CSC2457 3D & Geometric Deep Learning

ShapeAssembly: Learning to Generate Programs for 3D Shape Structure Synthesis

R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang,
Paul Guerrero, Niloy J. Mitra, Daniel Ritchie

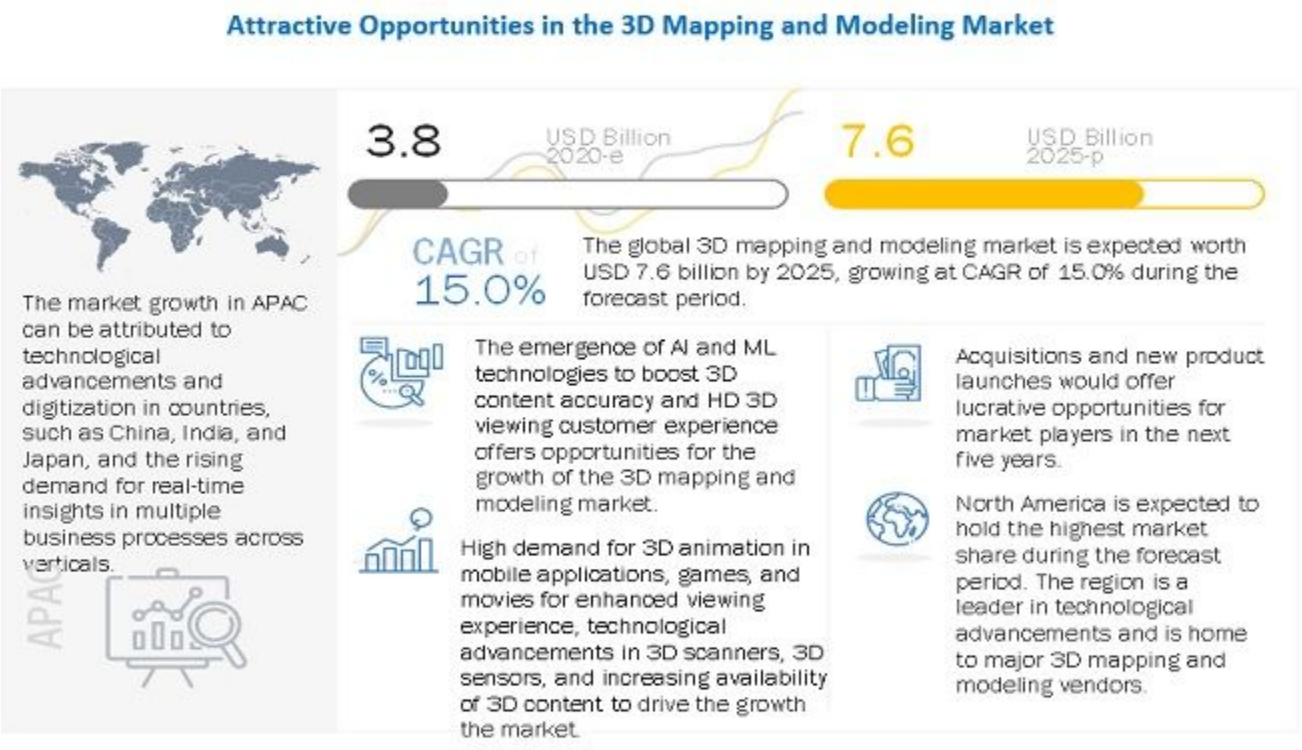
2021-02-09

Presenter: Zhoujie Zhao

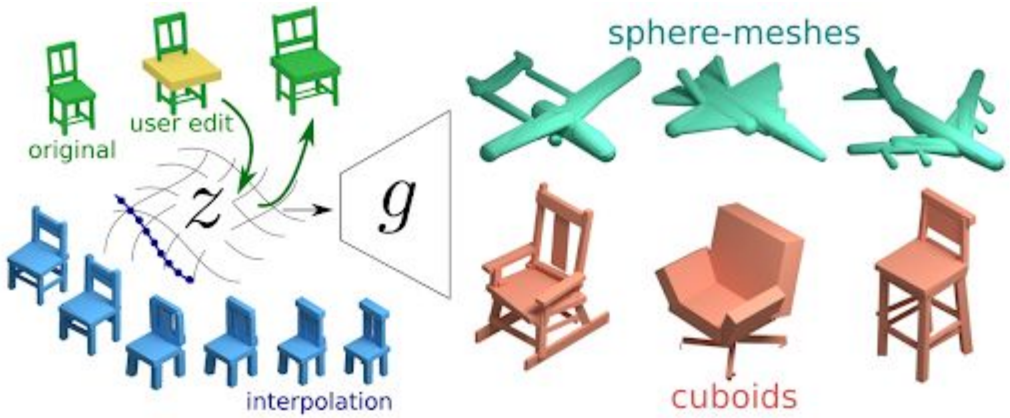
Instructor: Animesh Garg

UNIVERSITY OF TORONTO

Motivation



- Increasing demand for (high-quality) 3D objects
- The craft of 3d modeling remains difficult and time consuming



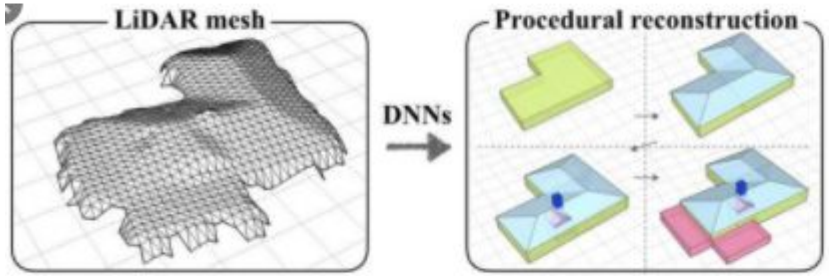
- One promising way: generative models of 3D shapes

- Ideal model: plausible output geometry, a wide range of shape, interpretable representation

- Procedural models: authoring a good procedural model from scratch is difficult

  - Procedural Modeling of Buildings (2006)

  -Procedural Modeling of a Building from a Single Image (2018,2016)



- Deep generative models: implausible geometry, hard to edit or manipulate

  -ComplementMe:Weakly-Supervised Component Suggestions for 3D Modeling (2017)

