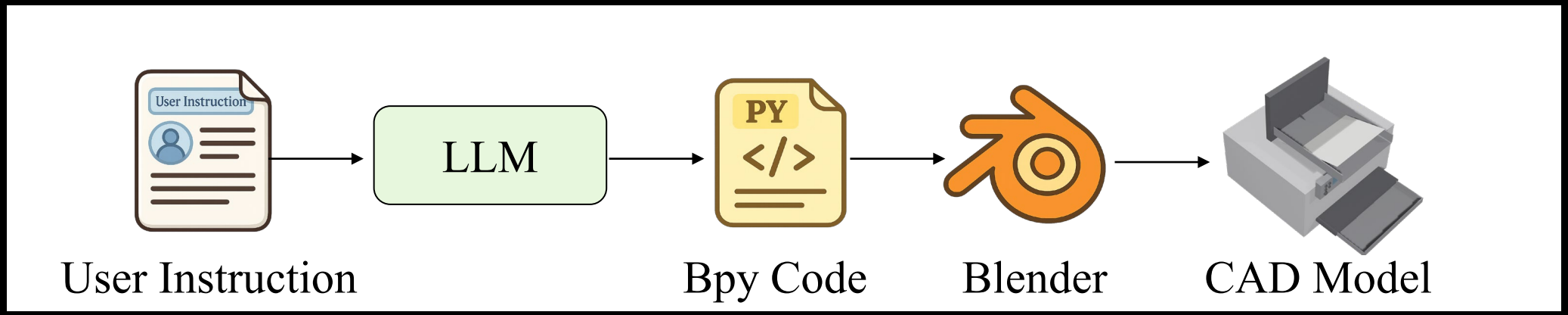
Learning Hierarchical and Geometry-Aware Graph Representations for Text-to-CAD

(Graph-CAD, ICLR 2026)

Shengjie Gong[1] Wenjie Peng[1] Hongyuan Chen[2] Gangyu Zhang[1]
Yunqing Hu[2] Huiyuan Zhang[2] Shuangping Huang[1*] Tianshui Chen[3]

[1]South China University of Technology
[2]Zhuzhou CRRC Times Electric Co., Ltd.
[3]Guangdong University of Technology

Method introduction of Graph-CAD



Task Description:

Generate executable CAD code from natural language instructions.

Existing Methods:

End-to-end Text-to-bpy Generation

No explicit assembly hierarchy

No explicit geometric constraints

Prone to error accumulation and cascading failures

Our Approach:

