

Overview of Supplementary Material

This supplementary material provides additional details in support of our main paper, *CAD-Coder: Text-to-CAD Generation with Chain-of-Thought and Geometric Reward*. The contents are organized as follows:

- In Section A, we describe the hardware and software configurations, training durations for different stages, and the Chamfer Distance (CD) evaluation protocol.
- In Section B, we show that using only Chamfer Distance as reward leads to training failure, emphasizing the importance of multi-faceted reward design.
- In Section C, we present a detailed breakdown of the CAD generation process, including the user prompt and CoT prompt, CoT reasoning steps, and the final CadQuery output.
- In Section D, we provide additional qualitative comparisons between different methods.
- In Section E, we present several examples to show that our CAD-Coder supports CAD editing.
- In Section F, we analyze failure cases where our method underperforms.

A Additional Implementation Details

All experiments were executed on a cluster equipped with 8 NVIDIA A800 (80GB) GPUs. The SFT stage was trained for 7 hours, while the GRPO stage required 146 hours, both utilizing distributed training with standard data parallelism techniques, facilitated by DeepSpeed and Ray. For efficient model inference, we employed vLLM, and used CadQuery (version 2.3.1) for CAD script execution and validation.

Chamfer Distance (CD) was computed using the same implementation as in Text2CAD [13], ensuring a fair and reproducible comparison. The CD calculation relies on the normalization of the generated 3D mesh models, which is critical for consistent metric evaluation. While CAD-Translator [16] and CAD-LLaMA [15] address similar tasks, their implementations are not open-sourced, and the details of their normalization procedures remain unclear. As a result, although their papers report CD scores, the values differ by an order of magnitude compared to those reported by Text2CAD, making direct comparison infeasible.

B Extended Ablation Studies

To further analyze the impact of our proposed reward design, we conducted an ablation study by disabling all auxiliary components and retaining only the Chamfer Distance (CD) as the reward function during the GRPO training phase. All other settings, including model architecture, optimization strategy, and data pipeline, remained unchanged.

However, as training progressed, we observed that the model began to generate invalid CadQuery code after approximately 200 steps. This led to frequent failures during the reward evaluation stage, as the CD computation relies on successful execution of the generated scripts to produce mesh outputs. Once invalid syntax or structural inconsistencies occurred, the reward pipeline could no longer provide feedback, which ultimately caused the reinforcement learning to halt prematurely.

These results highlight the critical role of robust code supervision and reward shaping beyond pure geometry-based metrics, especially in code-generation scenarios where executability directly impacts learning stability.

378 C Step-by-Step CoT Reasoning in CAD Generation

379 To illustrate the step-by-step reasoning performed by our model prior to generating CadQuery scripts,
 380 we present an example corresponding to the first CAD model shown in Fig. 3. The user-provided
 381 prompt specifies precise geometric operations, which our model parses into a structured CoT (Chain-
 382 of-Thought) reasoning sequence. This sequence includes part decomposition, coordinate planning,
 383 sketch construction, and extrusion logic. The CoT reasoning is followed by executable CadQuery
 384 scripts that constructs the final 3D model. This example demonstrates the model's ability to convert
 385 complex, multi-step textual descriptions into valid, parameterized CAD scripts through interpretable
 386 and systematic reasoning.

Input

User Prompt: "Start by creating a new coordinate system for the first part, setting the Euler angles to $[0.0, 0.0, -90.0]$ and the translation vector to $[0.0, 0.0, 0.0]$. Next, draw a two-dimensional sketch on a new face, forming a loop with four lines: the first line starts at $(0.0, 0.0)$ and ends at $(0.375, 0.0)$; the second line starts at $(0.375, 0.0)$ and ends at $(0.375, 0.375)$; the third line starts at $(0.375, 0.375)$ and ends at $(0.0, 0.375)$; and the fourth line starts at $(0.0, 0.375)$ and ends at $(0.0, 0.0)$. Apply a sketch scale of 0.375 to this loop. Rotate the scaled sketch using the Euler angles $[0.0, 0.0, -90.0]$ and translate it using the translation vector $[0.0, 0.0, 0.0]$. Finally, extrude the sketch 0.375 units in the direction opposite to the normal to create a solid body with final dimensions of 0.375 units in length, width, and height. For the second part, create another new coordinate system, setting the Euler angles to $[-90.0, 0.0, -90.0]$ and the translation vector to $[0.375, 0.125, 0.0]$. Draw a new two-dimensional sketch on a new face, forming a loop with three lines: the first line starts at $(0.0, 0.0)$ and ends at $(0.25, 0.0)$; the second line starts at $(0.25, 0.0)$ and ends at $(0.25, 0.25)$; and the third line starts at $(0.25, 0.25)$ and ends at $(0.0, 0.0)$. Apply a sketch scale of 0.25 to this loop. Rotate the scaled sketch using the Euler angles $[-90.0, 0.0, -90.0]$ and translate it using the translation vector $[0.375, 0.125, 0.0]$. Extrude the sketch 0.375 units in the direction of the normal and add this extruded shape to the existing cube. The final dimensions of this rectangular prism are 0.25 units in length, 0.25 units in width, and 0.375 units in height."

