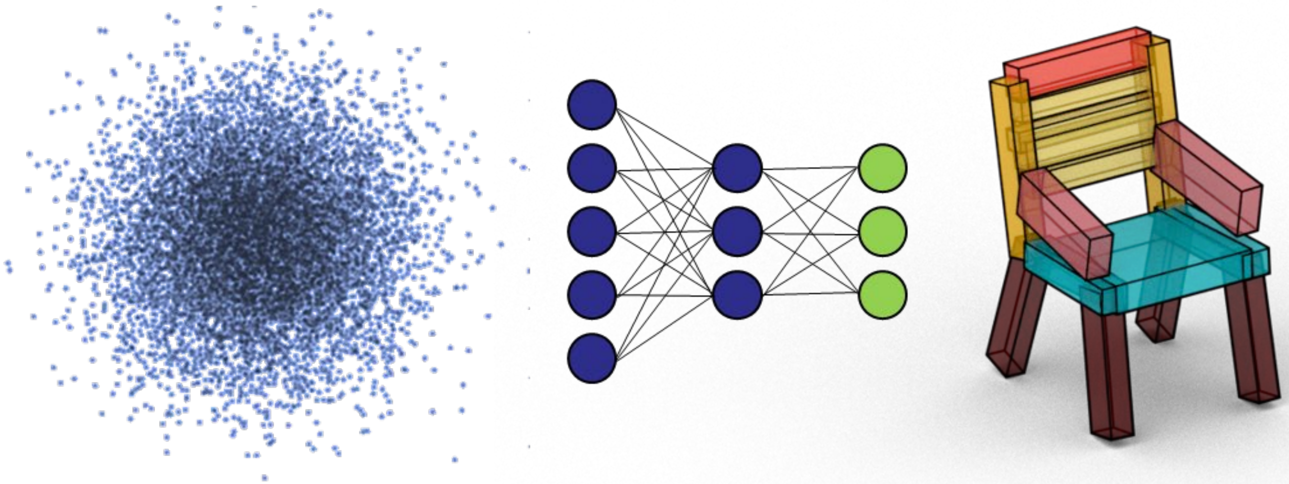
CS348n: Neural Representations and Generative Models for 3D Geometry



Leonidas Guibas
Computer Science Department
Stanford University

Leonidas Guibas Laboratory
Geometric Computing

Class Logistics

- Project proposals due today

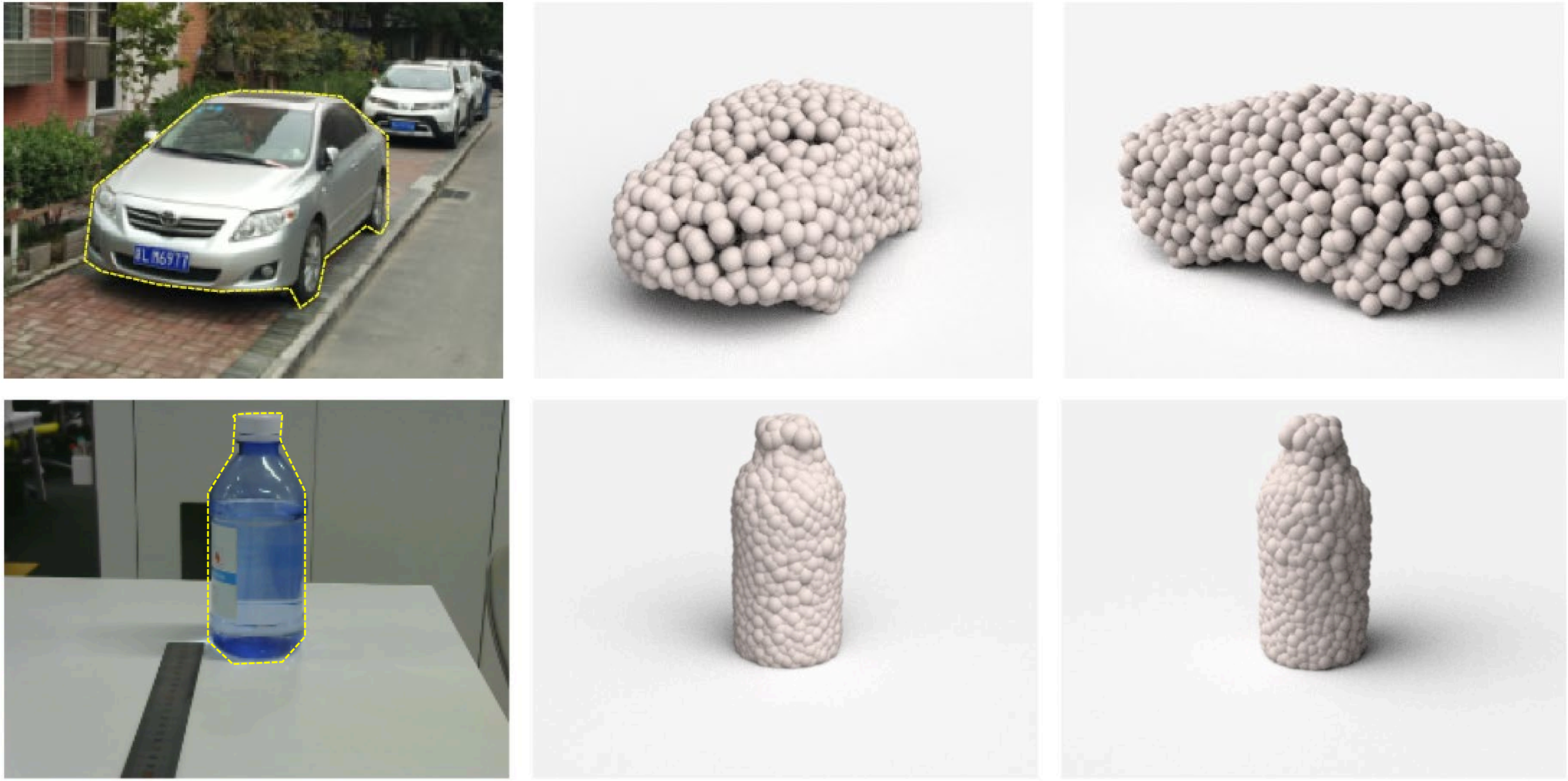
- Next lecture (Wed, Feb 23) on NeRFs will be by Ben Mildenhall and Pratul P. Srinivasan, two of the authors of the original NeRF paper. It will still be virtual on Zoom.

- Because of the special guests, the student presentations for Feb 23 will be moved to Feb 28 (two sets of student presentation then)

- From Feb 28 the class will be physical, in Clark S361

Last Time: Conditional Shape Generation Based on Image Data

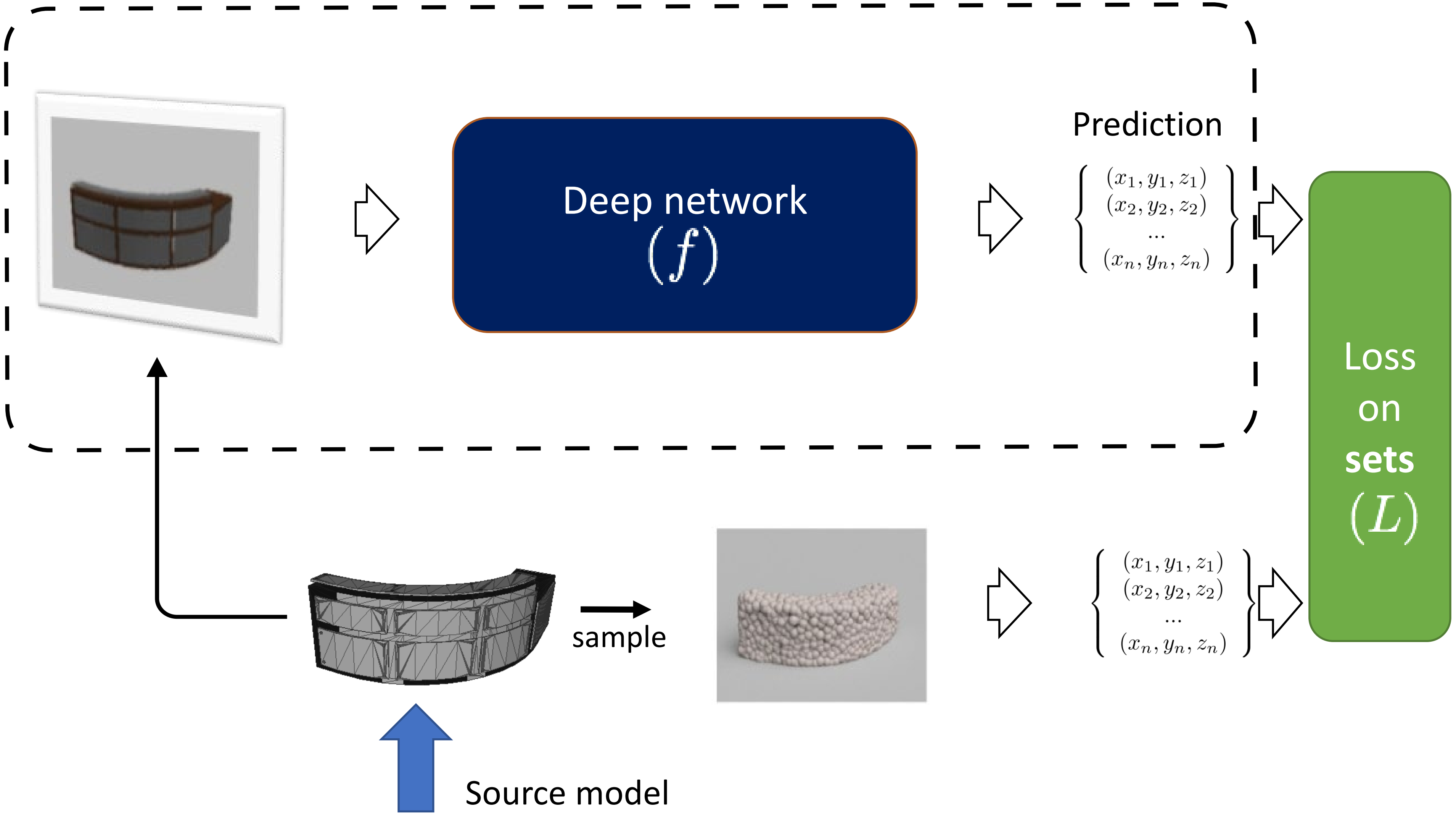
Point Cloud Synthesis from a Single Image



Input

Reconstructed 3D point cloud

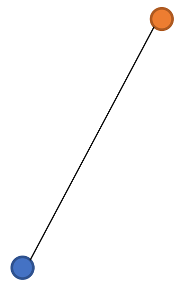
End-to-End Learning



Point Cloud Distance Metrics

Worst case: Hausdorff distance (HD)

$$d_{\mathrm{HD}}(S_1, S_2) = \max\{\max_{x_i \in S_1} \min_{y_j \in S_2} \|x_i - y_j\|, \max_{y_j \in S_2} \min_{x_i \in S_1} \|x_i - y_j\|\}$$

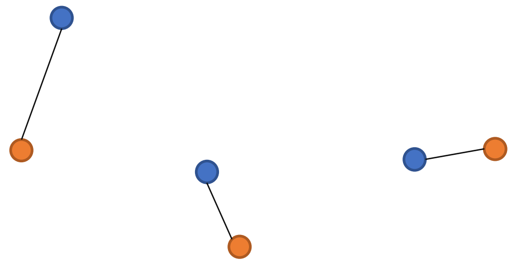


Average case: Chamfer distance (CD)

$$d_{CD}(S_1, S_2) = \frac{1}{n} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{m} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Optimal case: Earth Mover's distance (EMD)

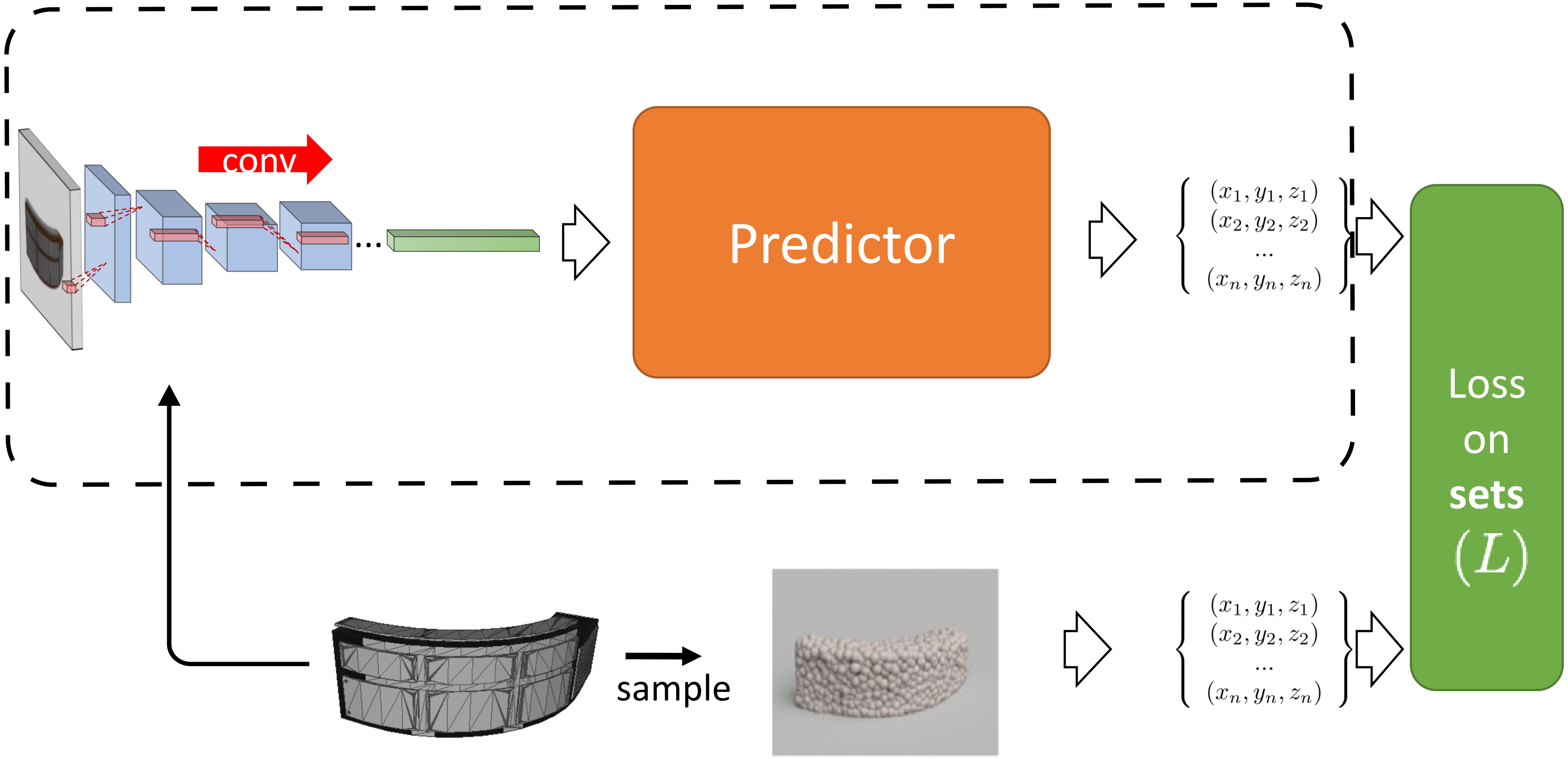
$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$



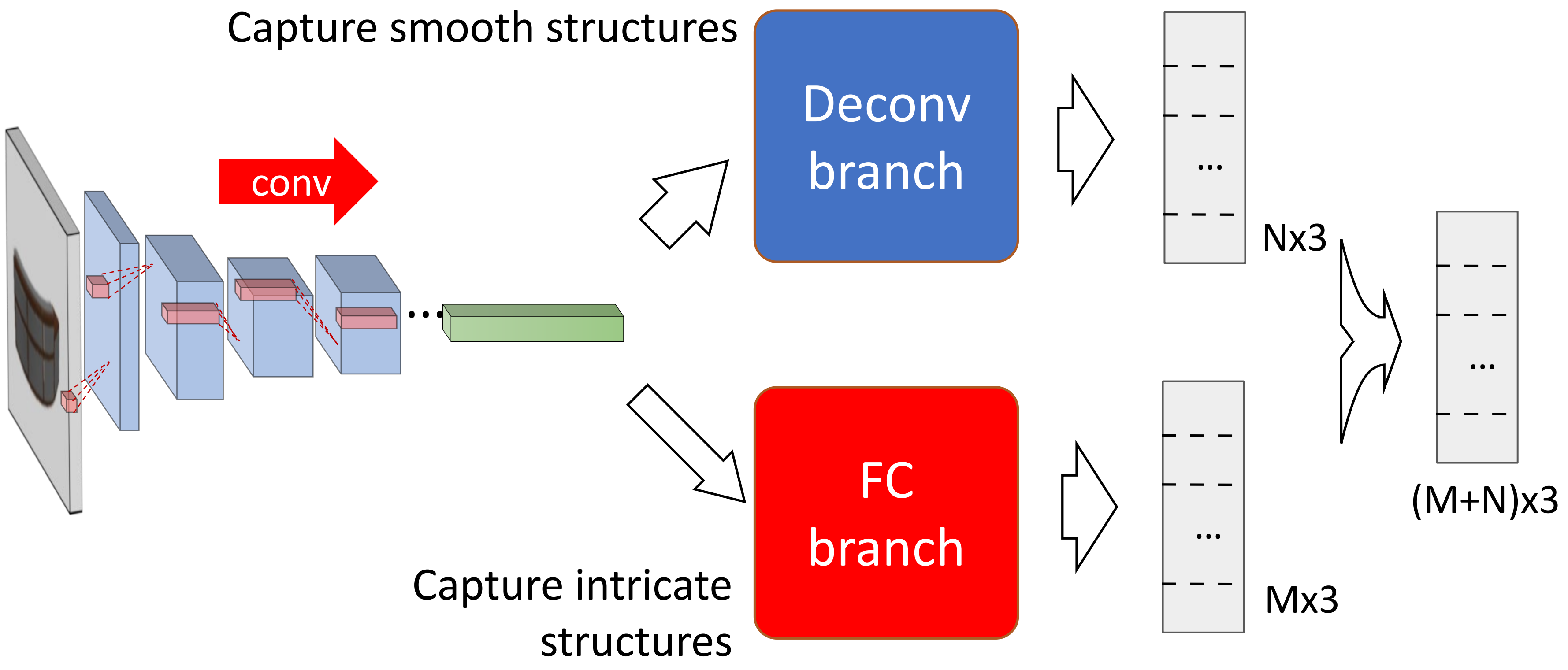
where $\phi : S_1 \to S_2$ is a bijection.

Solves the optimal transportation (bipartite matching) problem!

End-to-End Learning Architecture



Two-Branch Architecture



Set union by array concatenation

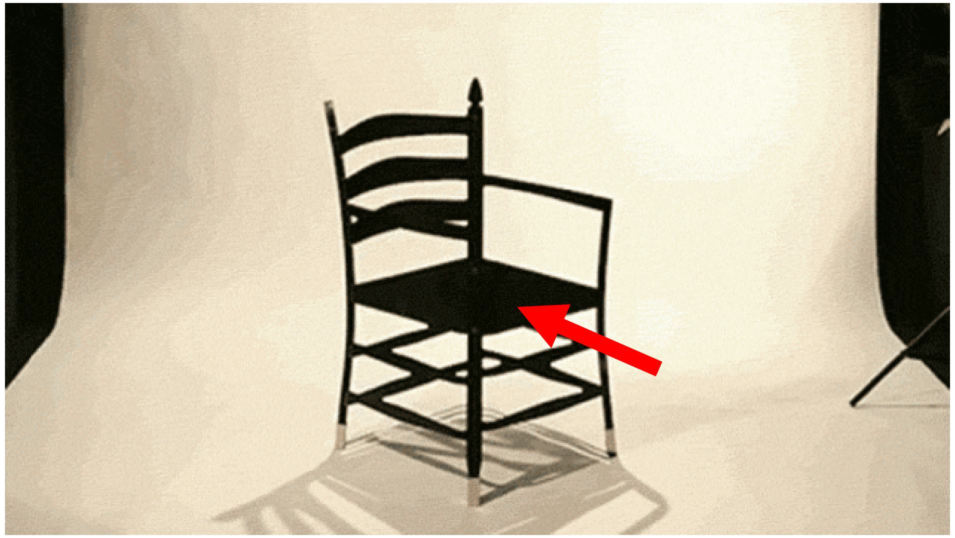
The Two Branches

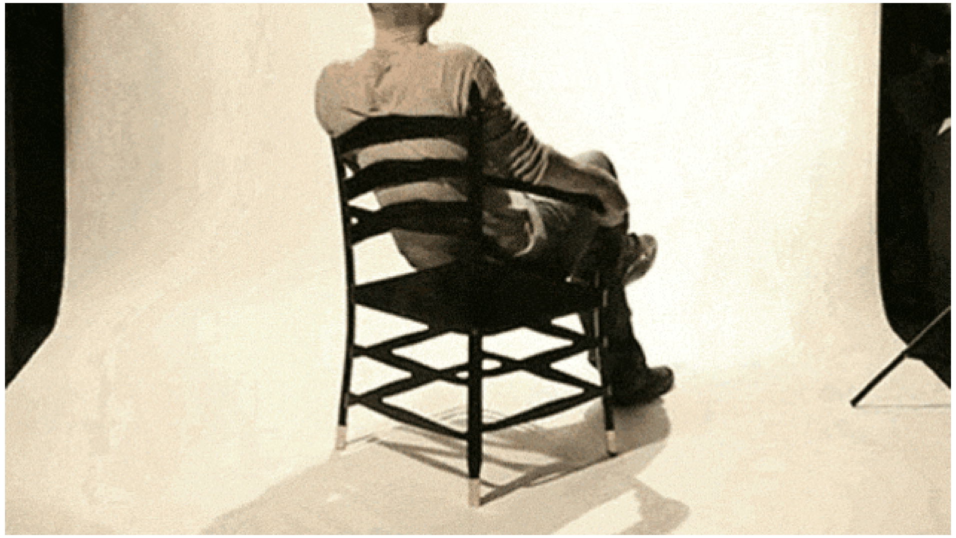
blue: deconv branch – **large, consistent, smooth** structures

red: fully-connected branch – **more intricate** structures

Ambiguity in Object Views

- A fundamental issue: inherent ambiguity in prediction



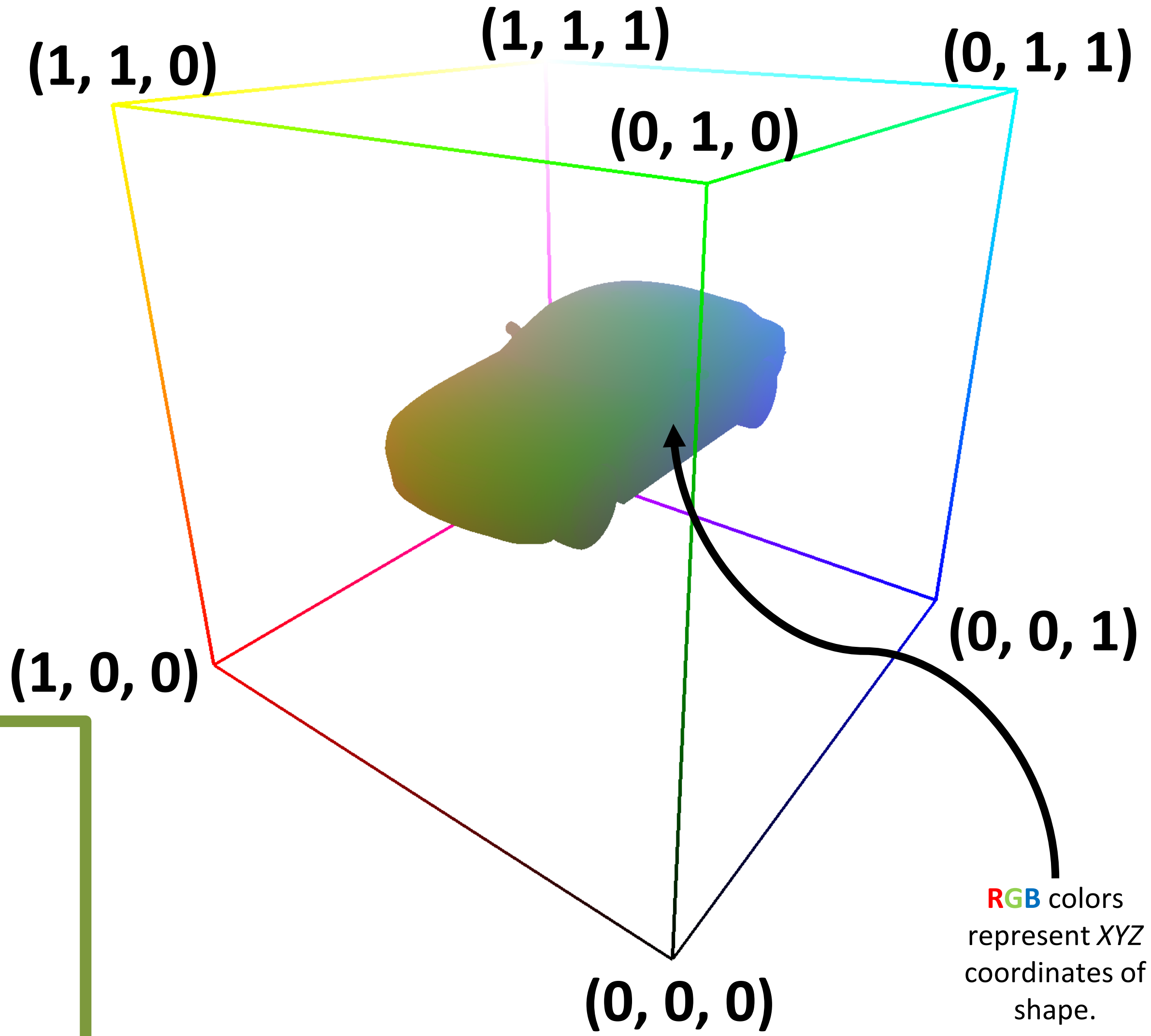


- By loss minimization, the network tends to predict a “**mean shape**” that **averages out** uncertainty

Canonical “Containers” for Object Categories

He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, Leonidas J. Guibas. *Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation*. CVPR 2019.

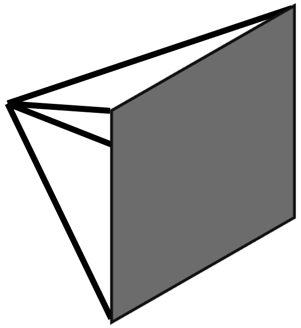
Normalized Object Coordinate Spaces (NOCS)

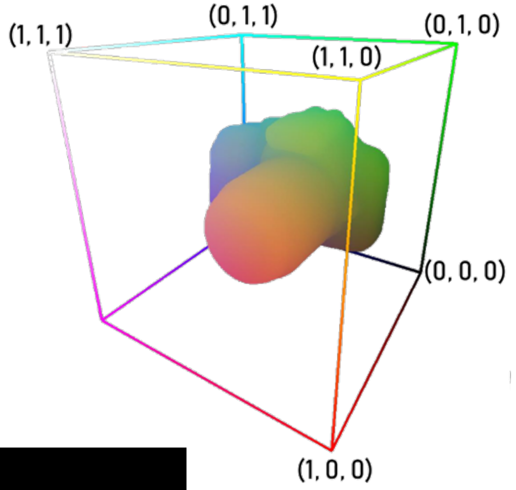


Canonicalize

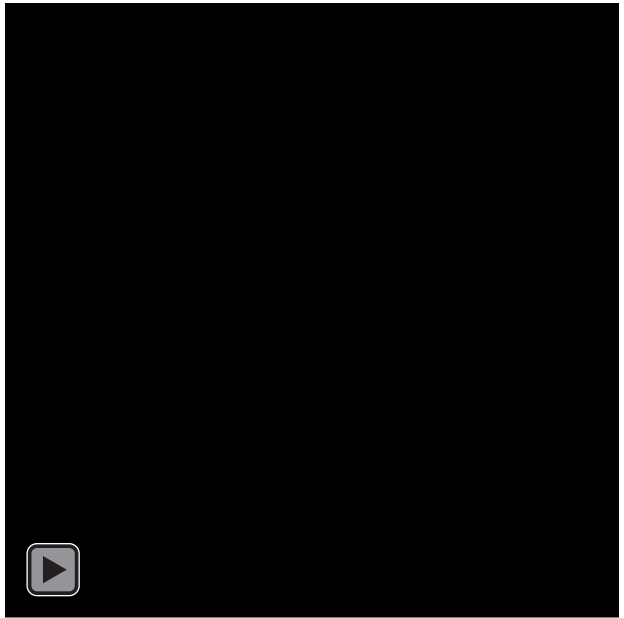
- Position
- Orientation
- Size

NOCS Lifting Map

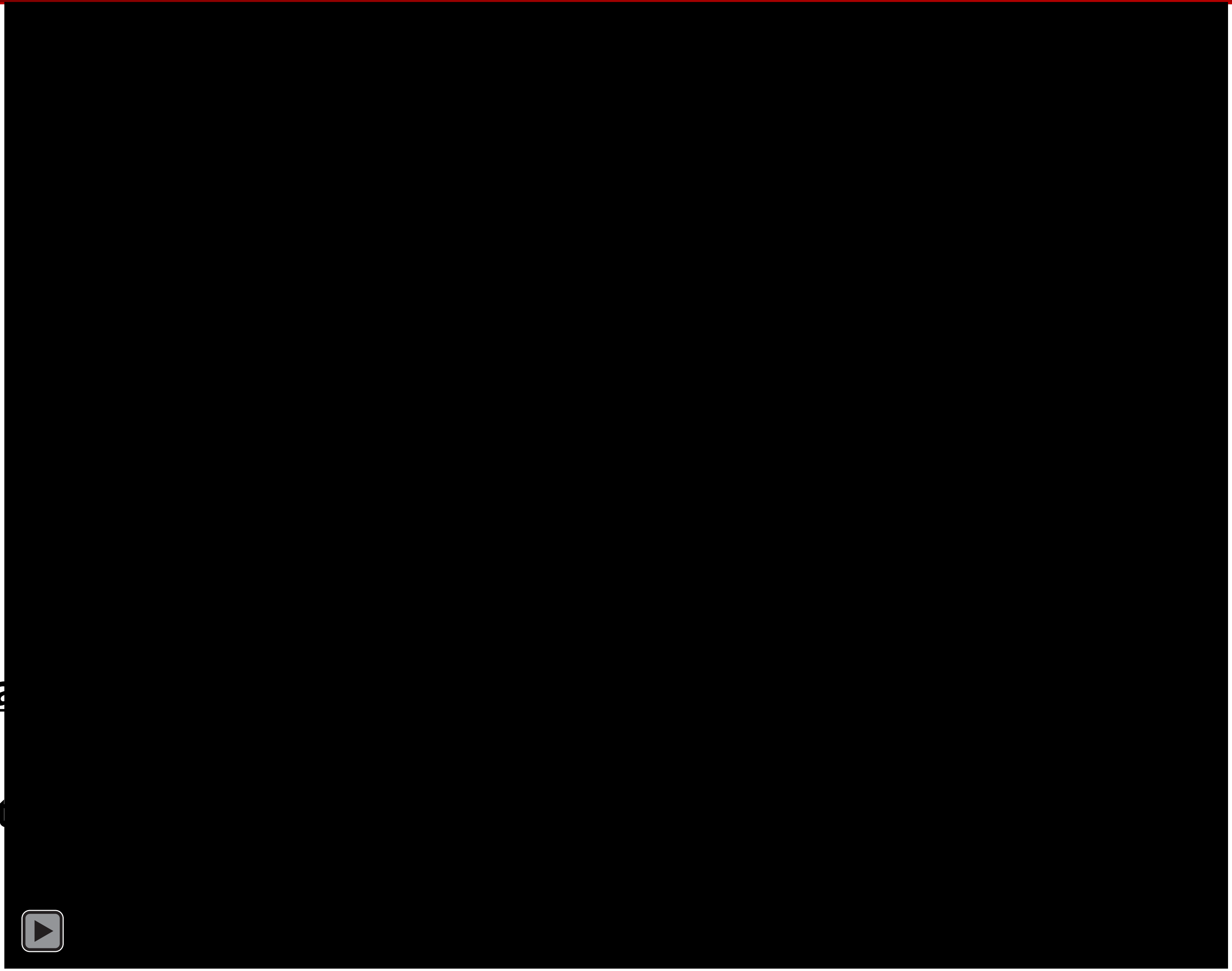




Readout



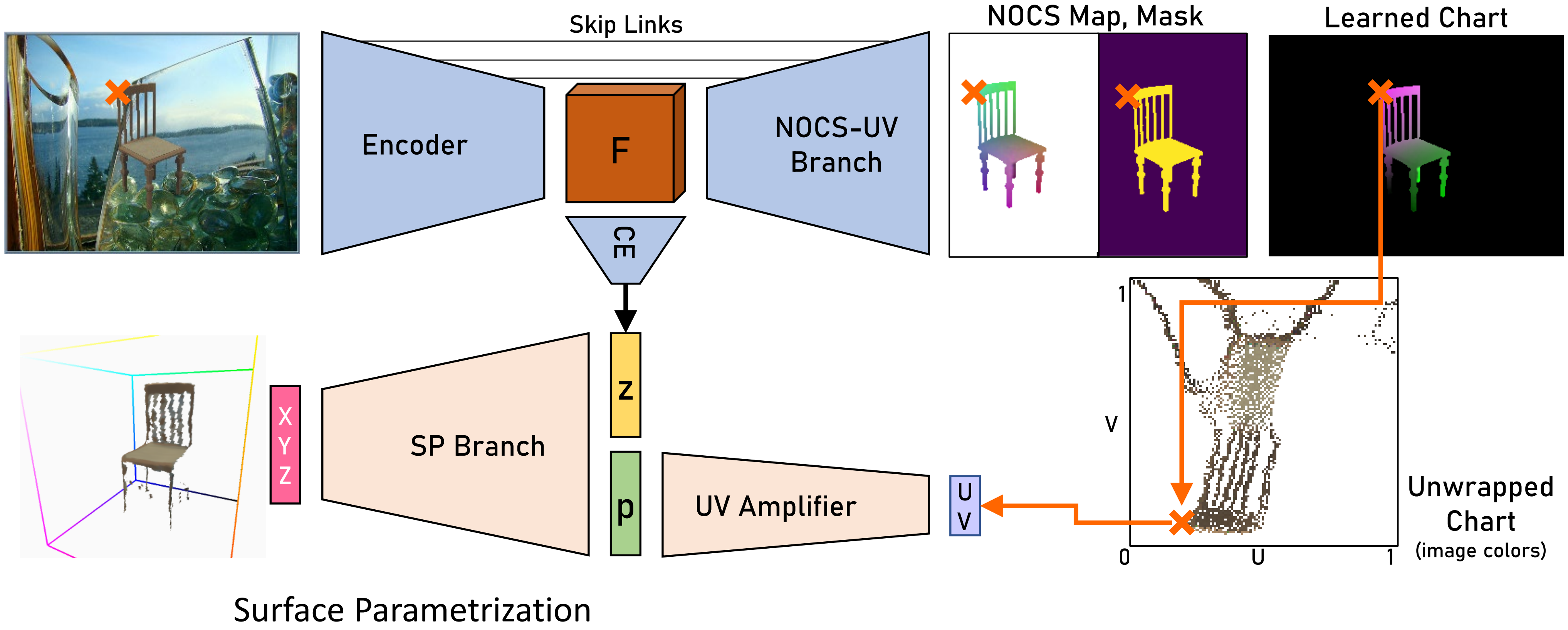
Multi-View NOCS Aggregation



Limita

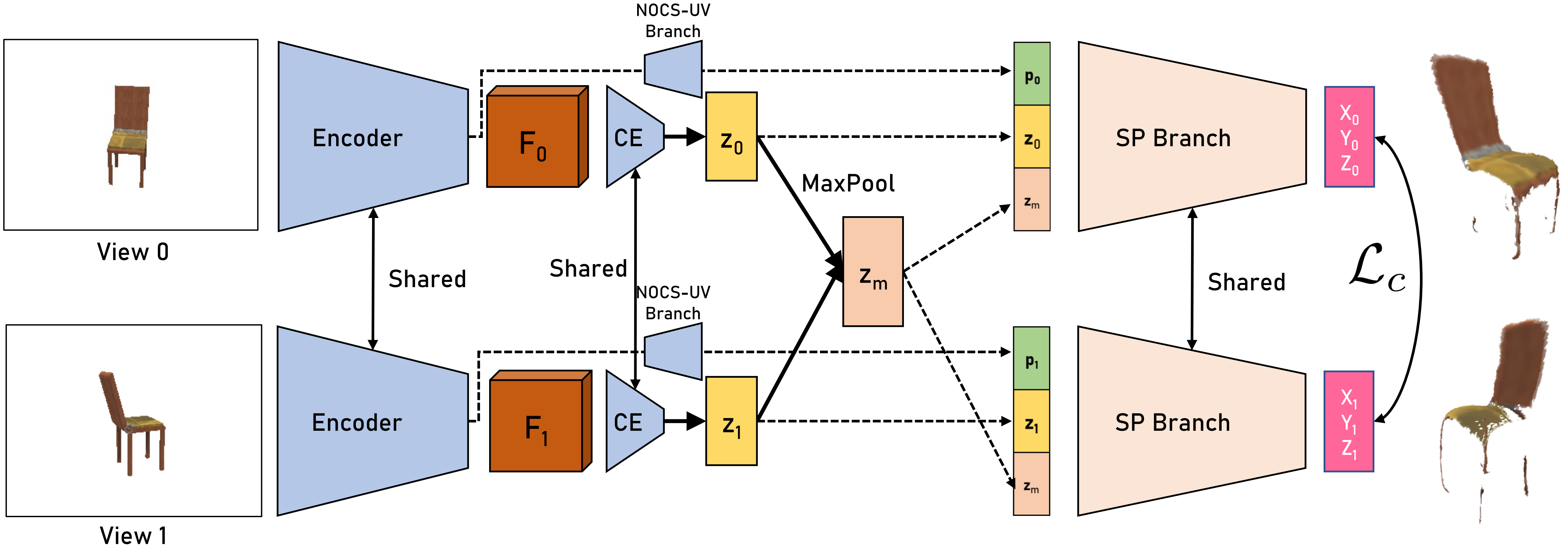
- Set union of
- No surface

Pix2Surf: Single-View Single-Chart



Lei, J., Sridhar, S., Guerrero, P., Sung, M., Mitra, N. and Guibas, L.J. Pix2surf: Learning parametric 3D surface models of objects from images. ECCV 2020.