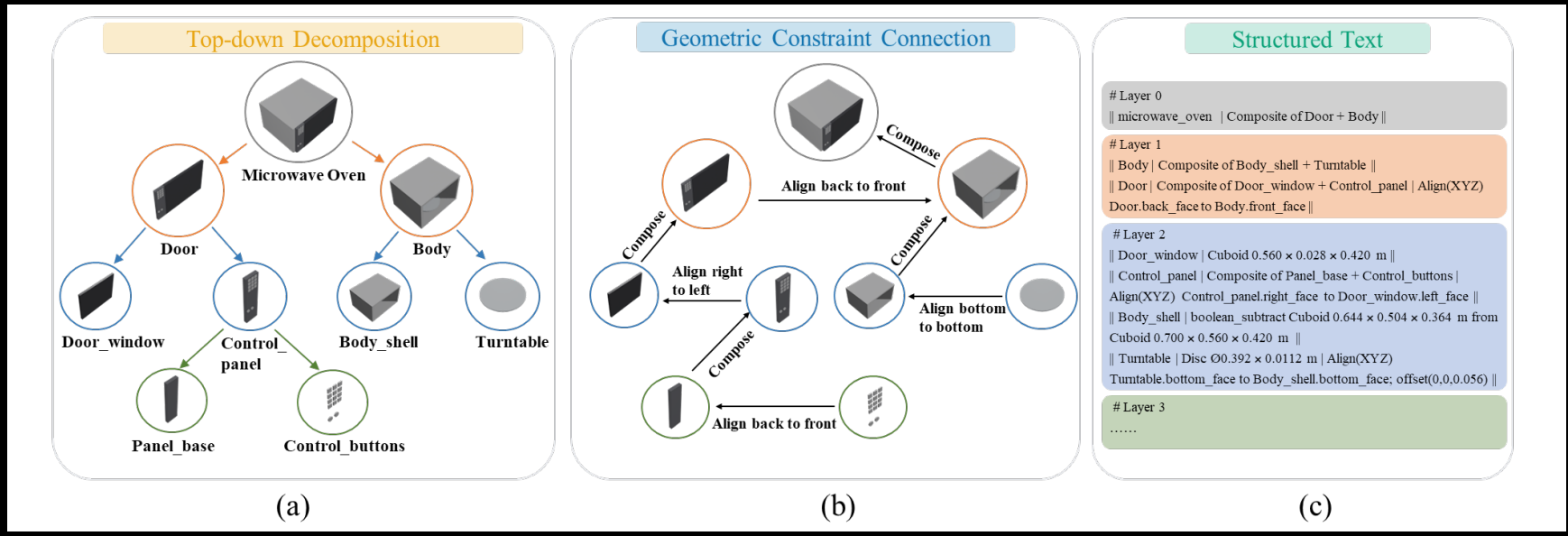
Graph-mediated Text-to-CAD

Hierarchical decomposition + geometric constraints

Three-stage generation for robust and structured CAD synthesis

Challenges in Text-to-CAD:

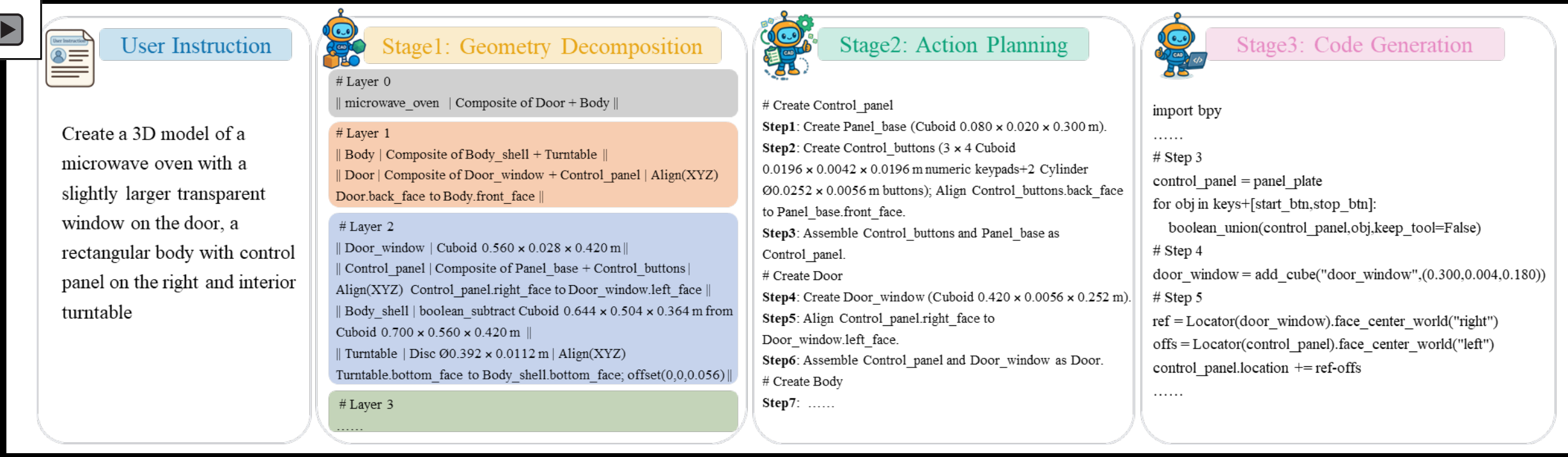
- Long-horizon generation with interdependent CAD operations makes the process highly fragile.
- Without explicit assembly hierarchy and geometric constraints, direct text-to-code generation easily leads toerror accumulation and cascading failures.



