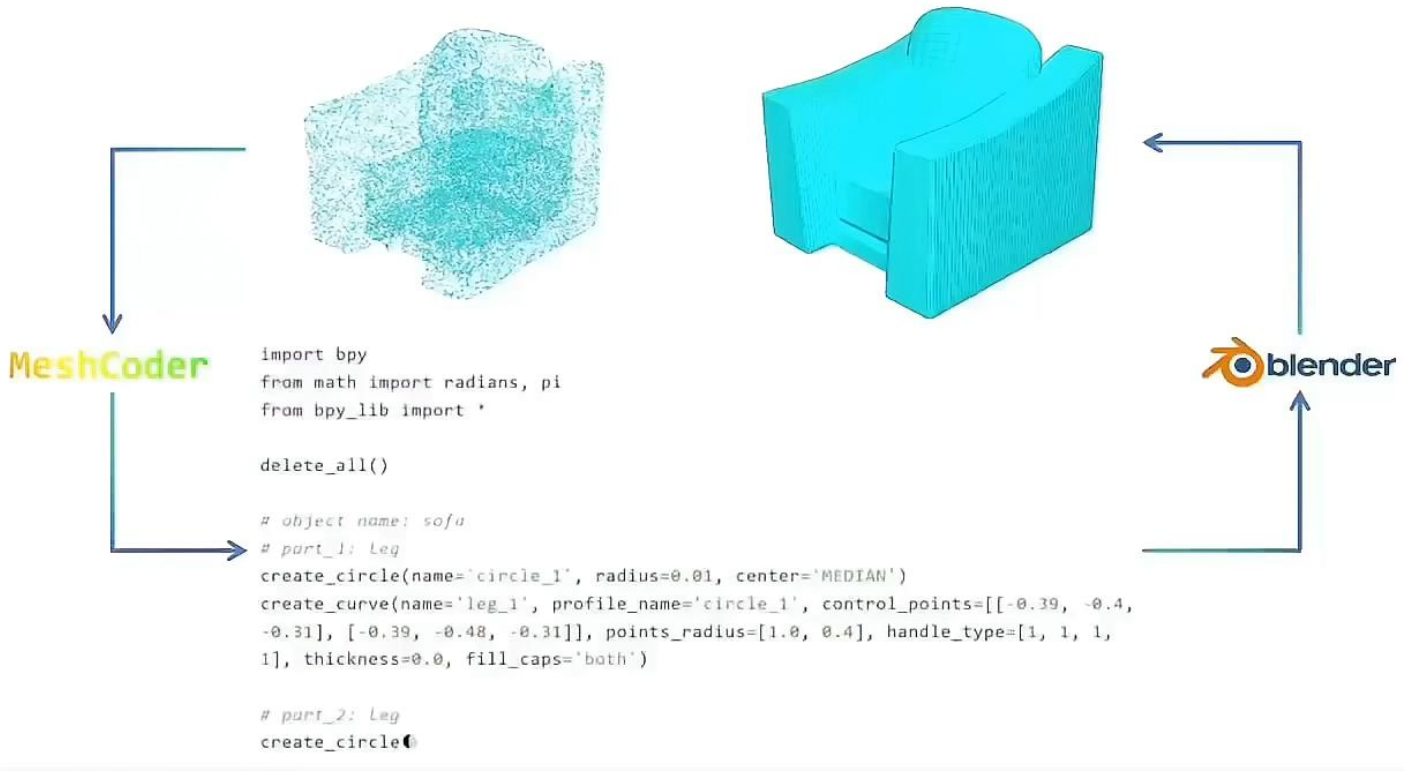
MeshCoder: LLM-Powered Structured Mesh Code Generation from Point Clouds

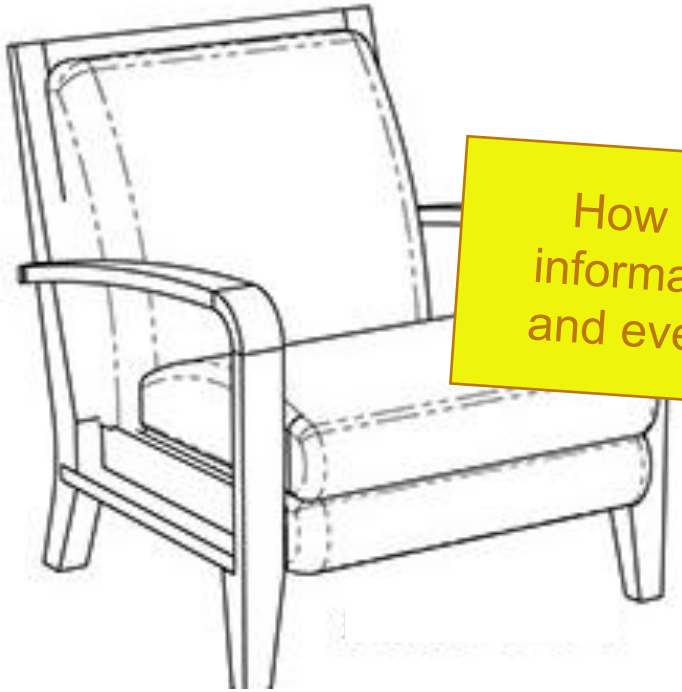
Bingquan Dai[1,2*], Li Luo[1, 3*], Qihong Tang[1,4], Jie Wang[1,5], Xinyu Lian[1], Hao Xu[1,6], Minghan Qin[2], Xudong Xu[1], Bo Dai[1,3], Haoqian Wang[2†], Zhaoyang Lyu[1†], Jiangmiao Pang[1]