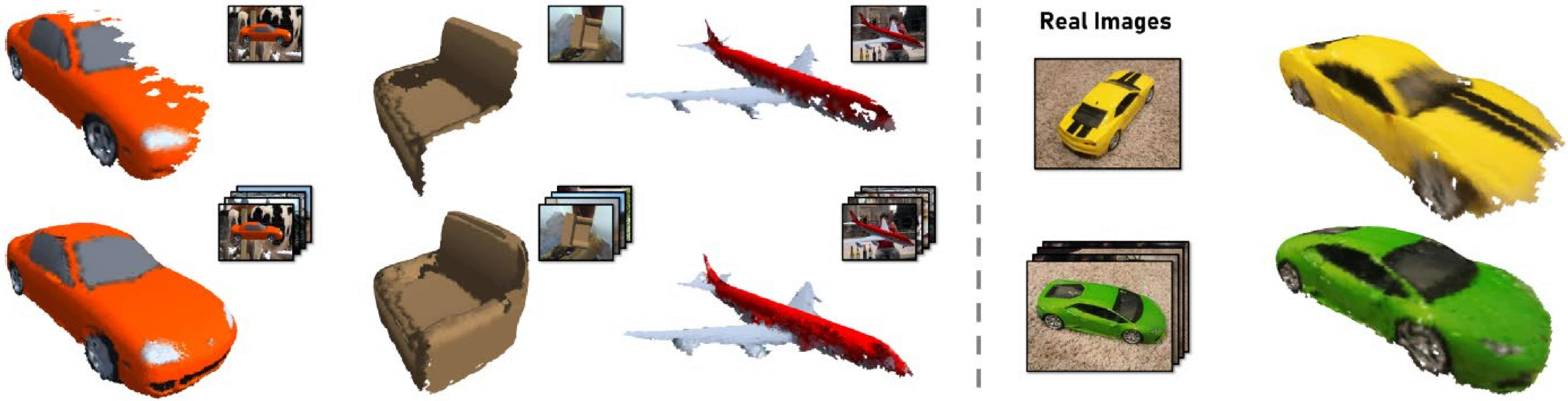
Pix2Surf: Multi-View Atlas



Multi-View consistency loss

$$L_c = \frac{1}{|P|} \sum_{(i,j) \in P} ||x_i - x_j||_2$$

Pix2Surf: Single-View Single-Chart

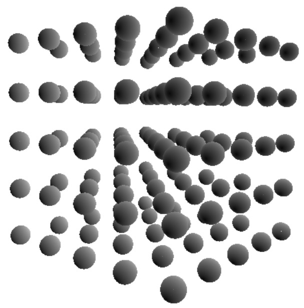


Lei, J., Sridhar, S., Guerrero, P., Sung, M., Mitra, N. and Guibas, L.J. Pix2surf: Learning parametric 3D surface models of objects from images. ECCV 2020.

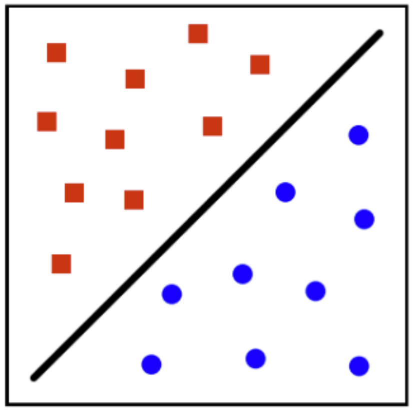
Occupancy Networks

- **Key Idea**:
  - Do not represent the 3D shape explicitly
  - Consider the surface implicitly, as **the decision boundary of a non-linear classifier,** parametrized by the neural network:

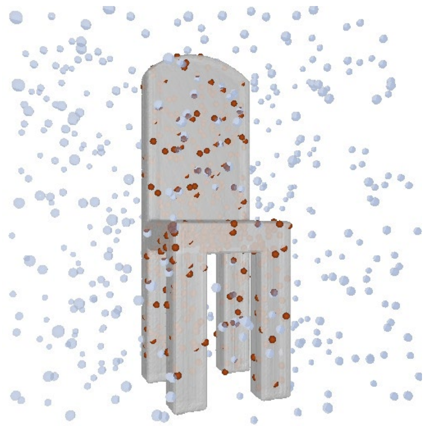
Condition



3D Locations

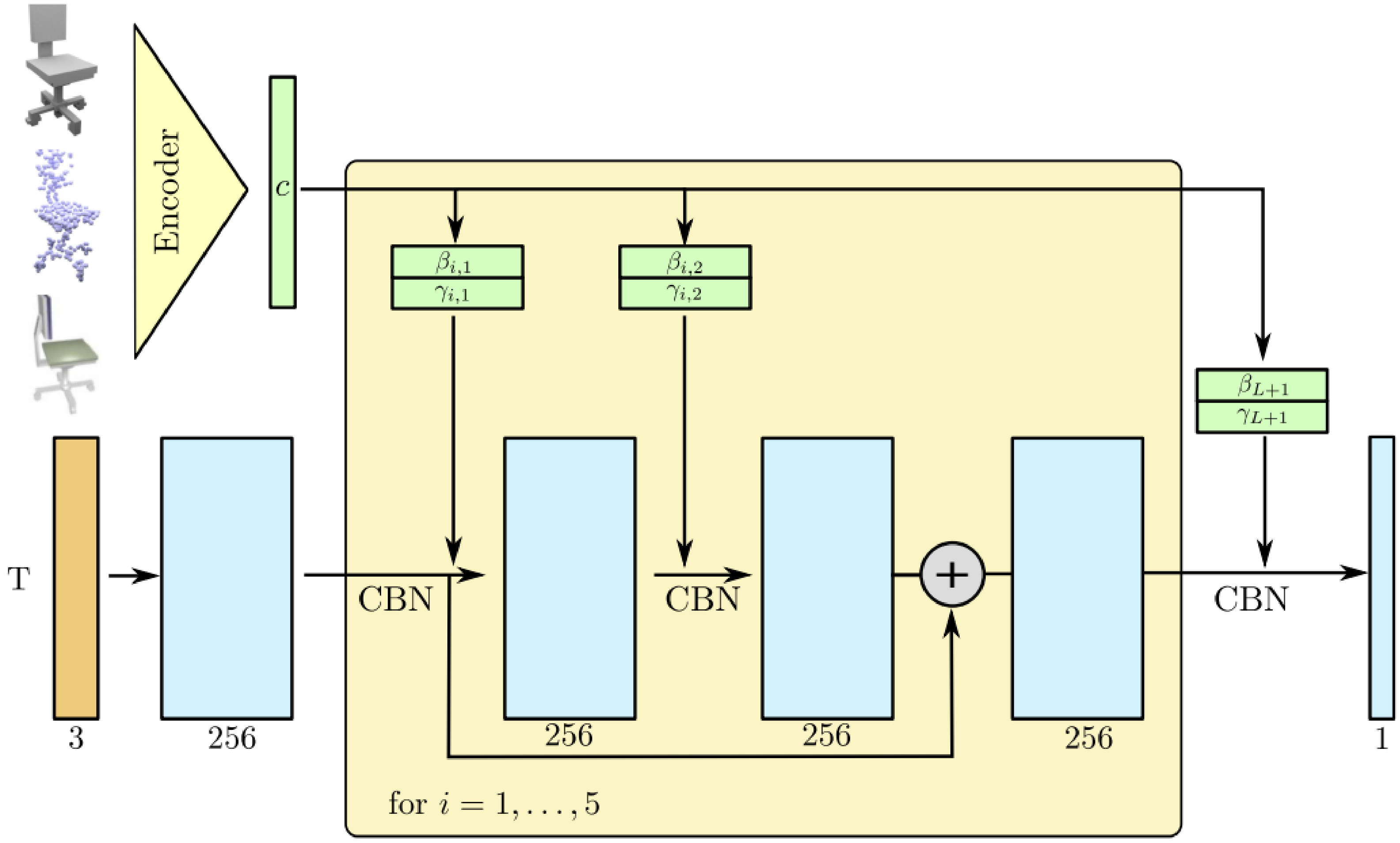


The **decision boundary of the classifier** models the **occupancy field.**



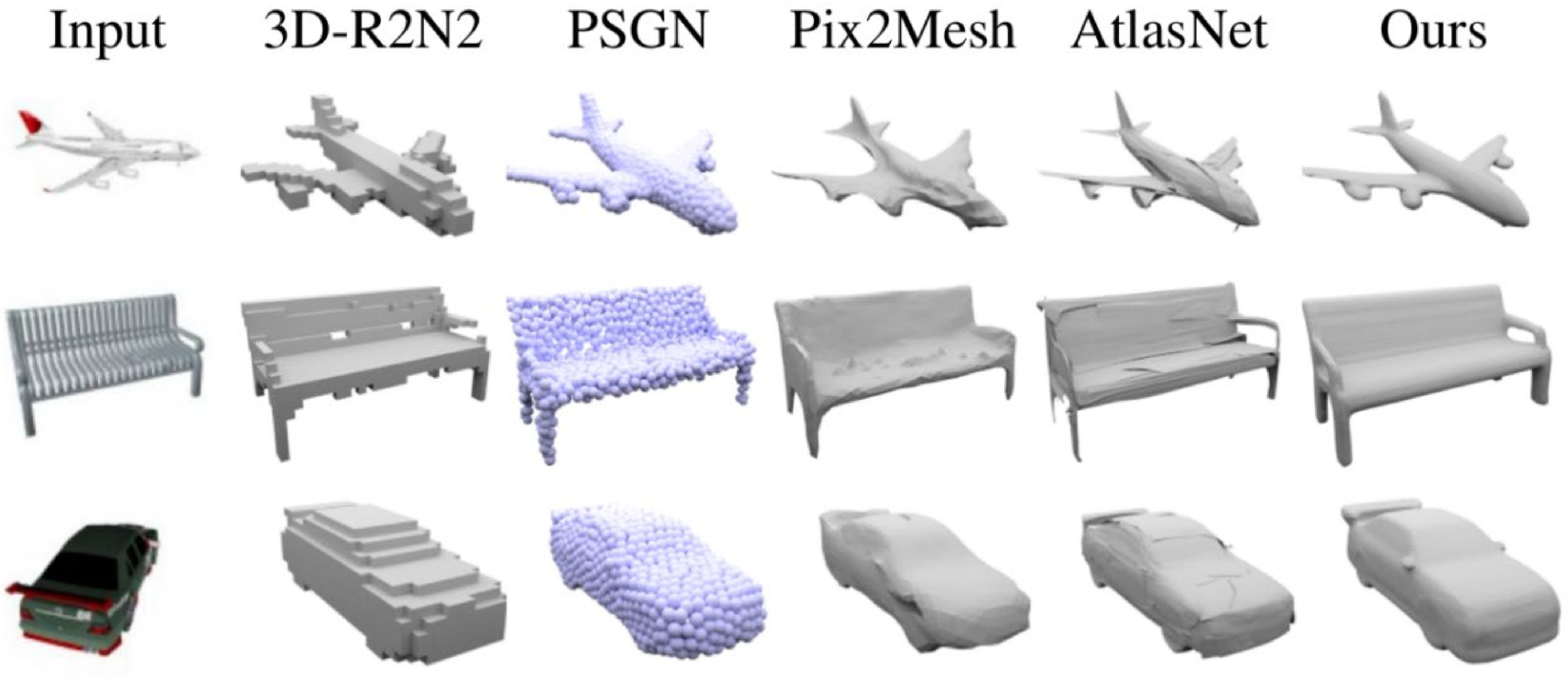
Occupancy Probability

Network Architecture



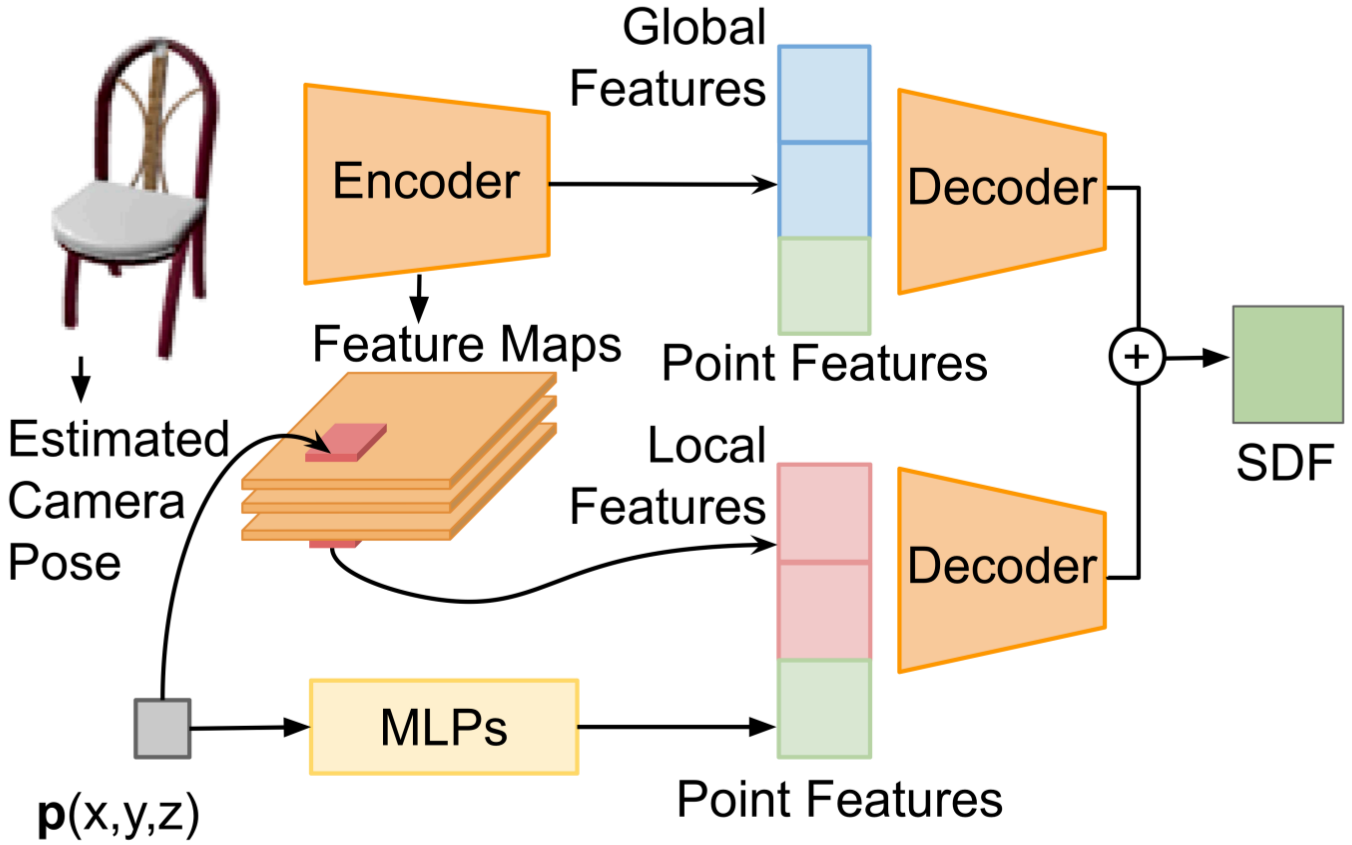
How can we train Occupancy Networks?

How well does it work?



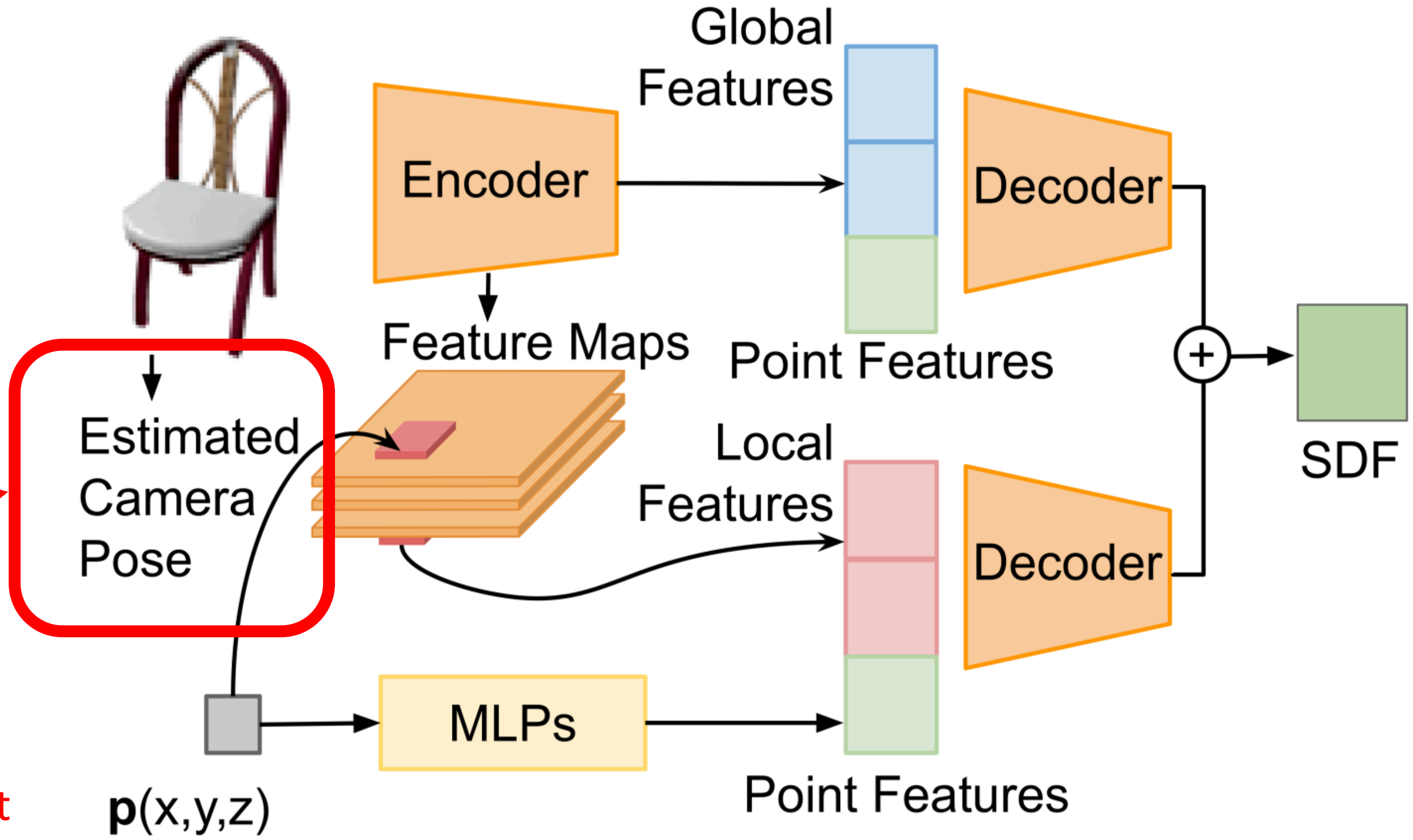
DISN – Model Overview

Key Idea: Use both global and local features for capturing both the overall shape and the fine-grained details.



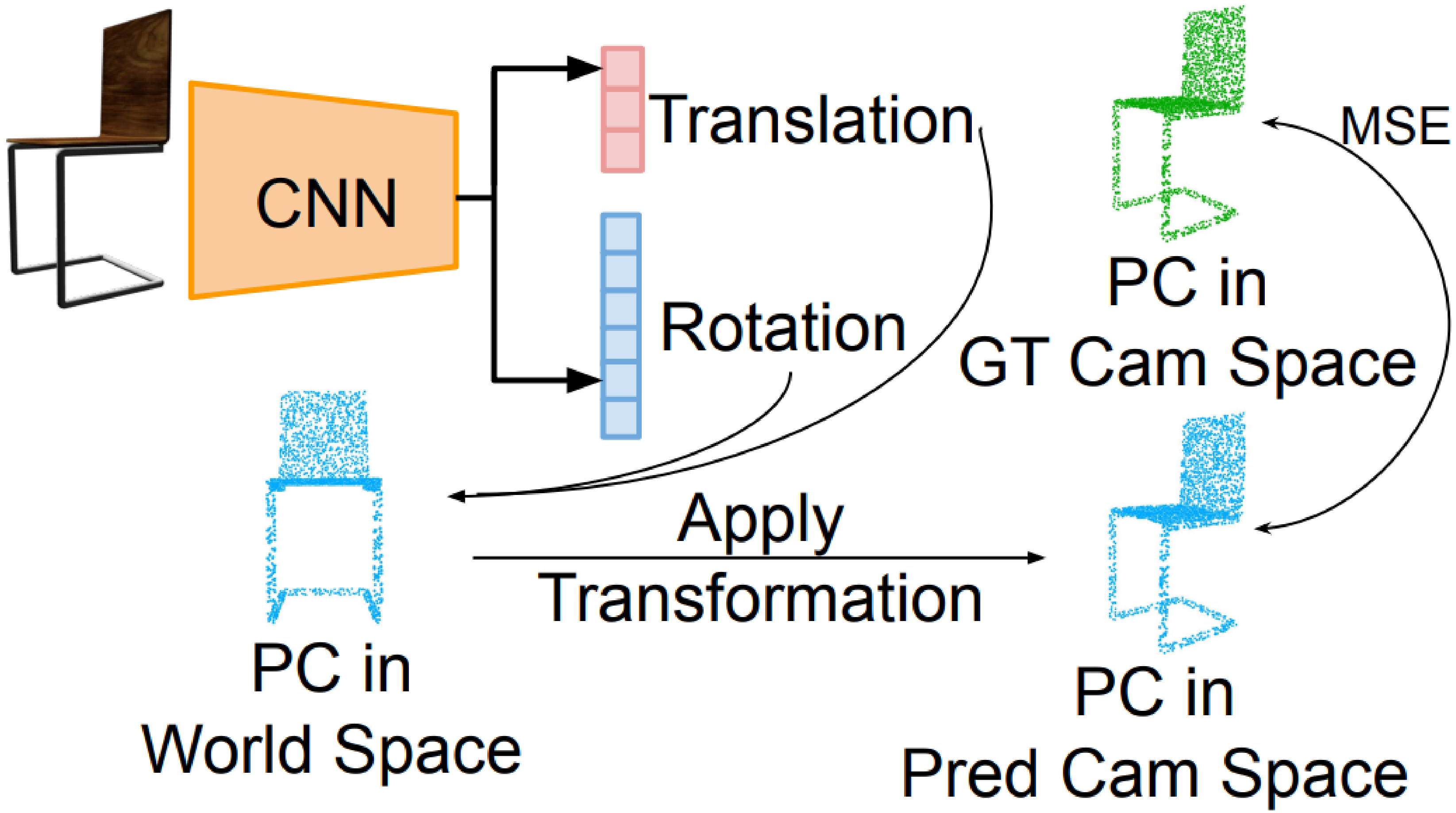
DISN – Model Overview

Key Idea: Use both global and local features for capturing both the overall shape and the fine-grained details.



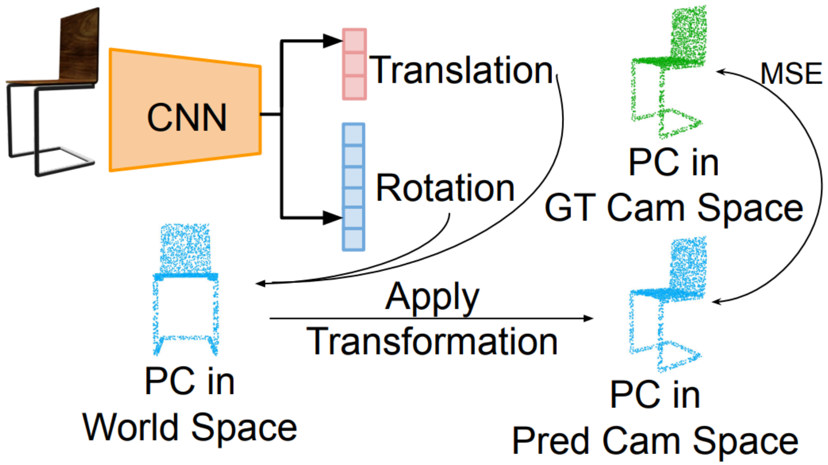
Given an image and a **3D point p in world coordinates**, we first need to **predict the camera parameters** that project the 3D point to the image plane.

DISN – Camera Pose Estimation

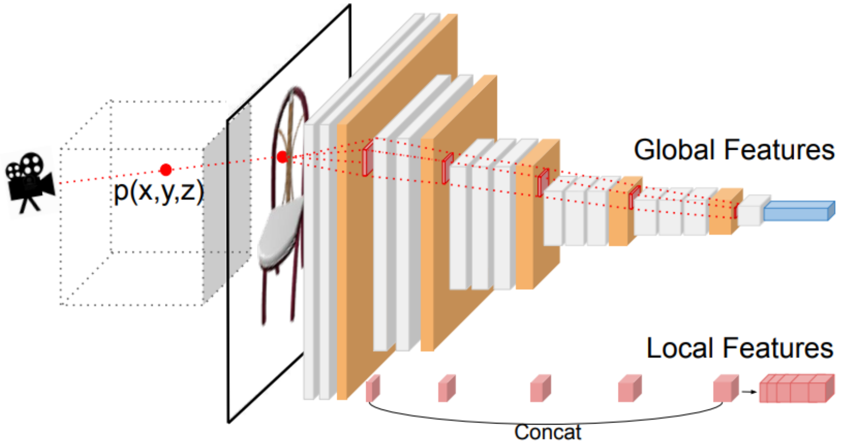


DISN – Model Overview

- **Camera Pose Network:** Estimate the camera pose, the 6 DoF transformation from the camera coordinate to world coordinate.
- **Local Feature Extraction Network:** Using the camera pose find a 3D point's 2D location on the image and **extract local feature patches from multiple network layers.**

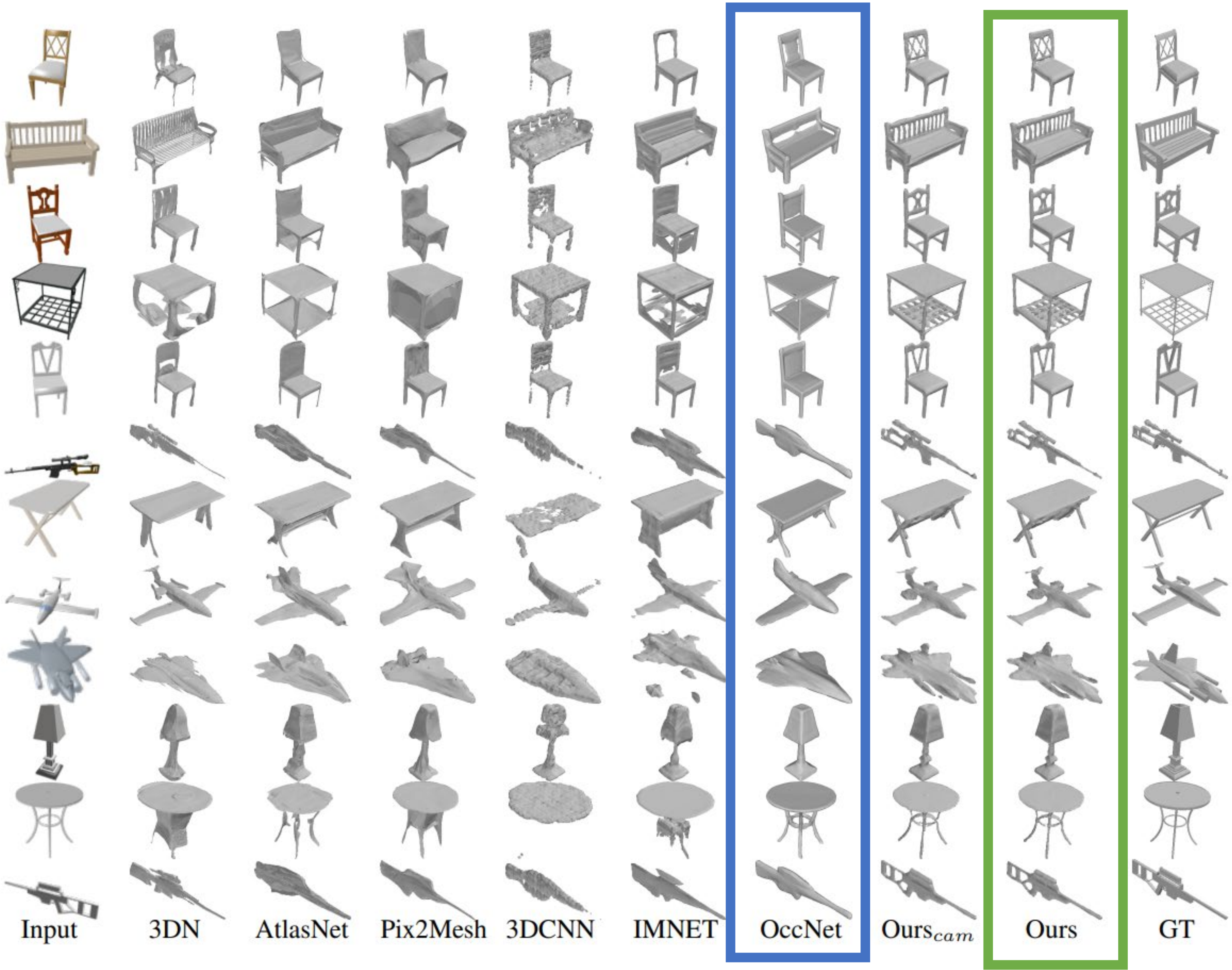


Camera Pose Network

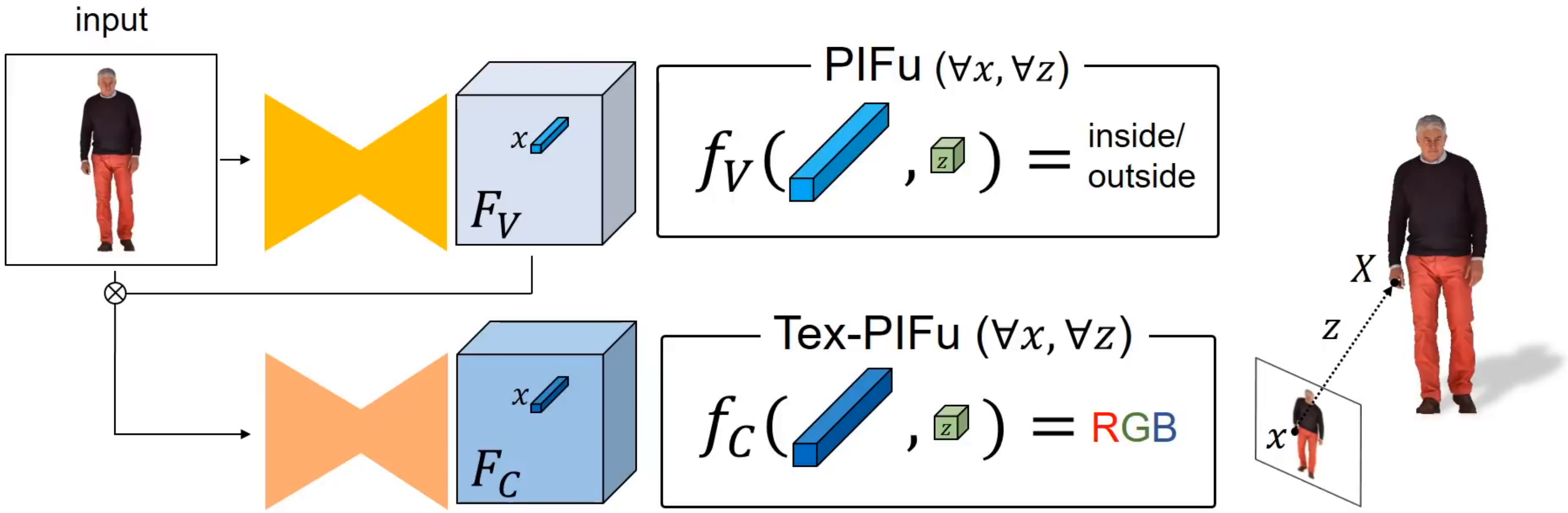


Local Feature Extraction Network

How well does it work?