Key Ideas:

(1) Introduce a hierarchical and geometry-aware graph to explicitly model assembly hierarchy and geometric constraints;

(2) Use a three-stage generation pipeline to guide robust and structured CAD synthesis. (Geometry Decomposition → Action Planning → Code Generation)



Stage 1: Geometry Decomposition:

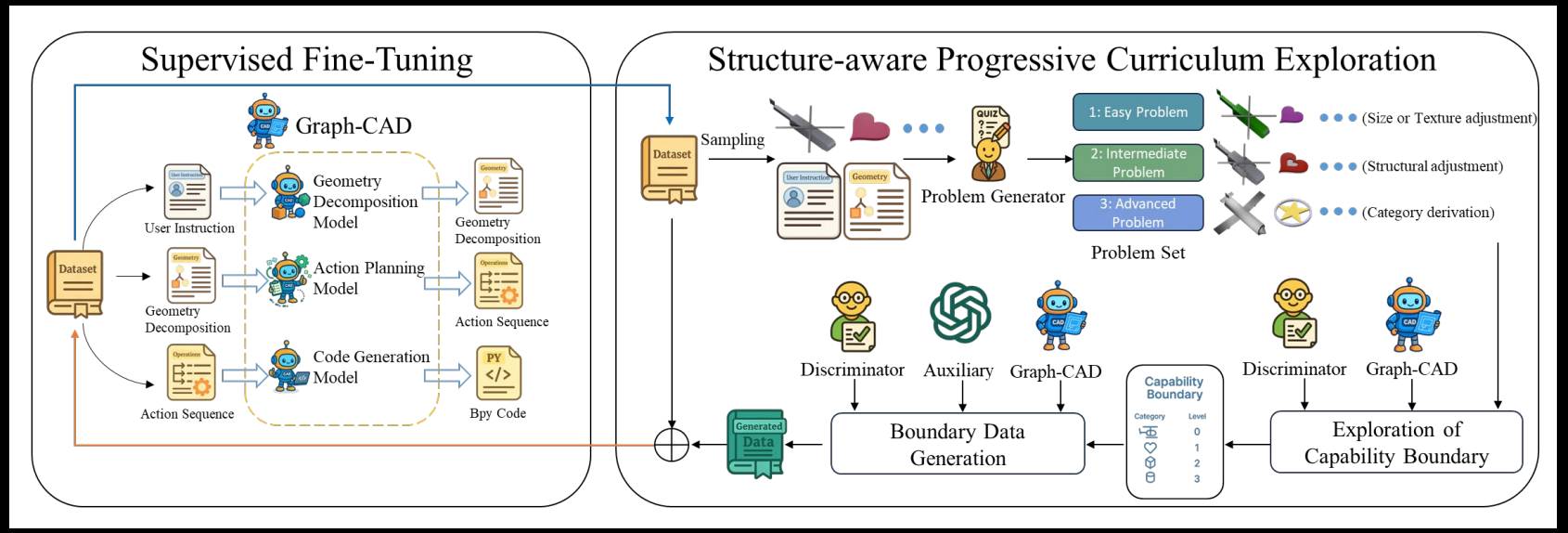
Build a hierarchical graph with explicit geometric constraints.

Stage 2: Action Planning:

Convert the graph into an ordered CAD operation sequence.

Stage 3: Code Generation:

Translate the planned operation sequence into executable bpy code.



Structure-Aware Progressive Curriculum Learning

Motivation:

As CAD assemblies become more complex, limited training data hinders generalization to diverse and structurally intricate designs.