  -StructureNet: Hierarchical Graph Networks for 3D Shape Generation (2019)

Contributions

Insight: Procedural models and deep generative models have complementary strengths

In this paper, the authors proposed:

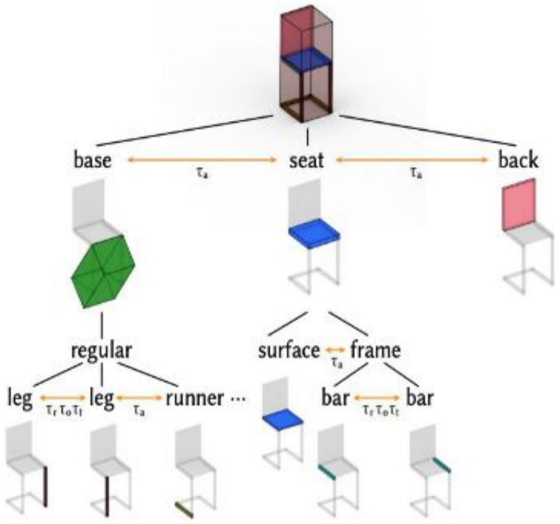
- An Assembly language for shapes, allowing the procedural specification of shape structures represented as connected part assemblies.
- A deep generative model for ShapeAssembly programs, coupling the ease-of-training and variability of generative networks with the precision and editability of procedural representations.

Problem Setting

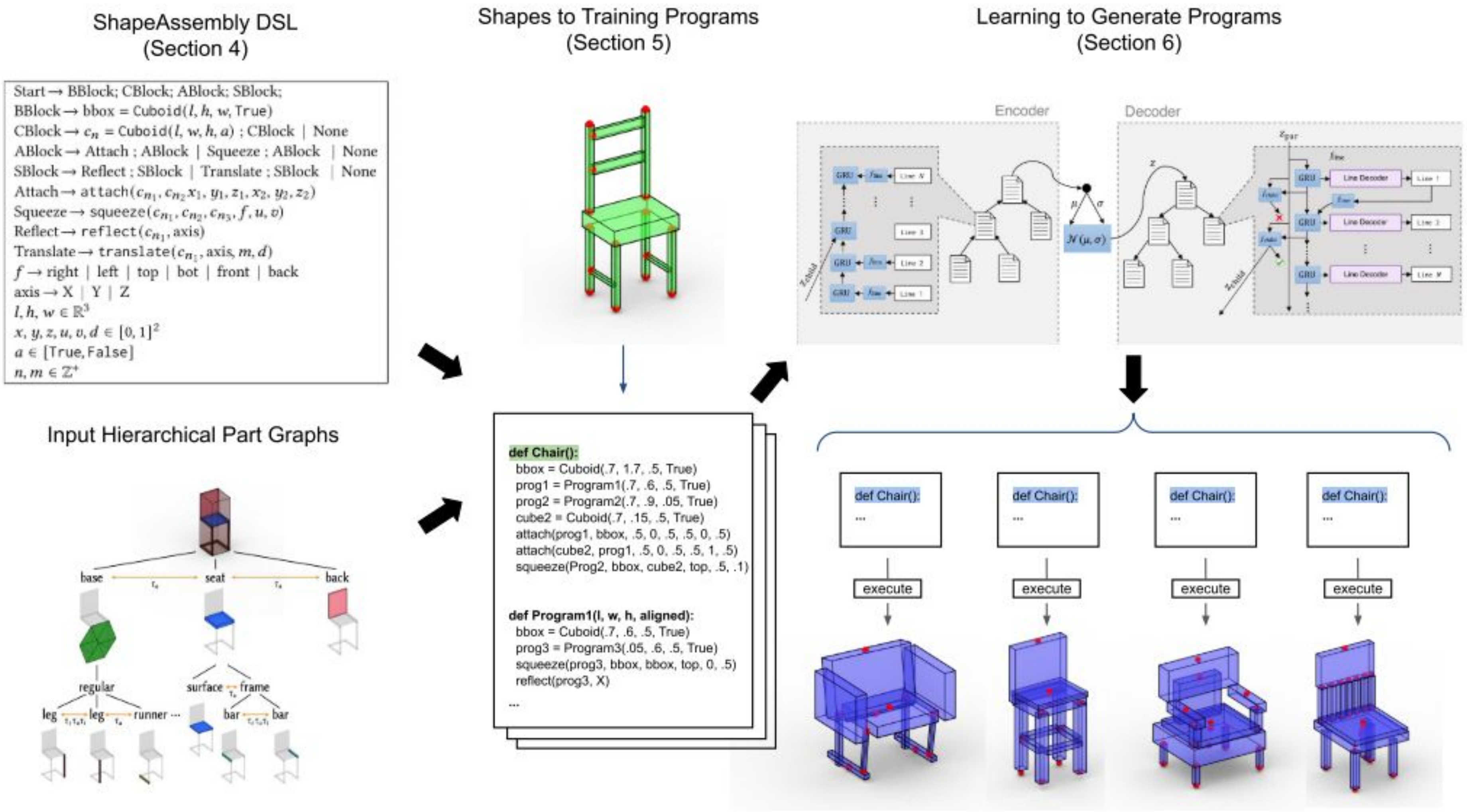
Input: dataset of hierarchical 3D graphs

Output: novel 3D shapes

- Uses hierarchical sequence VAE to generate a DSL program
- Uses that program to generate the ultimate 3D shape output