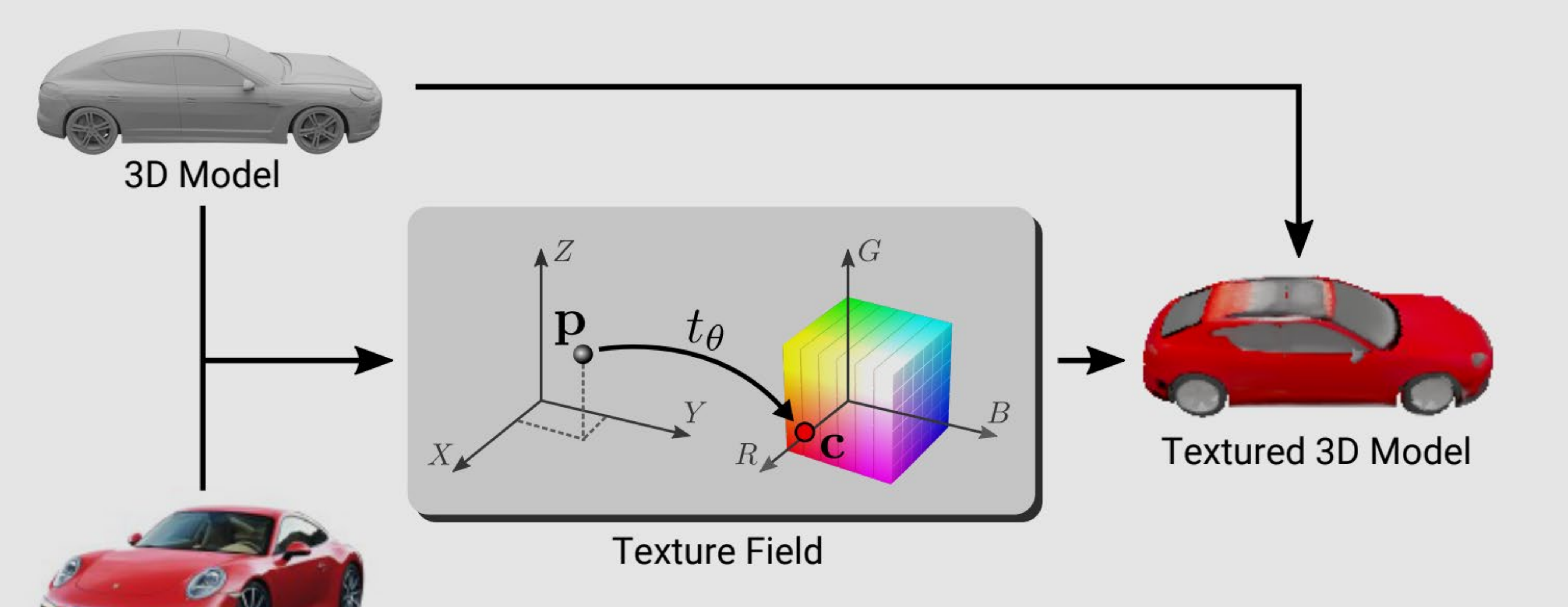


PIFu: Pixel-Aligned Implicit Function



What about Texture?

A Similar Idea: Texture Fields

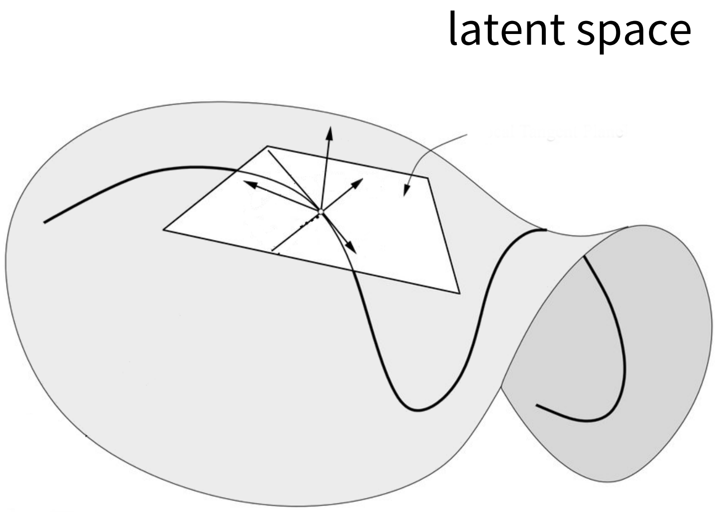


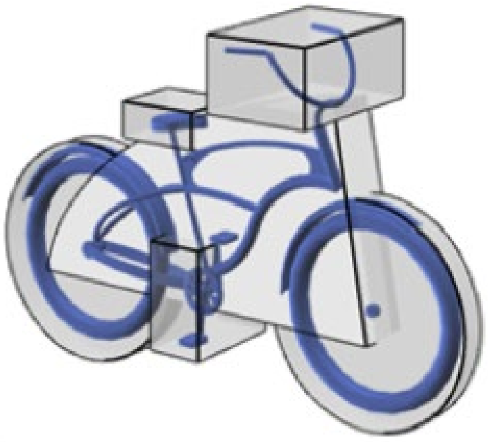
For every point on the corresponding surface predict its color value.

Shape Deformations / Edits and Variation Generation

Generative Models for Deformations / Edits

Learn possible variations of an input shape, meeting semantic constraints.

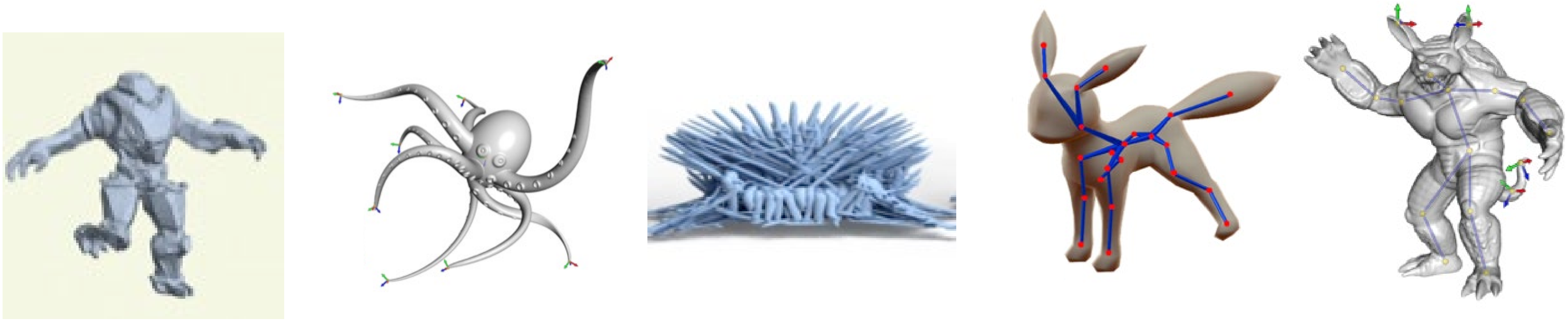




Motivation

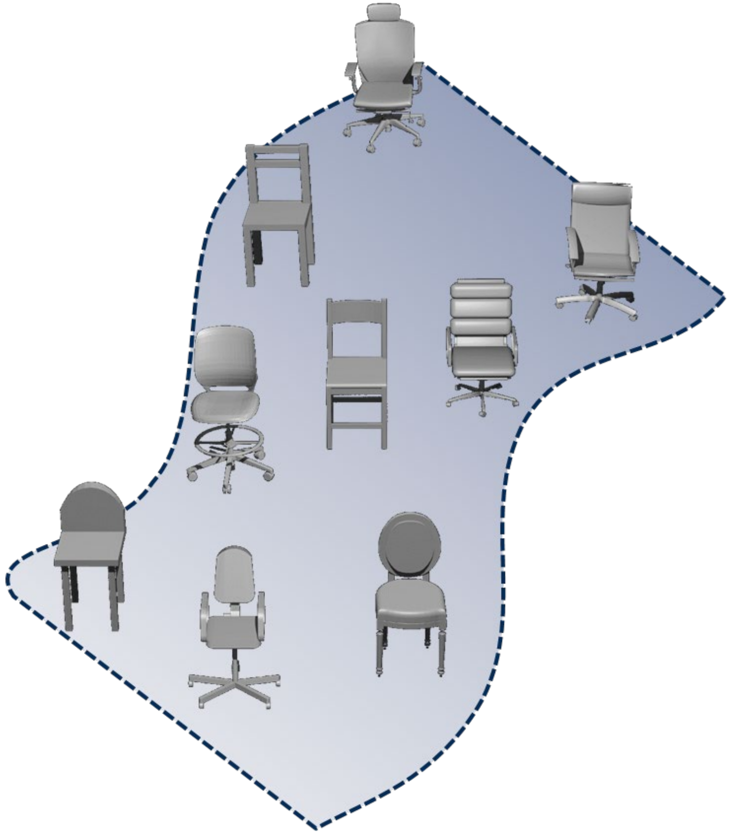
- Leverage on existing artist generated models

- Create new models through deformations and edits

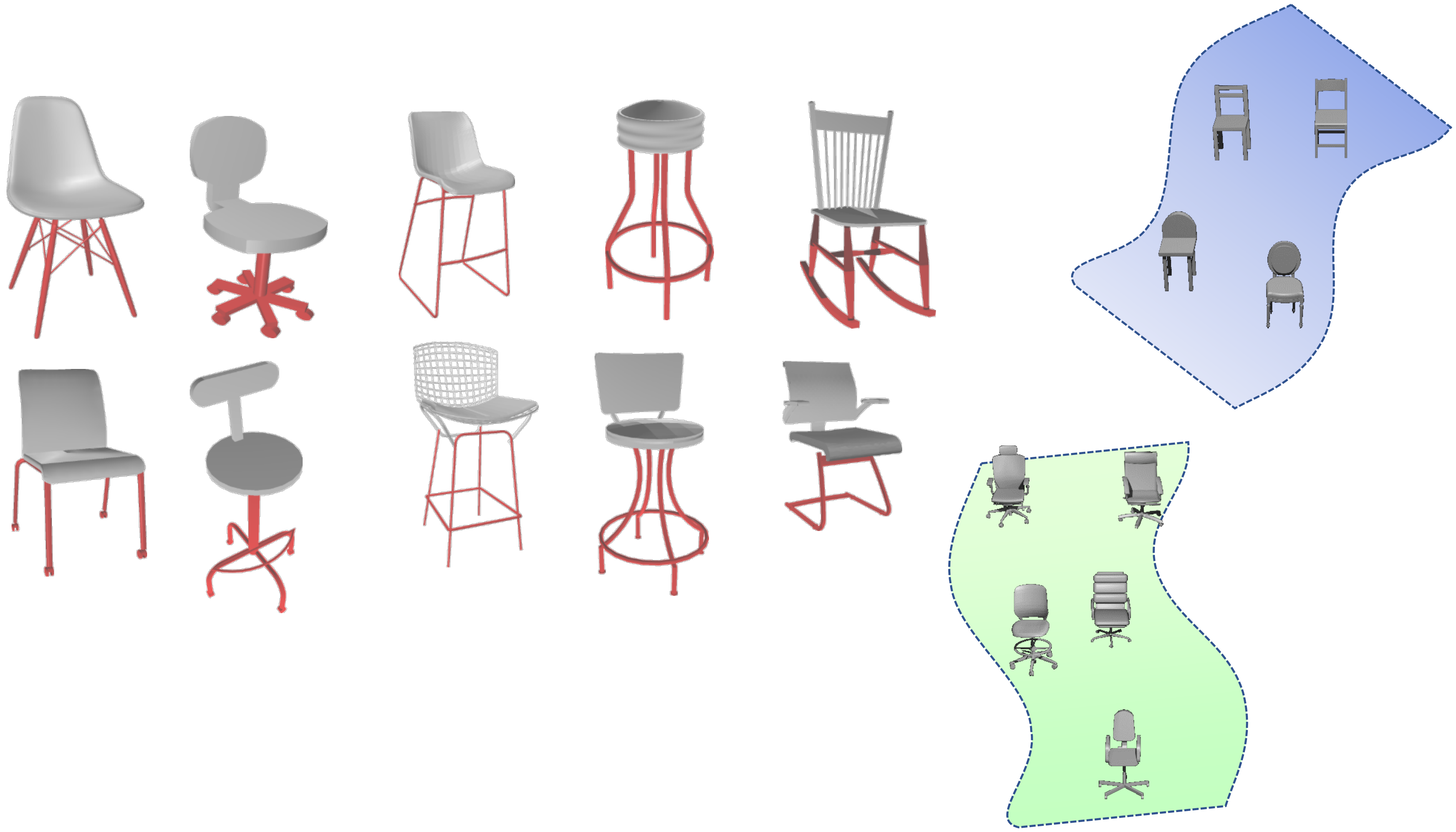


Continuous Shape Variability

A chair manifold?



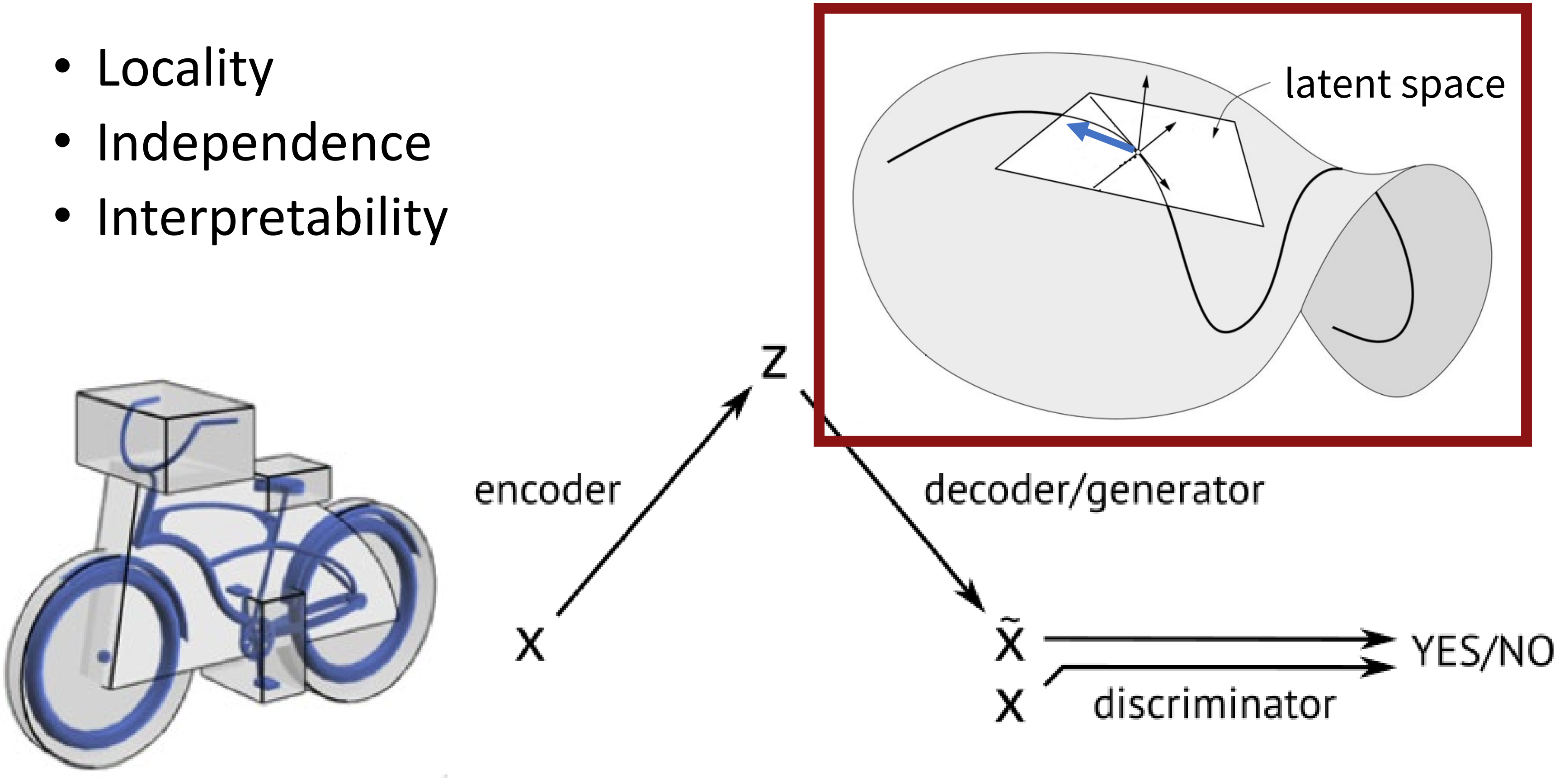
Combinatorial or Discrete Variability



Shapes are often Over-Parametrized for Edits

Learn a reduced parameter space and enforce that each learned parameter represents an intuitive deformation or change.

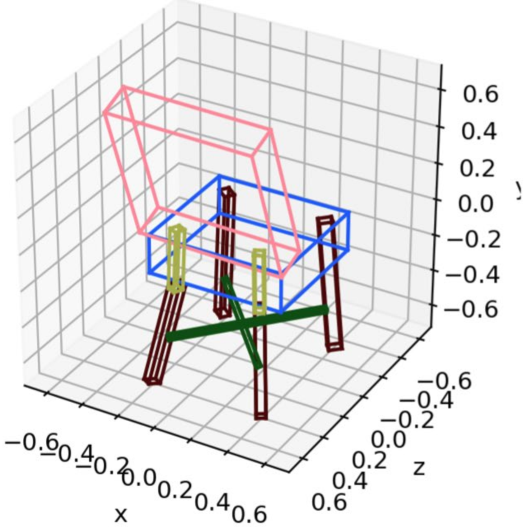
- Locality
- Independence
- Interpretability



Adversarial testing w/ the deformed shape

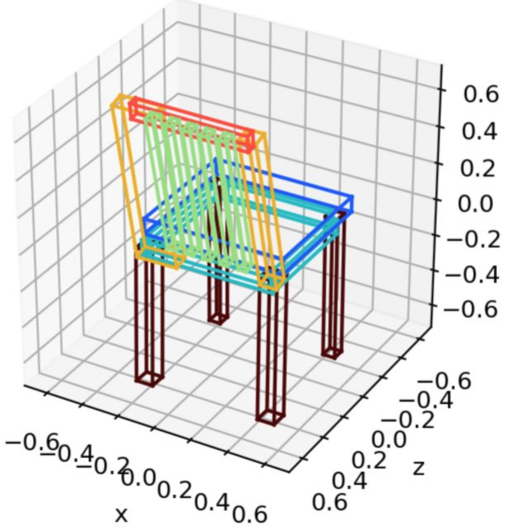
Shape Differences as First-Class Citizens

Shape A




```
0 chair    {0}
├0 chair_back    {1}
│  ├0 back_surface    {2}
│  │  ├0 back_single_surface [LEAF] {3}
│  ├1 back_connector [LEAF] {4}
│  ├2 back_connector [LEAF] {5}
├1 chair_seat    {6}
│  ├0 seat_surface    {7}
│  │  ├0 seat_single_surface [LEAF] {8}
├2 chair_base    {9}
│  ├0 regular_leg_base    {10}
│  │  ├0 leg [LEAF] {11}
│  │  ├1 leg [LEAF] {12}
│  │  ├2 leg [LEAF] {13}
│  │  ├3 leg [LEAF] {14}
│  │  ├4 bar_stretcher [LEAF] {15}
│  │  ├5 bar_stretcher [LEAF] {16}
```

Shape B




```
0 chair    {0}
├0 chair_back    {0}
│  ├0 back_surface    {0}
│  │  ├0 back_surface_vertical_bar [LEAF] {3}
│  │  ├1 back_surface_vertical_bar [LEAF] {4}
│  │  ├2 back_surface_vertical_bar [LEAF] {5}
│  │  ├3 back_surface_vertical_bar [LEAF] {6}
│  │  ├4 back_surface_vertical_bar [LEAF] {7}
│  ├1 back_frame    {8}
│  │  ├0 back_frame_vertical_bar [LEAF] {9}
│  │  ├1 back_frame_vertical_bar [LEAF] {10}
│  │  ├2 back_frame_horizontal_bar [LEAF] {11}
├1 chair_seat    {0}
│  ├0 seat_surface    {0}
│  │  ├0 seat_single_surface [LEAF] {0}
│  ├1 seat_frame    {15}
│  │  ├0 seat_frame_bar [LEAF] {16}
│  │  ├1 seat_frame_bar [LEAF] {17}
│  │  ├2 seat_frame_bar [LEAF] {18}
│  │  ├3 seat_frame_bar [LEAF] {19}
├2 chair_base    {0}
│  ├0 regular_leg_base    {0}
│  │  ├0 leg [LEAF] {0}
│  │  ├1 leg [LEAF] {0}
│  │  ├2 leg [LEAF] {0}
│  │  ├3 leg [LEAF] {0}
```

Shape Diff

```
0 [DiffNode: SAME]
 ├0 [DiffNode: SAME]
 │  ├0 [DiffNode: SAME]
 │  │  ├0 [DiffNode: DEL]
 │  │  ├1 [DiffNode: ADD]
 │  │  │  ├0 back_surface_vertical_bar [LEAF] {3}
 │  │  ├2 [DiffNode: ADD]
 │  │  │  ├0 back_surface_vertical_bar [LEAF] {4}
 │  │  ├3 [DiffNode: ADD]
 │  │  │  ├0 back_surface_vertical_bar [LEAF] {5}
 │  │  ├4 [DiffNode: ADD]
 │  │  │  ├0 back_surface_vertical_bar [LEAF] {6}
 │  │  ├5 [DiffNode: ADD]
 │  │  │  ├0 back_surface_vertical_bar [LEAF] {7}
 │  ├1 [DiffNode: DEL]
 │  ├2 [DiffNode: DEL]
 │  ├3 [DiffNode: ADD]
 │  │  ├0 back_frame    {8}
 │  │  │  ├0 back_frame_vertical_bar [LEAF] {9}
 │  │  │  ├1 back_frame_vertical_bar [LEAF] {10}
 │  │  │  ├2 back_frame_horizontal_bar [LEAF] {11}
 ├1 [DiffNode: SAME]
 │  ├0 [DiffNode: SAME]
 │  │  ├0 [DiffNode: LEAF]
 │  ├1 [DiffNode: ADD]
 │  │  ├0 seat_frame    {15}
 │  │  │  ├0 seat_frame_bar [LEAF] {16}
 │  │  │  ├1 seat_frame_bar [LEAF] {17}
 │  │  │  ├2 seat_frame_bar [LEAF] {18}
 │  │  │  ├3 seat_frame_bar [LEAF] {19}
 ├2 [DiffNode: SAME]
 │  ├0 [DiffNode: SAME]
 │  │  ├0 [DiffNode: LEAF]
 │  │  ├1 [DiffNode: LEAF]
 │  │  ├2 [DiffNode: LEAF]
 │  │  ├3 [DiffNode: LEAF]
 │  │  ├4 [DiffNode: DEL]
 │  │  ├5 [DiffNode: DEL]
```

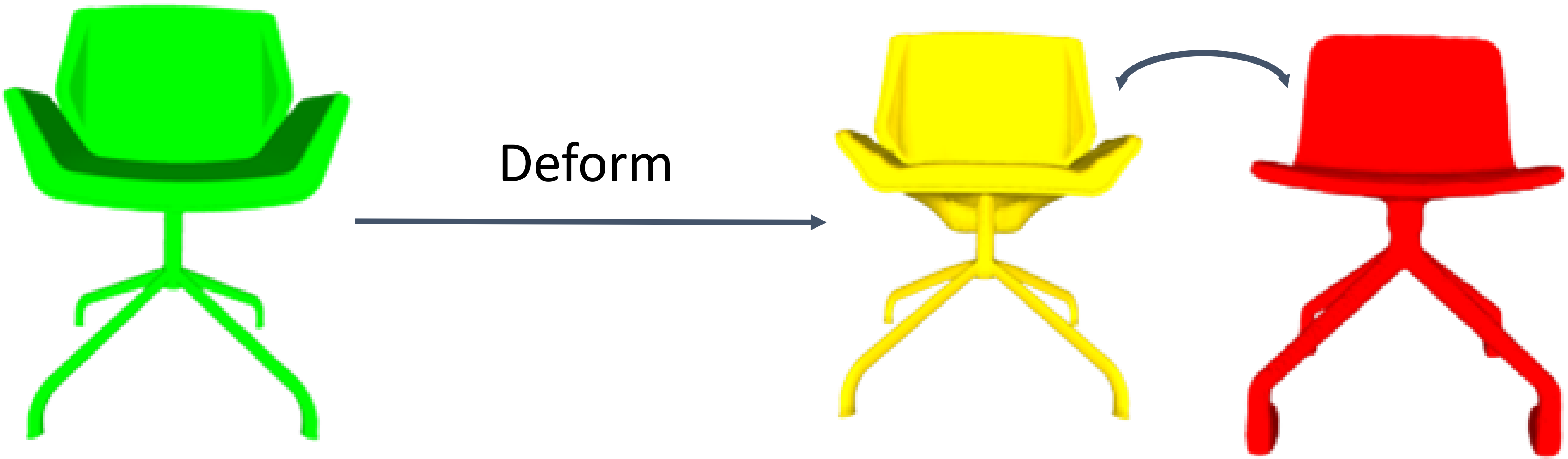
Continuous Shape Variations: Deformations

Neural ODEs for Shape Deformation

MeshFlow: A Robust and Scalable Shape Deformation Framework. Jingwei Huang, Max Jiang, Baiqiang Leng, Bin Wang, Leonidas Guibas.

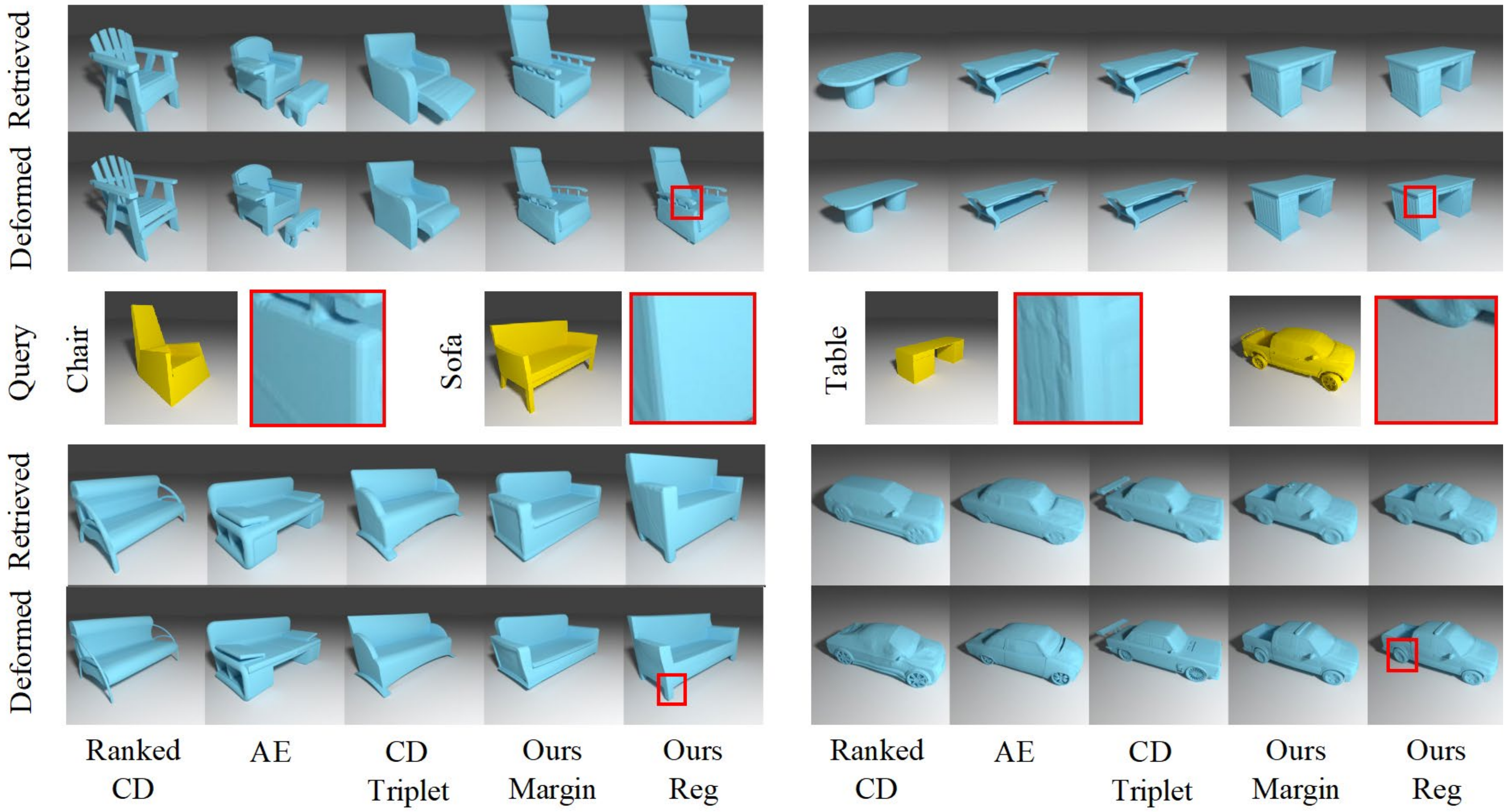
Problem Setup: Semantic Deformation

Pairwise Deformation:

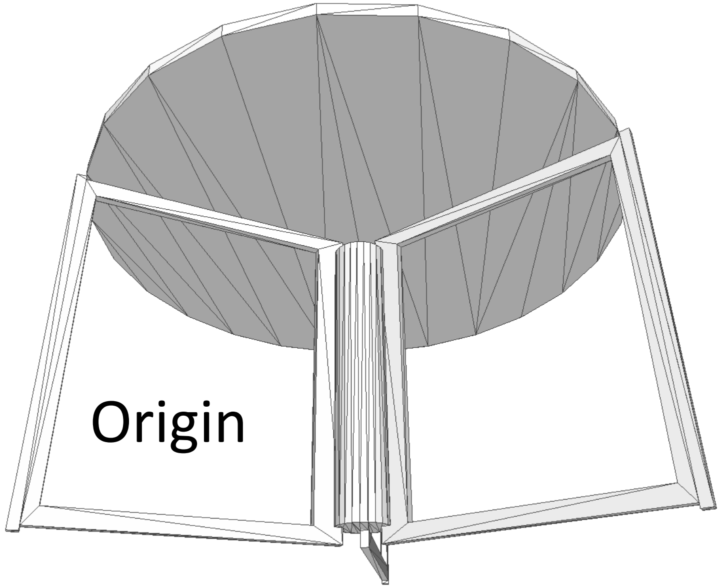


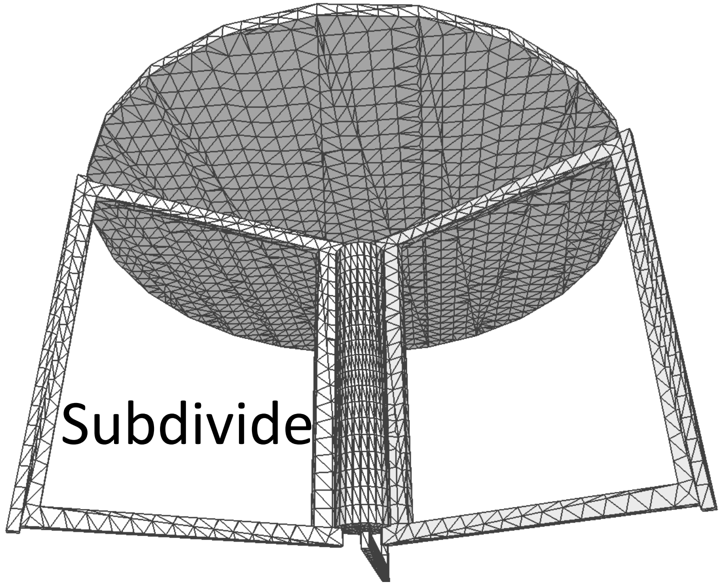
Source geometry provides “style” while target provides “pose”. Geometric “style” transfer.