CoT Prompt: Please create a CadQuery Python code to generate a model based on the following description. The reasoning process **MUST BE** enclosed within `<think>` `</think>` tags. The final CadQuery code **MUST BE** put in `“python code”` with **ONLY** the executable code inside the python box, nothing else. The final model is represented by `r`. In the `<think>` section, simulate the thought process of an engineer converting text descriptions into a CAD model. Follow these steps:

- **Stage 1: Description Analysis.** Break down the description into different parts or components. Identify key parameters for each part (coordinate systems, Euler angles, translation vectors). Understand the spatial relationships and assembly sequence between parts.
- **Stage 2: Coordinate System Planning.** Determine the coordinate systems used for each part. Parse how Euler angle rotations and translation vectors are applied. Ensure understanding of local-to-global coordinate system transformations.
- **Stage 3: Sketch Construction Strategy.** Analyze how to create each 2D sketch (loops, lines, points). Determine scaling factors for each sketch. Plan how to transform sketches into 3D space.
- **Stage 4: Extrusion Operation Planning.** Identify extrusion parameters for each part (direction, distance). Understand how to add extrusions to existing solids (new or merge). Verify dimensions after extrusion match the description.
- **Stage 5: Code Implementation Strategy.** Plan the sequence of CadQuery operations. Determine necessary CadQuery functions and methods. Consider how to organize code for clarity and readability.

387

- **Stage 6: When scaling in CadQuery.** Directly scale the size, define a scaling factor variable, and apply the scaling factor directly to all coordinates and dimensions. Don't try to use a non-existent `scale()` method on Workplane objects.

After your thinking process, provide clean, working CadQuery Python code to create the 3D model. Think step by step, but only keep a minimum draft for each thinking step, with 50 words at most. description:

388

Output

CoT Reasoning Steps (Generated by CAD-Coder):

1. ****Description Analysis**:**
 - Two parts: first rectangular prism, second triangular prism
 - First part has dimensions $0.375 \times 0.375 \times 0.375$, rotated -90 about Z
 - Second part is $0.25 \times 0.25 \times 0.375$, rotated -90 about X and Z, translated to $(0.375, 0.125, 0)$
2. ****Coordinate System Planning**:**
 - First part uses identity CS, second rotated -90 X/ -90 Z at $(0.375, 0.125, 0)$
 - Both parts translate to origin initially
3. ****Sketch Construction Strategy**:**
 - First part: 4-line rectangle sketch
 - Second part: 3-line triangular sketch
 - Apply scaling factors (0.375 and 0.25) to coordinates
4. ****Extrusion Operation Planning**:**
 - First part extruded $-0.375Z$
 - Second part extruded $+0.375Z$
 - Combine using union operation
5. ****Code Implementation Strategy**:**
 - Create base sketches
 - Apply rotations and translations
 - Extrude and combine

Generated CadQuery Script:

```
import cadquery as cq
# First part
part1 = (cq.Workplane("XY")
        .moveTo(0.0, 0.0)
        .lineTo(0.375, 0.0)
        .lineTo(0.375, 0.375)
        .lineTo(0.0, 0.375)
        .close()
        .extrude(-0.375)
        .rotate((0,0,0), (1,0,0), 90)
        .translate((0.0, 0.0, 0.0))
        )
# Second part
part2 = (cq.Workplane("XY")
        .moveTo(0.0, 0.0)
        .lineTo(0.25, 0.0)
        .lineTo(0.25, 0.25)
        .close()
        .extrude(0.375)
        .rotate((0,0,0), (1,0,0), 90)
        .rotate((0,0,0), (0,0,1), 90)
        .translate((0.375, 0.125, 0.0))
        )
# Combine parts
r = part1.union(part2)
```

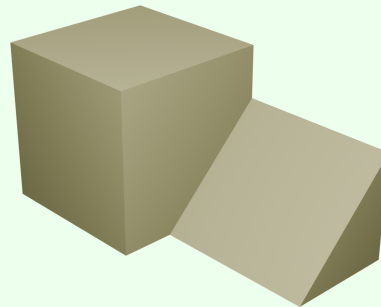


Figure S1: Generated CAD Model

389

390 D Additional Qualitative Comparisons Across Methods

391 Fig. S2 presents additional qualitative comparisons of CAD models generated by different methods.
 392 For methods that do not produce valid 3D models, placeholders are shown in their respective results.