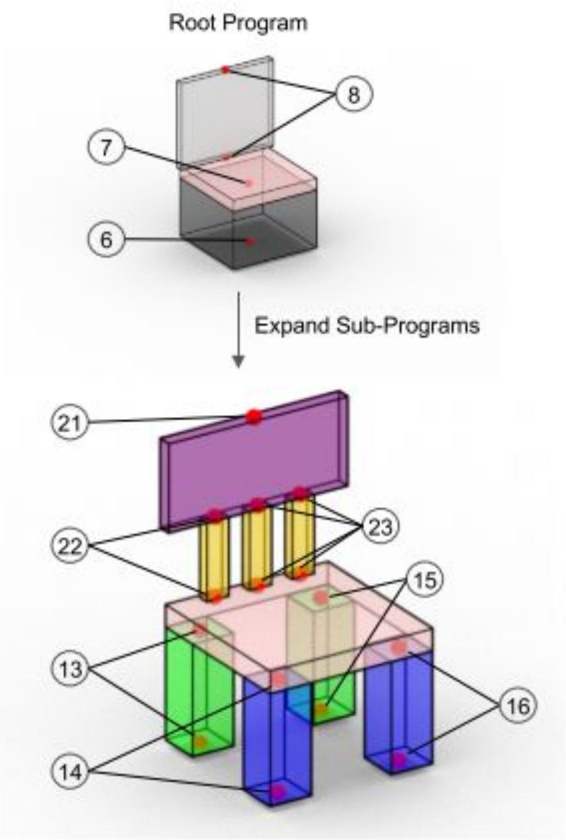


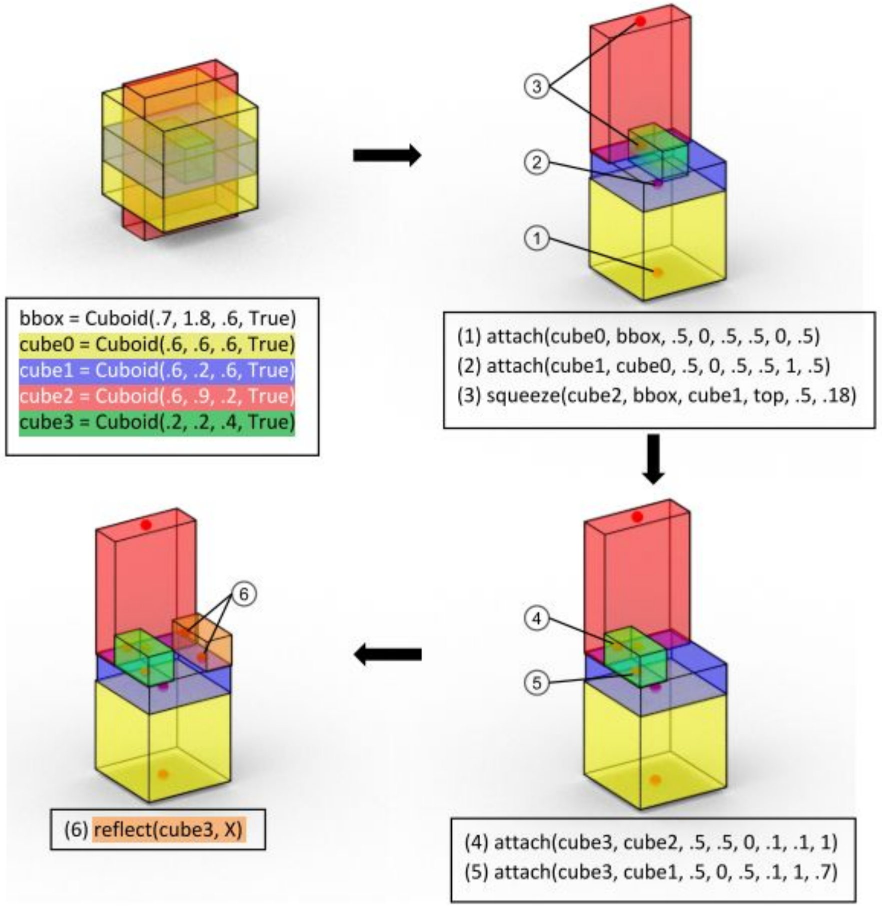
Approach


ShapeAssembly DSL
(Section 4)
Start → BBlock; CBlock; ABlock; SBlock;
BBlock → bbox = Cuboid(l, h, w, True)
CBlock → c_n = Cuboid(l, w, h, a) ; CBlock | None
ABlock → Attach ; ABlock | Squeeze ; ABlock | None
SBlock → Reflect ; SBlock | Translate ; SBlock | None
Attach → attach(c_n1, c_n2 x1, y1, z1, x2, y2, z2)
Squeeze → squeeze(c_n1, c_n2, c_n3, f, u, v)
Reflect → reflect(c_n1, axis)
Translate → translate(c_n1, axis, m, d)
f → right | left | top | bot | front | back
axis → X | Y | Z
l, h, w ∈ ℝ^3
x, y, z, u, v, d ∈ [0, 1]^2
a ∈ [True, False]
n, m ∈ ℤ+
Input Hierarchical Part Graphs
base
seat
back
regular
surface
frame
leg
leg
runner ...
bar
bar
Shapes to Training Programs
(Section 5)
def Chair():
bbox = Cuboid(.7, 1.7, .5, True)
prog1 = Program1(.7, .6, .5, True)
prog2 = Program2(.7, .9, .05, True)
cube2 = Cuboid(.7, .15, .5, True)
attach(prog1, bbox, .5, 0, .5, .5, 0, .5)
attach(cube2, prog1, .5, 0, .5, .5, 1, .5)
squeeze(Prog2, bbox, cube2, top, .5, .1)
def Program1(l, w, h, aligned):
bbox = Cuboid(.7, .6, .5, True)
prog3 = Program3(.05, .6, .5, True)
squeeze(prog3, bbox, bbox, top, 0, .5)
reflect(prog3, X)
...
Learning to Generate Programs
(Section 6)
Encoder
Decoder
GRU
𝒩(μ, σ)
Line Decoder
def Chair():
...
execute