[1]Shanghai AI Lab [2]Tsinghua University [3]The University of Hong Kong



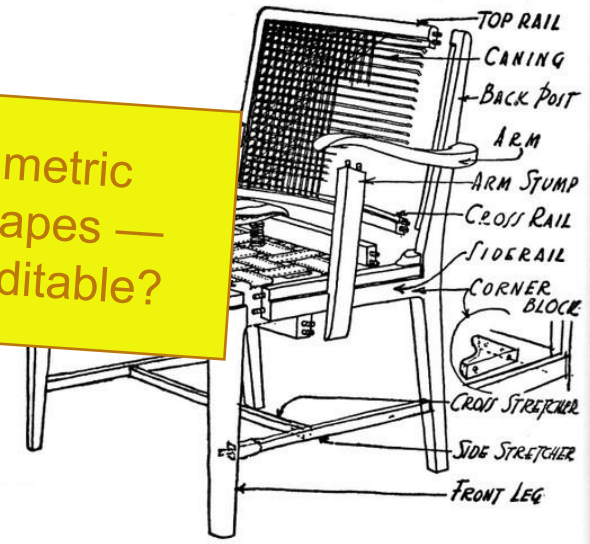
Motivating Problem

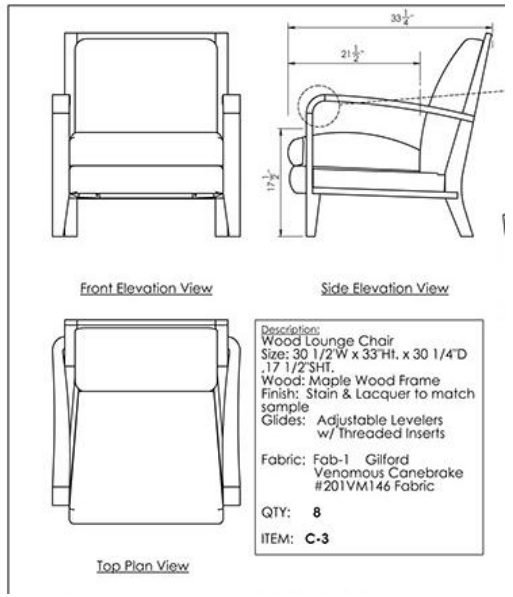
A nice sofa



How can we get parametric information from 3D shapes — and even make them editable?

The sofa with parametric details





reverse engineering

MeshCoder

Inference Model

Professional 3d software python codes

Execute in Blender

```
# object name: sofa
# part_1: leg
create_circle(name='circle_1', radius=0.01, center='MEDIAN')
create_curve(name='leg_1', profile_name='circle_1', control_points=[[-0.39
```

⋮

```
# part_8: back sofa board
create_primitive(name='back sofa board_8', primitive_type='cube', locatio
bevel(name='back sofa board_8', width=0.27, segments=4)
```

```
# part_9: sofa board
create_primitive(name='sofa board_9', primitive_type='cube', location=
bevel(name='sofa board_9', width=0.28, segments=4)
```

```
# part_10: arm
create_quad(name=['quad_1_10', 'quad_2_10', 'quad_3_10', 'quad_4_10'],
bridge_edge_loops(name='arm_10', profile_name=['quad_1_10', 'quad_2_10
bevel(name='arm_10', width=0.06, segments=2)
```

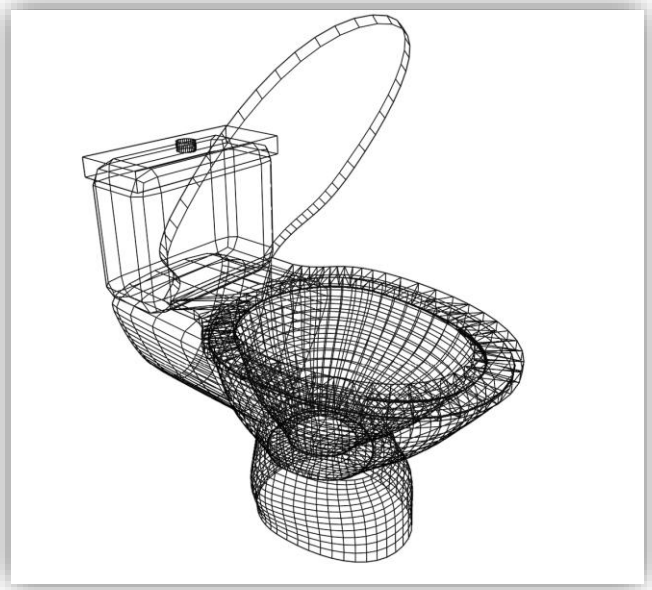
```
# part_11: cushion
create_primitive(name='cushion_11', primitive_type='cube', location=
bevel(name='cushion_11', width=0.28, segments=4)
```

```
# part_12: cushion
create_curve(name=['curve_1_12', 'curve_2_12', 'curve_3_12', 'curve_4_
bridge_edge_loops(name='cushion_12', profile_name=['curve_1_12', 'curv
```