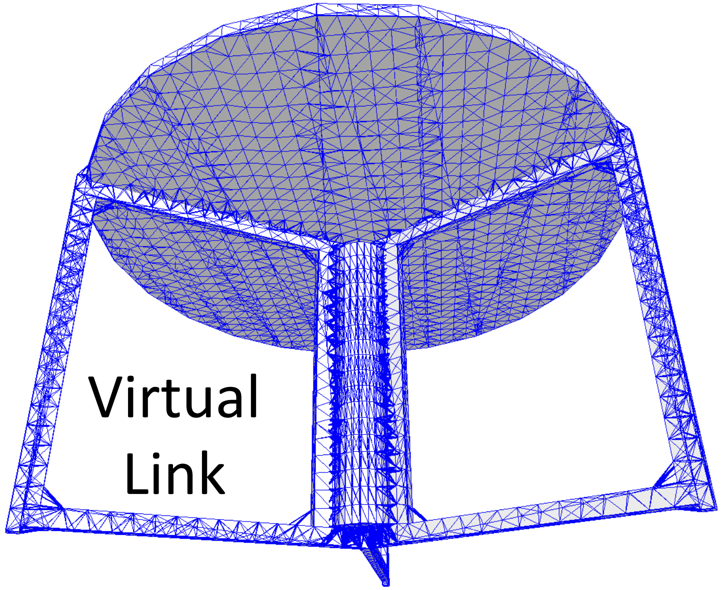
“As Rigid As Possible” (ARAP) Deformations

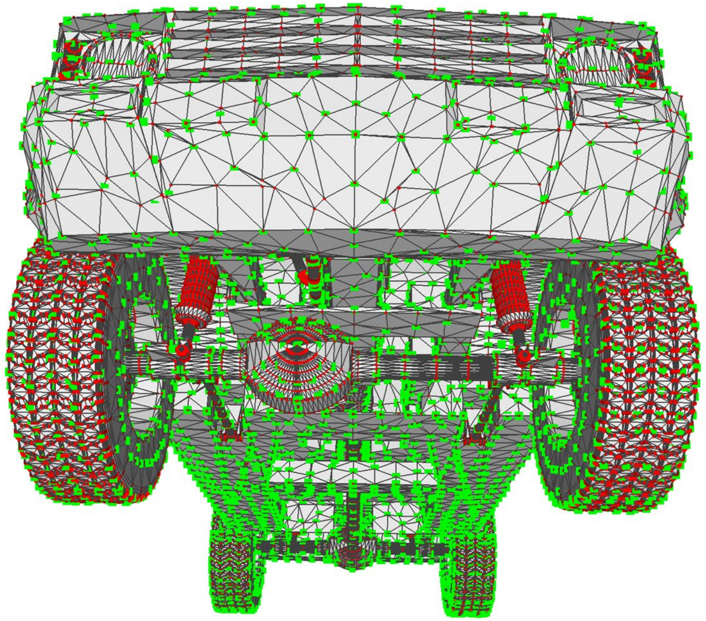


Remeshing: Mesh to Uniform Skeleton Graph









Uniform Skeleton
(red->green)

Uniform Skeleton Graph Representation

(a) Source (b) Target (c) Deform with Edges (d) Deform with Virtual Links

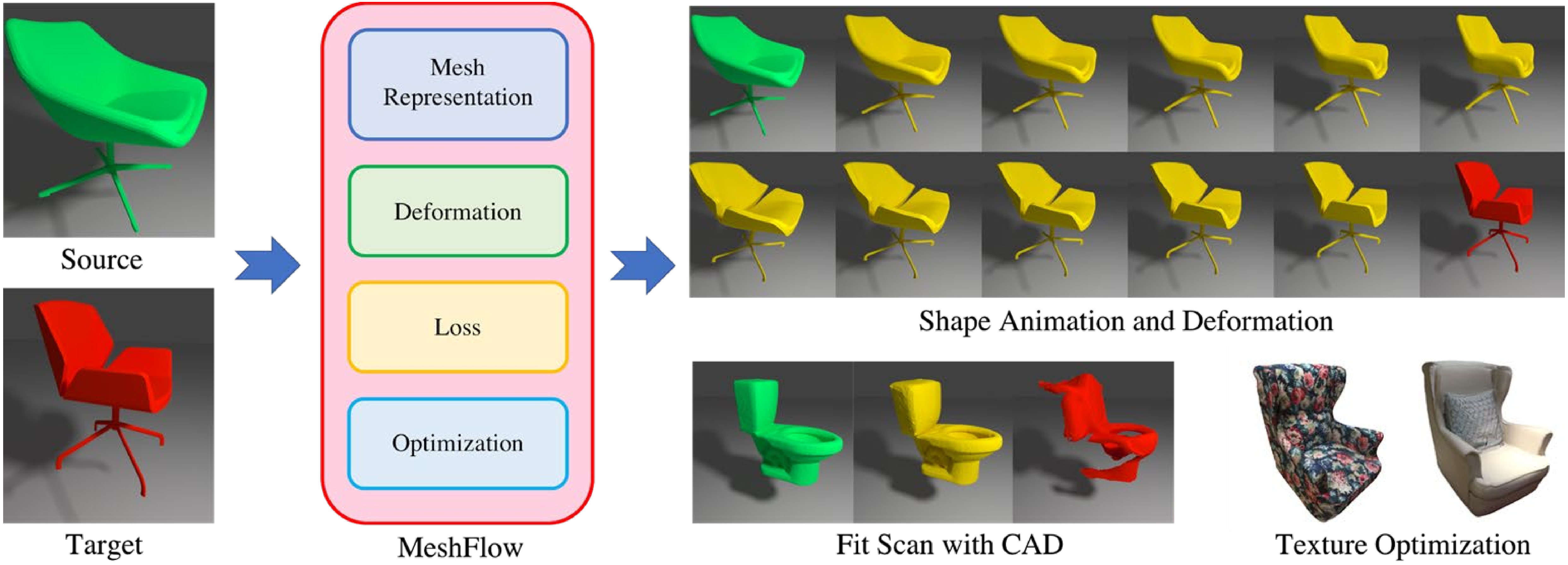
Methodology

- A general interface for shape deformation optimization
  - Mesh representation to support effective Non-rigid ICP
  - Support general deformation functions

Fitness to destination + Preservation of constraints

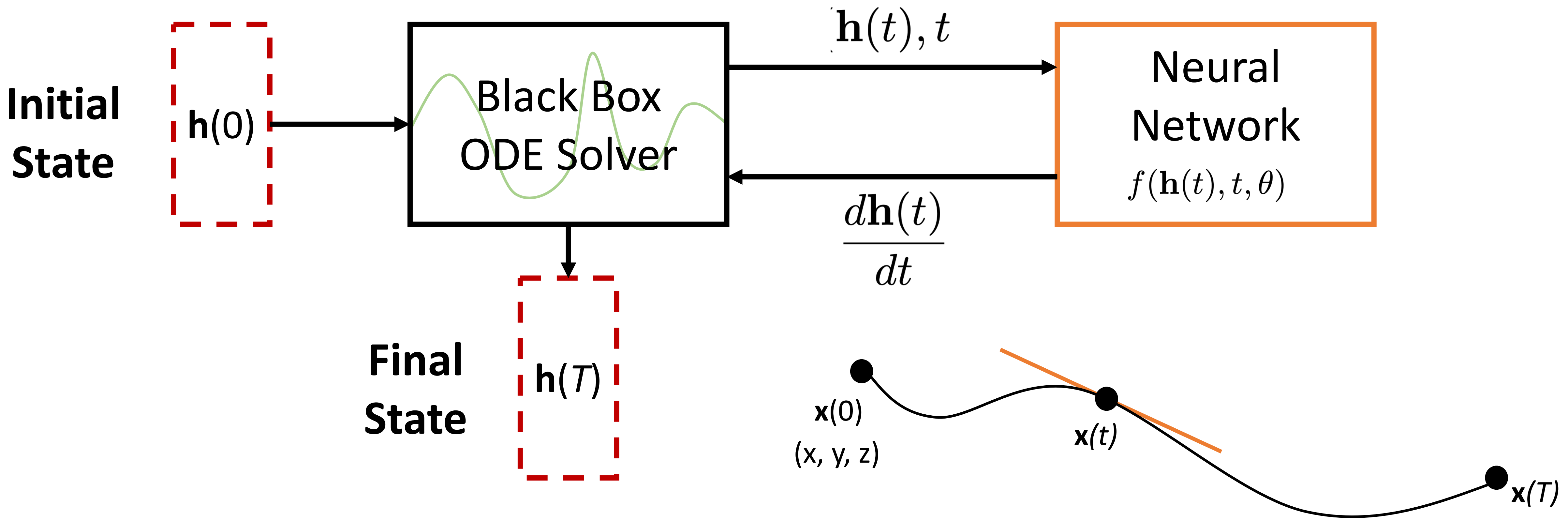
- Deep flow-based method
  - A bijective mapping
  - No self-intersection
  - Encourages but does not rely on rigidity during optimization
  - Better alignment

MeshFlow Key Modules



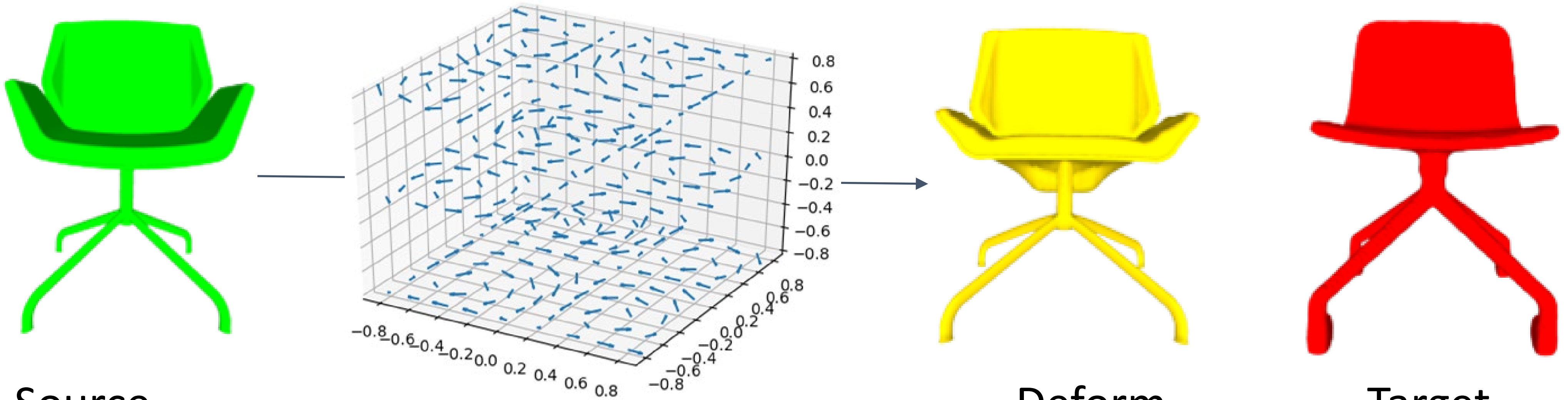
Neural Ordinary Differential Equations

- **Key idea**: neural network gives derivative, ODE solver to integrate
- Gives **“infinite depth” nets** and **continuous representation** of time series



Deep Flow-Based Deformation

Use Neural ODE + learned flow model to deform geometries.



Parameterize a learnable flow field $\mathbf{f}_\theta : \mathbb{R}^4 \mapsto \mathbb{R}^3$ using a fully connected neural network that outputs three velocity components for every point at every time. Advecting the source shape by integrating an ODE produces the resulting deformed shape.

Deep Flow-Based Deformation

Denote the mapping induced by convecting the geometry using Neural ODE as:

$$\Phi_\theta(\mathbf{X}_0) = \mathbf{X}_0 + \int_0^1 \mathbf{f}_\theta(\mathbf{X}(t), t)dt$$

The mapping is bijective since the inverse of the mapping can be easily acquired by inverting the integration order.

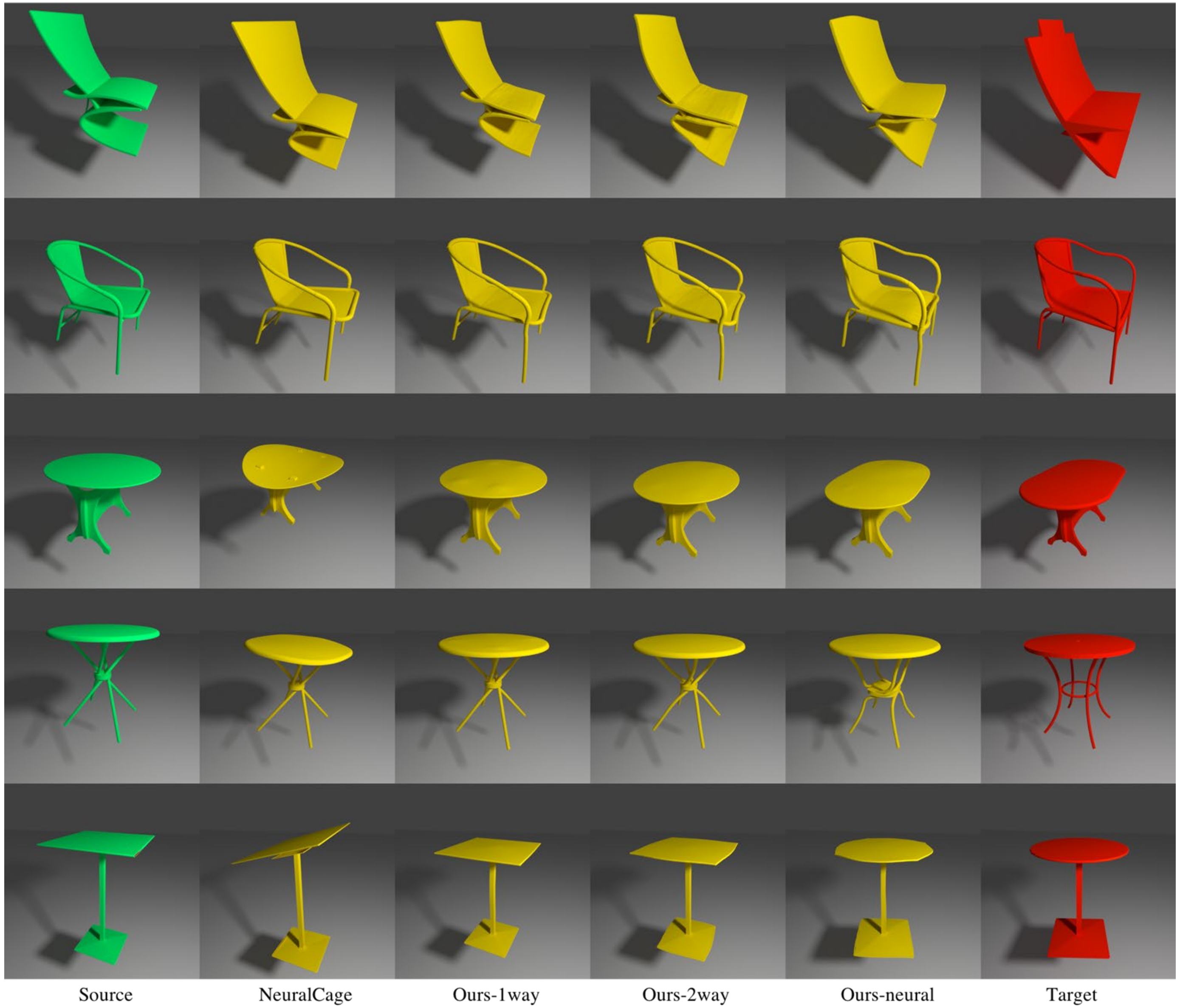
$$\Phi_\theta^{-1}(\mathbf{X}_1) = \mathbf{X}_1 + \int_1^0 \mathbf{f}_\theta(\mathbf{X}(t), t)dt$$

Therefore the deformation field between two shapes can be learned via optimizing for a symmetric deformation loss between the two shapes:

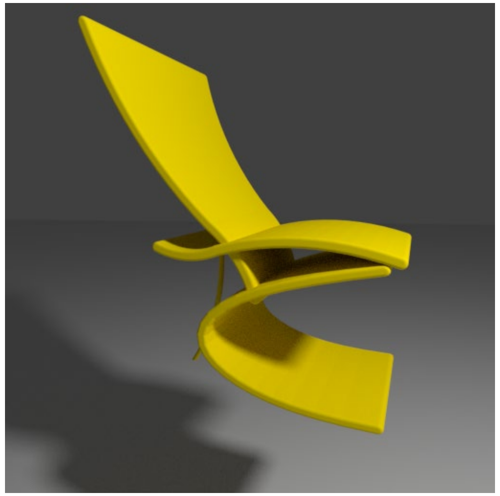
$$\underset{\theta}{\operatorname{argmin}} \quad \mathcal{L}(\theta) = \mathcal{C}(\Phi_\theta(\mathbf{X}_0), \mathbf{X}_1) + \mathcal{C}(\Phi_\theta^{-1}(\mathbf{X}_1), \mathbf{X}_0)$$

Here $\mathcal{C}$ is a differentiable geometric loss. A simple choice is Chamfer loss.

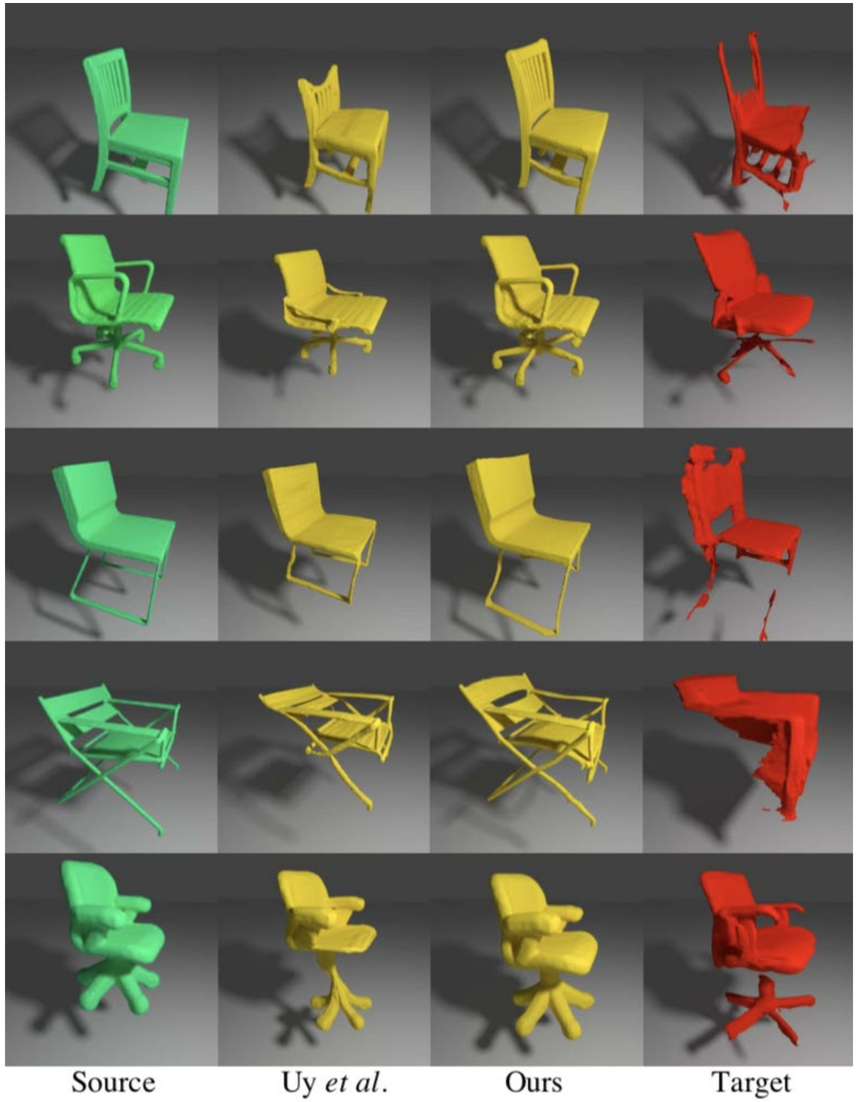
Deformation Comparison

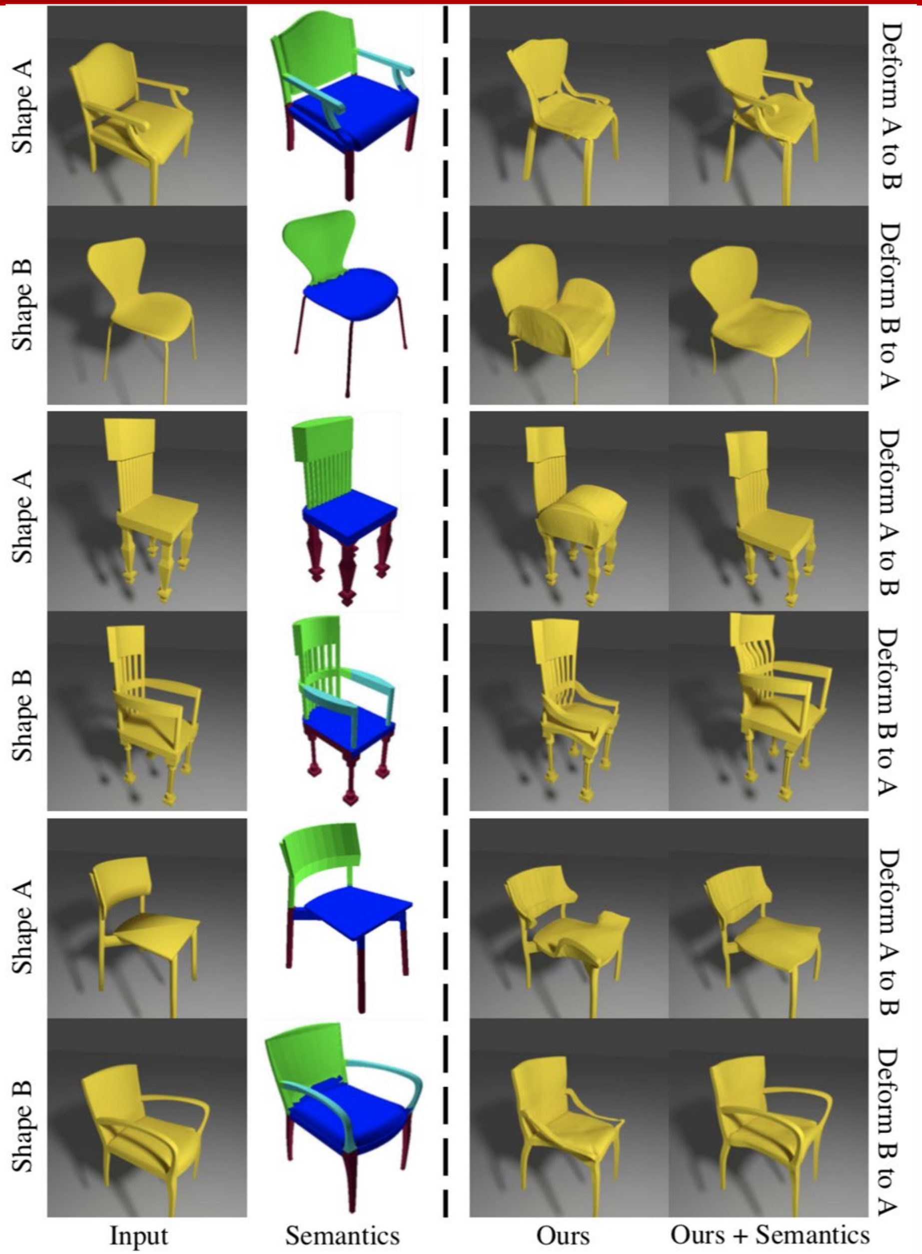


Deformation Video



Scan2CAD & Part Deformation





2020

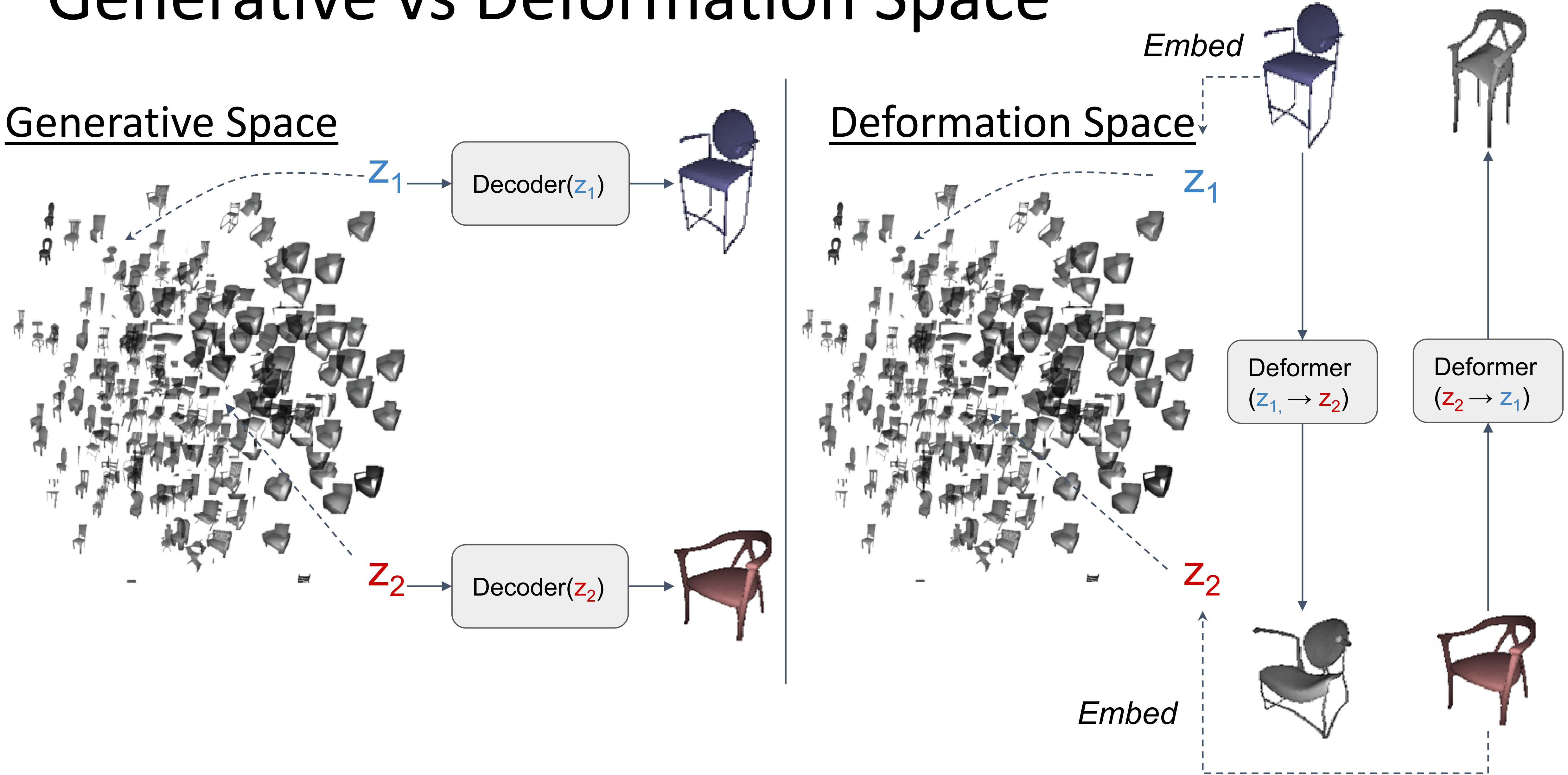
ShapeFlow

Learnable Deformations Among 3D Shapes

Chiyu "Max" Jiang [1], Jingwei Huang [2],
Andrea Tagliasacchi [3], Leonidas Guibas [2]

[1] UC Berkeley, [2] Stanford University, [3] Google Brain

Generative vs Deformation Space



Deformation Space

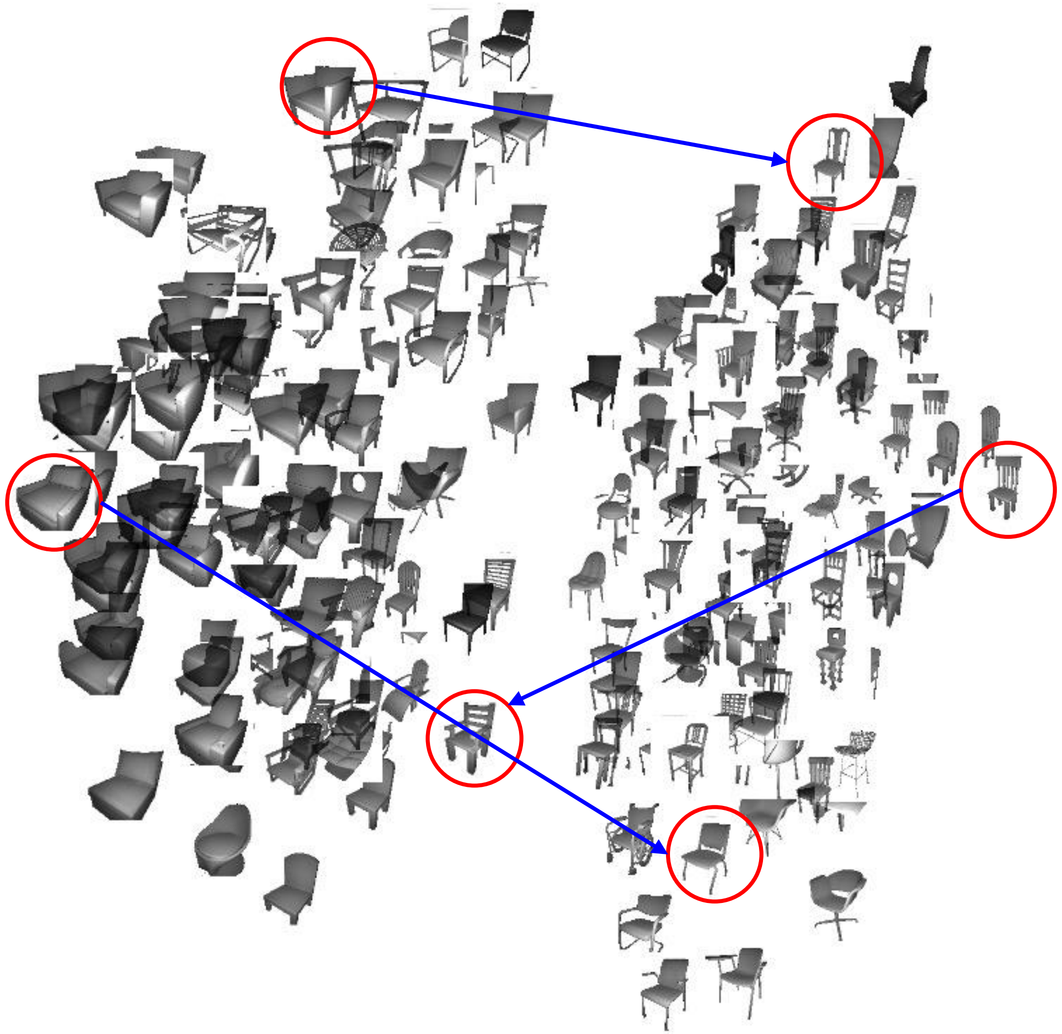
A deformation space naturally allows the disentanglement of geometric style (coming from the source) and structural pose (conforming to the target).

Regularizing Deformation Flows

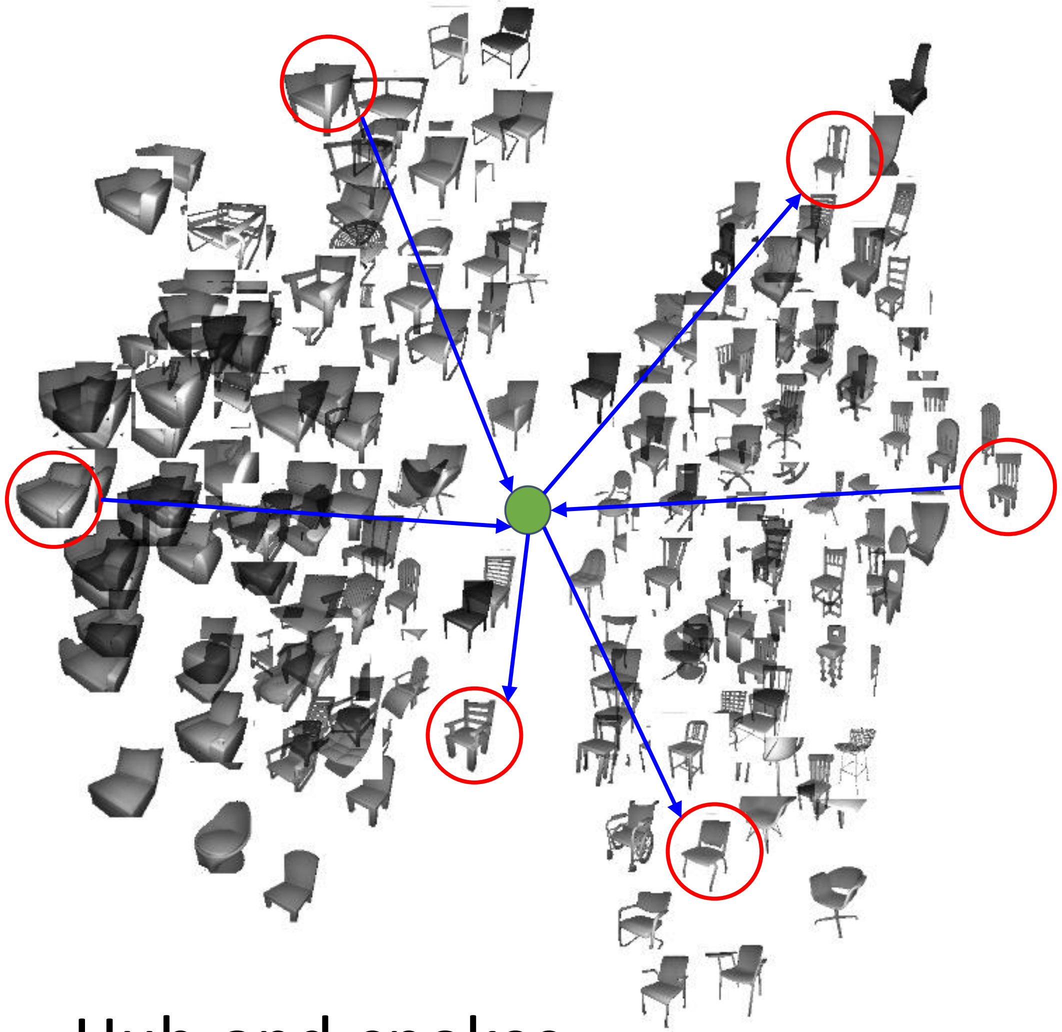
Furthermore, we can apply implicit and explicit flow regularization to ensure various desirable deformation properties.