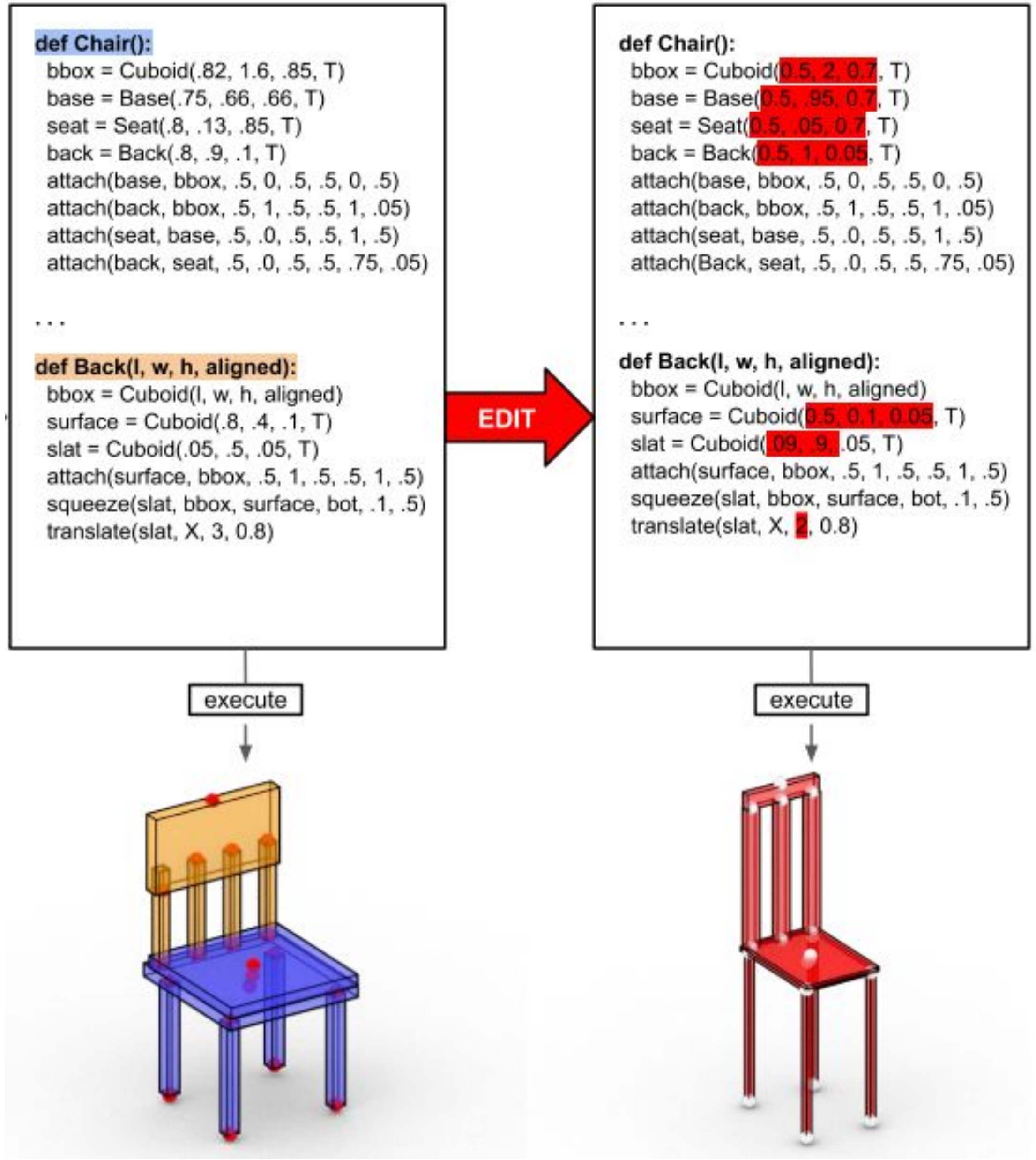
ShapeAssembly

```
1.  def Chair():
2.    bbox = Cuboid(1, 1.5, .8, True)
3     base = Base(.8, .5, .8, True)
4.    cube1 = Cuboid(.8, .1, .8, True)
5.    back = Back(.9, .8, .07, True)
6.    attach(base, bbox, .5, 0, .5, .5, 0, .5)
7.    attach(cube1, base, .5, 0, .5, .5, 1, .5)
8.    squeeze(back, bbox, cube1, top, .5, .05)

9.  def Base(l, w, h, aligned):
10.   bbox = Cuboid(l, w, h, aligned)
11.   cube0 = Cuboid(.2, .5, .2, True)
12    cube1 = Cuboid(.2, .5, .2, True)
13.   squeeze(cube0, bbox, bbox, top, .1, .1)
14.   squeeze(cube1, bbox, bbox, top, .1, .8)
15.   reflect(cube0, X)
      reflect(cube1, X)

17. def Back(l, w, h, aligned):
18.   bbox = Cuboid(l, w, h, aligned)
19.   cube0 = Cuboid(.9, .4, .07, True)
20.   cube1 = Cuboid(.1, .4, .05, True)
21.   attach(cube0, bbox, .5, 1, .5, .5, 1, .5)
22.   squeeze(cube1, bbox, cube0, bot, .3, .5)
23.   translate(cube1, X, 2, .5)
```





ShapeAssembly



ShapeAssembly

- BBlock
- CBlock
- ABlock
- SBlock

Start $\rightarrow$ BBlock; CBlock; ABlock; SBlock;
BBlock $\rightarrow$ bbox = $\texttt{Cuboid}(l,\ h,\ w,\ \texttt{True})$
CBlock $\rightarrow$ c_n = $\texttt{Cuboid}(l,\ w,\ h,\ a)$; CBlock | None
ABlock $\rightarrow$ Attach ; ABlock | Squeeze ; ABlock | None
SBlock $\rightarrow$ Reflect ; SBlock | Translate ; SBlock | None
Attach $\rightarrow$ $\texttt{attach}(c_{n_1},\ c_{n_2},\ x_1,\ y_1,\ z_1,\ x_2,\ y_2,\ z_2)$
Squeeze $\rightarrow$ $\texttt{squeeze}(c_{n_1},\ c_{n_2},\ c_{n_3},\ f,\ u,\ v)$
Reflect $\rightarrow$ $\texttt{reflect}(c_n,\ \text{axis})$
Translate $\rightarrow$ $\texttt{translate}(c_n,\ \text{axis},\ m,\ d)$
$f \rightarrow$ right | left | top | bot | front | back
axis $\rightarrow$ X | Y | Z
$l,\ h,\ w \in \mathbb{R}^+$
$x,\ y,\ z,\ u,\ v,\ d \in [0,\ 1]^2$
$a \in [\texttt{True},\ \texttt{False}]$
$n,\ m \in \mathbb{Z}^+$