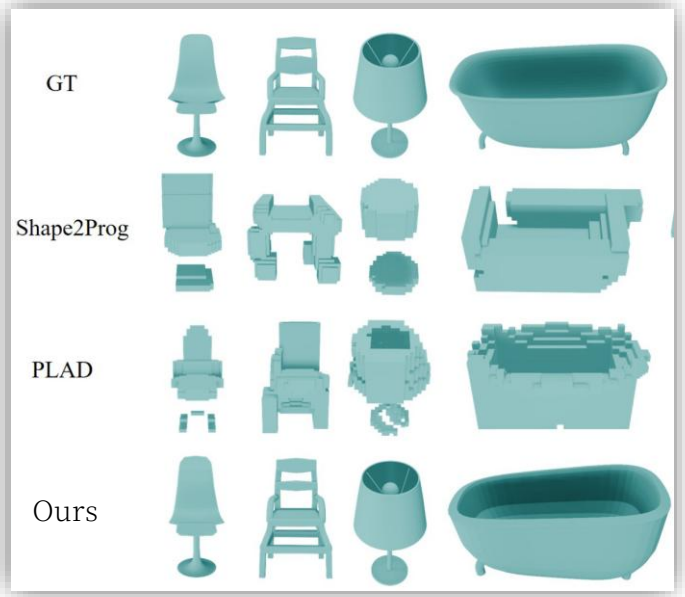
Our method achieves great performance

Diverse Categories

Clear Topology

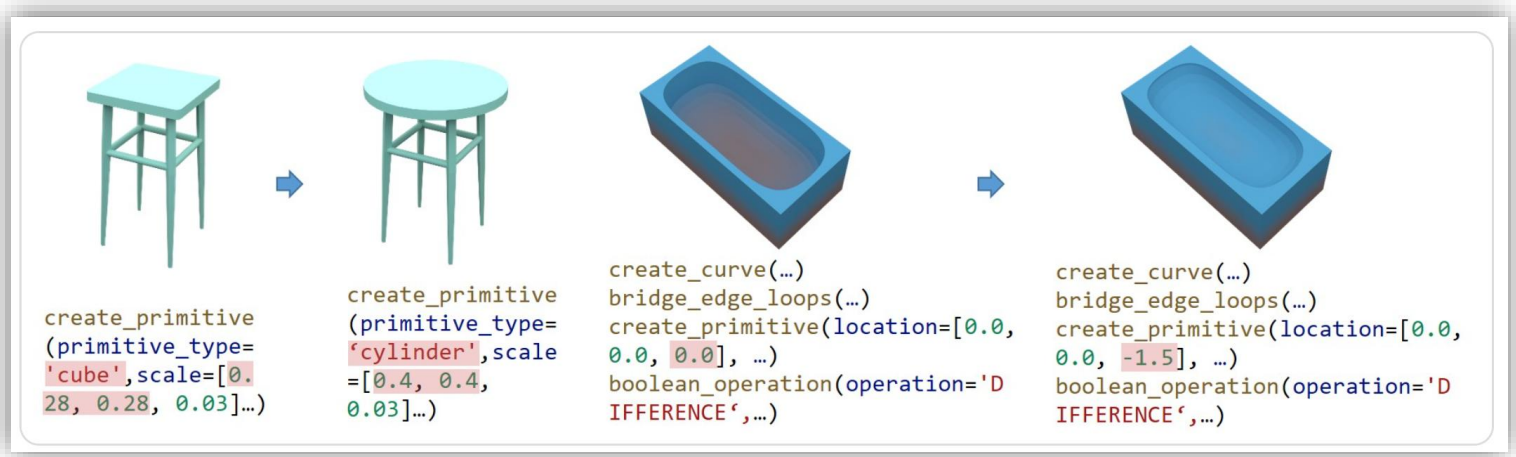


Accurate Reconstruction

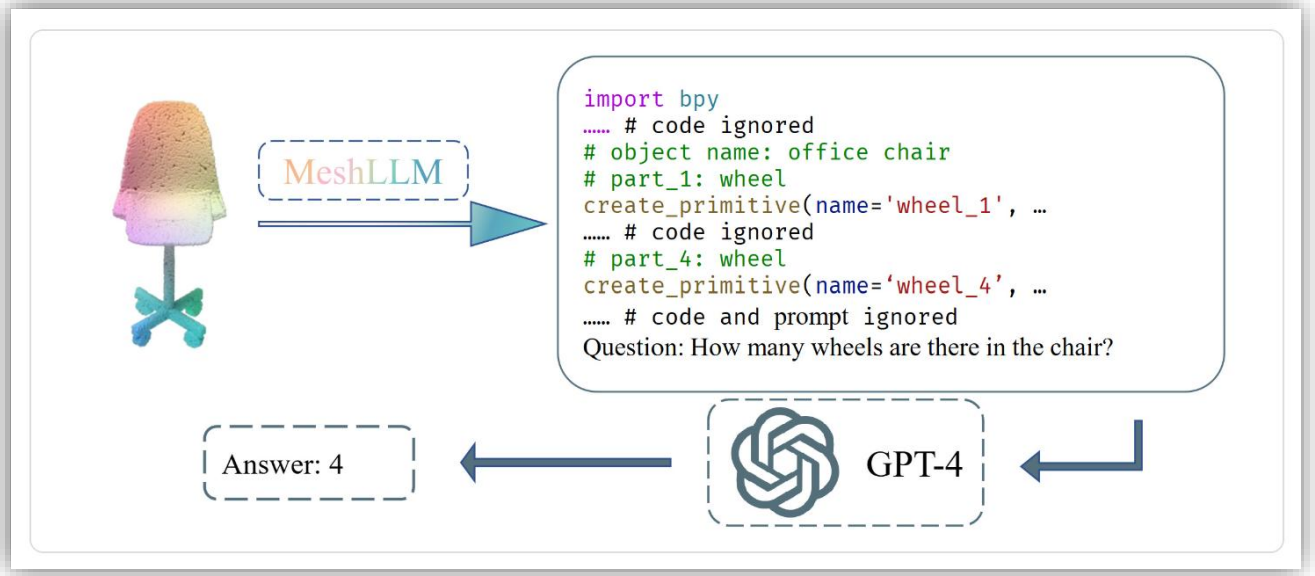


Unique features

Human editability

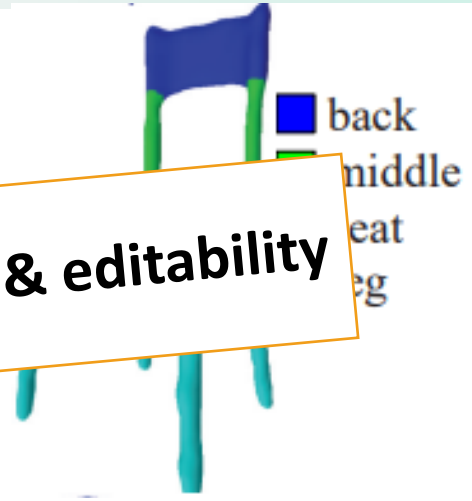


LLM understanding



Existing Methods

3D Scanning + Mesh Segmentation

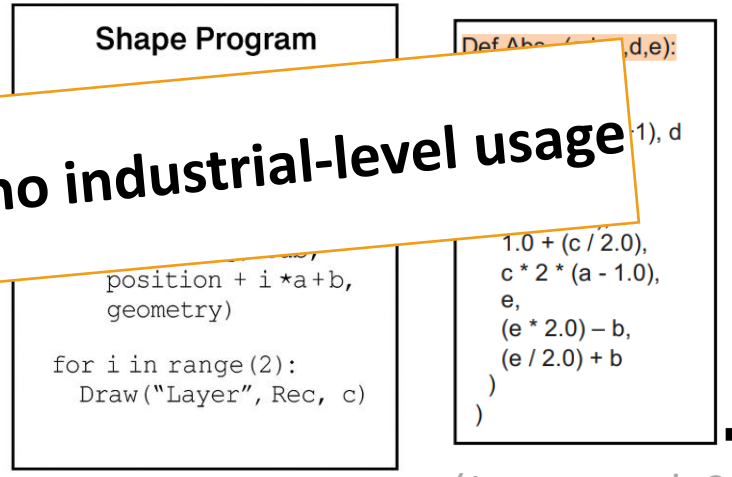


(Kalogerakis et al, 2010) [1]

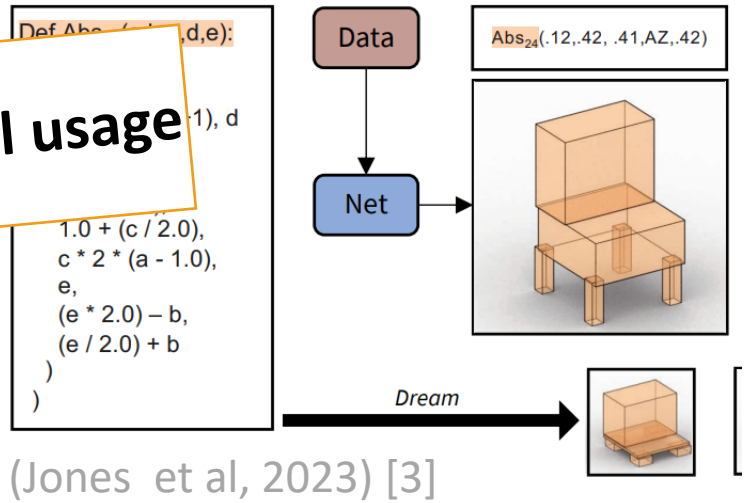
no parametric details & editability

Shape Program-based

shape logic only, no industrial-level usage



(Tian et al, 2019) [2]



(Jones et al, 2023) [3]