- Implicit regularization: volume conservation
- Implicit regularization: symmetries.
- Explicit regularization: surface metrics

Hub-and-Spoke Deformation

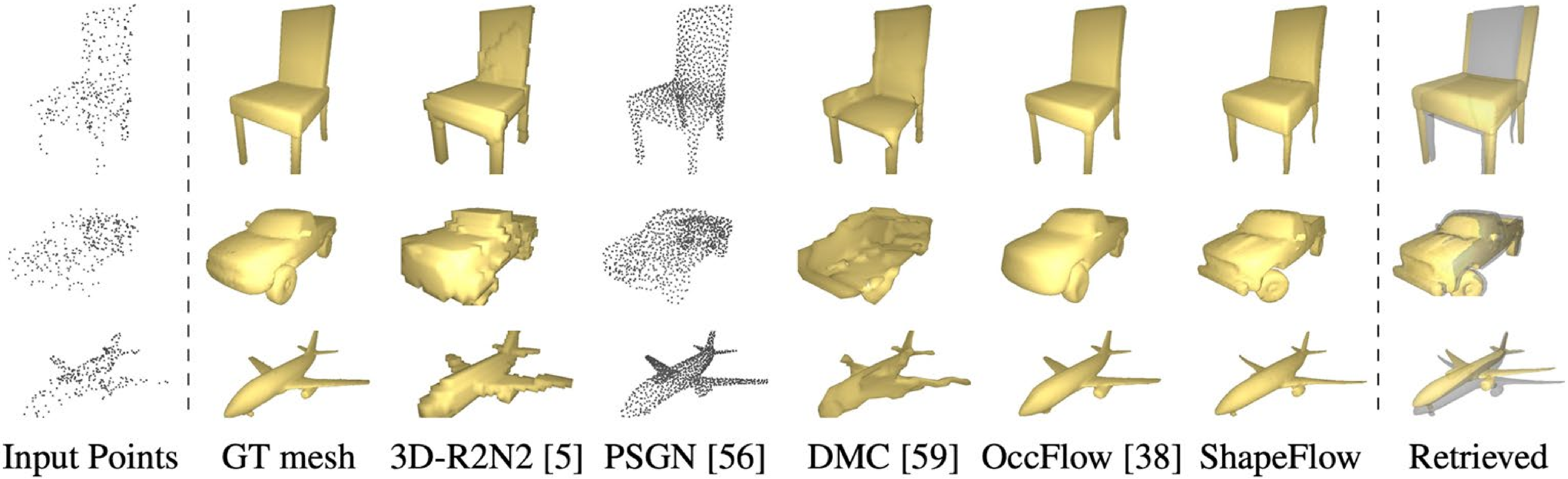


All-To-All



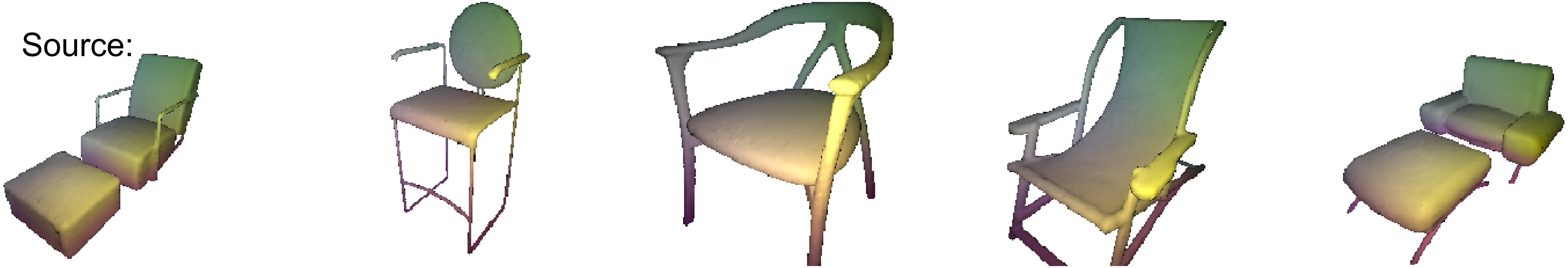
Hub-and-spokes

Experiment - Reconstruction via Deformation

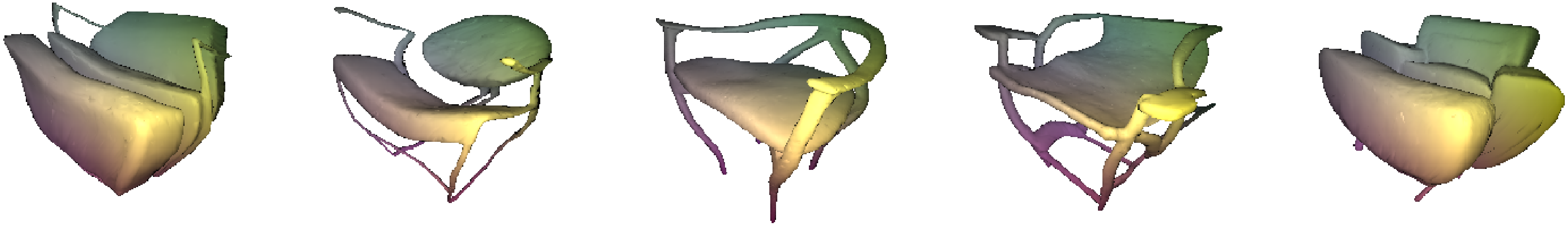


Deformation to “Mean Shape” in Canonical Space

Source:

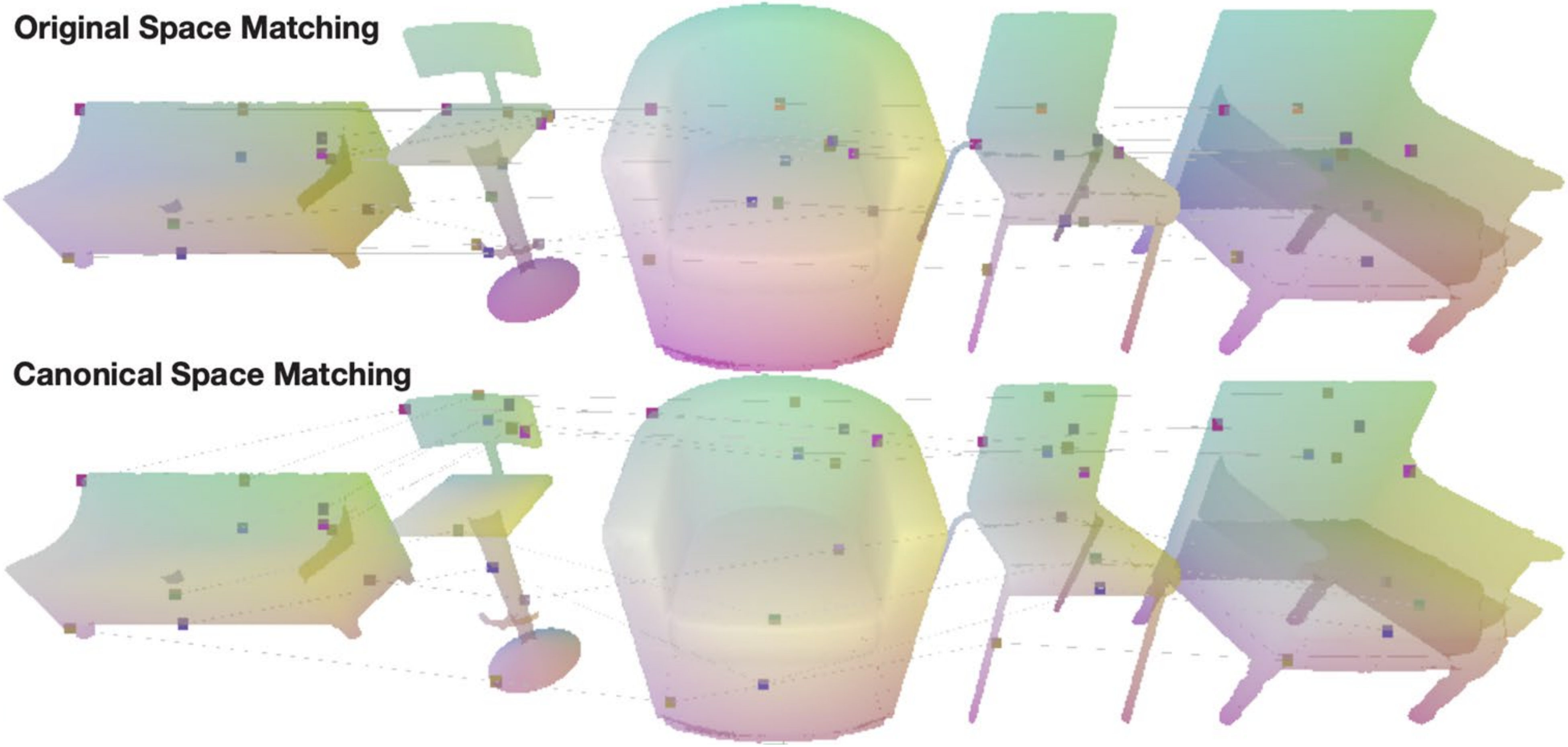


Mean-Shape at “hub”:



Shapes naturally align when deformed to the common “hub” in an unsupervised manner.

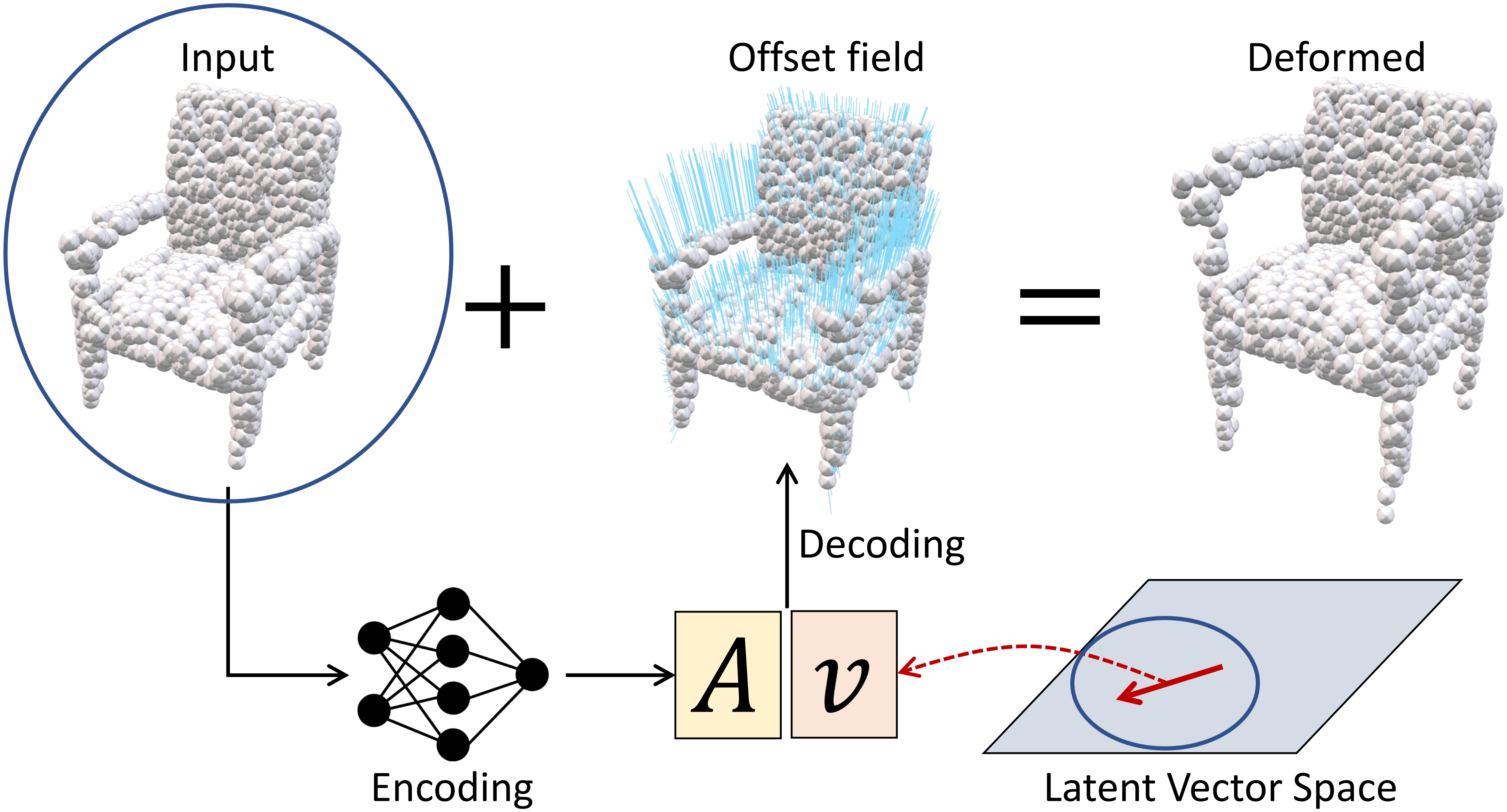
Experiment: Unsupervised Shape Correspondences



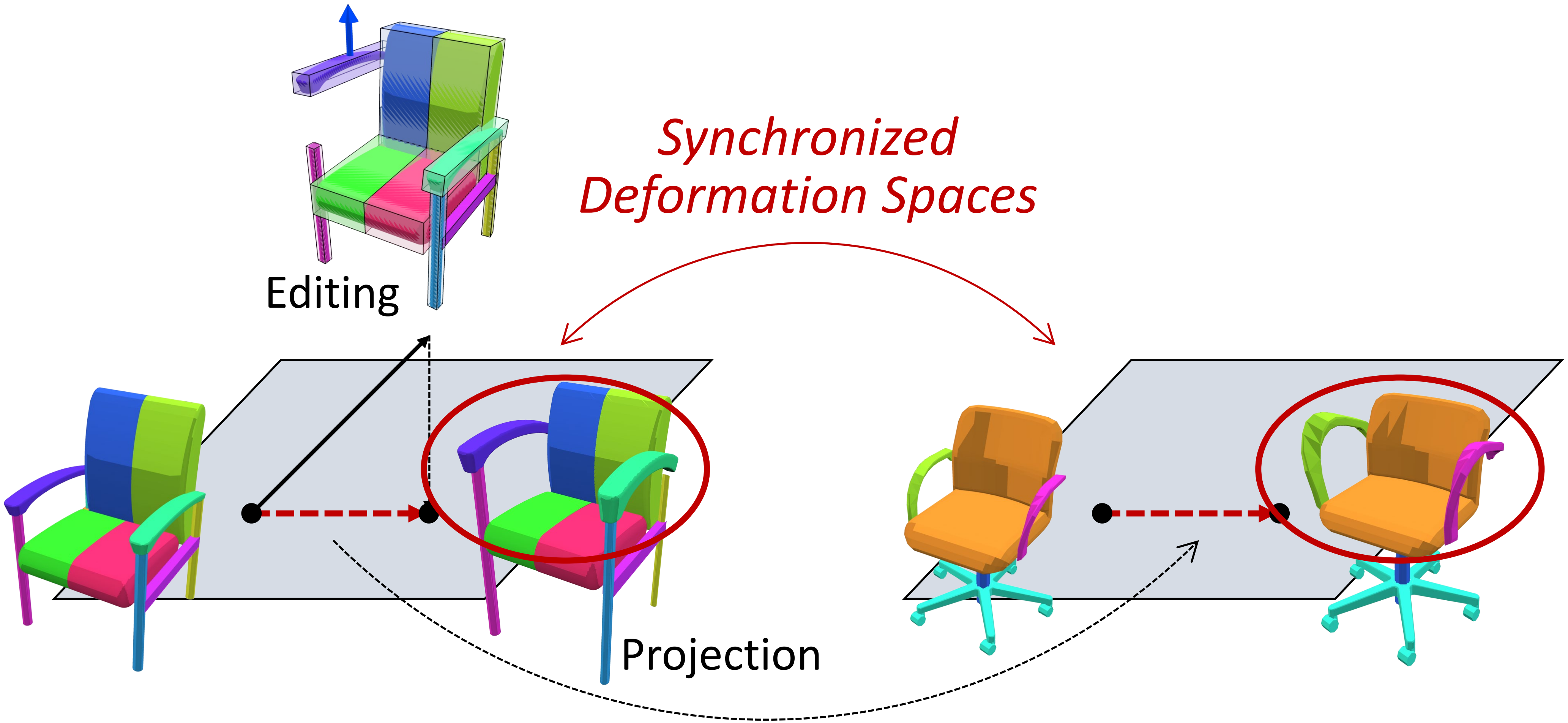
DeformSyncNet

DeformSyncNet: Deformation Transfer via Synchronized Shape Deformation Spaces . Minhyuk Sung, Zhenyu Jiang, Panos Achlioptas, Niloy J. Mitra, and Leonidas J. Guibas. SIGGRAPH Asia 2020

Embedding Space for Deformations and Variations

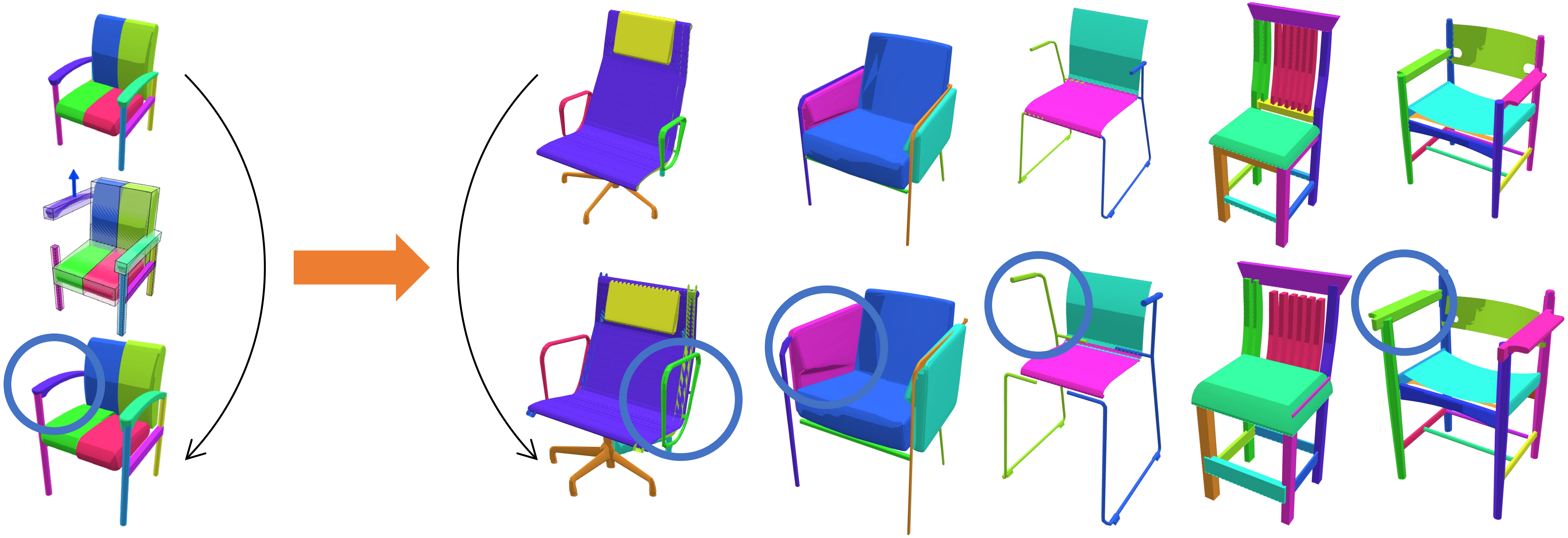


Synchronized Shape Edit/Deformation Spaces



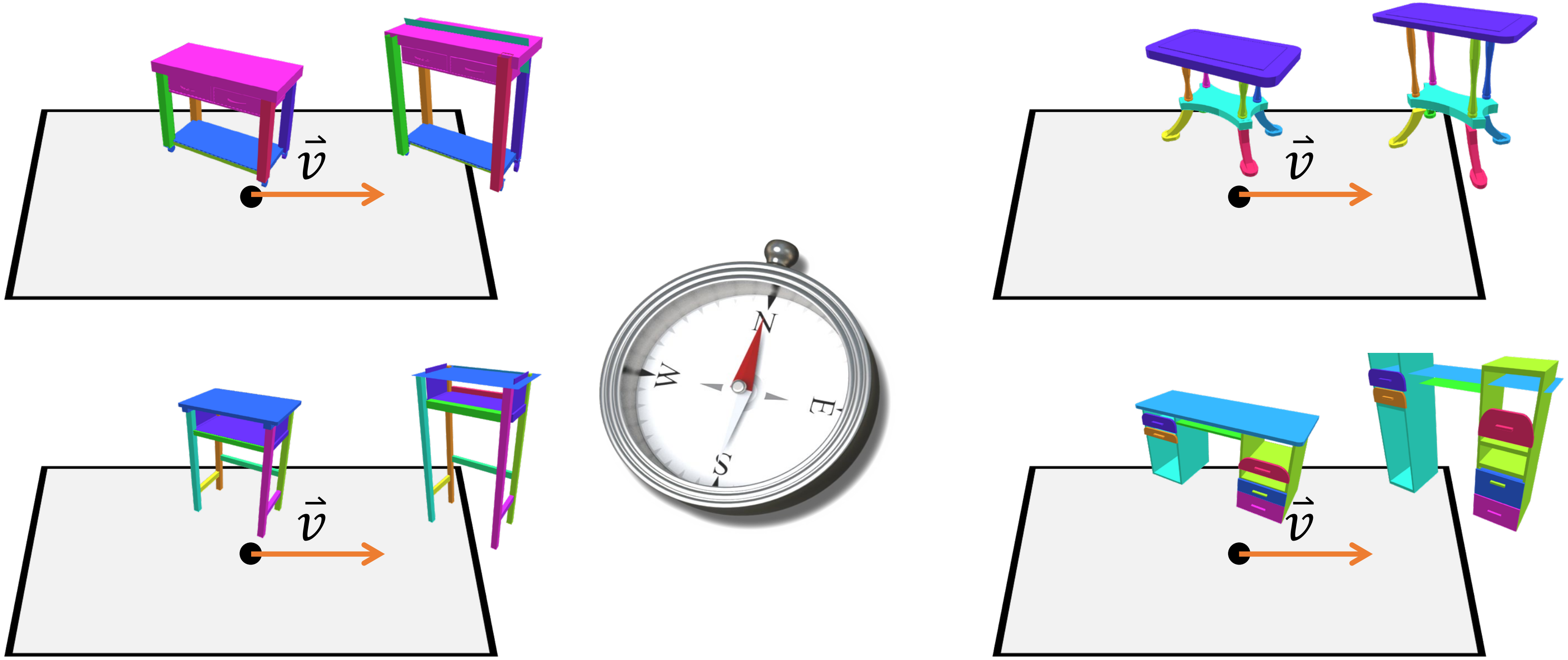
Learning and Exploiting Correlations in Deformations

Transfer deformations across shapes without correspondences.



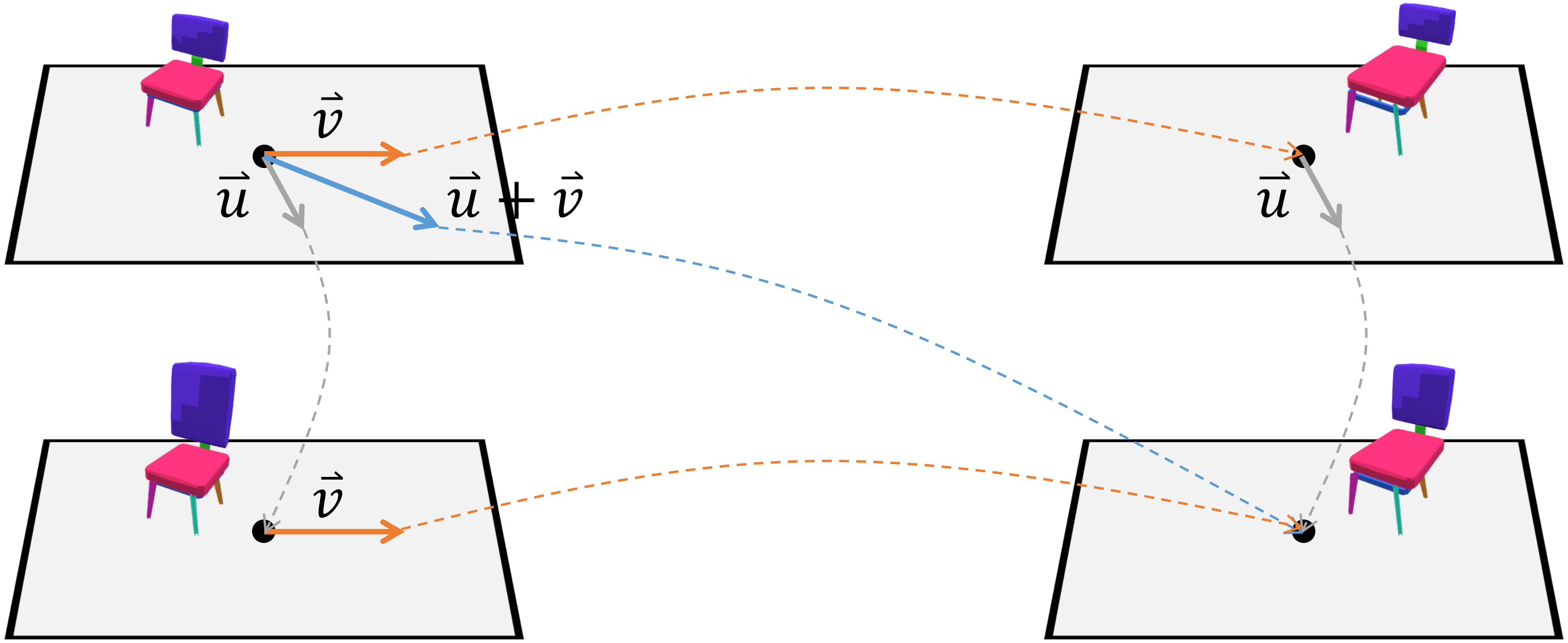
Consistency

We aim to have a latent vector meaning the same thing **everywhere**:
e.g., $\vec{v} = \langle 1, 0, \cdots, 0 \rangle$ Indicates "elongate legs".



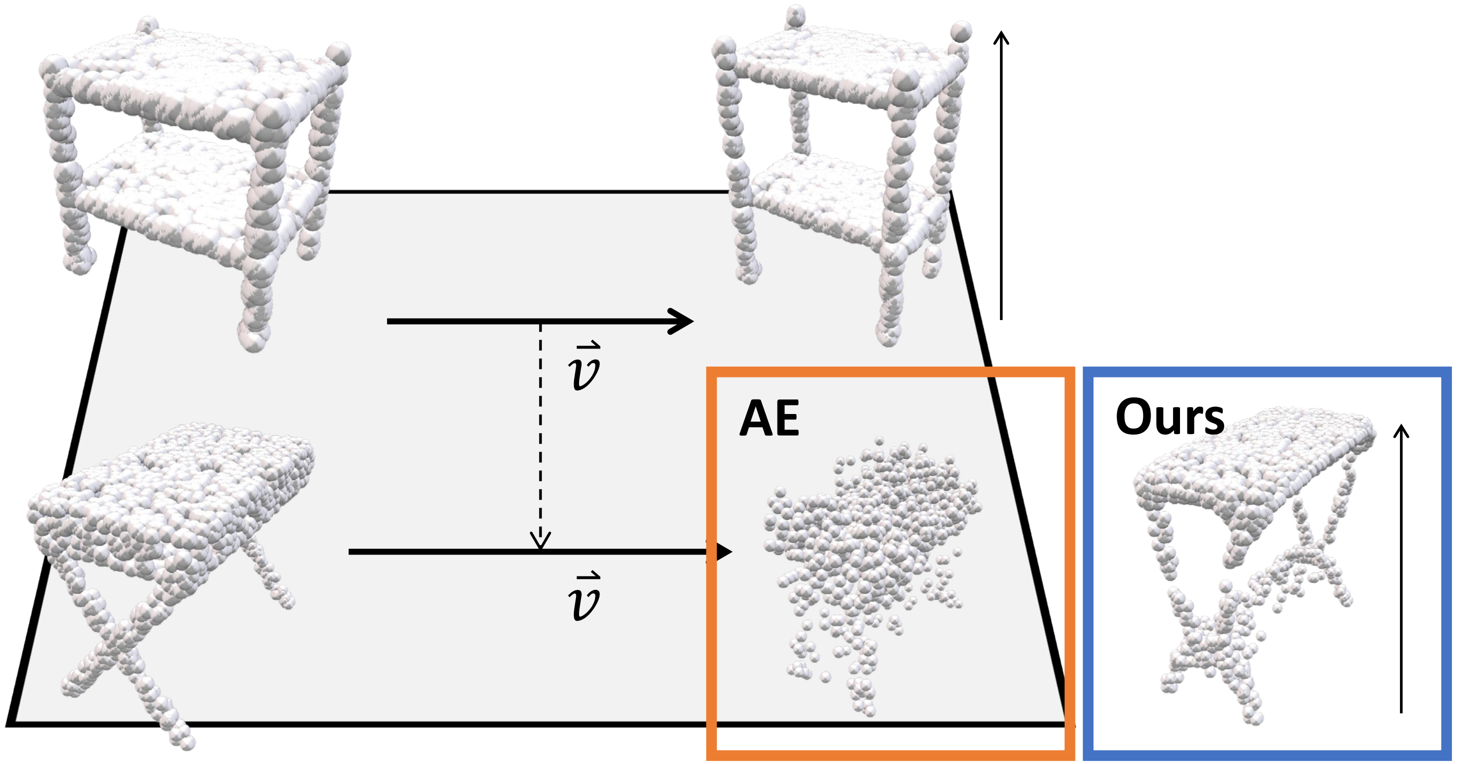
Path Invariance

We want to reach to the same destination, no matter which route we choose.



Autoencoder Latent Space

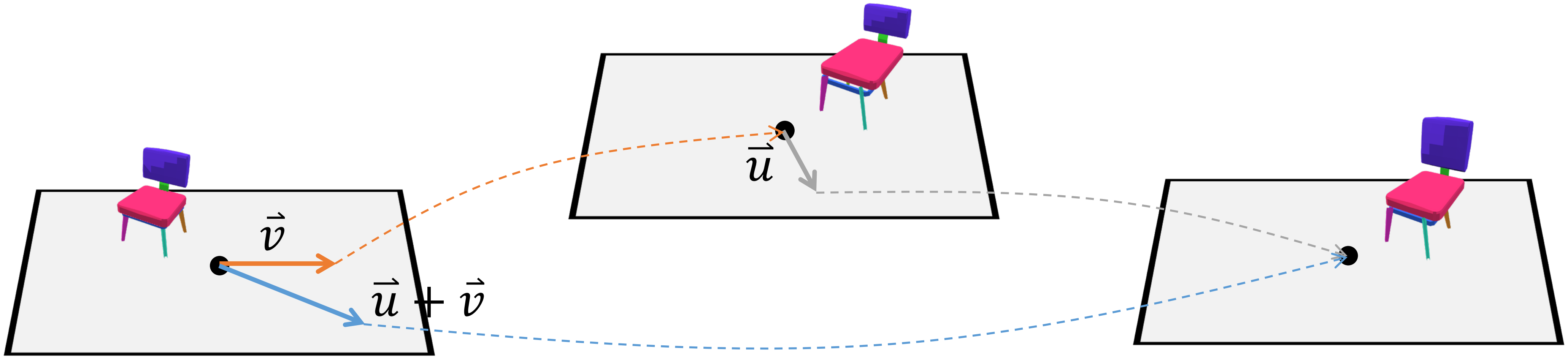
The axes of autoencoder latent spaces are not typically associated with semantically meaningful shape changes.



A Latent Space for Deformations/Actions

An affine latent action space satisfies the following property:

Additive action: $x \in X,\ \vec{u}, \vec{v} \in V,\ (x \oplus \vec{u}) \oplus \vec{v} = x \oplus (\vec{u} + \vec{v})$.



Autoencoder

An action defined with an antoencoder:

$$\boxed{x \oplus \vec{v} \coloneqq \mathcal{D}(\mathcal{E}(x) + \vec{v}).}$$

does not guarantee additivity and transitivity.

- A vector $\vec{v}$ can act differently given the shape.
- Multiple vectors can be decoded to the same deformation.

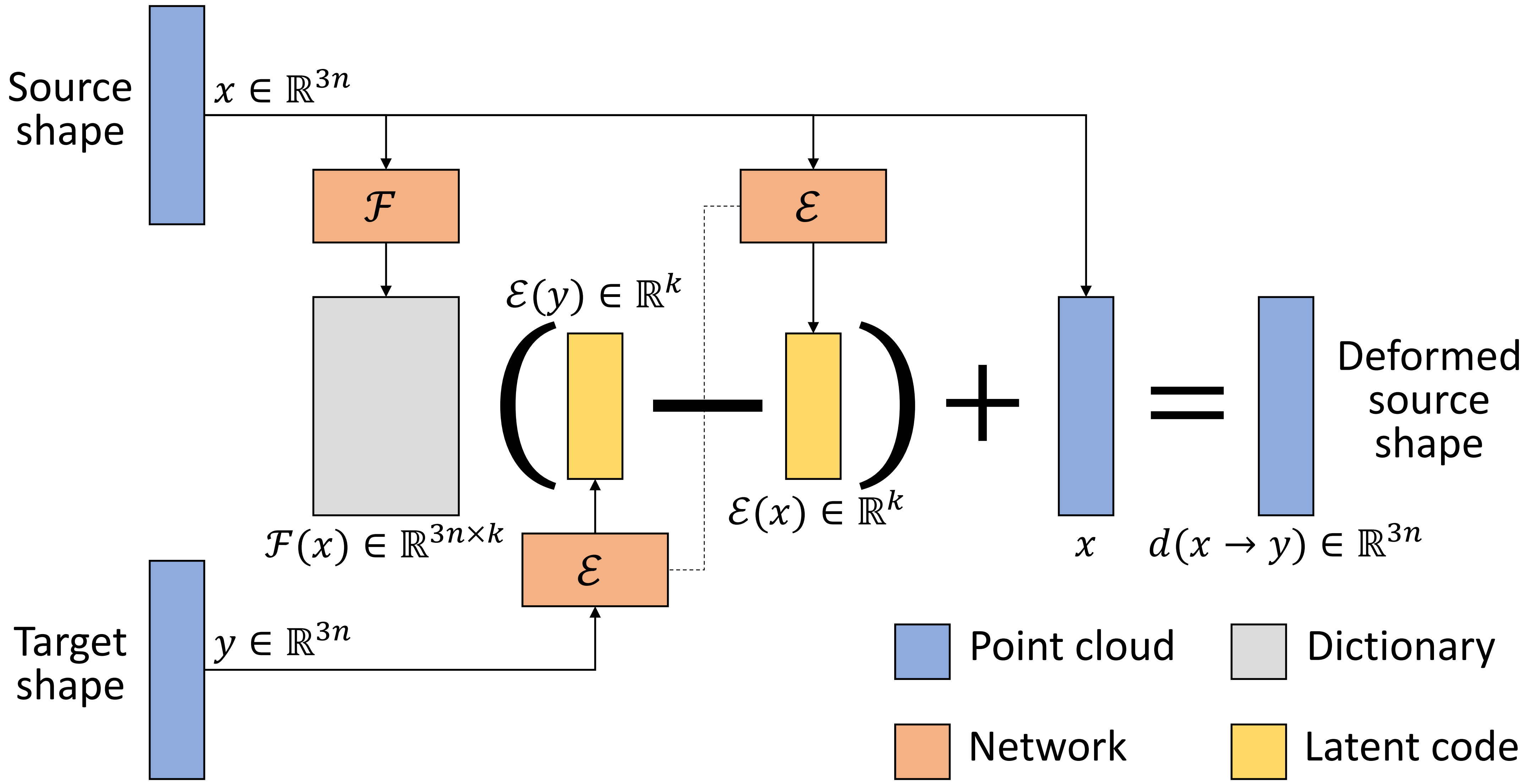
Another Solution

We predict the deformation dictionary for each shape using another dictionary prediction network $\mathcal{F} \in \mathbb{R}^{3n} \rightarrow \mathbb{R}^{3n\times k}$.