Program Extraction Procedure

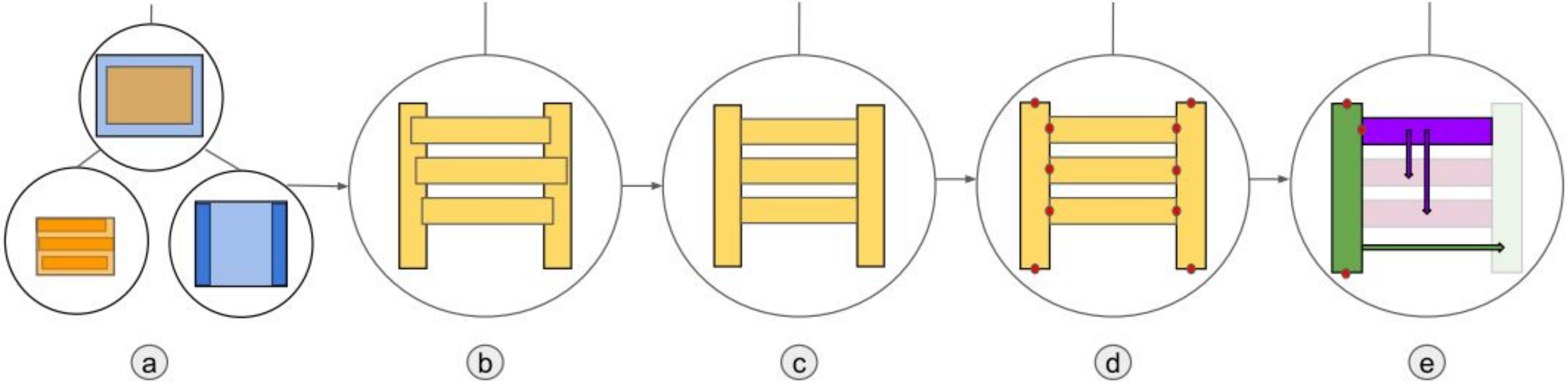
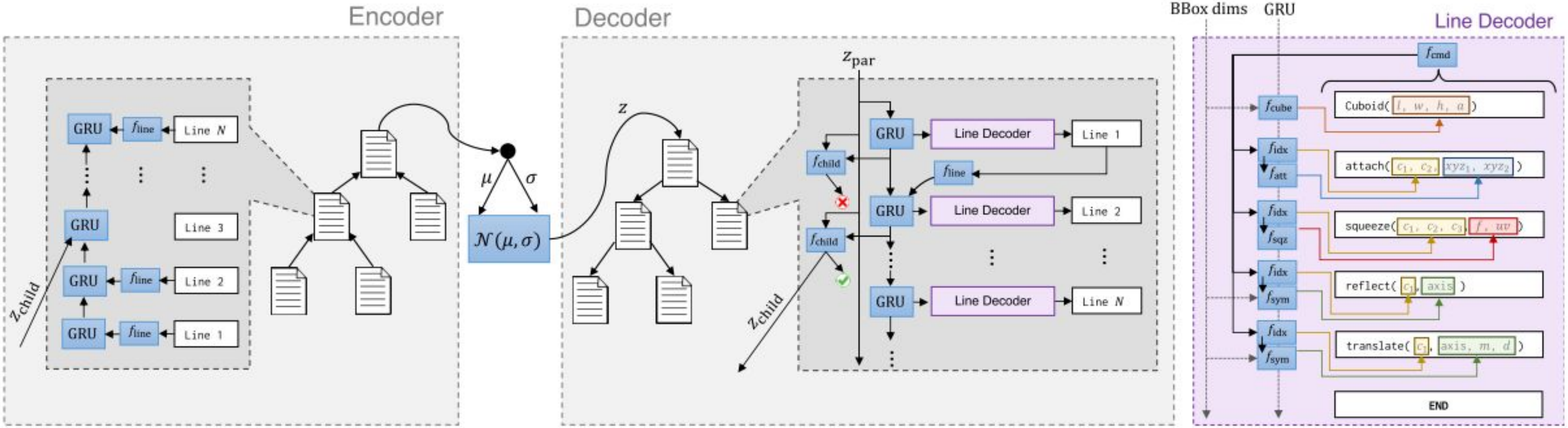
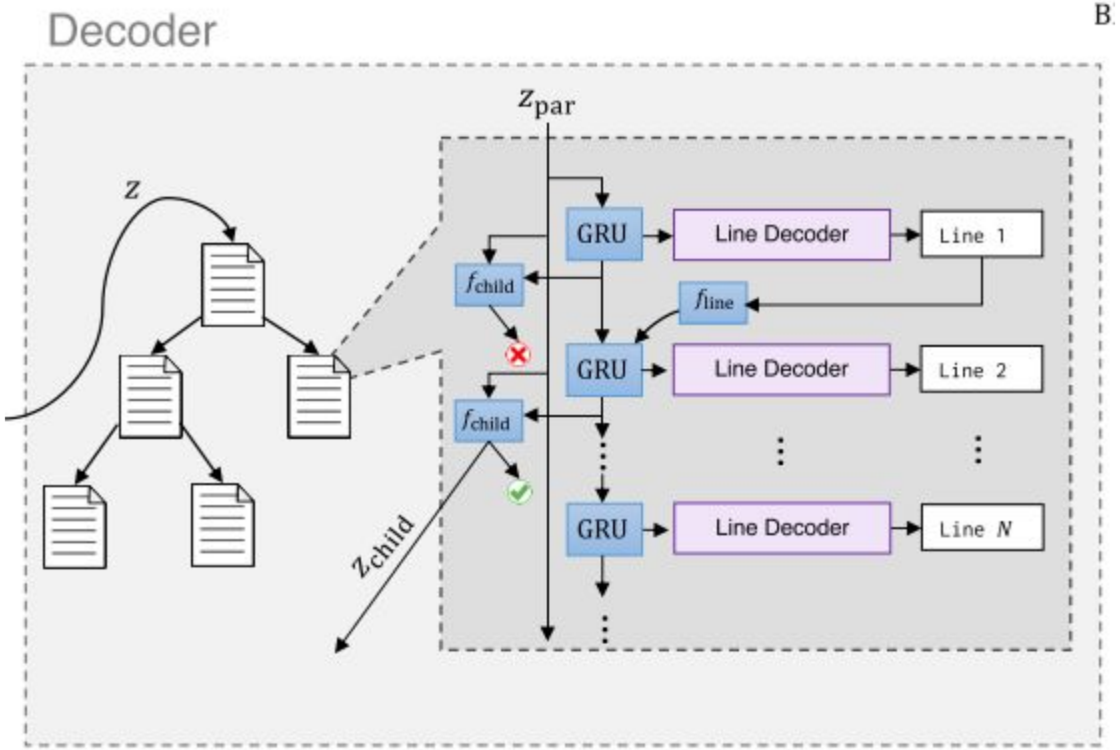


Fig. 5. The steps of our program extraction pipeline. *(a)* Fragment of an input hierarchical part graph showing chair back (parent node), chair back frame (blue child), and chair back surface (orange child). *(b)* Locally flattening the hierarchy so that physically interacting leaf parts become siblings. *(c)* Shortening leaf parts that intersect other leaf parts. *(d)* Locating attachment points between parts. *(e)* Forming leaf parts into symmetry groups.