[1] Kalogerakis et al. 2010. Learning 3D Mesh Segmentation and Labeling. ACM Transactions on Graphics (TOG).
[2] Tian et al. 2019. Learning to Infer and Execute 3D Shape Programs. International Conference on Learning Representations (ICLR).
[3] Jones et al. 2023. ShapeCoder: Discovering Abstractions for Visual Programs. ACM Transactions on Graphics (TOG)

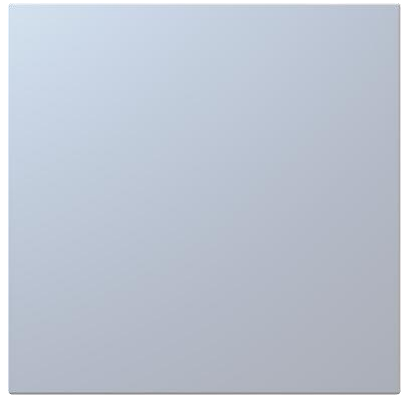
Step 1

Structural Shape Expression

Build our own DSL

Based on Blender API

Operation name	Pseudocode
Primitive	primitive(‘cube', loc=…, rot=…, scale=…)

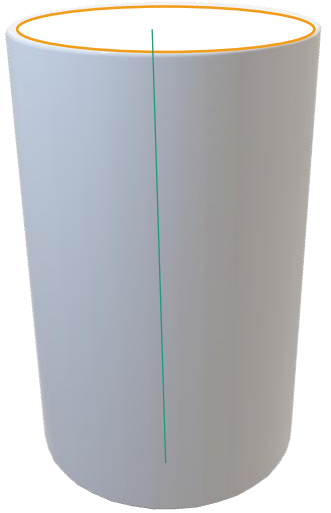


Build our own DSL

Based on Blender API

Operation name	Pseudocode
Primitive	primitive(‘cube', loc=…, rot=…, scale=…)