Figure S2: Additional qualitative comparisons of CAD models generated by different methods.

393 E Performance on CAD Editing

394 Although our model was not explicitly trained on CAD editing data, it demonstrates promising
395 capabilities in handling simple CAD editing tasks based on user instructions. This suggests that
396 the model has acquired a degree of structural understanding of CadQuery code and can generalize
397 beyond the training objective of generation-from-scratch.

398 As shown in Fig. S3, the model successfully performs lightweight operations such as modifying object
399 dimensions, removing a component, or adjusting translation and rotation parameters in response to
400 natural language prompts. These preliminary results highlight the model’s potential to be extended
401 toward interactive or instruction-following CAD editing scenarios.

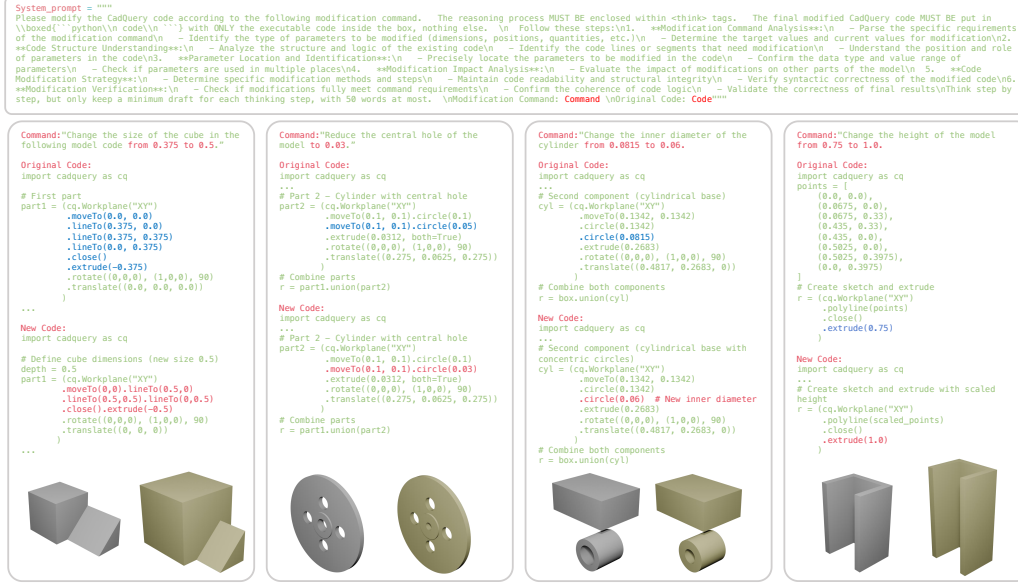


Figure S3: Examples of simple CadQuery code editing based on instructions.

402 F Failure Cases

403 As shown in Figure S4, our method still struggles with certain challenging cases. As illustrated in
404 Fig. S4(a), our method still struggles with complex structures composed of multiple sub-components,
405 where inaccurate spatial alignment between modules can lead to visible dislocations or offsets.
406 Moreover, as shown in Fig. S4(b), the model may misclassify operations such as extrusion and cutting,
407 resulting in geometries that deviate from the intended design. In addition, Fig. S4(c) highlights the
408 challenge posed by very thin structures or internal cavities, where sparse point sampling may induce
409 reward hacking behavior, revealing limitations in handling overlapping features and tight geometric
410 tolerances.

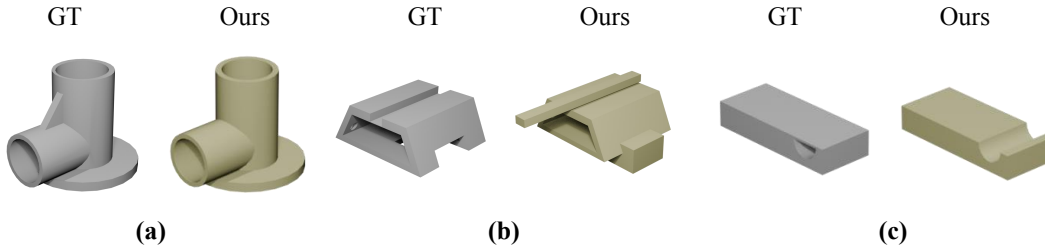


Figure S4: Failure cases in CAD-Coder generation