Generative Model



Generative Model



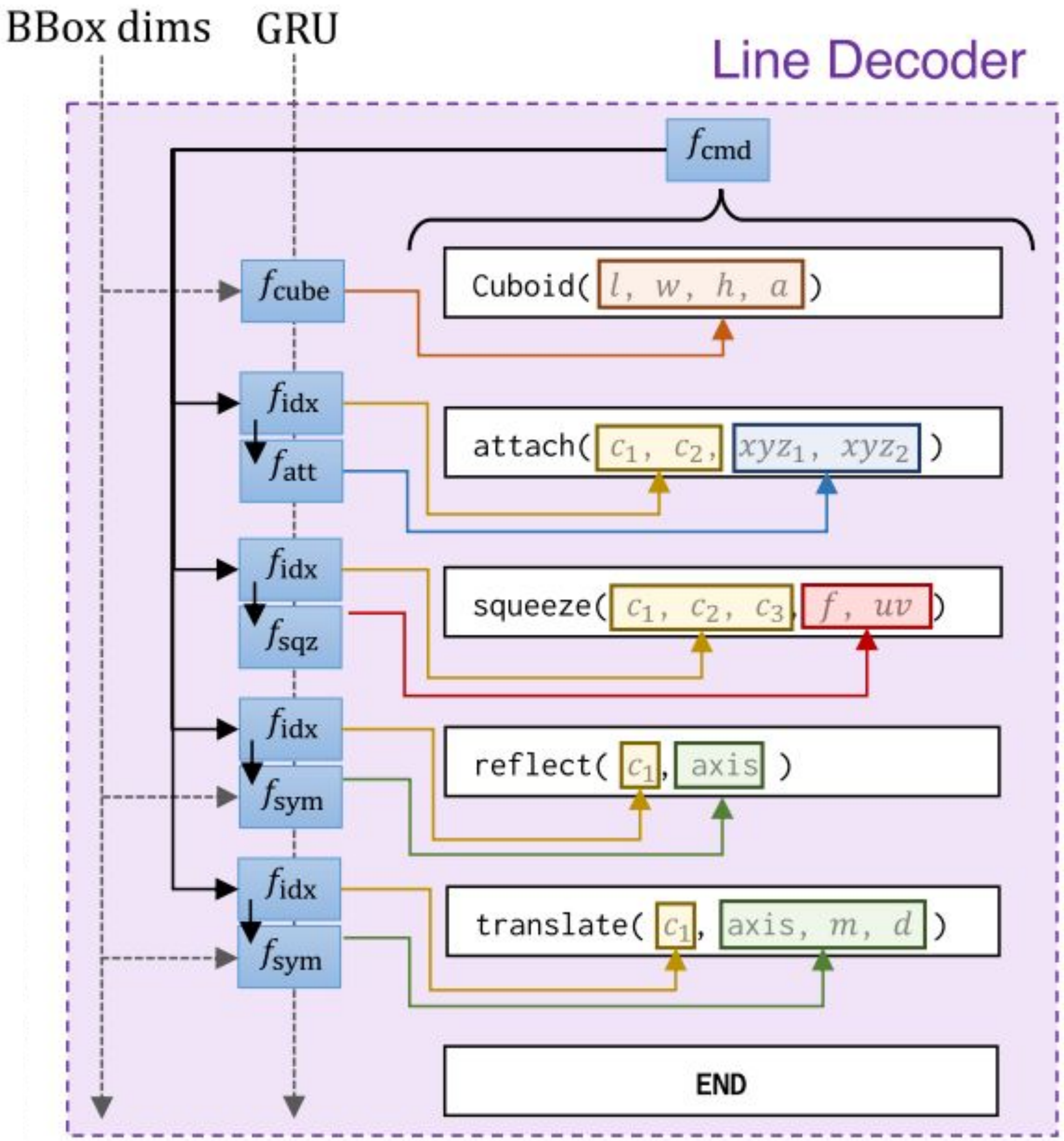
Gated Recurrent Unit (GRU)

f_{child}

z_{child}

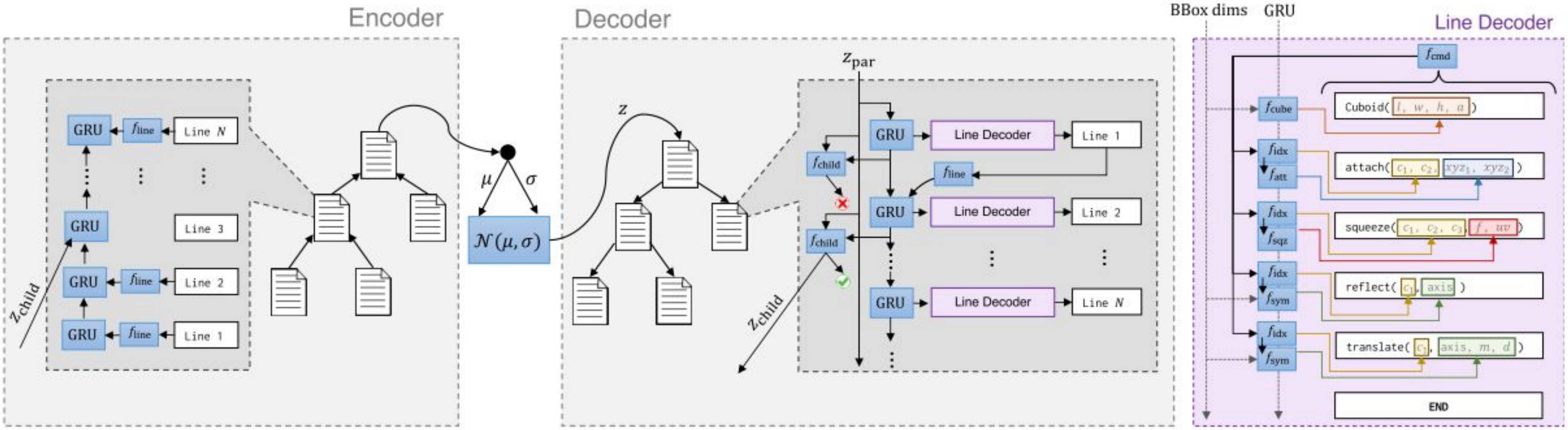
Generative Model

- $f_{\mathbf{cmd}}$: (7)
- $f_{\mathbf{cube}}$: (4)
- $f_{\mathbf{idx}}$: (11×3)
- $f_{\mathbf{att}}$: (3×2)
- $f_{\mathbf{sqz}}$: (8)
- $f_{\mathbf{sym}}$: (5)