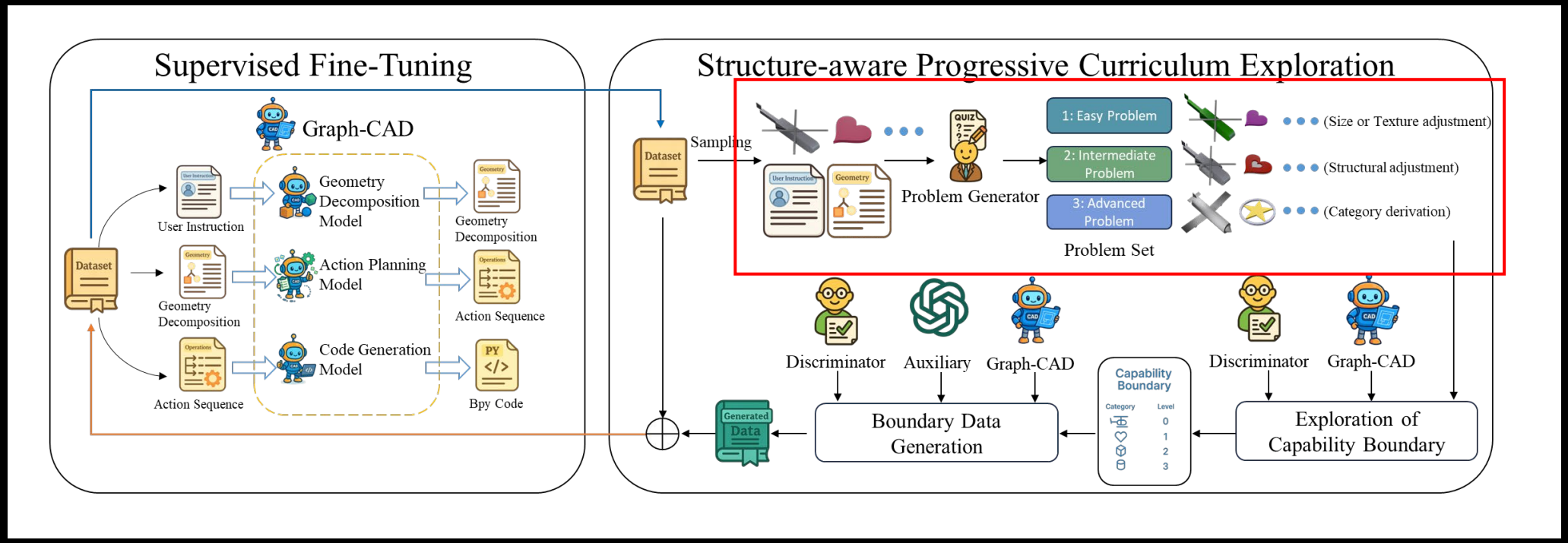
Core Idea:

Progressively improve the model by first identifying its current capability boundary, then generating new training instances near that boundary.

Core Modules:

(1) Capability Boundary Exploration

(2) Boundary Data Generation



Problem Generator:

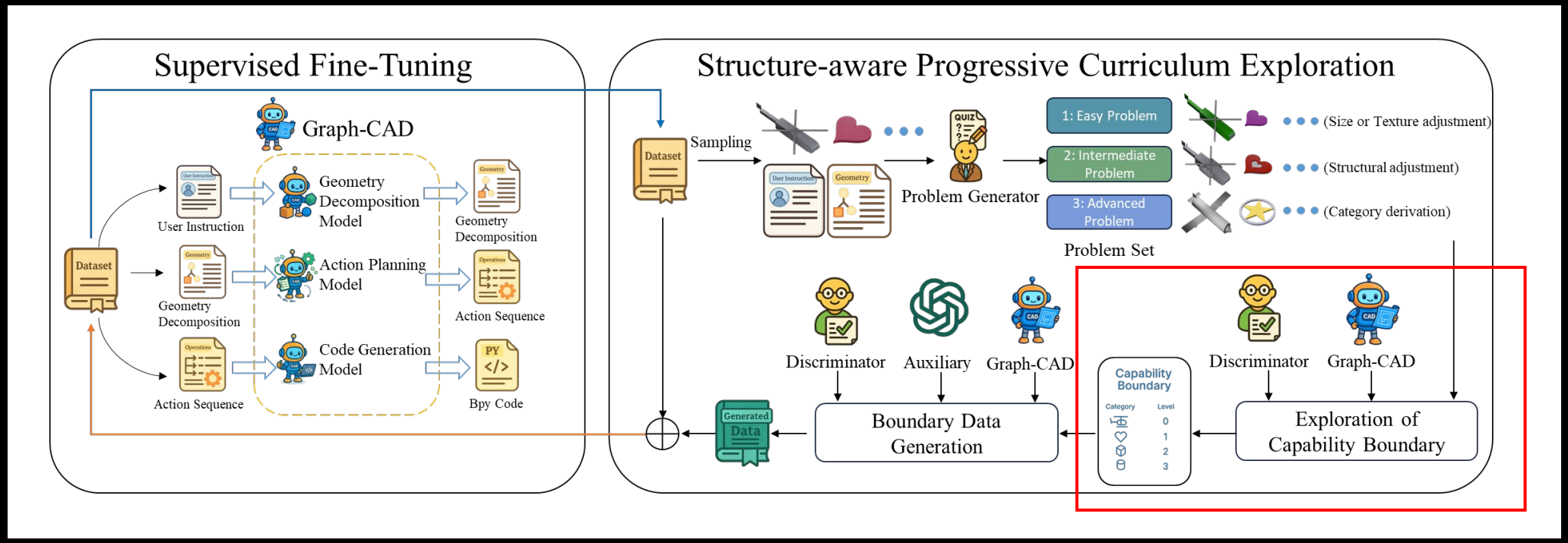
Generate difficulty-graded variants from each seed example:

(1) Easy: size or texture adjustment

(2) Intermediate: structural adjustment

(3) Advanced: category derivation

These variants are used to probe the model's current capability boundary.

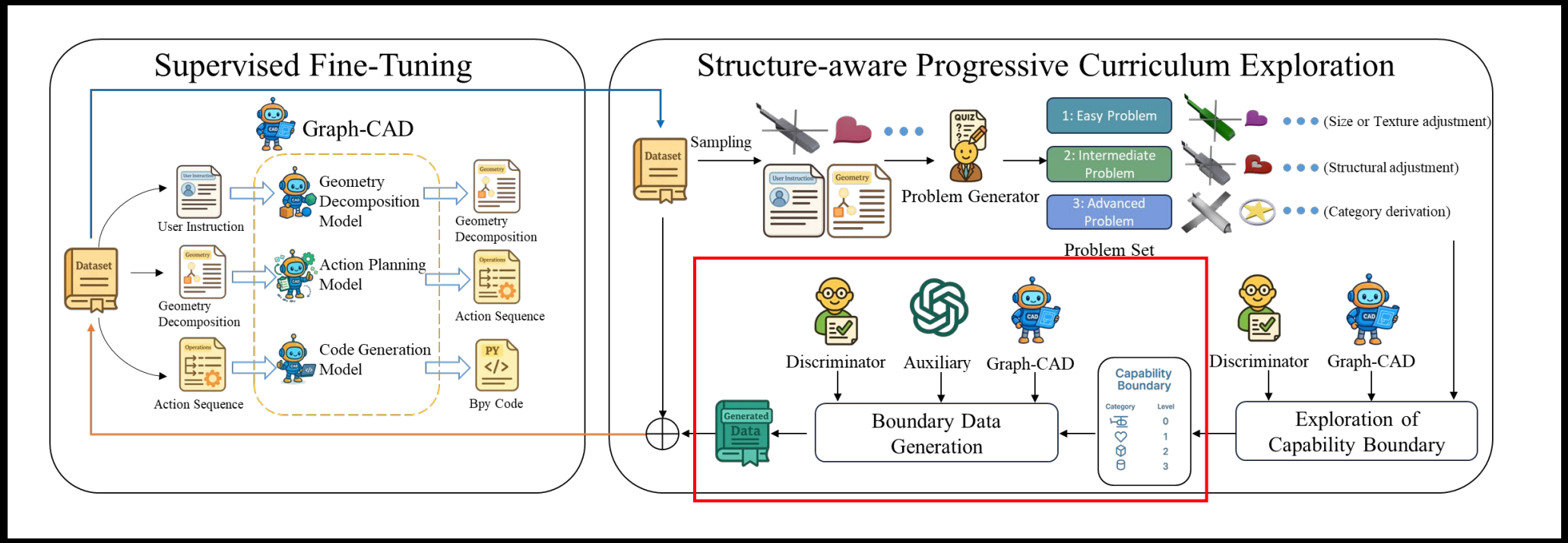


Exploration of Capability Boundary:

Test the model on graded variants from easy to advanced.

The highest difficulty level it can reliably solve is treated as the current capability boundary.

This boundary determines where to generate new training samples in the next iteration.



Boundary Data Generation:

Generate new samples near the current capability boundary.

If the model solves Intermediate, new data is synthesized for Intermediate + Advanced levels.

Validated samples are then added to the training set for the next iteration.

Experiment introduction of Graph-CAD

Experimental Setups

Datasets:

BlendGeo (Training Set)

A 12K dataset constructed for Graph-CAD, where each sample contains: user instruction + geometric decomposition graph + action sequence + executable bpy code.