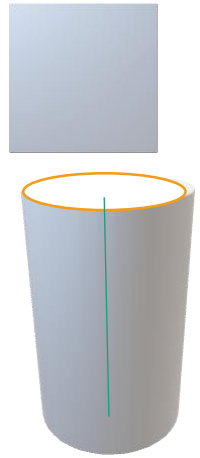
Sweep / Translation sweep(profile='bezier', path='curve_spine', caps='both')

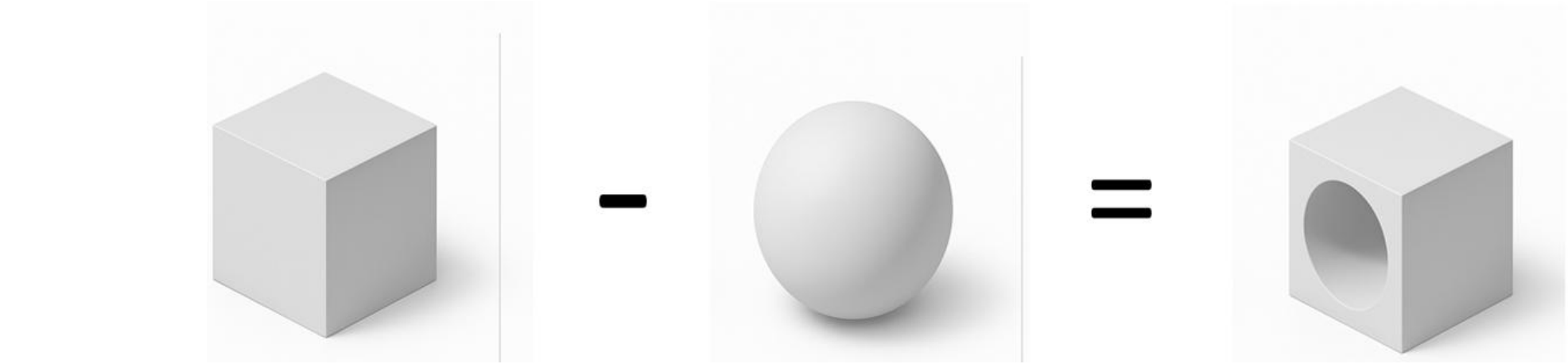


Build our own DSL

Based on Blender API

Operation name	Pseudocode
Primitive	primitive(‘cube', loc=…, rot=…, scale=…)
Sweep / Translation	sweep(profile='bezier', path='curve_spine', caps='both')
Boolean	boolean(A='cube', B='sphere', op='difference')

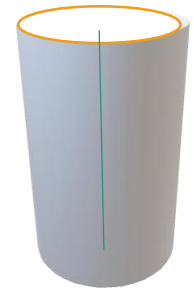


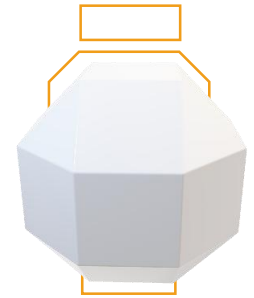


Build our own DSL

Based on Blender API

Operation name	Pseudocode
Primitive	primitive(‘cube', loc=…, rot=…, scale=…)
Sweep / Translation	sweep(profile='bezier', path='curve_spine', caps='both')
Boolean	boolean(A='cube', B='sphere', op='difference')
Loft / Bridge Loop	bridge_loops([c1,c2,c3,c4], smooth=0.6)

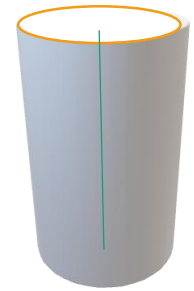


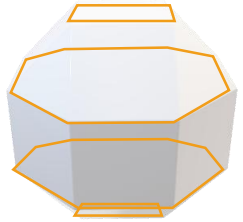


Build our own DSL

Based on Blender API

Operation name	Pseudocode
Primitive	primitive(‘cube', loc=…, rot=…, scale=…)
Sweep / Translation	sweep(profile='bezier', path='curve_spine', caps='both')
Boolean	boolean(A='cube', B='sphere', op='difference')
Loft / Bridge Loop	bridge_loops([c1,c2,c3,c4], smooth=0.6)
Array	array('cube', type='grid', counts=(4,4), step=(0.2,0.2))





Build our own DSL

Based on Blender API

Operation name	Pseudocode
Primitive	primitive(‘cube', loc=…, rot=…, scale=…)
Sweep / Translation	sweep(profile='bezier', path='curve_spine', caps='both')
Boolean	boolean(A='cube', B='sphere', op='difference')
Loft / Bridge Loop	bridge_loops([c1,c2,c3,c4], smooth=0.6)
Array	array('cube', type='grid', counts=(4,4), step=(0.2,0.2))

Five types of code blocks to express almost every 3d object.

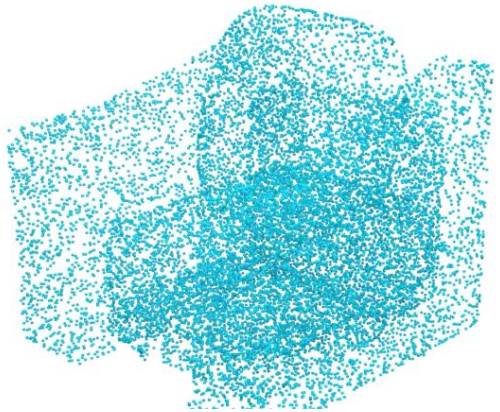
Step 2

Dataset Construction

Basic Idea

Aim: Construct an (object point cloud, code) dataset for the inference model to learn from.