The deformation $\mathrm{d}(x \rightarrow y)$ from shape x to y is computed as:

$$\boxed{\mathrm{d}(x \rightarrow y) = \mathcal{F}(x)\big(E(y) - E(x)\big) + x.}$$

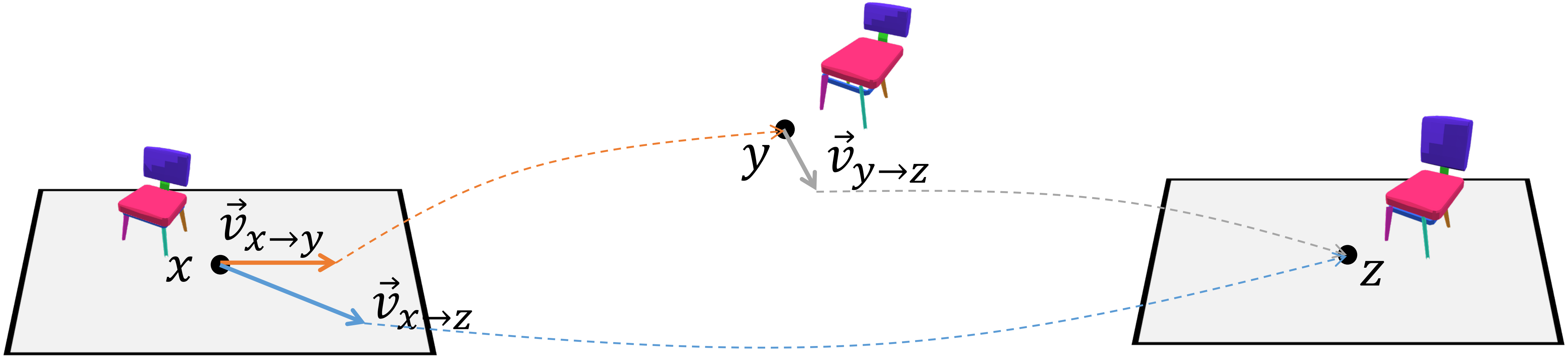
Neural Network



Path Invariance

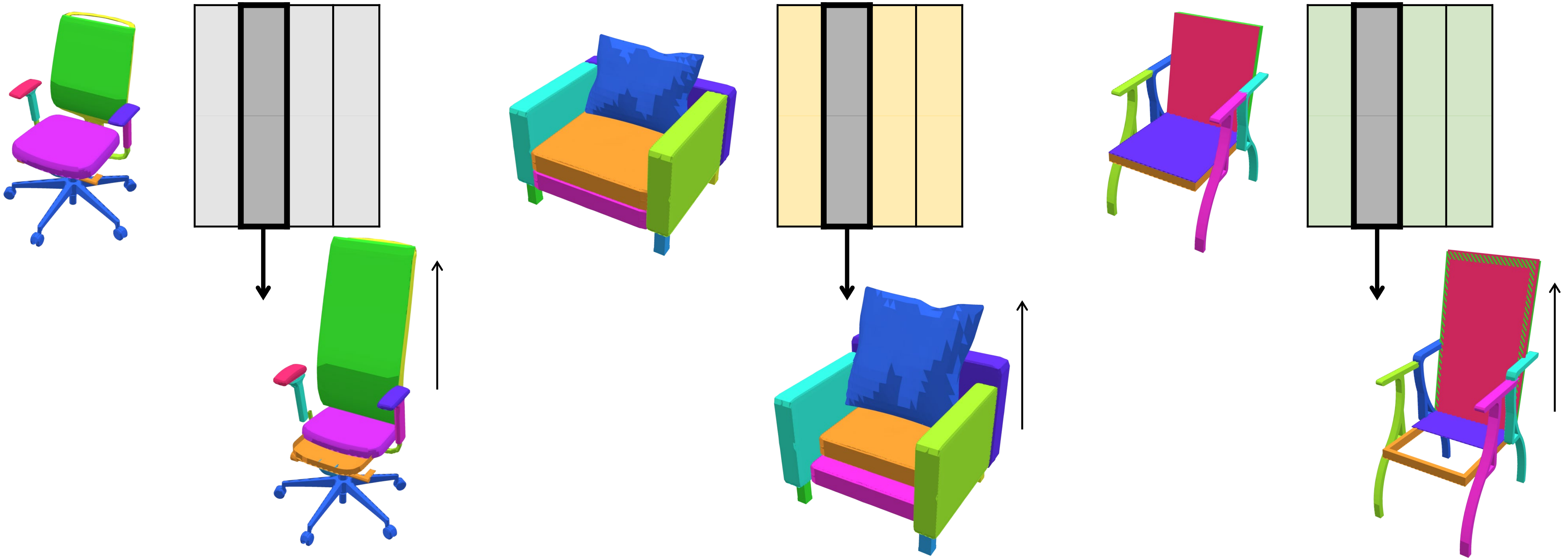
Reinforce consistency by imposing path invariance.

$$\boxed{\vec{\boldsymbol{v}}_{x\to y} + \vec{\boldsymbol{v}}_{y\to z} = \vec{\boldsymbol{v}}_{x\to z}}$$



Consistent Deformation Dictrionaries $\mathcal{F}$

Consistency across the deformation dictionaries **emerges** during training.

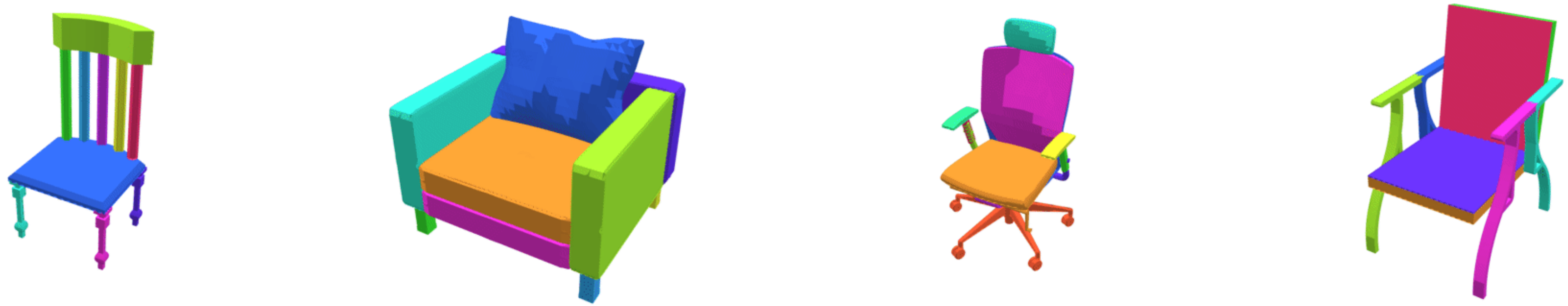


Deformation Dictionary

Translating **seat** along the up/down direction.

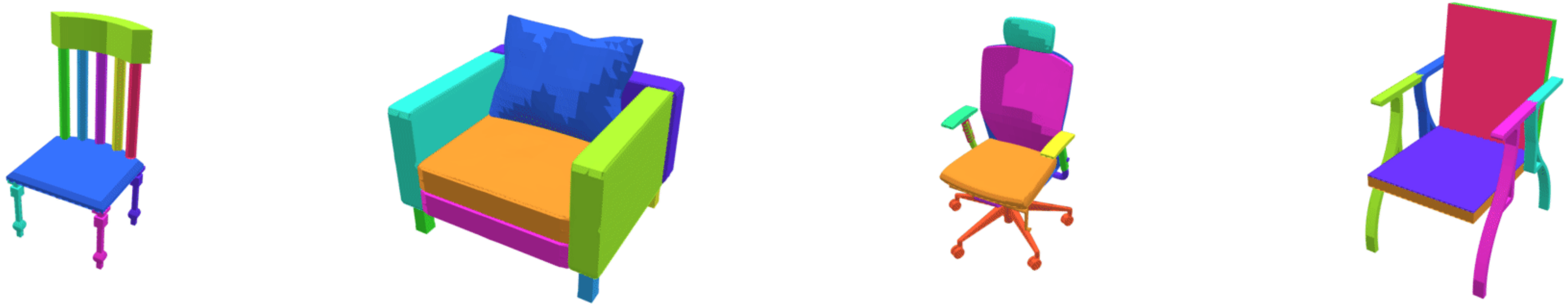
Deformation Dictionary

Translating **back** along the front/back direction.

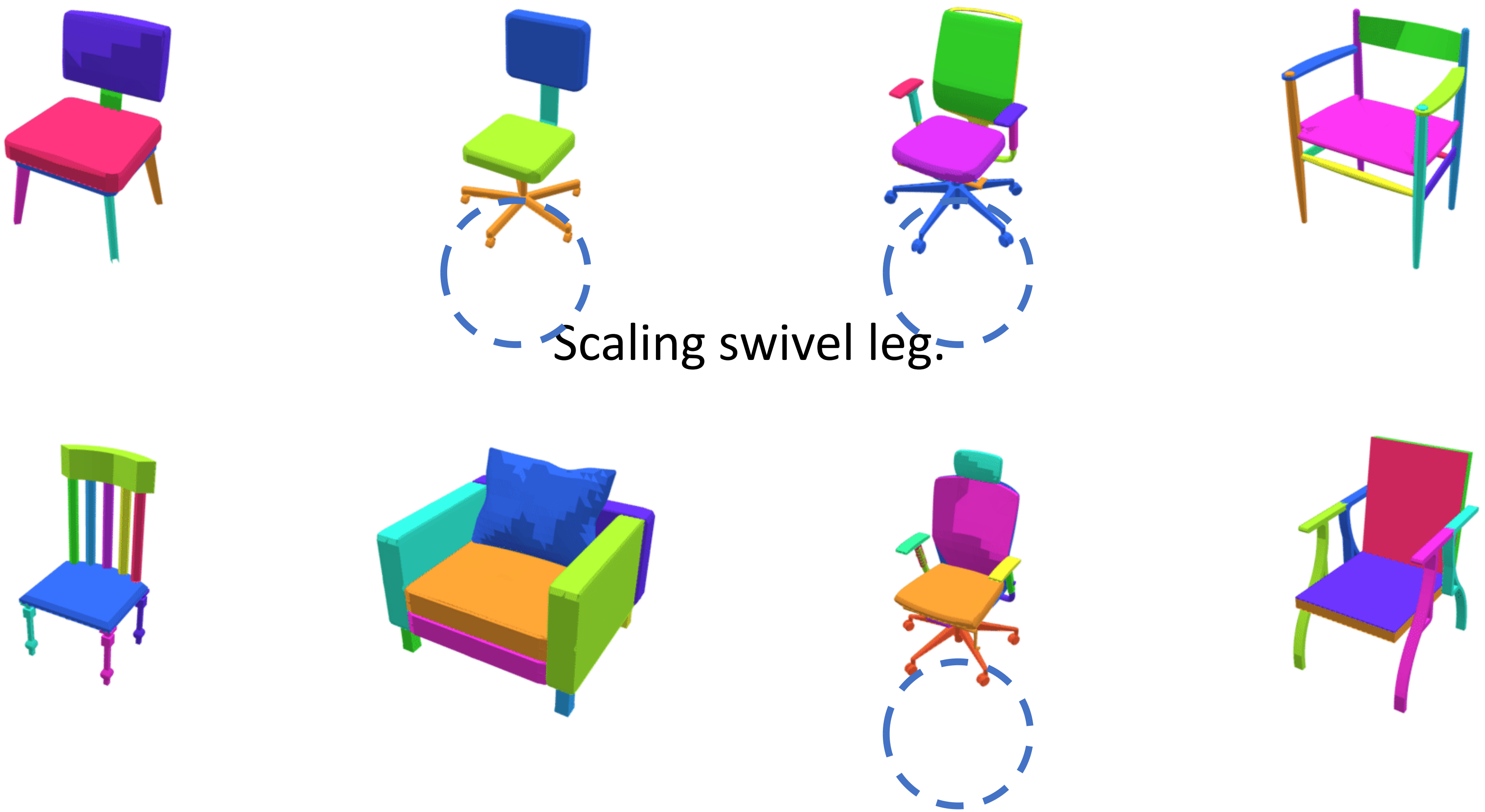


Deformation Dictionary

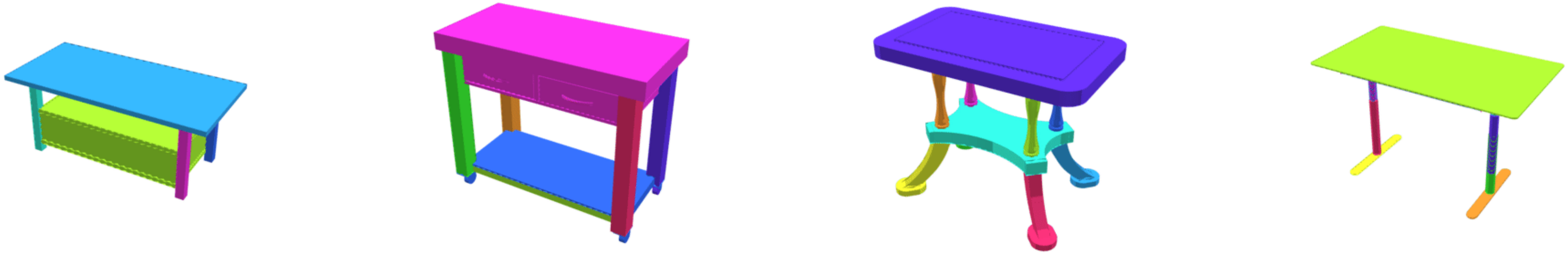
Scaling **back** along the up/down direction.



Deformation Dictionary

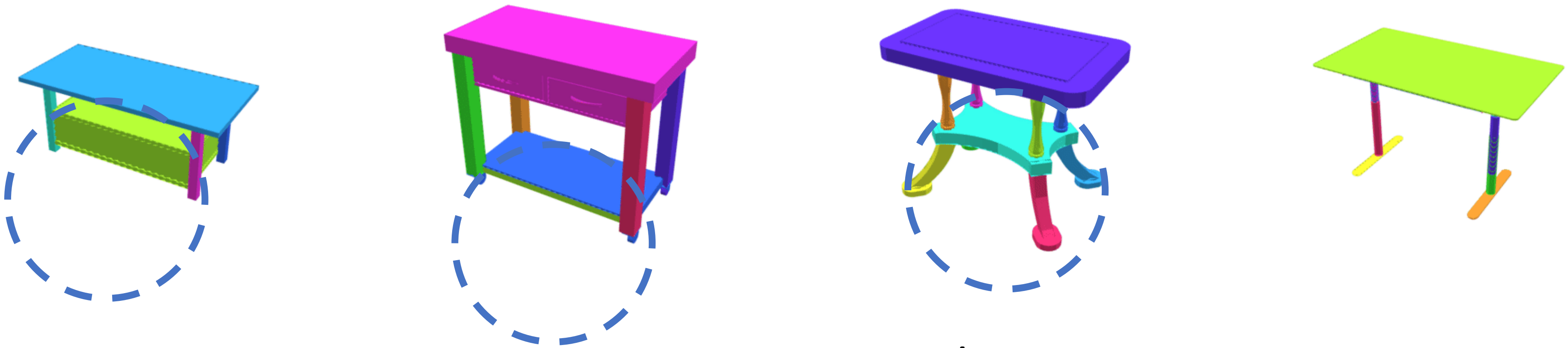


Deformation Dictionary

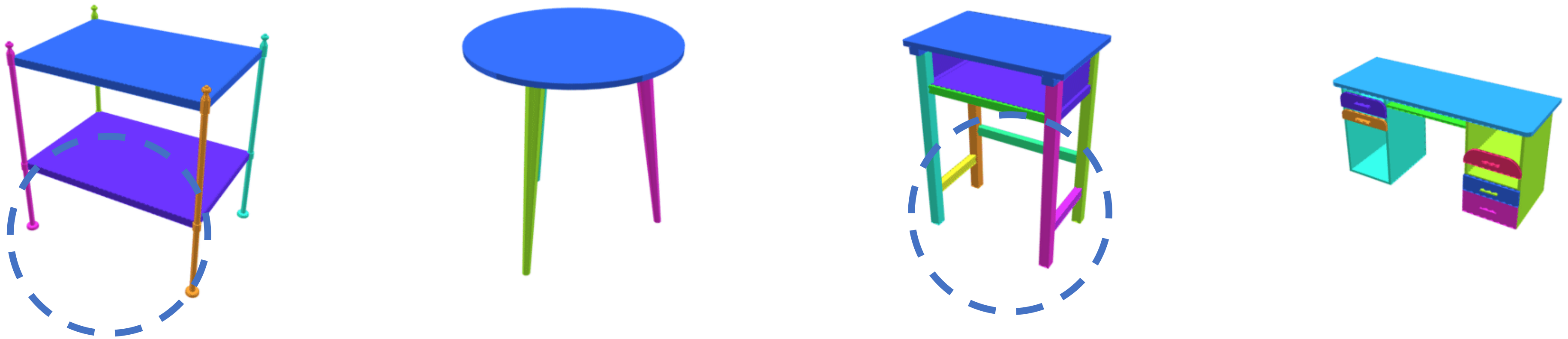


Scaling along the front/back direction.

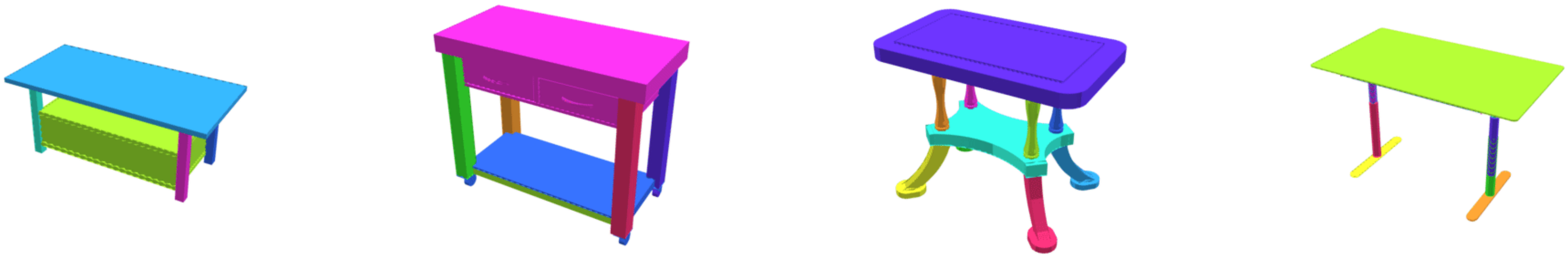
Deformation Dictionary



Translating **shelf** along the up/down direction.

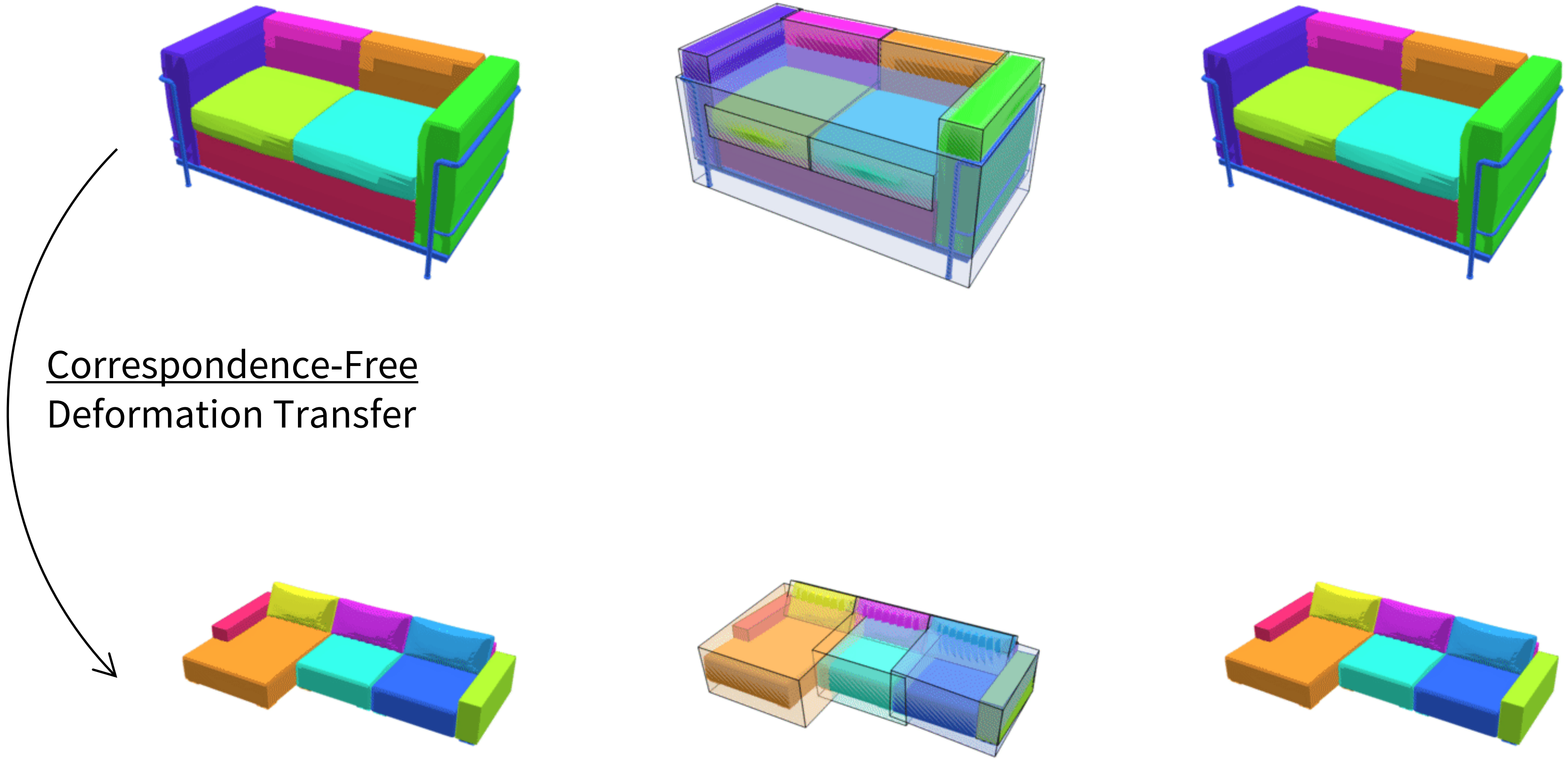


Deformation Dictionary

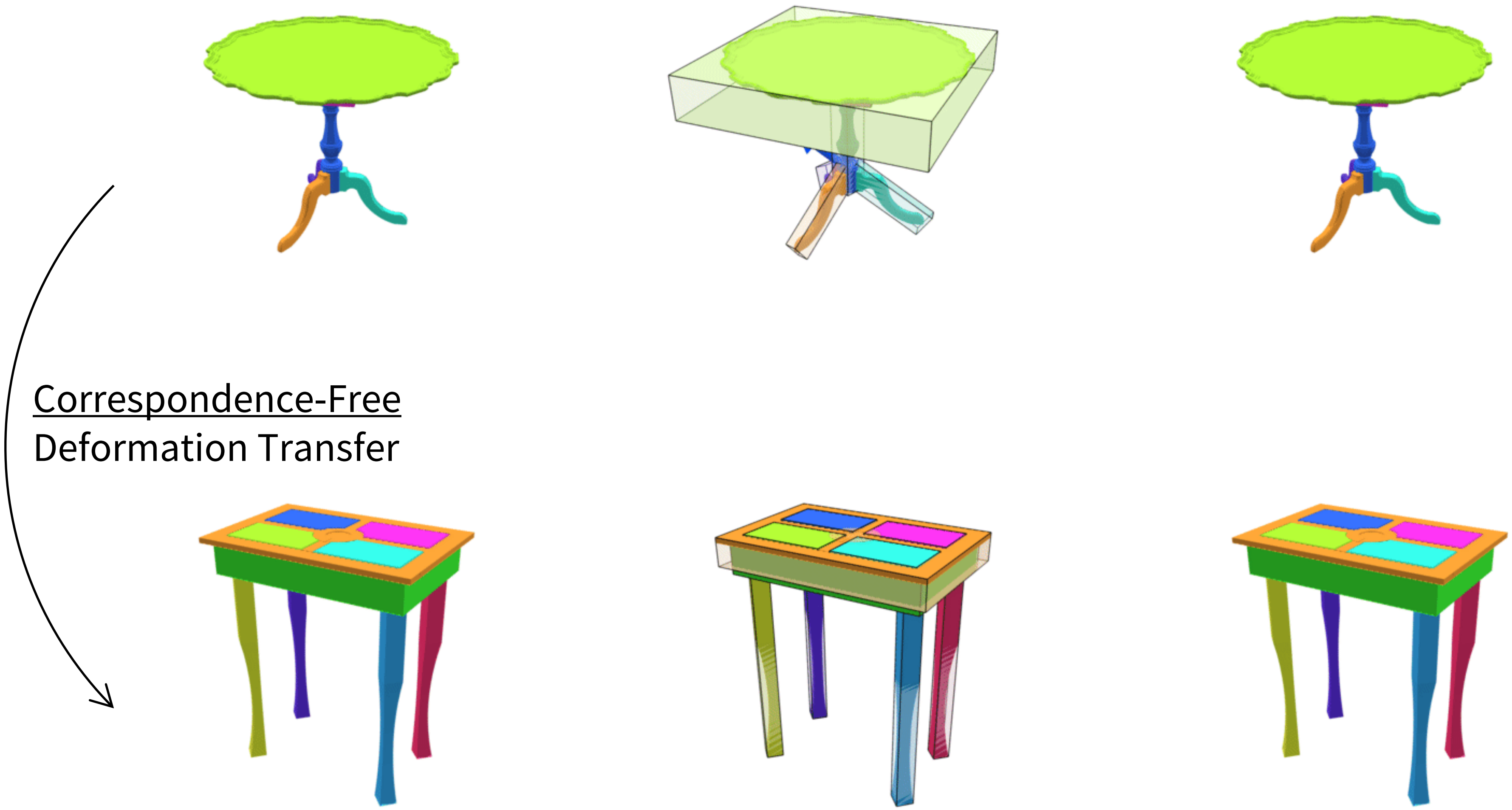


Translating **top** along the up/down direction.

Projection & Deformation Transfer



Projection & Deformation Transfer

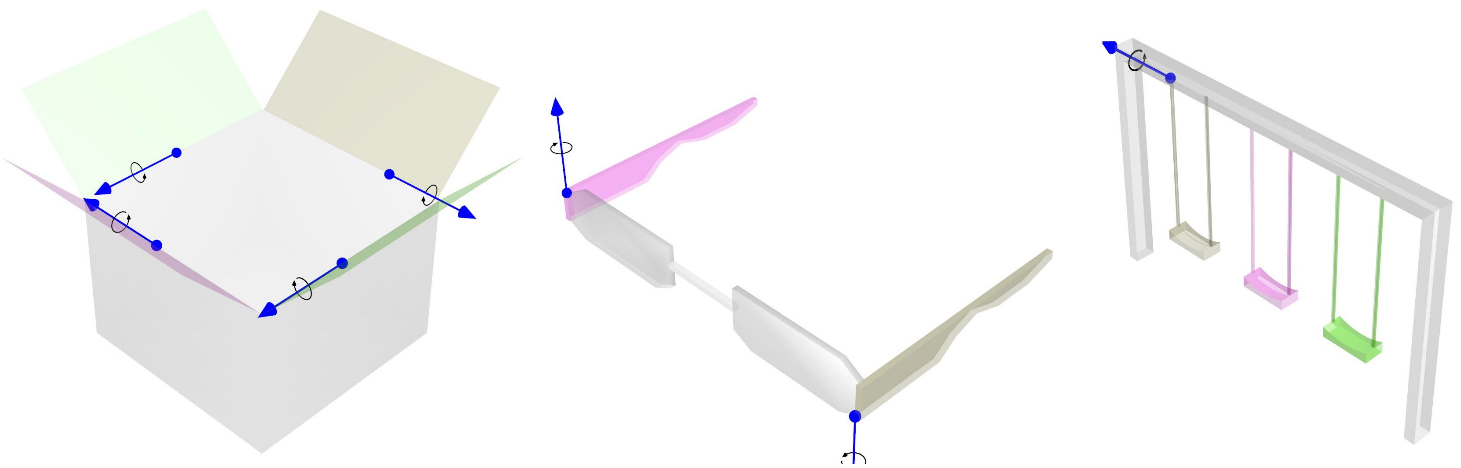


Projection & Deformation Transfer

Extension to Circular Motion

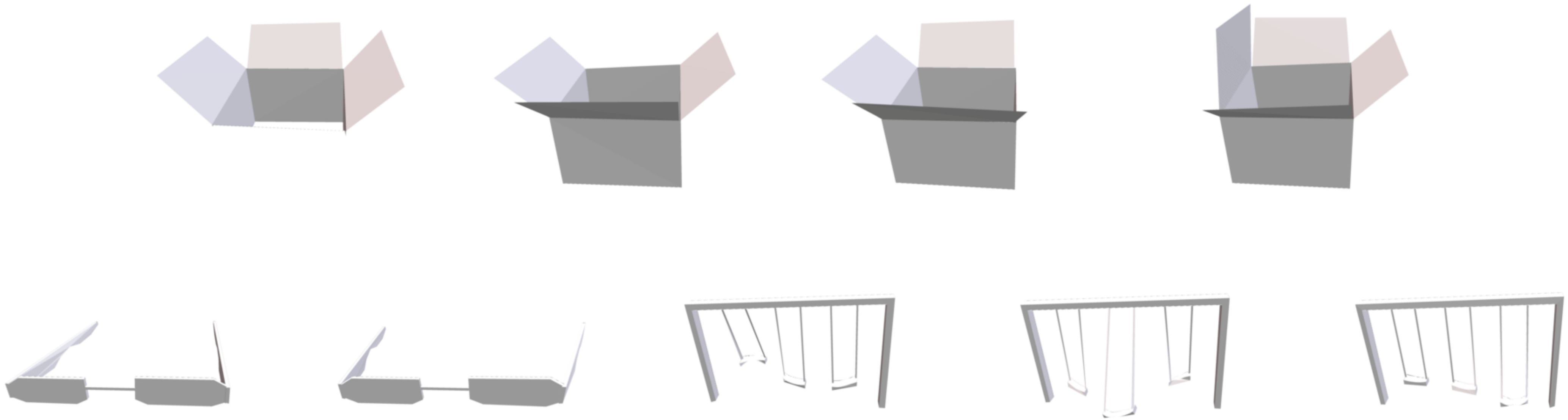
- Each item in the dictionary indicates a linear motion for each point.
- We change our formulation to predict a circular motion per point.

Linear Motion



Shape2Motion Dataset (Wang et al., 2019)

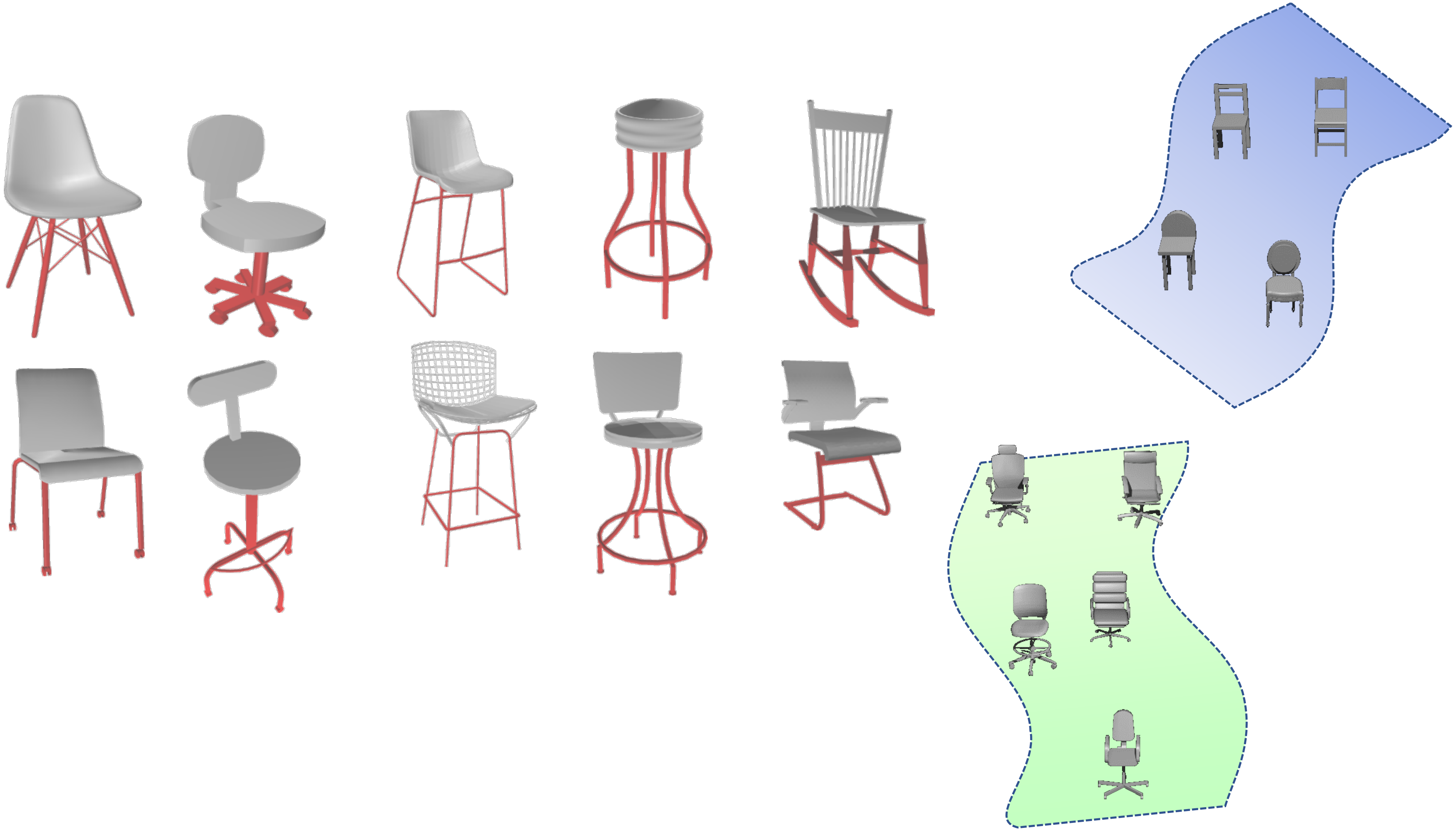
Extension to Circular Motion



NOTE: Transitivity is not guaranteed — there can exist multiple latent codes describing a specific deformation.