CADBench (Evaluation Benchmark)

We evaluate on the public CADBench benchmark, which contains two subsets:

- CADBench-Sim: in-distribution instructions
- CADBench-Wild: out-of-distribution instructions

Experimental Setups

Evaluation Metrics

Original CADBench Metrics:

Used to evaluate visual quality and code executability:

Attr.: attribute accuracy

Spat.: spatial relation accuracy

Inst.: instruction-following accuracy

Esyntax: syntax error rate

Our Added Metrics:

GCS: Geometric Constraint Satisfaction

NLA: Node-Level Accuracy

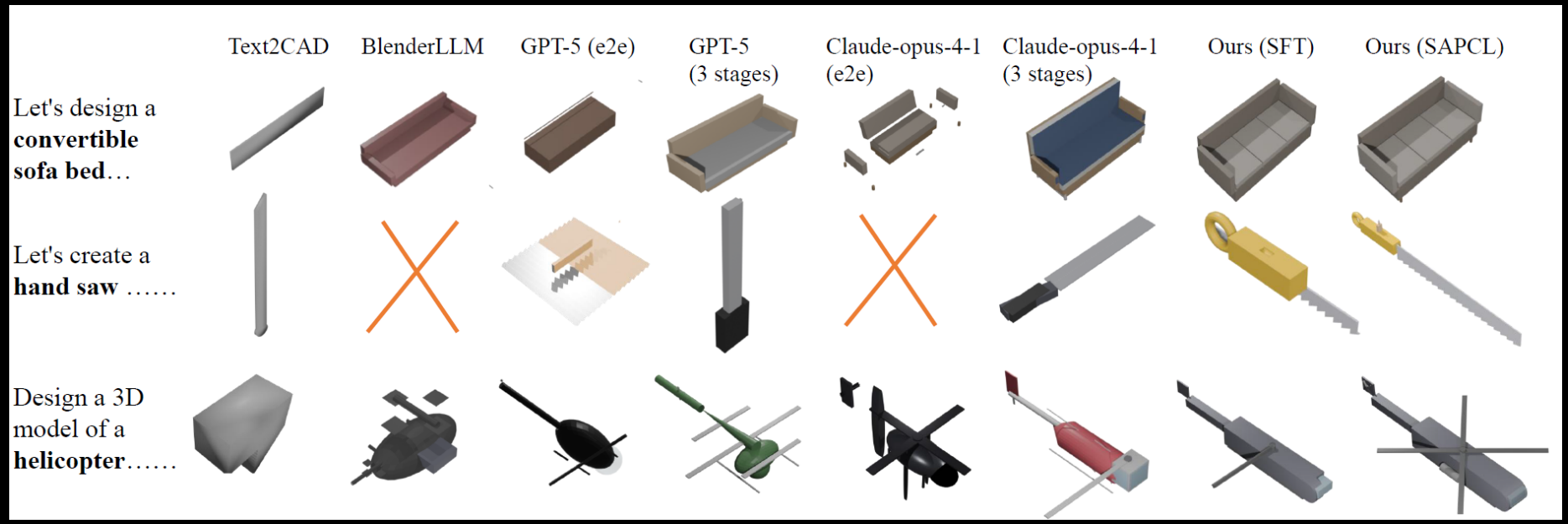
HLA: Hierarchy-Level Accuracy

Quantitative Results

Models	CADBench-Sim							CADBench-Wild						
	Attr.↑	Spat.↑	Inst.↑	Avg.↑	E_{syntax}↓	CLIP↑	GCS↑	Attr.↑	Spat.↑	Inst.↑	Avg.↑	E_{syntax}↓	CLIP↑	GCS↑
Specifically Text-to-CAD open-source models														
BlenderLLM	0.6893	0.6953	0.3650	0.5832	2.4%	0.6409	0.5513	0.6782	0.6363	0.4581	0.5909	5.3%	0.6056	0.4983
Text2CAD	0.3278	0.2084	0.0446	0.1936	6.6%	0.5707	-	0.4198	0.3082	0.1323	0.2868	14.0%	0.5211	-
CADFusion	0.3566	0.2258	0.0674	0.2166	6.2%	0.5578	-	0.3822	0.3716	0.1496	0.3011	11.5%	0.5278	-
General-purpose Large Language Models														
Qwen-Plus	0.3604	0.3777	0.2072	0.3151	48.4%	0.3362	0.2379	0.2596	0.2722	0.1951	0.2423	61.0%	0.2446	0.1305