(



,

```
# object name: sofa
# part_1: leg
create_circle(name='circle_1', radius=0.01, center='MEDIAN')
create_curve(name='leg_1', profile_name='circle_1', control_points=[[-0.39
```

⋮

```
# part_8: back sofa board
create_primitive(name='back sofa board_8', primitive_type='cube', locatio
bevel(name='back sofa board_8', width=0.27, segments=4)
```

```
# part_9: sofa board
create_primitive(name='sofa board_9', primitive_type='cube', location=[
bevel(name='sofa board_9', width=0.28, segments=4)
```

```
# part_10: arm
create_quad(name=['quad_1_10', 'quad_2_10', 'quad_3_10', 'quad_4_10'],
bridge_edge_loops(name='arm_10', profile_name=['quad_1_10', 'quad_2_10
bevel(name='arm_10', width=0.06, segments=2)
```

```
# part_11: cushion
create_primitive(name='cushion_11', primitive_type='cube', location=
bevel(name='cushion_11', width=0.28, segments=4)
```

```
# part_12: cushion
create_curve(name=['curve_1_12', 'curve_2_12', 'curve_3_12', 'curve_4_
bridge_edge_loops(name='cushion_12', profile_name=['curve_1_12', 'curv
```

)

Structural, part by part

Basic Idea


part level
dataset
part level
inference model
object level
datasets

Basic Idea

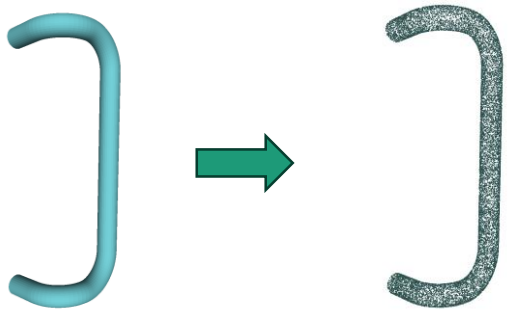
Part-level

Code generation Programs

GPT draft + Human refine

Part-level python codes

```
7   create_circle(name='curve', radius=0.04)
8   create_curve(name='translation', profile_name='curve', control_points=
```



Basic Idea

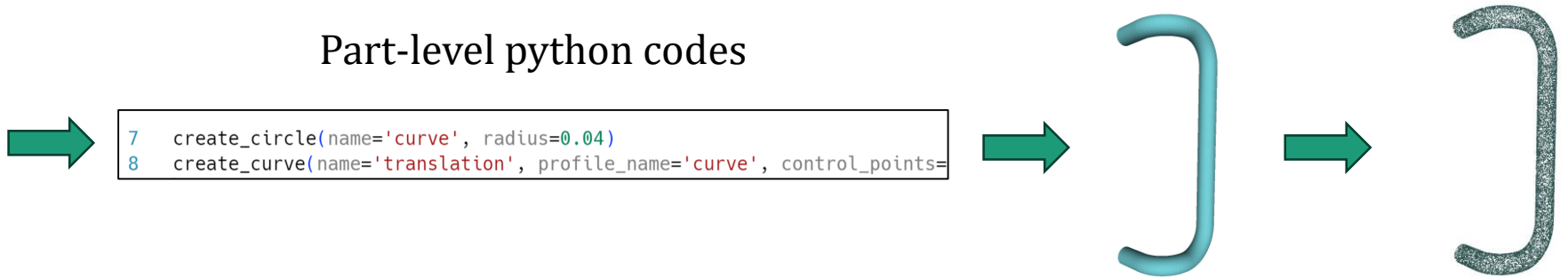
Part-level

Code generation Programs

GPT draft + Human refine

Part-level python codes

```
7   create_circle(name='curve', radius=0.04)
8   create_curve(name='translation', profile_name='curve', control_points=
```



( ,

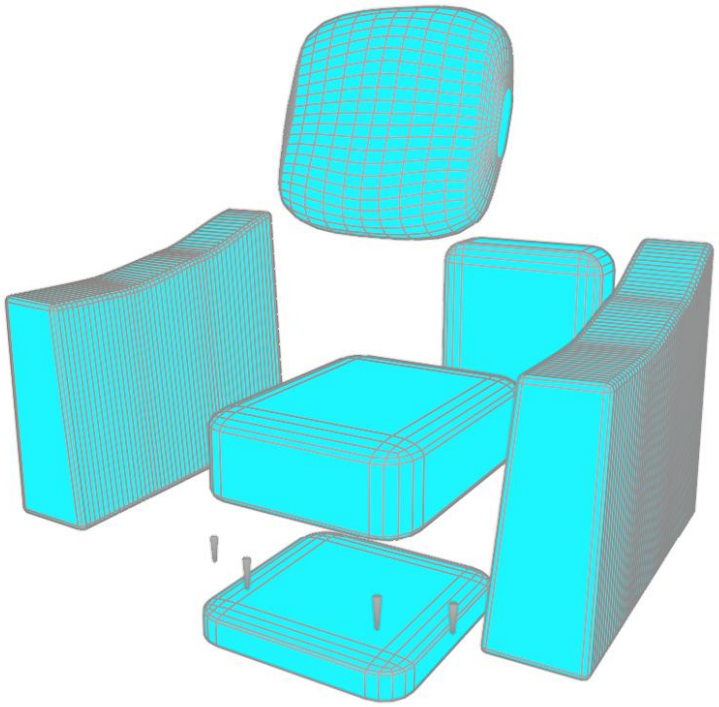
```
7   create_circle(name='curve', radius=0.04)
8   create_curve(name='translation', profile_name='curve', control_points=
```