Discrete Shape Variations

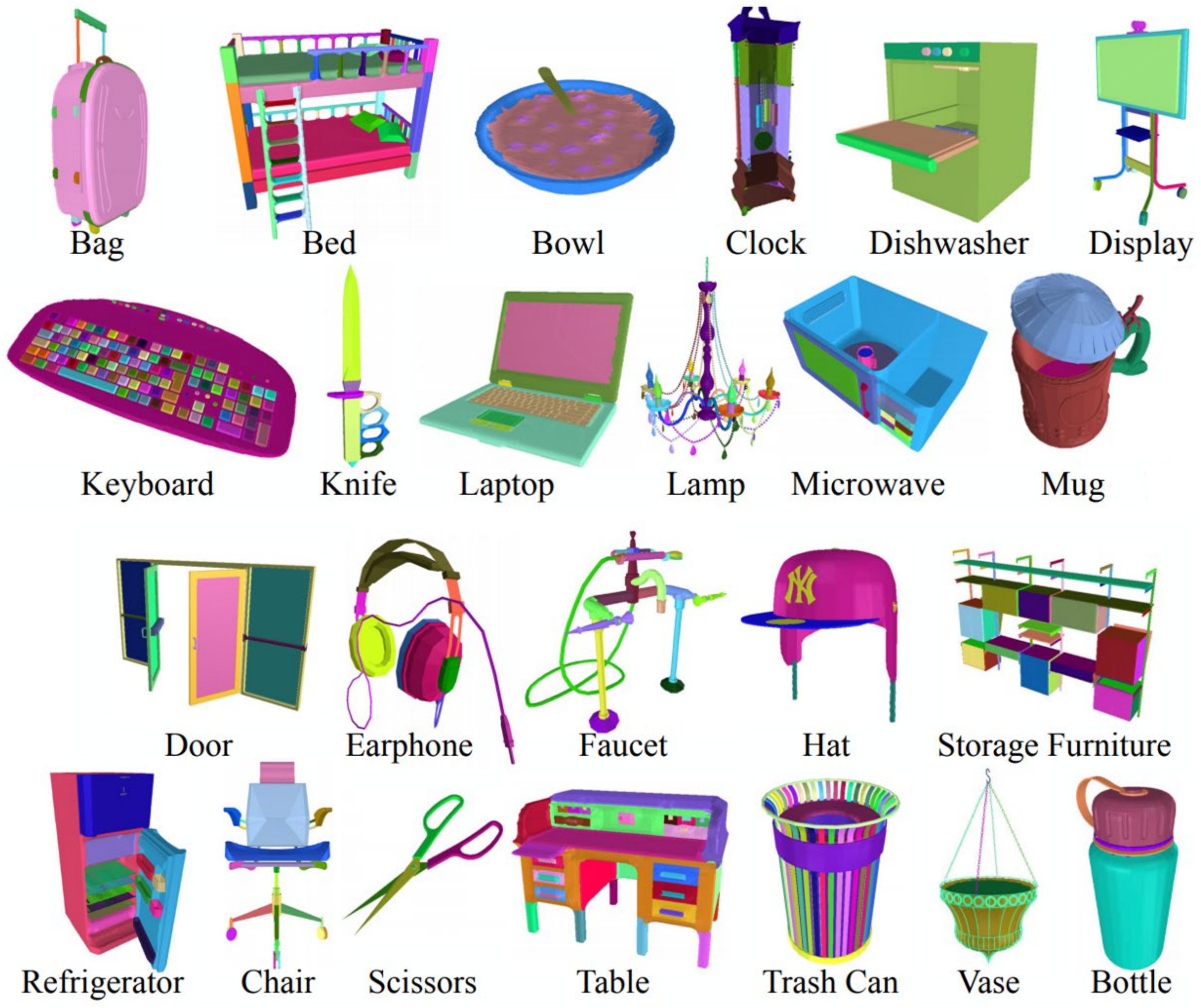
Combinatorial or Discrete Variability



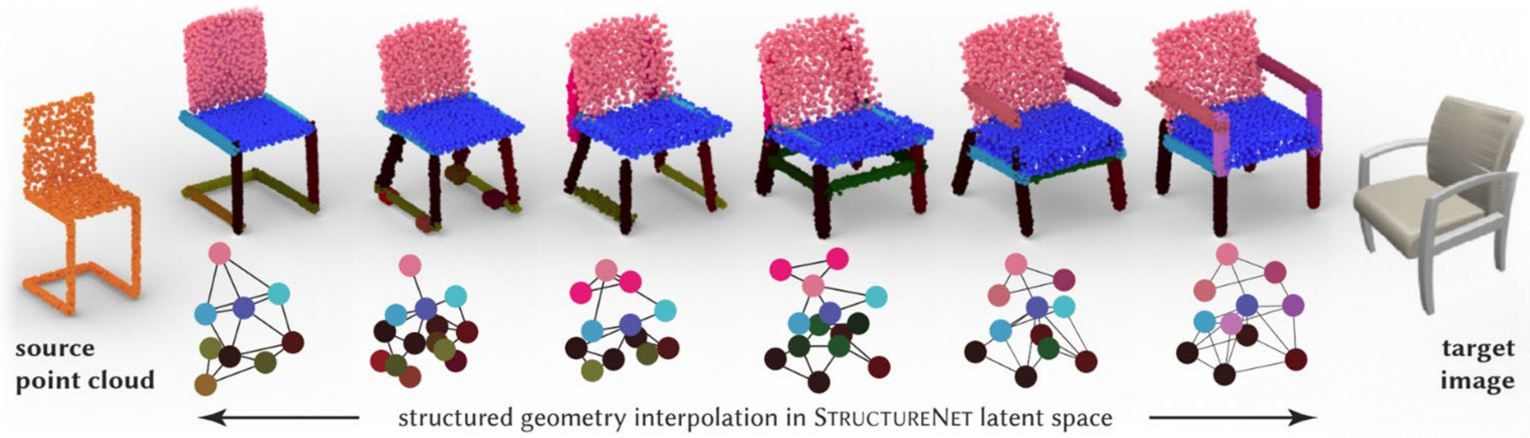
StructEdit

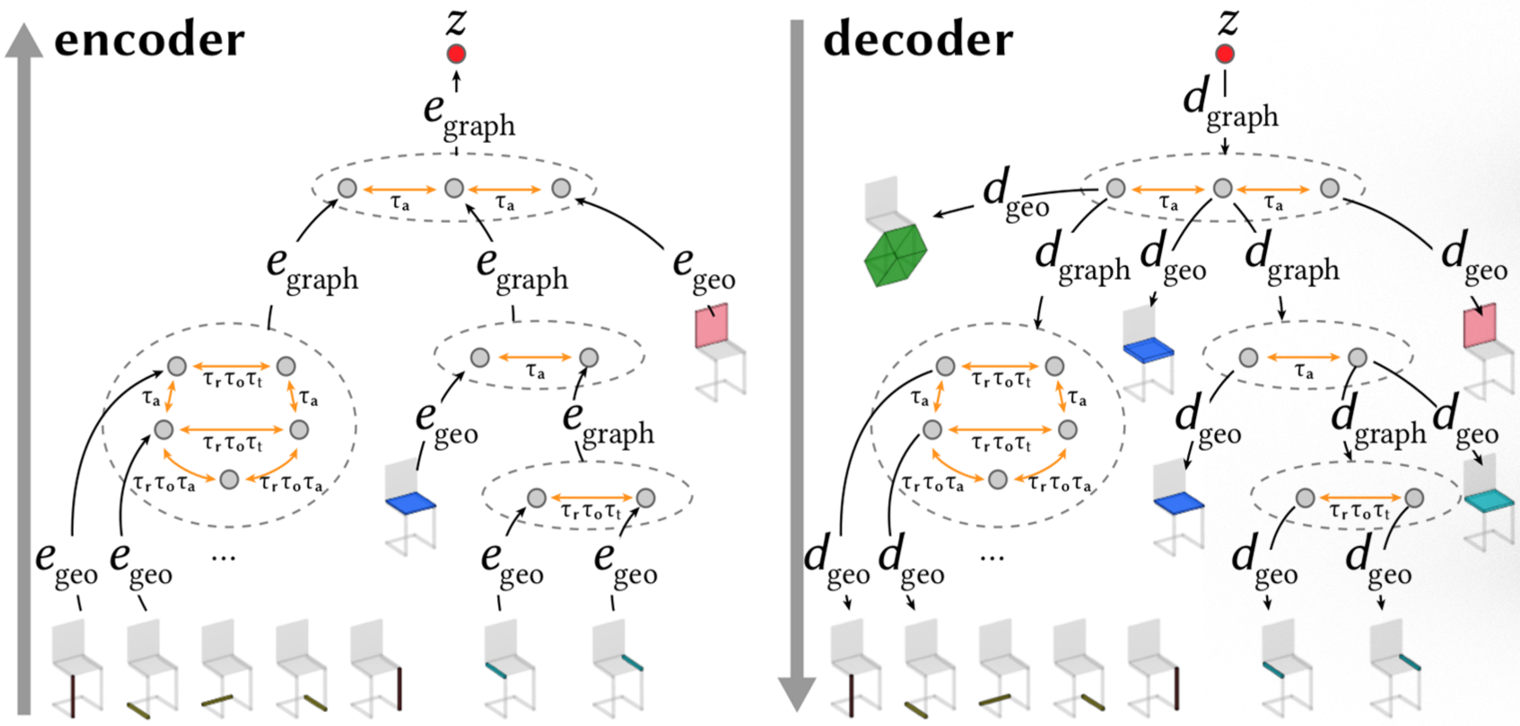
StructEdit: Learning Structural Shape Variations. Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, Leonidas J. Guibas. CVPR 2020.

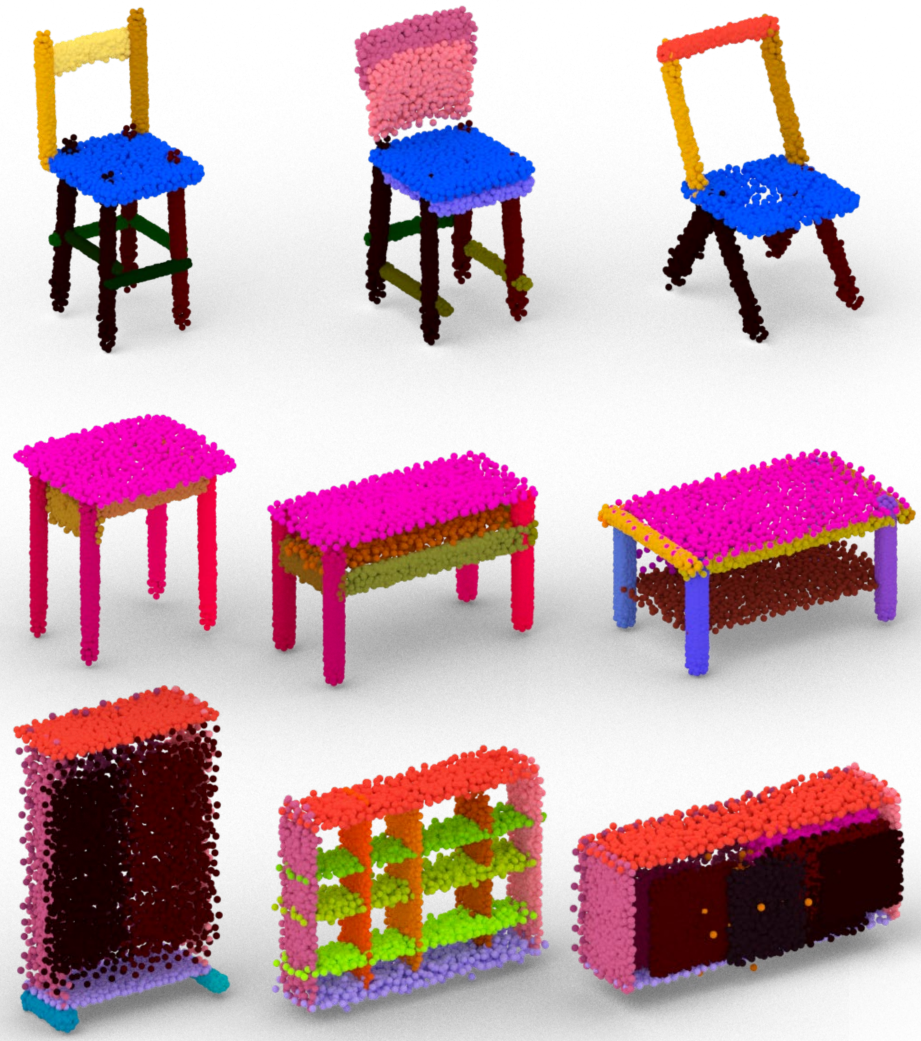
Recap: PartNet (*CVPR 2019*)



Recap: StructureNet (*Siggraph Asia 2019*)



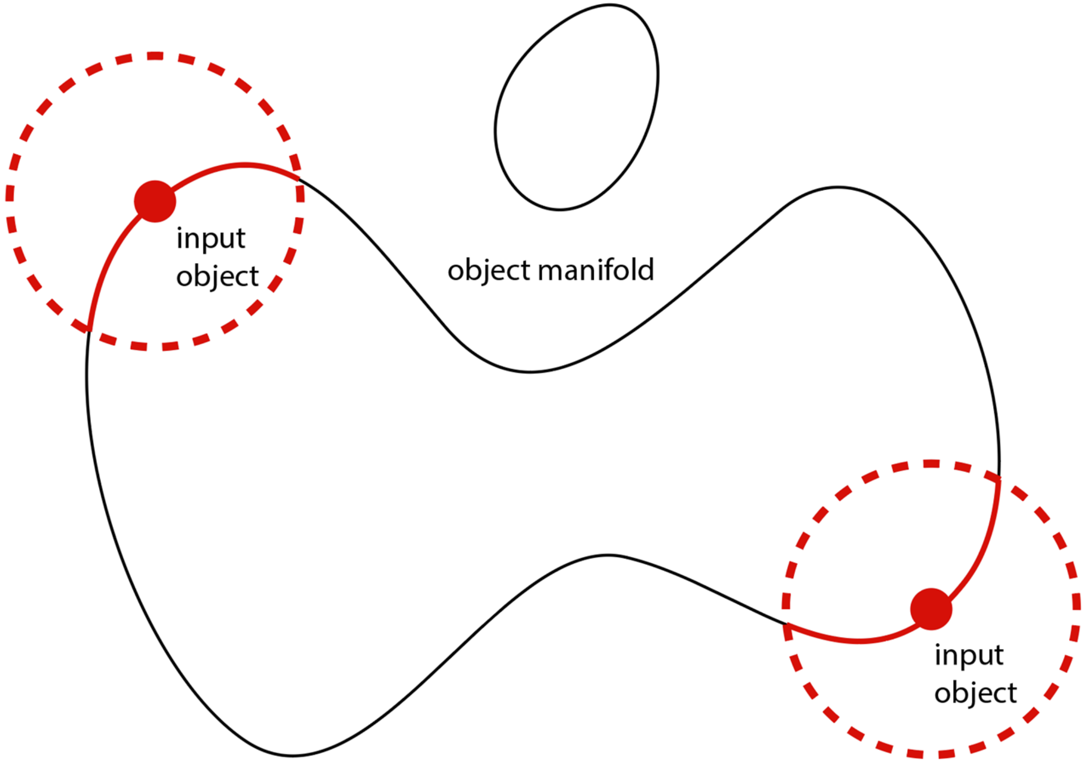




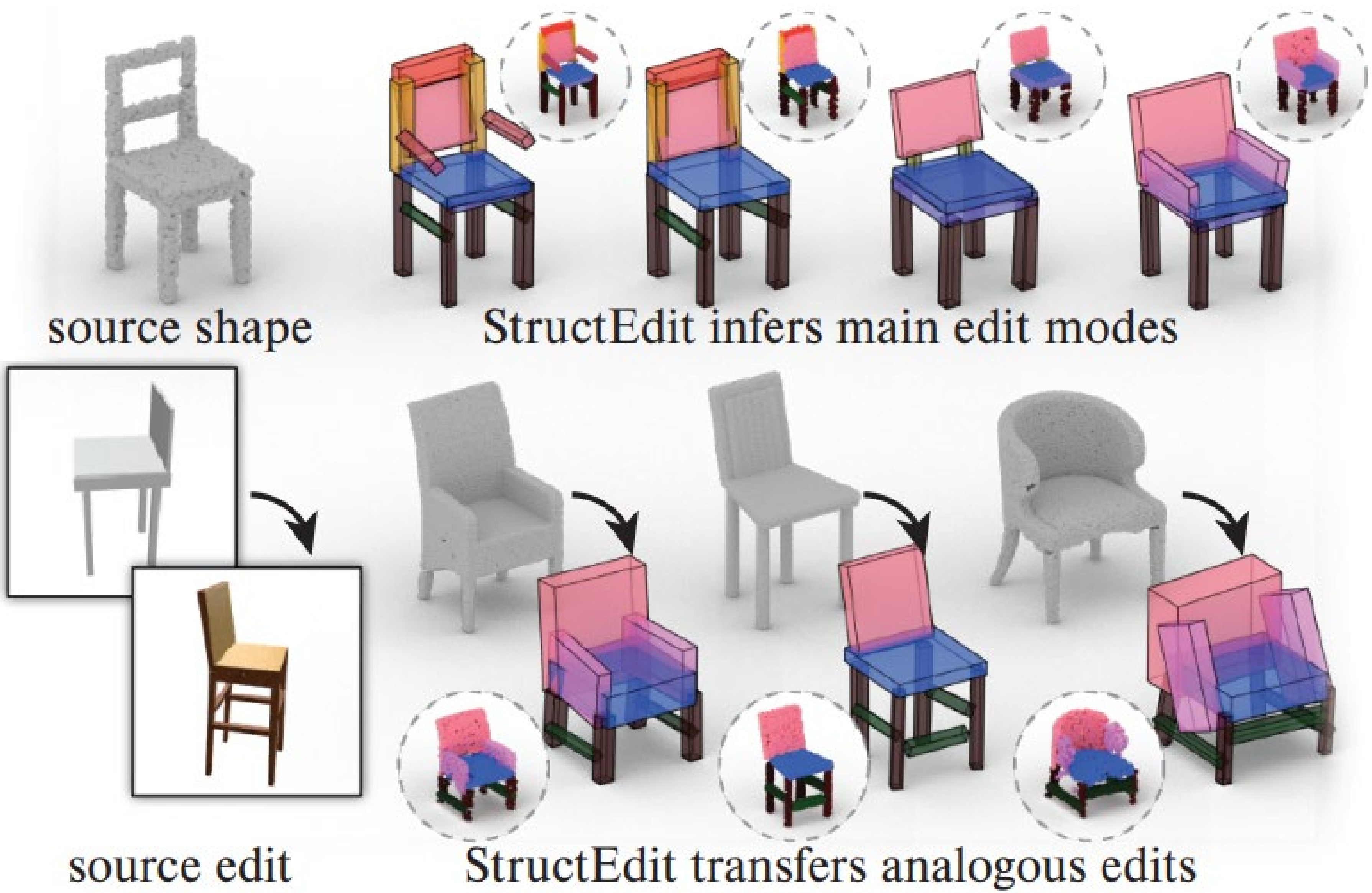
Learn to Embed Local Shape Neighborhoods

Learn a Structural Shape-diff Space, which is

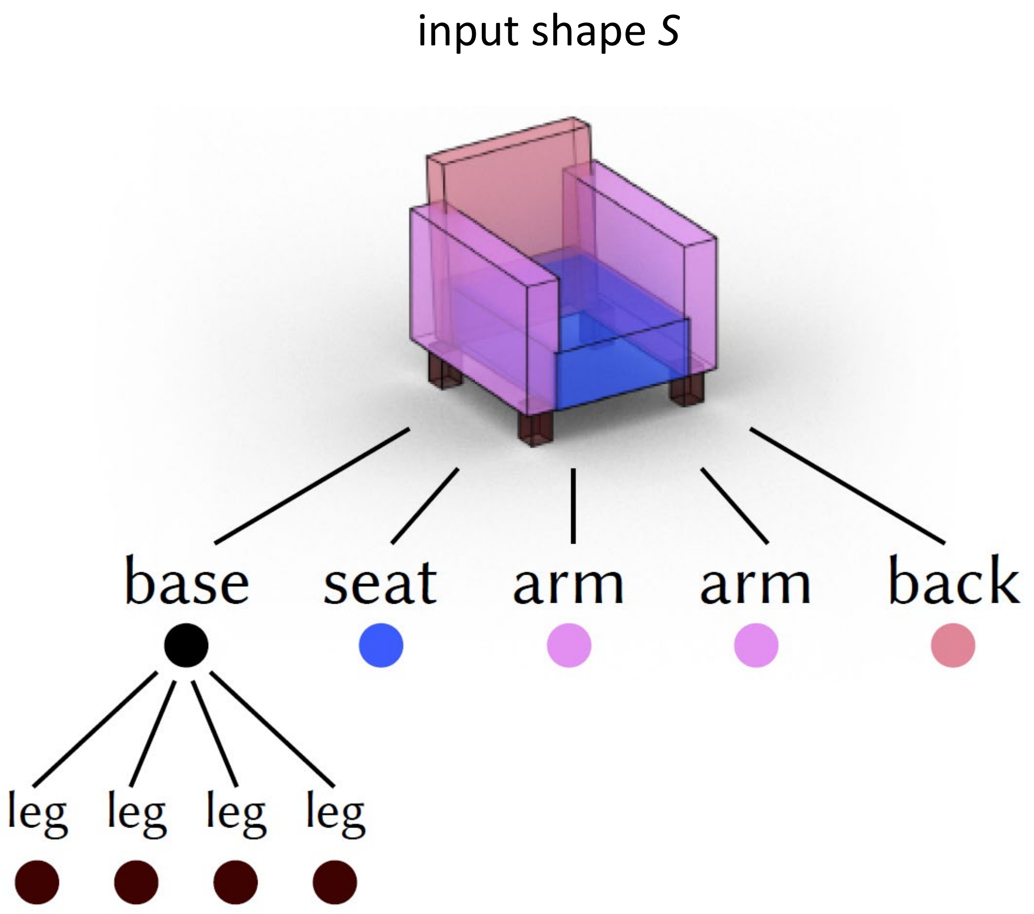
- **Unified**: learn one space that can be applied for any input shape
- **Specialized**: suggest different plausible shape-diff's for different input shapes
- **Coherent**: suggest similar plausible shape-diff's for similar input shapes

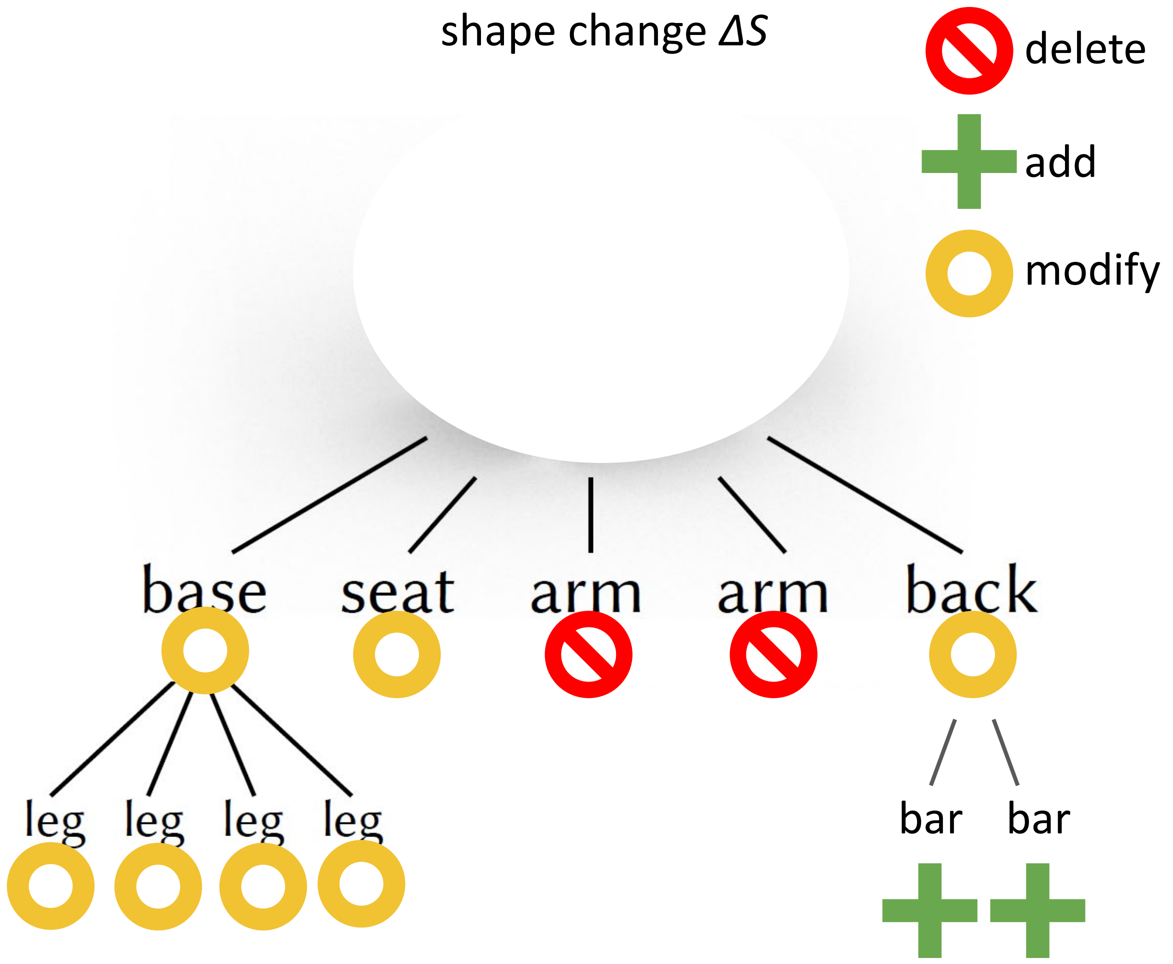


Learn Shape Structural Variations: StructEdit



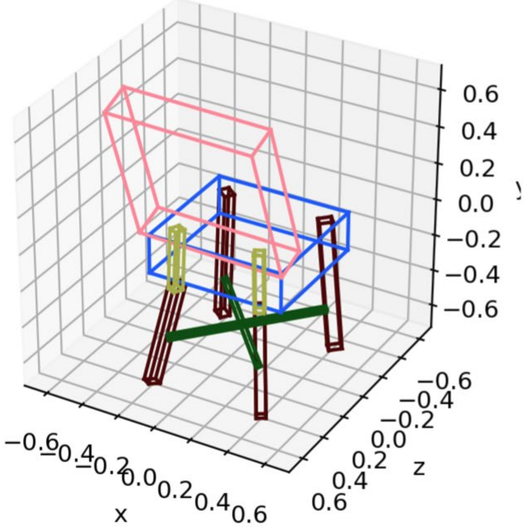
Shape Difference (Structure + Deformation)





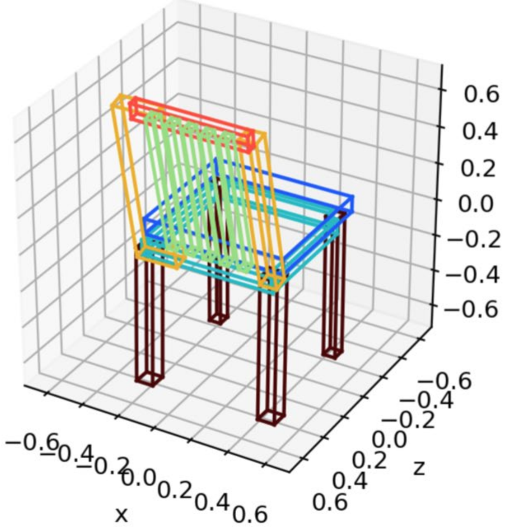
Shape Hierarchy Differences

Shape A




```
0 chair    {0}
├0 chair_back    {1}
│  ├0 back_surface     {2}
│  │  ├0 back_single_surface [LEAF] {3}
│  ├1 back_connector [LEAF] {4}
│  ├2 back_connector [LEAF] {5}
├1 chair_seat     {6}
│  ├0 seat_surface     {7}
│  │  ├0 seat_single_surface [LEAF] {8}
├2 chair_base     {9}
│  ├0 regular_leg_base     {10}
│  │  ├0 leg [LEAF] {11}
│  │  ├1 leg [LEAF] {12}
│  │  ├2 leg [LEAF] {13}
│  │  ├3 leg [LEAF] {14}
│  │  ├4 bar_stretcher [LEAF] {15}
│  │  ├5 bar_stretcher [LEAF] {16}
```

Shape B



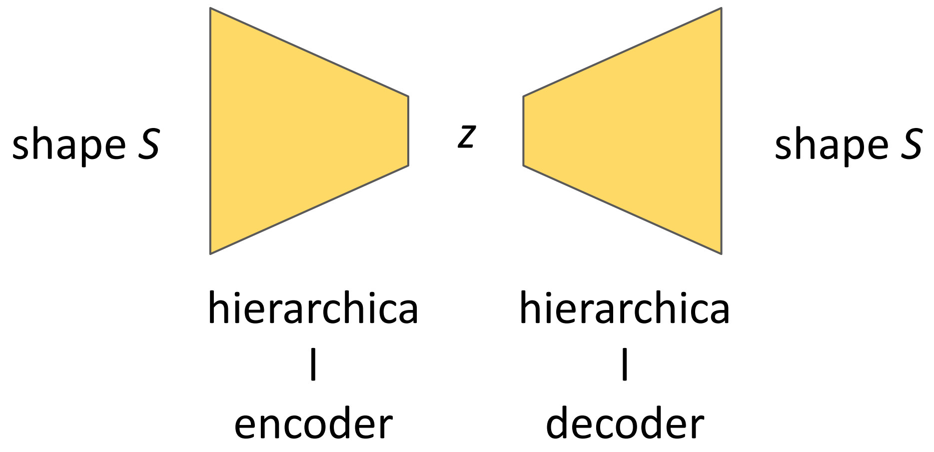

```
0 chair    {0}
├0 chair_back    {0}
│  ├0 back_surface     {0}
│  │  ├0 back_surface_vertical_bar [LEAF] {3}
│  │  ├1 back_surface_vertical_bar [LEAF] {4}
│  │  ├2 back_surface_vertical_bar [LEAF] {5}
│  │  ├3 back_surface_vertical_bar [LEAF] {6}
│  │  ├4 back_surface_vertical_bar [LEAF] {7}
│  ├1 back_frame     {8}
│  │  ├0 back_frame_vertical_bar [LEAF] {9}
│  │  ├1 back_frame_vertical_bar [LEAF] {10}
│  │  ├2 back_frame_horizontal_bar [LEAF] {11}
├1 chair_seat     {0}
│  ├0 seat_surface     {0}
│  │  ├0 seat_single_surface [LEAF] {0}
│  ├1 seat_frame     {15}
│  │  ├0 seat_frame_bar [LEAF] {16}
│  │  ├1 seat_frame_bar [LEAF] {17}
│  │  ├2 seat_frame_bar [LEAF] {18}
│  │  ├3 seat_frame_bar [LEAF] {19}
├2 chair_base     {0}
│  ├0 regular_leg_base     {0}
│  │  ├0 leg [LEAF] {0}
│  │  ├1 leg [LEAF] {0}
│  │  ├2 leg [LEAF] {0}
│  │  ├3 leg [LEAF] {0}
```

Shape Diff

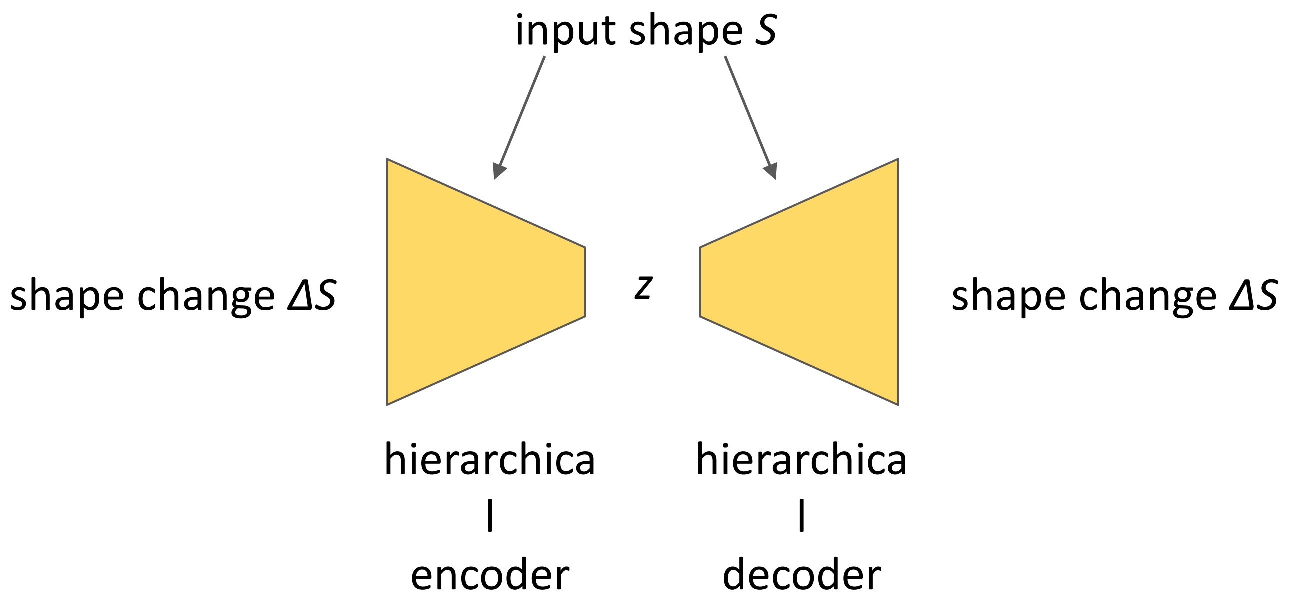
```
0 [DiffNode: SAME]
├0 [DiffNode: SAME]
│  ├0 [DiffNode: SAME]
│  │  ├0 [DiffNode: DEL]
│  │  ├1 [DiffNode: ADD]
│  │  │  ├0 back_surface_vertical_bar [LEAF] {3}
│  │  ├2 [DiffNode: ADD]
│  │  │  ├0 back_surface_vertical_bar [LEAF] {4}
│  │  ├3 [DiffNode: ADD]
│  │  │  ├0 back_surface_vertical_bar [LEAF] {5}
│  │  ├4 [DiffNode: ADD]
│  │  │  ├0 back_surface_vertical_bar [LEAF] {6}
│  │  ├5 [DiffNode: ADD]
│  │  │  ├0 back_surface_vertical_bar [LEAF] {7}
│  ├1 [DiffNode: DEL]
│  ├2 [DiffNode: DEL]
│  ├3 [DiffNode: ADD]
│  │  ├0 back_frame     {8}
│  │  │  ├0 back_frame_vertical_bar [LEAF] {9}
│  │  │  ├1 back_frame_vertical_bar [LEAF] {10}
│  │  │  ├2 back_frame_horizontal_bar [LEAF] {11}
├1 [DiffNode: SAME]
│  ├0 [DiffNode: SAME]
│  │  ├0 [DiffNode: LEAF]
│  ├1 [DiffNode: ADD]
│  │  ├0 seat_frame     {15}
│  │  │  ├0 seat_frame_bar [LEAF] {16}
│  │  │  ├1 seat_frame_bar [LEAF] {17}
│  │  │  ├2 seat_frame_bar [LEAF] {18}
│  │  │  ├3 seat_frame_bar [LEAF] {19}
├2 [DiffNode: SAME]
│  ├0 [DiffNode: SAME]
│  │  ├0 [DiffNode: LEAF]
│  │  ├1 [DiffNode: LEAF]
│  │  ├2 [DiffNode: LEAF]
│  │  ├3 [DiffNode: LEAF]
│  │  ├4 [DiffNode: DEL]
│  │  ├5 [DiffNode: DEL]
```

Network Architecture

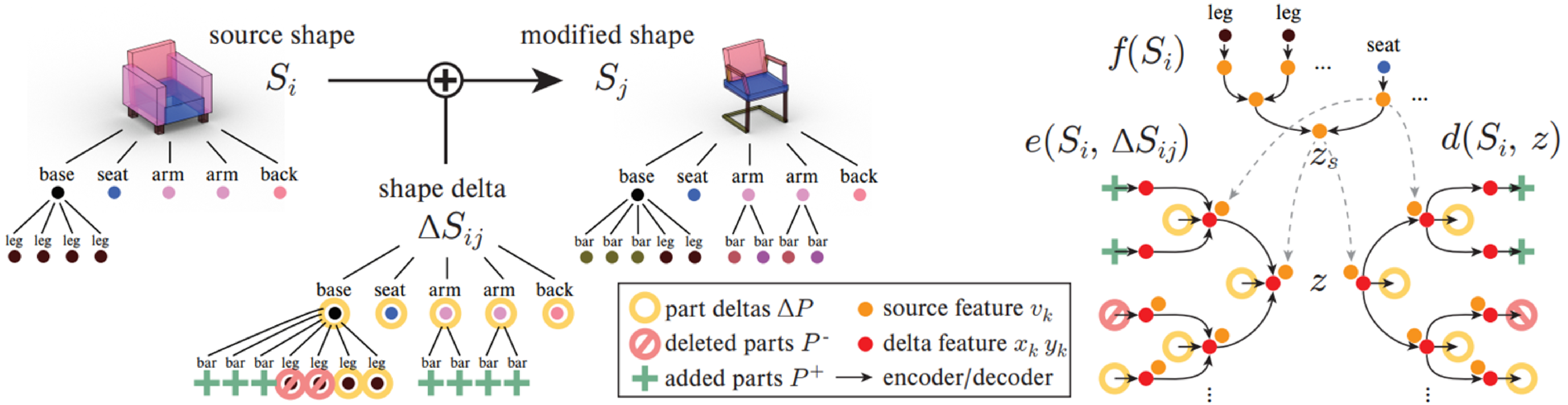
StructureNet (VAE)



