CAD model using the **Hierarchical Structure Tree** described in Task 3. **Note:** 1. The CAD model consists of multiple Sketch–Extrude pairs, each representing one CAD component; 2. Each Sketch contains multiple Faces, each Face contains multiple Loops , and each Loop is composed of multiple Curves (e.g., Line, Arc, Circle). 3. Only the structural hierarchy needs to be represented, no assumptions or values related to Extrude operations or other parameters are required. 4. CAD components should be placed at the same level as their corresponding Sketch–Extrude pair, they should not form a separate hierarchy level.
Step 5: Pre-order Traversal
Perform a pre-order traversal on the **Hierarchical Structure Tree** constructed in Step 4, and assign the traversal order as the ** unique index ** for each node in the tree.
Step 6: Adjacency Relation Extraction
Based on the **Hierarchical Structure Tree** and the ** unique index **, evaluates the adjacency relationships between parent and child, forming a structured adjacency triplet representation.
Step 7: Structured Output Generation
Finally, organize all results into a structured output format according to the specified requirements.
</Chain-of-Thought Guidance>

<Example>
...
</Example >

You may follow the template provided between the <Example> and <\Example> tags for your thinking process. Then output the content between <Output> and </Output> tags, excluding the <Output> and </Output> tags themselves. Please ensure all necessary information is included and the format is strictly followed. **Note:** Do not output the full content of the "Adjacency Matrix", and do not use ellipsis (...) to omit content.

<Output>
Text Redescription:
Instruction Class:
Pre-order Traversal :
Adjacency Matrix:
</Output>

Figure 20. The prompt templates for CAD Insighter.

One-shot example for CAD Insighter

<One-shot example>
Input Reference Image: { }
Input Text Description: { "This is an object composed of a cylinder on top and a hexagonal prism at the bottom."}
Input Instruction: { "Please replace the cylinder above with a cube." }

** Step 1: Prompt Recognition**
Input Text Description: "This is an object composed of a cylinder on top and a hexagonal prism at the bottom."
Input Instruction: "Please replace the cylinder above with a cube."
Step 2: Instruction Identification
Instruction Class: 5
Step 3: Text Redescription
Text Redescription: "This is an object composed of a cube on top and a hexagonal prism at the bottom"
Step 4: Construct the Hierarchical Structure Tree of the CAD Model
CAD model
- Sketch-Extrude 1 (Cube)
  - Sketch
    - Face1
      - Loop1
        - Curve1: Line
        - Curve2: Line
        - Curve3: Line
        - Curve4: Line
  - Extrude
- Sketch-Extrude 2 (Hexagonal Prism)
  - Sketch
    - Face1
      - Loop1
        - Curve1: Line
        - Curve2: Line
        - Curve3: Line
        - Curve4: Line
        - Curve5: Line
        - Curve6: Line
  - Extrude