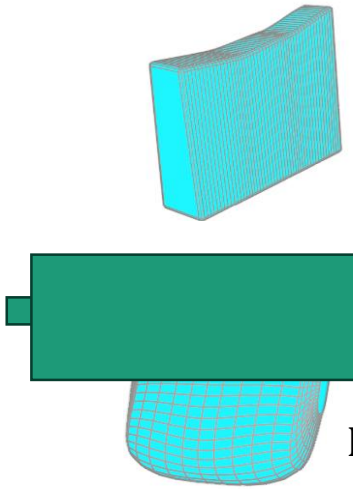
)

(Point Cloud, Code) data pair

Basic Idea

Object-level





Extract Point Cloud

Part-level Inference model

.....

```
# object name: sofa
# part_1: leg
create_circle(name='circle_1', radius=0.01, center='MEDIAN')
create_curve(name='leg_1', profile_name='circle_1', control_points=[[-0.39
```

```
# part_6: leg
create_circle(name='circle_6', radius=0.01, center='MEDIAN')
create_curve(name='leg_6', profile_name='circle_6', control_points=[[0.39
```

```
# part_7: arm
create_quad(name=['quad_1_7', 'quad_2_7', 'quad_3_7', 'quad_4_7'], control_
bridge_edge_loops(name='arm_7', profile_name=['quad_1_7', 'quad_2_7', 'quad
bevel(name='arm_7', width=0.06, segments=2)
```

```
# part_8: back sofa board
create_primitive(name='back sofa board_8', primitive_type='cube', locatio
bevel(name='back sofa board_8', width=0.27, segments=4)
```

```
# part_9: sofa board
create_primitive(name='sofa board_9', primitive_type='cube', location=
bevel(name='sofa board_9', width=0.28, segments=4)
```

```
# part_10: arm
create_quad(name=['quad_1_10', 'quad_2_10', 'quad_3_10', 'quad_4_10'],
bridge_edge_loops(name='arm_10', profile_name=['quad_1_10', 'quad_2_10
bevel(name='arm_10', width=0.06, segments=2)
```

```
# part_11: cushion
create_primitive(name='cushion_11', primitive_type='cube', location=
bevel(name='cushion_11', width=0.28, segments=4)
```

```
# part_12: cushion
create_curve(name=['curve_1_12', 'curve_2_12', 'curve_3_12', 'curve_4_
bridge_edge_loops(name='cushion_12', profile_name=['curve_1_12', 'curv
```

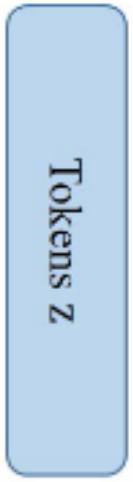
.....