Llama-3.1-405b	0.3302	0.3355	0.1537	0.2731	36.4%	0.3943	0.3269	0.3331	0.3530	0.1943	0.2934	47.2%	0.3242	0.2903
Deepseek-r1	0.4124	0.4366	0.2179	0.3556	19.2%	0.5011	0.5556	0.4814	0.5141	0.3735	0.4564	20.50%	0.4858	0.4275
Gemini-2.5-pro	0.2173	0.2180	0.1565	0.1972	42.4%	0.2050	0.4048	0.2002	0.1880	0.1667	0.1850	48.7%	0.1750	0.2584
GPT-5	0.7013	0.7347	0.4250	0.6203	2.8%	0.6449	0.3846	0.6858	0.7091	0.5595	0.6515	5.5%	0.6003	0.4017
Claude-opus-4-1	0.7216	0.7368	0.5403	0.6662	7.4%	0.6151	0.4932	0.6847	0.7218	0.5997	0.6687	14.5%	0.5550	0.5062
Graph-CAD (Ours)														
Graph-CAD (SFT)	0.7295	0.7265	0.4733	0.6431	2.4%	0.6544	0.7830	0.6944	0.7270	0.5861	0.6692	4.5%	0.6358	0.8025
Graph-CAD (SAPCL)	**0.7681**	**0.7423**	**0.5546**	**0.6883**	**2.0%**	**0.6693**	**0.9018**	**0.7695**	**0.7590**	**0.6057**	**0.7114**	**2.5%**	**0.6577**	**0.8943**
SAPCL vs SFT	(5.29%↑)	(2.17%↑)	(17.18%↑)	(7.03%↑)		(2.28%↑)	(15.17%↑)	(10.82%↑)	(4.40%↑)	(3.34%↑)	(6.31%↑)		(3.44%↑)	(11.44%↑)

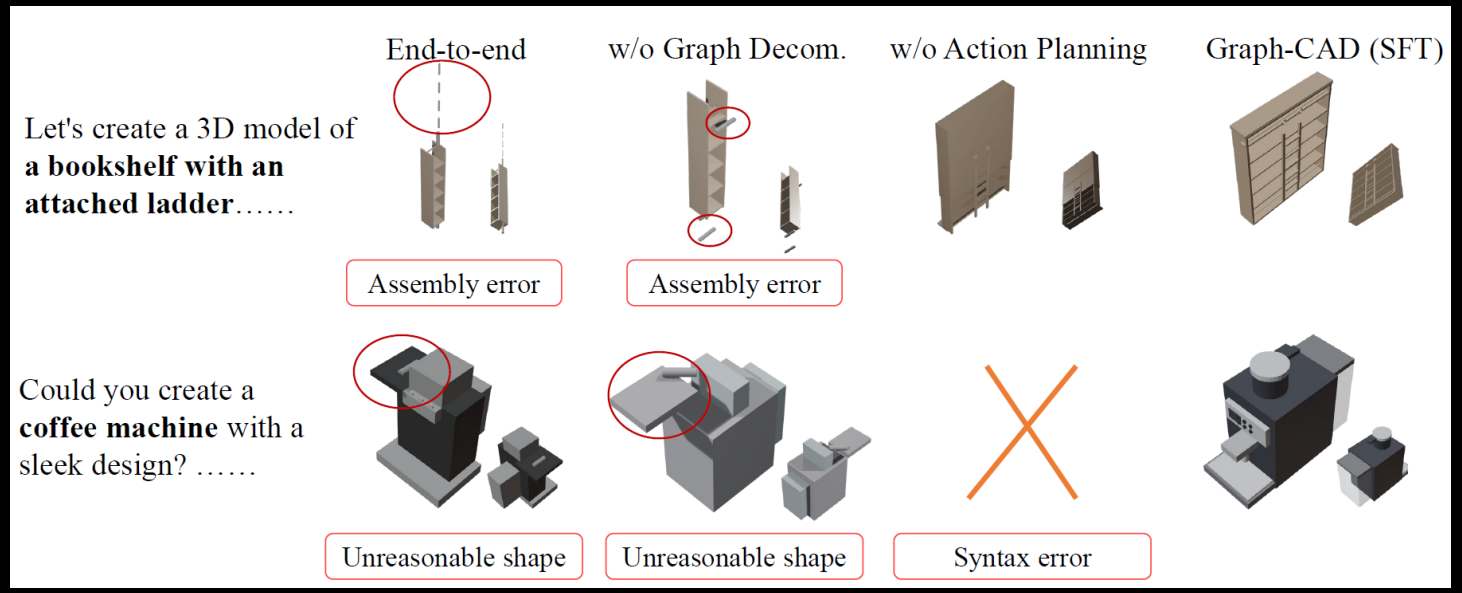
- Graph-CAD (SAPCL) sets the best results on both CADBench-Sim and CADBench-Wild.
- This validates the effectiveness of graph-mediated generation and structure-aware curriculum learning for robust Text-to-CAD synthesis.