Generative Model

Gated Recurrent Unit (GRU)



- f_{cmd}: (7)
- f_{cube}: (4)
- f_{idx}: (11×3)
- f_{att}: (3×2)
- f_{sqz}: (8)
- f_{sym}: (5)

f_{child} z_{child}

Results and Discussion

Novel Shape Synthesis

PartNet dataset

Results and Discussion

Analysis of Shape Quality

- **Rootedness ⥣ (% rooted)**
- **Stability ⥣ (% stable)**
- **Realism ⥣ (% fool)**
- **Frechet Distance ⥥ (FD)**
- **Flat**
- **No Order**
- **No Align**
- **No Macros**
- **No Reject**

- **StructureNet**
- **3D-PRNN**

Category	Method	% rooted ⥣	% stable ⥣	% fool ⥣	FD ⥥
Chair	3D-PRNN	73.1	50.9	12.60	39.30
	StructureNet	89.7	74.9	4.04	64.79
	Ours (Flat)	**95.0**	60.0	11.58	77.45
	Ours (No Order)	82.4	58.4	12.36	64.17
	Ours (No Align)	94.6	84.6	**28.68**	29.32
	Ours (No Macros)	92.0	77.9	19.56	36.78
	Ours (No Reject)	92.9	79.7	23.36	**20.63**
	Ours	94.5	**84.7**	25.06	22.34
	Ground Truth	100	88.0	—	—
Table	3D-PRNN	71.2	29.4	2.12	140.07
	StructureNet	94.4	76.8	3.94	173.35
	Ours (Flat)	87.0	66.0	29.84	148.63
	Ours (No Order)	84.5	56.0	27.38	114.10
	Ours (No Align)	92.2	61.5	23.64	**46.64**
	Ours (No Macros)	95.9	85.0	33.16	53.21
	Ours (No Reject)	94.1	76.4	29.20	52.78
	Ours	**96.2**	**85.9**	**33.21**	49.07
	Ground Truth	100	93.1	—	—
Storage	3D-PRNN	44.8	20.8	4.62	94.08
	StructureNet	**96.2**	75.0	5.04	92.85
	Ours (Flat)	95.9	74.0	7.44	81.17
	Ours (No Order)	87.9	63.4	8.70	107.42
	Ours (No Align)	89.7	49.3	11.04	**30.15**
	Ours (No Macros)	87.5	69.9	5.92	72.80
	Ours (No Reject)	94.3	80.9	11.66	31.69
	Ours	95.3	**83.7**	**13.50**	31.72
	Ground Truth	100	87	—	—

Results and Discussion

Program editability

Category	Method	Lines ⇓	Refl ⇑	Trans ⇑	Squeeze ⇑	Total ⇑
			Macros Per Line			
Chair	3D-PRNN	**15.7**	**0.1100**	0.0020	0.0240	0.1430
	StructureNet	27.1	0.0600	0.0004	0.0700	0.1330
	Ours	20.4	0.0880	**0.0054**	**0.0920**	**0.1860**
	Ground Truth	24.4	0.0800	0.0090	0.1130	0.2070
Table	3D-PRNN	**13.1**	**0.1300**	**0.0010**	0.0680	0.1990
	StructureNet	24.8	0.0270	0.0006	0.0620	0.0900
	Ours	19.0	0.0990	0.0002	**0.1440**	**0.2440**
	Ground Truth	20.0	0.0950	0.0050	0.1450	0.2460
Storage	3D-PRNN	22.6	0.0170	0.0060	0.0530	0.0770
	StructureNet	30.7	0.0390	0.0040	0.0770	0.1200
	Ours	**19.8**	**0.0820**	**0.0080**	**0.1440**	**0.2340**
	Ground Truth	24.7	0.0650	0.0147	0.1510	0.2320

More macros can make the program be more concise, easier to edit.