Step 5: Pre-order Traversal
The pre-order traversal sequence of the hierarchical structure tree is as follows:
Pre-order Traversal: ["0": "CAD model", "1": "Sketch-Extrude 1(Cube)", "2": "Sketch", "3": "Face1", "4": "Loop1", "5": "Curve1: Line", "6": "Curve2: Line", "7": "Curve3: Line", "8": "Curve4: Line", "9": "Extrude", "10": "Sketch-Extrude 2 (Hexagonal Prism)", "11": "Sketch", "12": "Face1", "13": "Loop1", "14": "Curve1: Line", "15": "Curve2: Line", "16": "Curve3: Line", "17": "Curve4: Line", "18": "Curve5: Line", "19": "Curve6: Line", "20": "Extrude"]
Step 6: Adjacency Relation Extraction
Based on the adjacency rules and the hierarchical structure, the non-zero elements in the adjacency matrix are:
Structured Adjacency Triplet**:** [[0, 1, 1], [0, 10, 1], [1, 0, 1], [1, 2, 1], [1, 9, 5], [2, 1, 1], [2, 3, 1], [3, 2, 1], [3, 4, 1], [4, 3, 1], [4, 5, 2], [4, 6, 2], [4, 7, 2], [4, 8, 2], [5, 4, 2], [6, 4, 2], [7, 4, 5], [8, 4, 2], [9, 1, 5], [10, 0, 1], [10, 11, 1], [10, 20, 1], [11, 10, 1], [11, 12, 1], [12, 11, 1], [12, 13, 1], [13, 12, 1], [13, 14, 2], [13, 15, 2], [13, 16, 2], [13, 17, 2], [13, 18, 2], [13, 19, 2], [14, 13, 2], [15, 13, 2], [16, 13, 2], [17, 13, 2], [18, 13, 2], [19, 13, 2], [20, 10, 5]]
Step 7: Structured Output Generation
Text Redescription: "This is an object composed of a cube on top and a hexagonal prism at the bottom."
Instruction Class: 5
Pre-order Traversal: ["0": "CAD model", "1": "Sketch-Extrude 1(Cube)", "2": "Sketch", "3": "Face1", "4": "Loop1", "5": "Curve1: Line", "6": "Curve2: Line", "7": "Curve3: Line", "8": "Curve4: Line", "9": "Extrude", "10": "Sketch-Extrude 2 (Hexagonal Prism)", "11": "Sketch", "12": "Face1", "13": "Loop1", "14": "Curve1: Line", "15": "Curve2: Line", "16": "Curve3: Line", "17": "Curve4: Line", "18": "Curve5: Line", "19": "Curve6: Line", "20": "Extrude"]
Structured Adjacency Triplet: [[0, 1, 1], [0, 10, 1], [1, 0, 1], [1, 2, 1], [1, 9, 5], [2, 1, 1], [2, 3, 1], [3, 2, 1], [3, 4, 1], [4, 3, 1], [4, 5, 2], [4, 6, 2], [4, 7, 2], [4, 8, 2], [5, 4, 2], [6, 4, 2], [7, 4, 5], [8, 4, 2], [9, 1, 5], [10, 0, 1], [10, 11, 1], [10, 20, 1], [11, 10, 1], [11, 12, 1], [12, 11, 1], [12, 13, 1], [13, 12, 1], [13, 14, 2], [13, 15, 2], [13, 16, 2], [13, 17, 2], [13, 18, 2], [13, 19, 2], [14, 13, 2], [15, 13, 2], [16, 13, 2], [17, 13, 2], [18, 13, 2], [19, 13, 2], [20, 10, 5]]

</One-shot example>

Figure 21. One-shot CoT prompt template for for CAD Insighter.

Prompt Template for Scoring Text Descriptions

Please first understand the given image.Then, you need to verify whether the CAD model is consistent with the following description in terms of its components, the relationships between the components, whether each component has holes, the shape of the holes, and the number of holes.

Text description: {batch_data[0]["text_description"]}

I will follow the three evaluation criteria as shown below for you:

Poor (Bad):
The overall structure is incorrect, the description of holes is disorganized, and the decomposition of complex objects is unclear. The main holes are not identified, and it is seriously inconsistent with the model.
Average (Fair):
The overall structure is basically correct, and the main defects can be identified. However, there are errors in the decomposition of complex objects, such as incorrect relationships between components. In terms of details, the identification of holes on the main body or components is incorrect.
Good (Good):
The overall structure is accurate, the description of holes is clear, the decomposition of complex objects and their relationships are basically correct, and the details are relatively complete.

First, you need to output your score for the description of this text.
Then output your detailed text description of the object. Just output your description of the object. No need to provide a detailed analysis of the given text description.

Don't make statements like "Overall, the structure of the object fits the description provided very well.

For example:
1. Score : poor or average or good.
2. Detailed Description: The object is...

Figure 22. The prompt template for scoring textual descriptions.