Qualitative Comparison Results



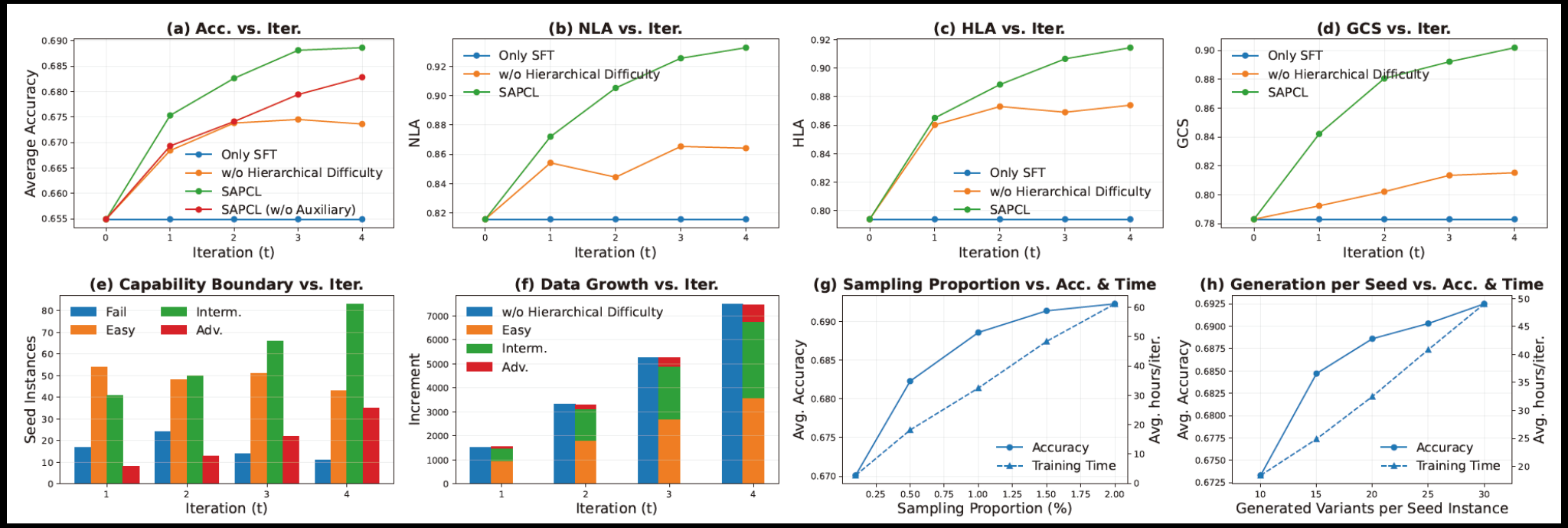
- Graph-CAD generates more visually plausible and instruction-aligned CAD models than existing methods.
- Compared with end-to-end generation, the three-stage graph-mediated pipeline produces more orderly part arrangements and more geometrically coherent assemblies, further validating the robustness of our framework.

Ablation Studies: Three-Stage Pipeline of Graph-CAD



- Removing either graph decomposition or action planning leads to typical failure modes such as assembly errors, unreasonable shapes, and even syntax errors, highlighting the necessity of both intermediate representations in robust Text-to-CAD generation.

Ablation Studies: Structure-Aware Progressive Curriculum Learning



- The figure above shows that SAPCL consistently outperforms both standard supervised fine-tuning and difficulty-unaware data expansion.

Thank you for watching!