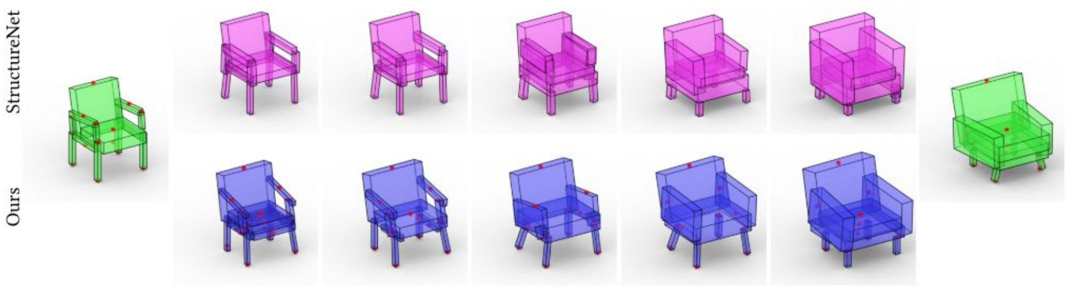
Geometric variability

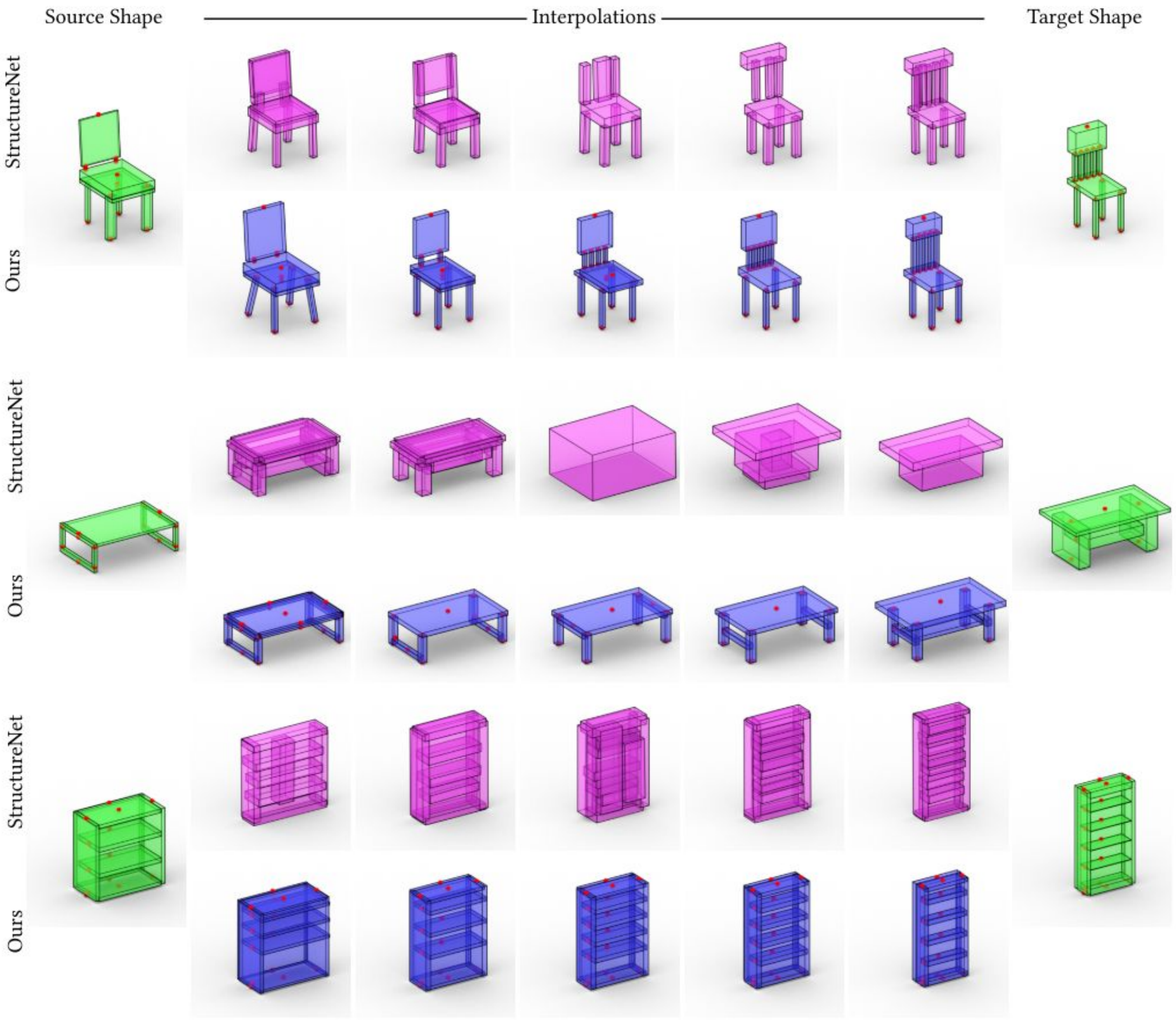
		Generalization	*Coverage*	*Variety*
		NND to Train ⇑	NND from Val ⇓	NND to Self ⇑
Category	Method	CD	CD	CD
Chair	3D-PRNN	**0.111**	0.123	**0.104**
	StructureNet	0.104	0.119	0.087
	Ours	0.108	**0.118**	**0.104**
	Validation	0.105	—	0.114
Table	3D-PRNN	0.095	0.130	0.086
	StructureNet	**0.129**	0.141	0.0925
	Ours	0.101	**0.108**	**0.102**
	Validation	0.09	—	0.099
Storage	3D-PRNN	**0.134**	0.132	**0.119**
	StructureNet	0.129	0.135	0.107
	Ours	0.125	**0.129**	**0.119**
	Validation	0.11	—	0.125

Best on the coverage and variety metrics.

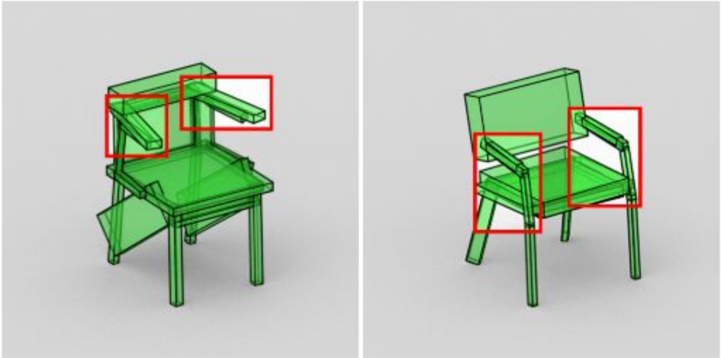
Results and Discussion

Latent Space





Limitations

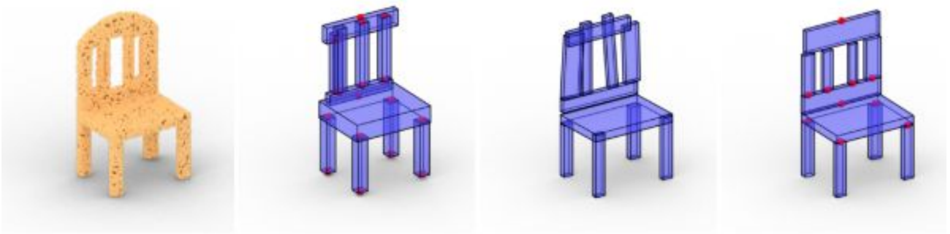


- Assumption of program extraction procedure

- Discard training programs with more than 12 total Cuboid declarations (a trade-off between variability and quality)

- Not guarantee the leaf-to-leaf connectivity

- Only supports cuboids



Contributions (Recap)

Insight: Procedural models and deep generative models have complementary strengths

In this paper, the authors proposed:

- An Assembly language for shapes, allowing the procedural specification of shape structures represented as connected part assemblies.
- A deep generative model for ShapeAssembly programs, coupling the ease-of-training and variability of generative networks with the precision and editability of procedural representations.