Step 3

Inference Model Overview

Model

shape tokenizer

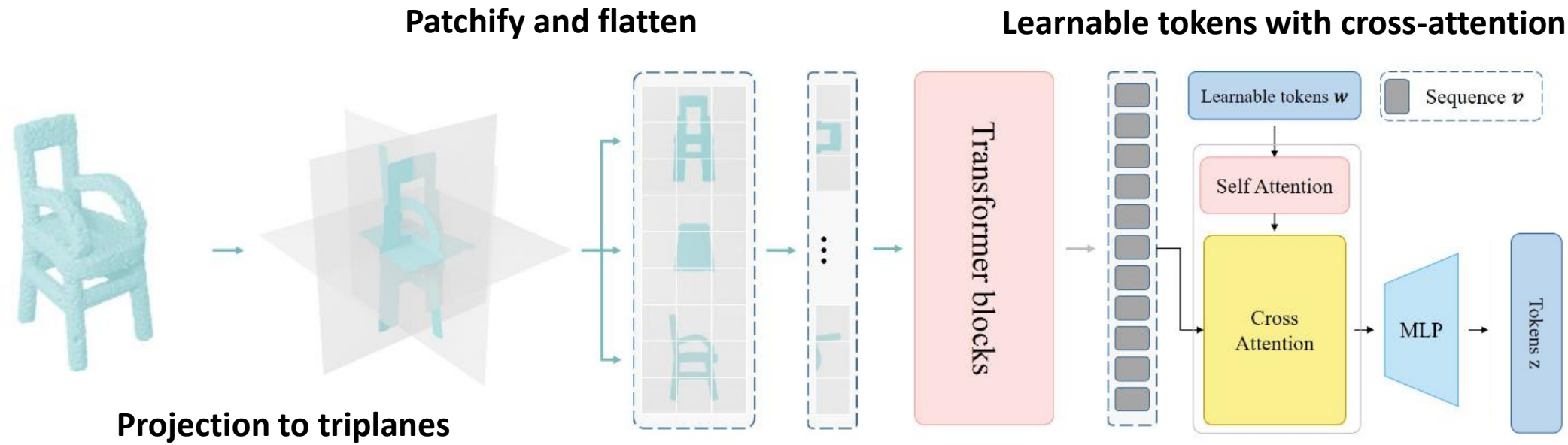


code **generation model**

```
# object name: sofa

# part_1: leg
create_circle(name='circle_1', radius=0.01, center='MEDIAN')
create_curve(name='leg_1', profile_name='circle_1', control_points=[[-0.39

•
•
•

# part_6: leg
create_circle(name='circle_6', radius=0.01, center='MEDIAN')
create_curve(name='leg_6', profile_name='circle_6', control_points=[[0.39

# part_7: arm
create_quad(name=['quad_1_7', 'quad_2_7', 'quad_3_7', 'quad_4_7'], control_
bridge_edge_loops(name='arm_7', profile_name=['quad_1_7', 'quad_2_7', 'quad
bevel(name='arm_7', width=0.06, segments=2)

# part_8: back sofa board
create_primitive(name='back sofa board_8', primitive_type='cube', locatio
bevel(name='back sofa board_8', width=0.27, segments=4)

# part_9: sofa board
create_primitive(name='sofa board_9', primitive_type='cube', location=[
bevel(name='sofa board_9', width=0.28, segments=4)

# part_10: arm
create_quad(name=['quad_1_10', 'quad_2_10', 'quad_3_10', 'quad_4_10'],
bridge_edge_loops(name='arm_10', profile_name=['quad_1_10', 'quad_2_10
bevel(name='arm_10', width=0.06, segments=2)

# part_11: cushion
create_primitive(name='cushion_11', primitive_type='cube', location=
bevel(name='cushion_11', width=0.28, segments=4)

# part_12: cushion
create_curve(name=['curve_1_12', 'curve_2_12', 'curve_3_12', 'curve_4_
bridge_edge_loops(name='cushion_12', profile_name=['curve_1_12', 'curv
```

Model

shape tokenizer



Model

code generation model

Llama-3.2-1B with LoRA finetuning

```
# object name: sofa
# part_1: leg
create_circle(name='circle_1', radius=0.01, center='MEDIAN')
create_curve(name='leg_1', profile_name='circle_1', control_points=[[-0.39
```

```
# part_6: leg
create_circle(name='circle_6', radius=0.01, center='MEDIAN')
create_curve(name='leg_6', profile_name='circle_6', control_points=[[0.39
```

```
# part_7: arm
create_quad(name=['quad_1_7', 'quad_2_7', 'quad_3_7', 'quad_4_7'], control_
bridge_edge_loops(name='arm_7', profile_name=['quad_1_7', 'quad_2_7', 'quad
bevel(name='arm_7', width=0.06, segments=2)
```

```
# part_8: back sofa board
create_primitive(name='back sofa board_8', primitive_type='cube', locatio
bevel(name='back sofa board_8', width=0.27, segments=4)
```

```
# part_9: sofa board
create_primitive(name='sofa board_9', primitive_type='cube', location=[
bevel(name='sofa board_9', width=0.28, segments=4)
```

```
# part_10: arm
create_quad(name=['quad_1_10', 'quad_2_10', 'quad_3_10', 'quad_4_10'],
bridge_edge_loops(name='arm_10', profile_name=['quad_1_10', 'quad_2_10
bevel(name='arm_10', width=0.06, segments=2)
```

```
# part_11: cushion
create_primitive(name='cushion_11', primitive_type='cube', location=
bevel(name='cushion_11', width=0.28, segments=4)
```

```
# part_12: cushion
create_curve(name=['curve_1_12', 'curve_2_12', 'curve_3_12', 'curve_4_
bridge_edge_loops(name='cushion_12', profile_name=['curve_1_12', 'curv
```

Experiment

Method	CD($\times 10^{-2}$)↓						IoU (%) ↑					
	Lamp	Chair	Sofa	TableDining	Toilet	All	Lamp	Chair	Sofa	TableDining	Toilet	All
Shape2Prog	32.04	5.19	3.05	1.77	5.73	4.50	18.67	27.83	54.88	62.17	51.17	42.38
PLAD	6.00	2.45	2.73	4.06	2.53	2.82	57.26	36.96	44.98	43.19	39.64	48.42
Ours	**0.008**	**0.075**	**0.057**	**0.033**	**0.027**	**0.069**	**81.53**	**77.68**	**90.84**	**82.34**	**86.85**	**83.87**

(Tian et al, 2019) [1]

(Jones et al, 2023) [2]

[1] Tian et al. 2019. Learning to Infer and Execute 3D Shape Programs. International Conference on Learning Representations.
[2] Jones et al. 2023. PLAD: Learning to Infer Shape Programs with Pseudo-Labels and Approximate Distributions

Contributions

- Expressive Blender **API** Library

  Embedded DSL for procedural modeling (primitives, patterns, sweeps, lofts).

- Large-Scale **Dataset** Pipeline

  Synthesized 10M+ part & 1M object point cloud–code pairs.

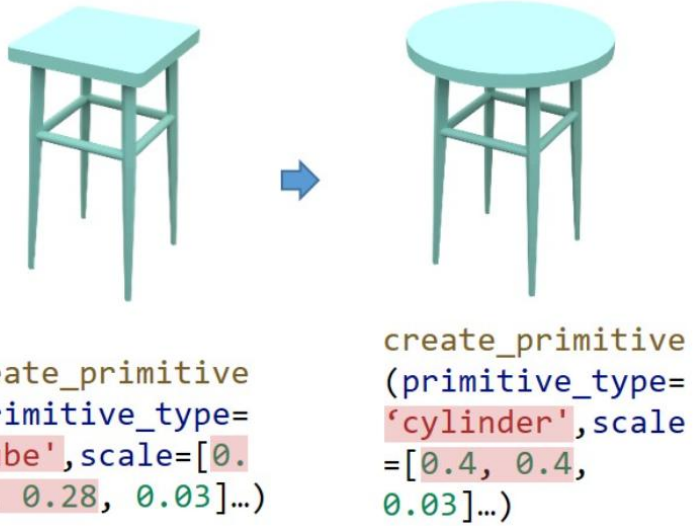
- Shape-to-Code Inference **Model**

  Encoder (shape tokenizer) + decoder (finetuned LLM) → executable code, enabling reconstruction, editing, and reasoning.

Contributions

- A new representation paradigm: one that makes 3D shapes **editable**, **interpretable**, and semantically meaningful — all thanks to code."

Code-Based Editing



Semantic Reasoning