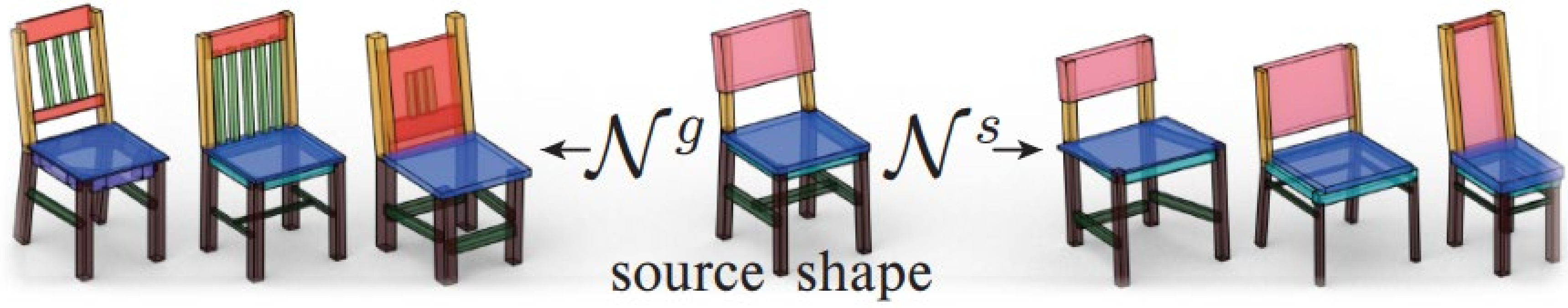
StructEdit (cVAE)



Network Architecture

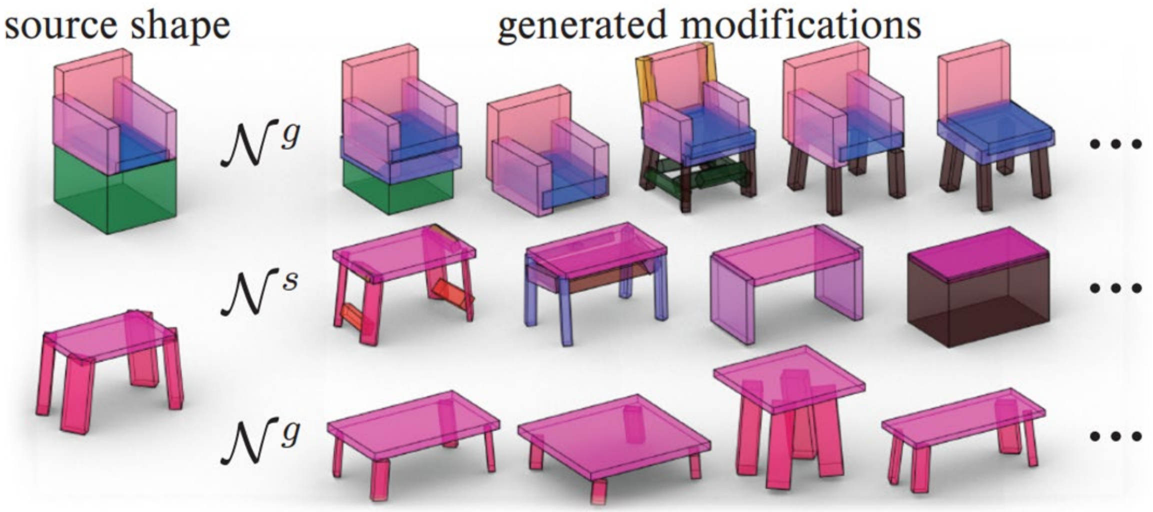


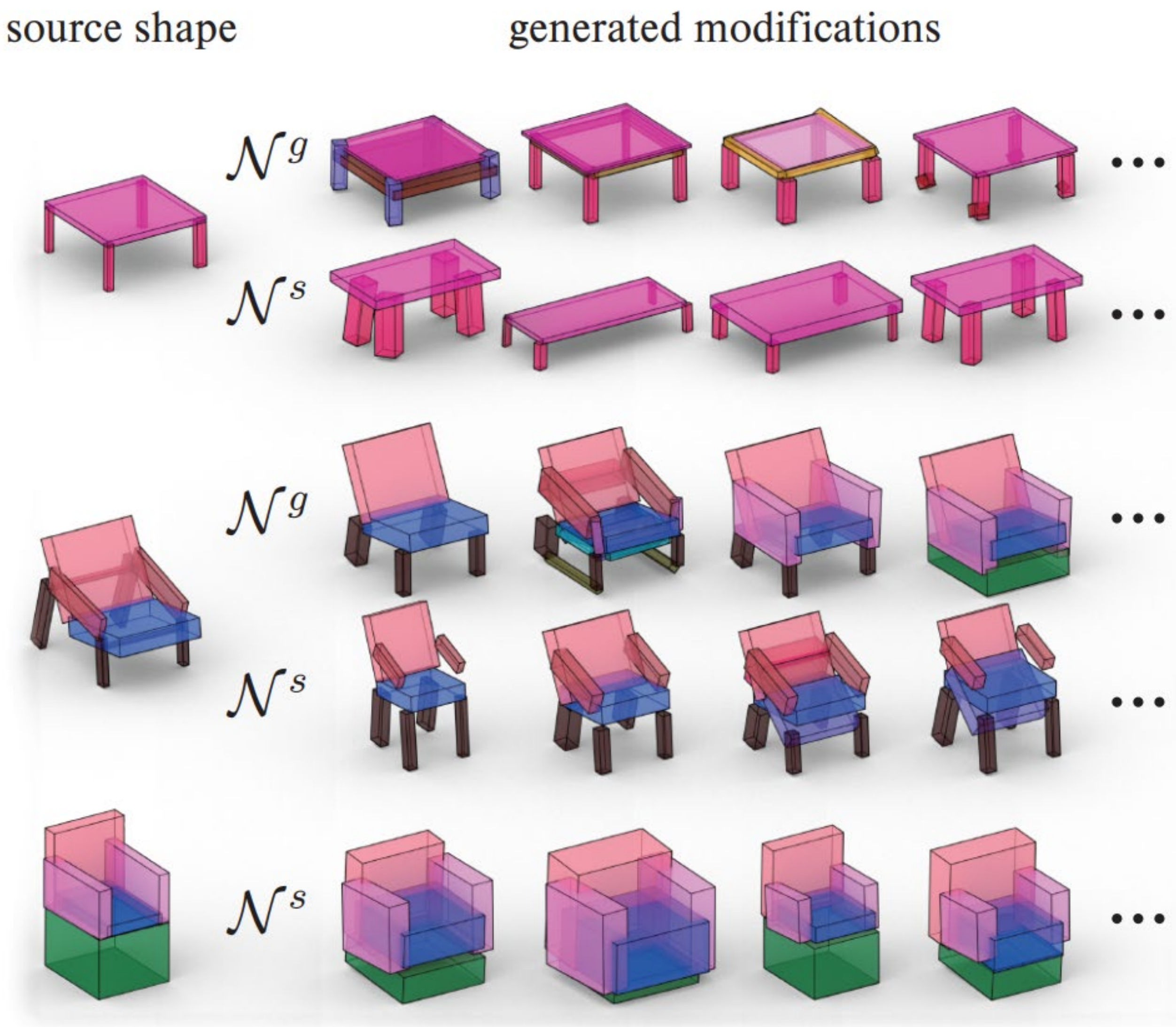
Two Types of Shape Neighborhood



Experiment: Edit Generation

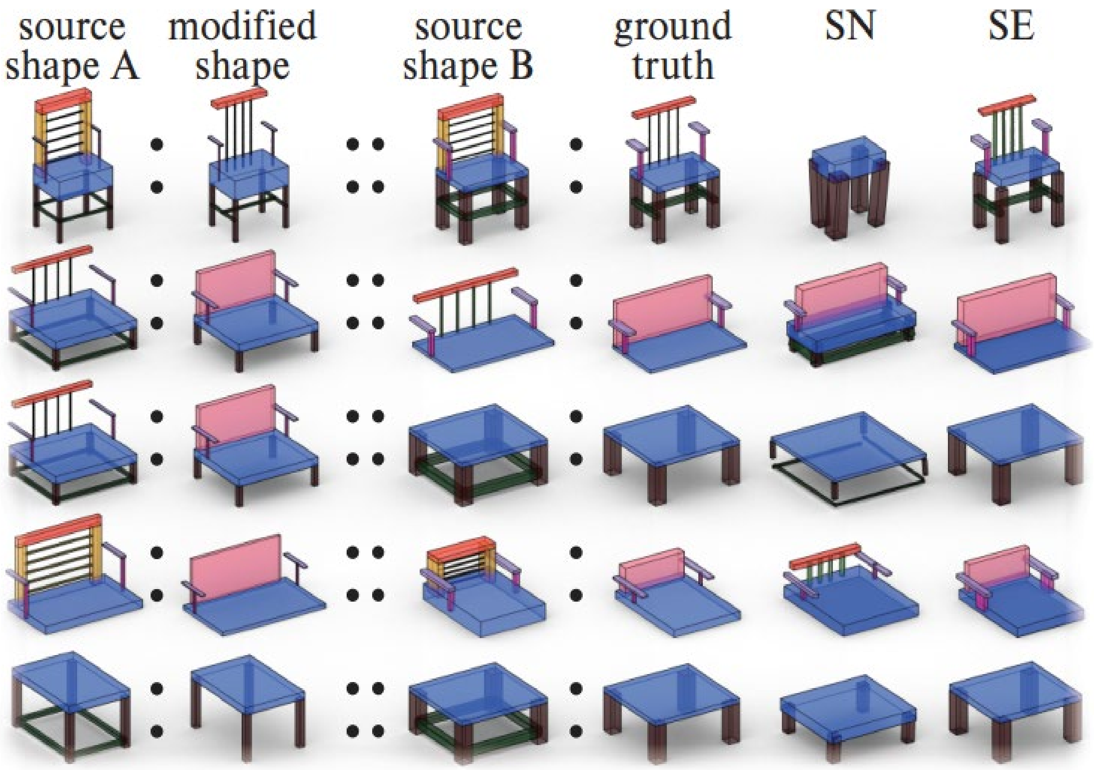
		$\mathcal{N}^g$				$\mathcal{N}^s$			
		chair	table	furn.	avg.	chair	table	furn.	avg.
$E_{\text{qc}}^{\text{geo}}$	ID	1.822	1.763	1.684	1.756	1.629	1.479	1.446	1.518
	$\text{SN}_{0.2}$	1.760	2.076	1.626	1.821	1.308	1.208	1.243	1.253
	$\text{SN}_{0.5}$	1.722	2.068	1.558	1.783	1.241	1.103	1.135	1.160
	$\text{SN}_{1.0}$	1.768	2.189	**1.554**	1.837	1.232	1.057	1.017	1.102
	SE	**1.593**	**1.655**	1.561	**1.603**	**1.218**	**1.000**	**1.015**	**1.078**
$E_{\text{qc}}^{\text{st}}$	ID	1.281	1.215	1.288	1.261	1.437	1.303	1.442	1.394
	$\text{SN}_{0.2}$	1.081	0.878	1.015	0.991	1.466	3.484	1.414	2.121
	$\text{SN}_{0.5}$	0.871	0.729	0.873	0.824	1.373	3.300	1.204	1.959
	$\text{SN}_{1.0}$	0.751	0.667	**0.726**	0.715	1.763	3.622	1.167	2.184
	SE	**0.559**	**0.524**	0.741	**0.608**	**0.609**	**0.451**	**0.676**	**0.579**

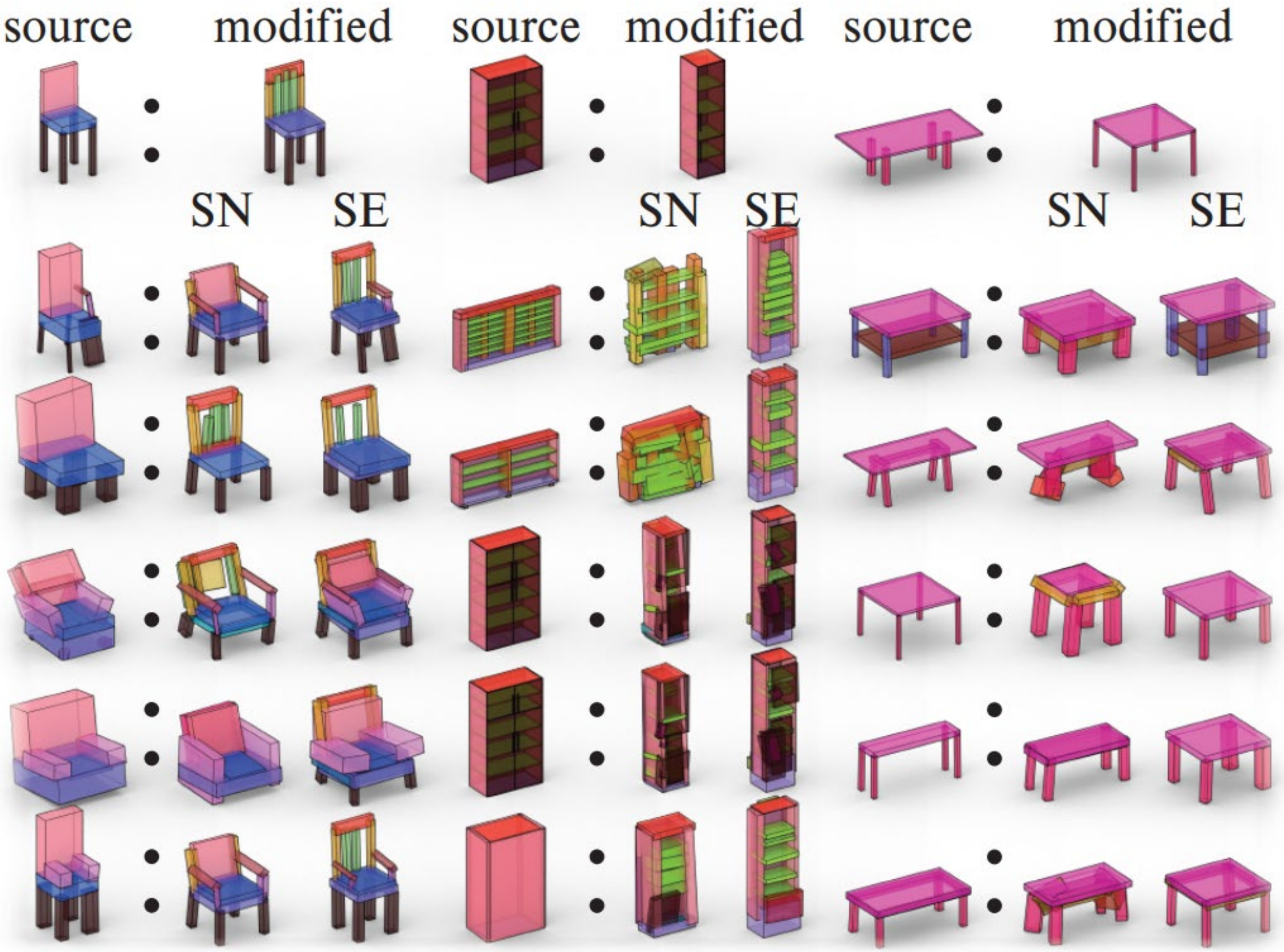




Experiment: Edit Transfer

$\mathcal{N}$		chair	sofa	stool	c. → s.	c. → st.	avg.
E_t^{geo}	Identity	1.002	0.938	0.892	0.892	0.938	0.932
	StructureNet	0.868	0.764	0.721	0.888	1.307	0.910
	StructEdit	**0.586**	**0.566**	**0.599**	**0.572**	**0.698**	**0.604**
E_t^{st}	Identity	0.941	1.328	0.333	0.333	1.328	0.853
	StructureNet	0.208	0.161	0.025	0.671	0.871	0.387
	StructEdit	**0.005**	**0.001**	**0.003**	**0.002**	**0.123**	**0.027**

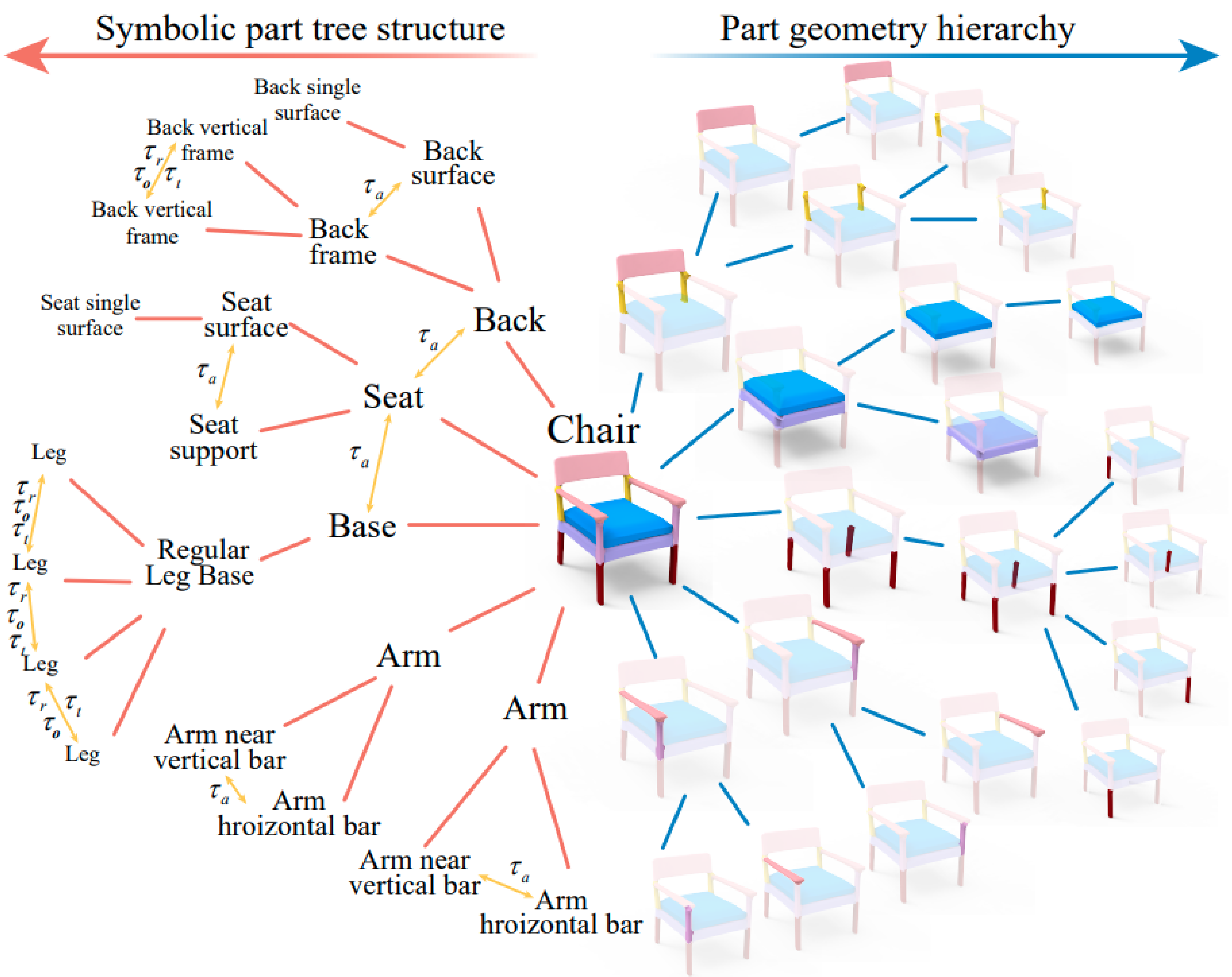




DSG-Net

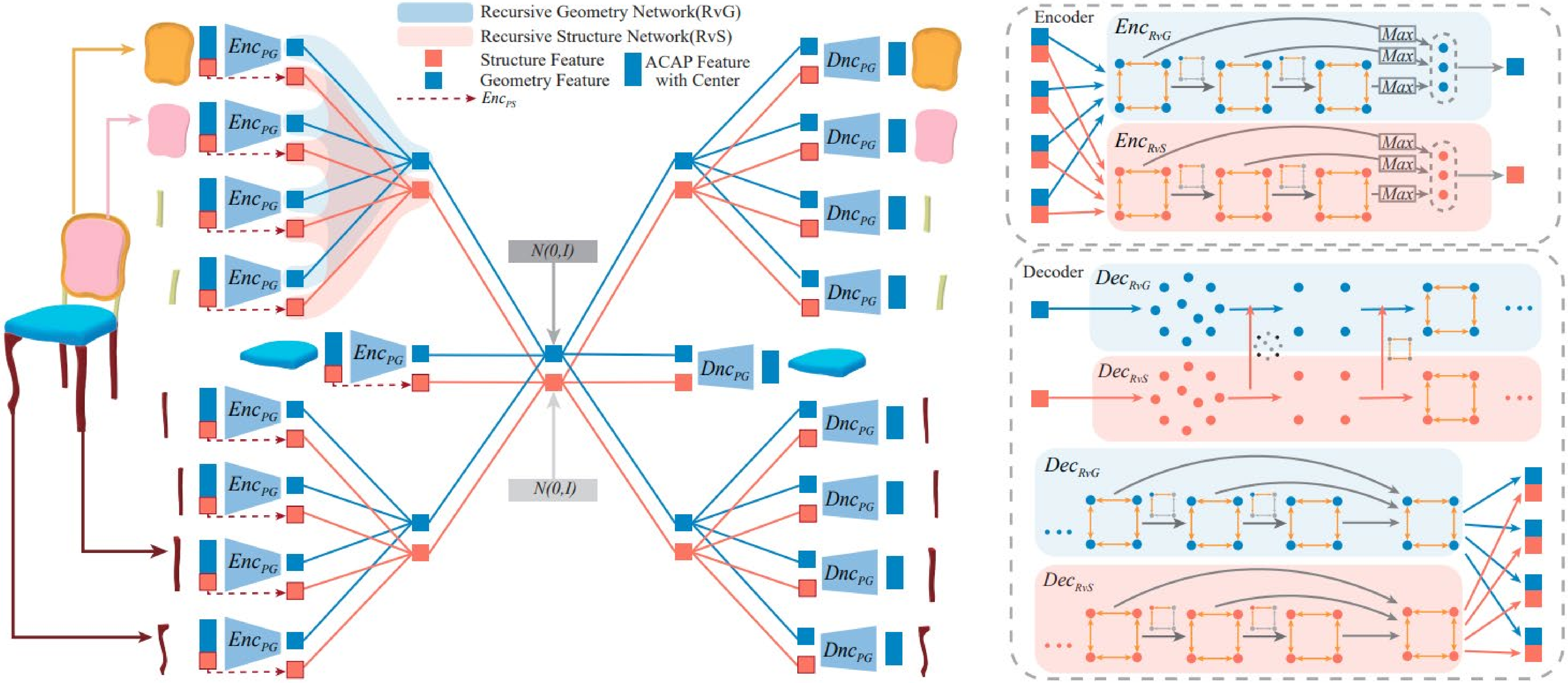
DSG-Net: Learning Disentangled Structure and Geometry for 3D Shape Generation. Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, Lin Gao

DSM-Net: Disentangled Tightly-Coupled Hierarchies



[Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, Lin Gao, 2020]

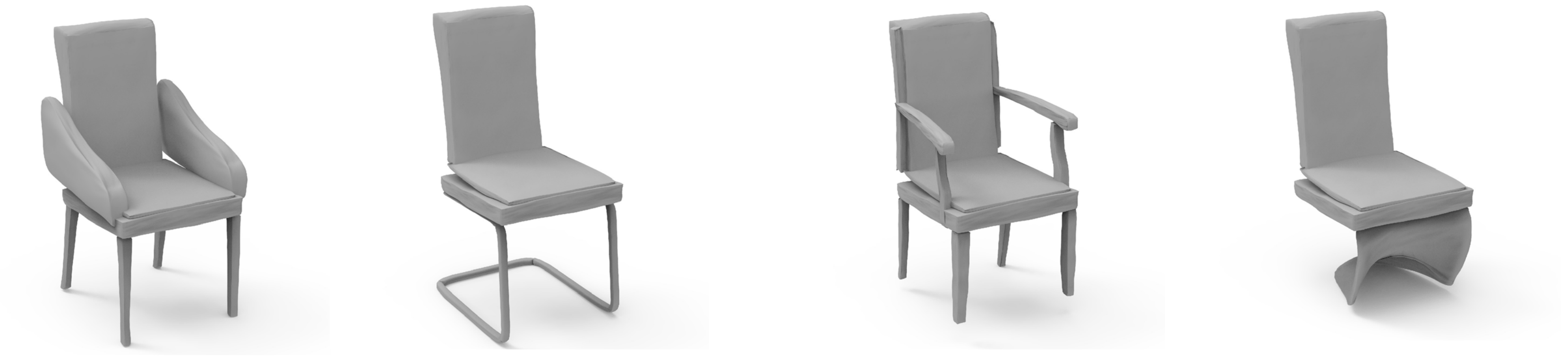
Coupled Network Architecture: Double-VAEs



Shape Generation

Disentangled Shape Generation

Fix Shape Structure



Fix Shape Geometry

Variation Generation

Learning to Vary

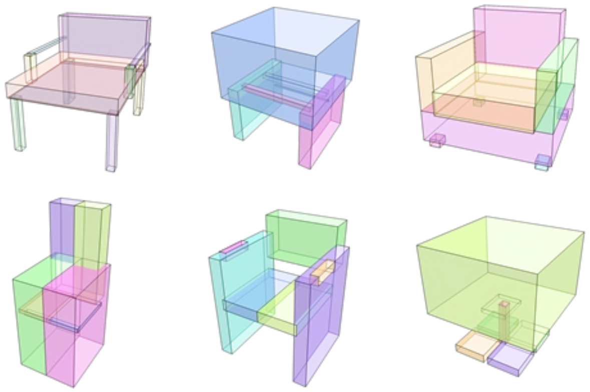
- Re-use what we already have
- Populate sparsely sampled regions

Geometry,
Arrangement,
Appearance,
Motion

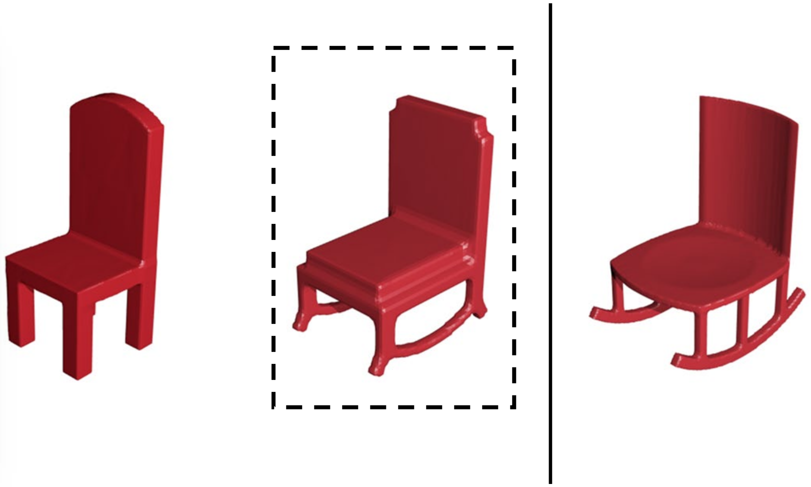
for Objects and
Scenes

Varying to Learn

- Provide generation diversity
- Create training data tailored for hard concepts



RGB image



Motivation

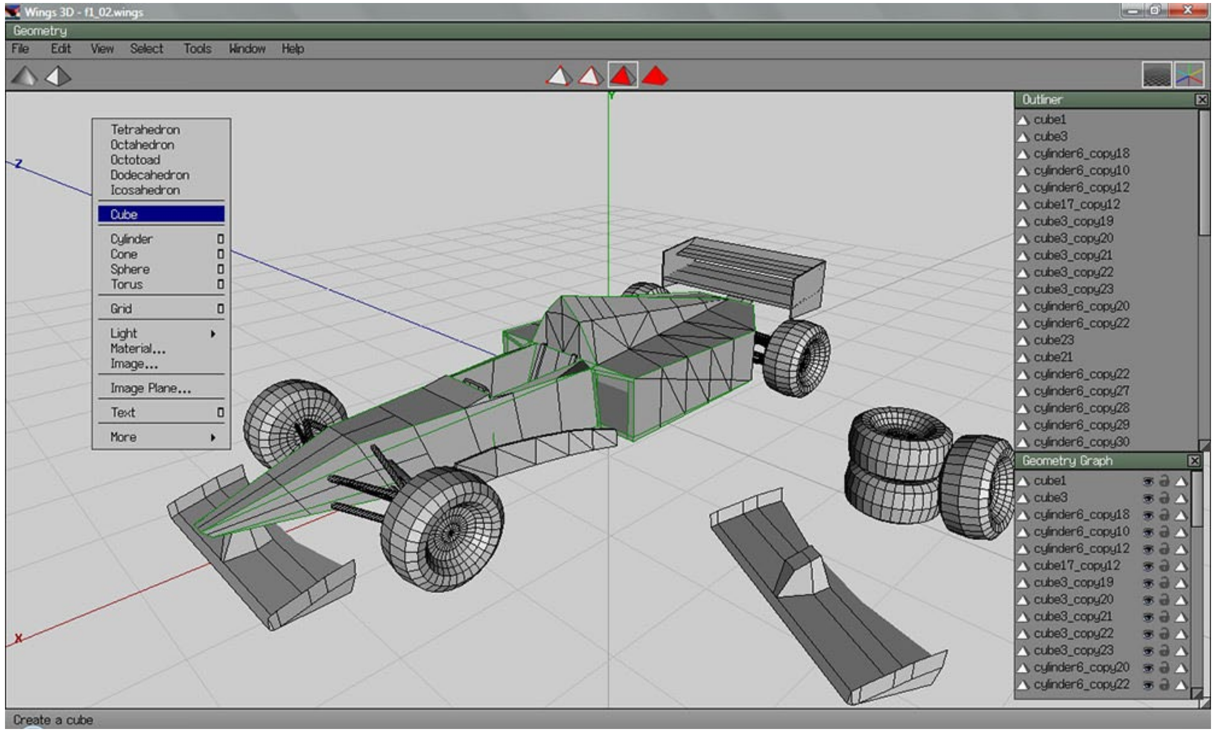


Photo taken from DeefSDF

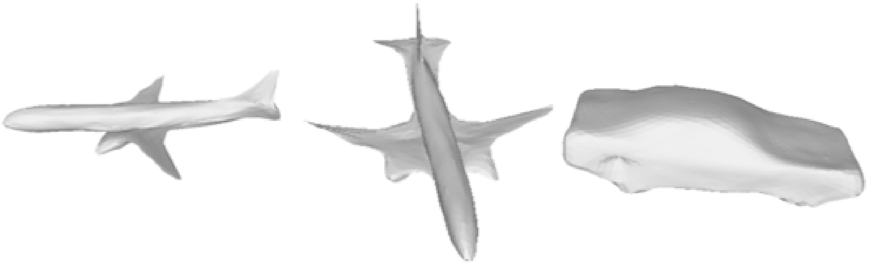


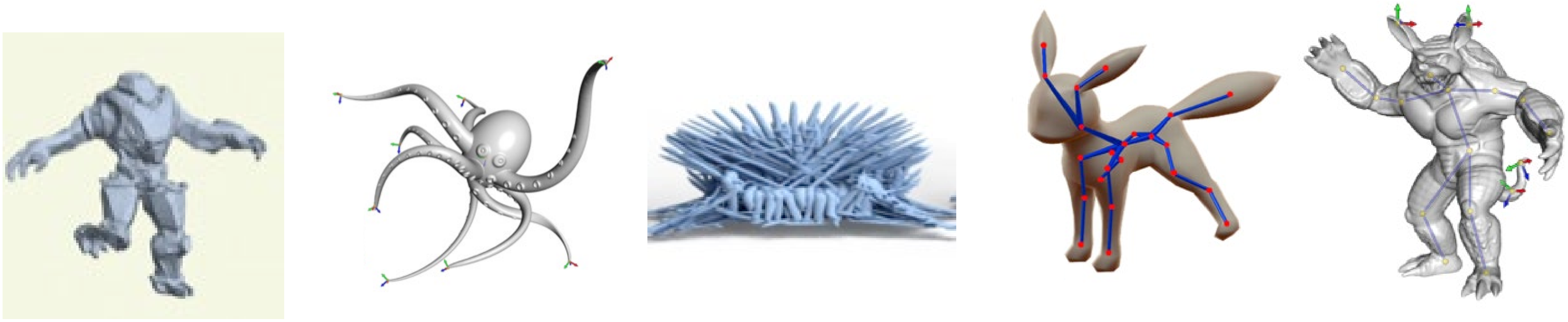
Photo taken from Pixel2Mesh++

[1] DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. Park, et. al., CVPR 2019.
[2] Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation. Wen, et. al., ICCV 2019.

Motivation

- Leverage on existing (**discrete**) artist generated models

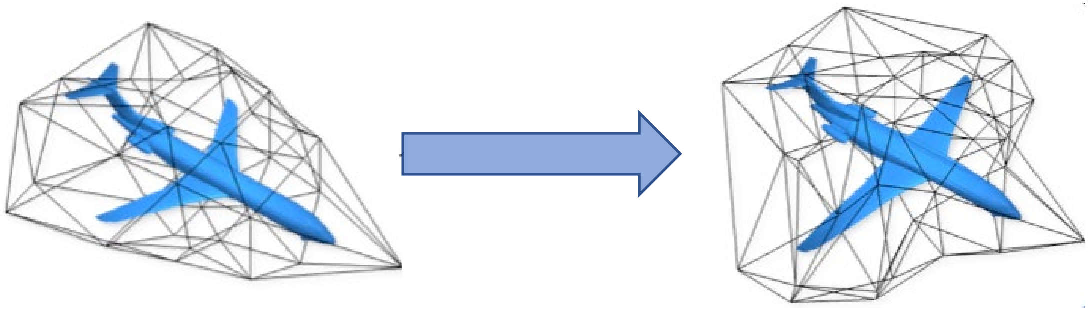
- Create new models through (**continuous**) variations/deformations



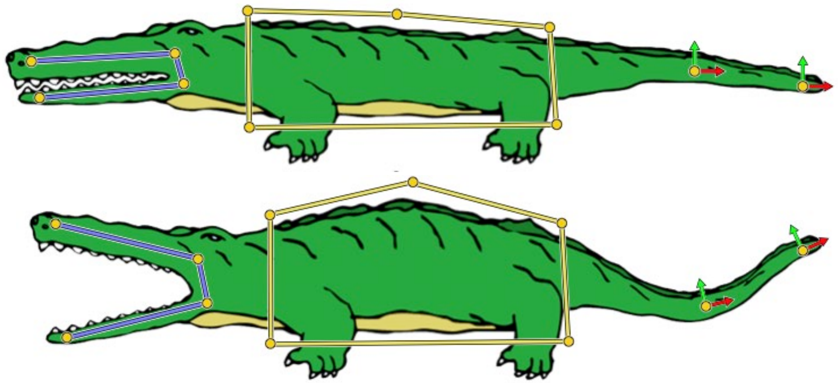
Deformation Models

- Leverage classical geometry techniques to define deformation.
  - Preserves local geometric features
  - In other words, ensures the quality of the output

- Cage-based deformation
  - Neural Cages (CVPR 2020)



- Control point-based / biharmonic coordinates
  - DeepMetaHandles (CVPR 2021)



Neural Cages for Detail-Preserving 3D Deformations

Wang Yifan, Noam Aigerman, Vladimir Kim, Siddhartha Chaudhuri, Olga Sorkine-Hornung

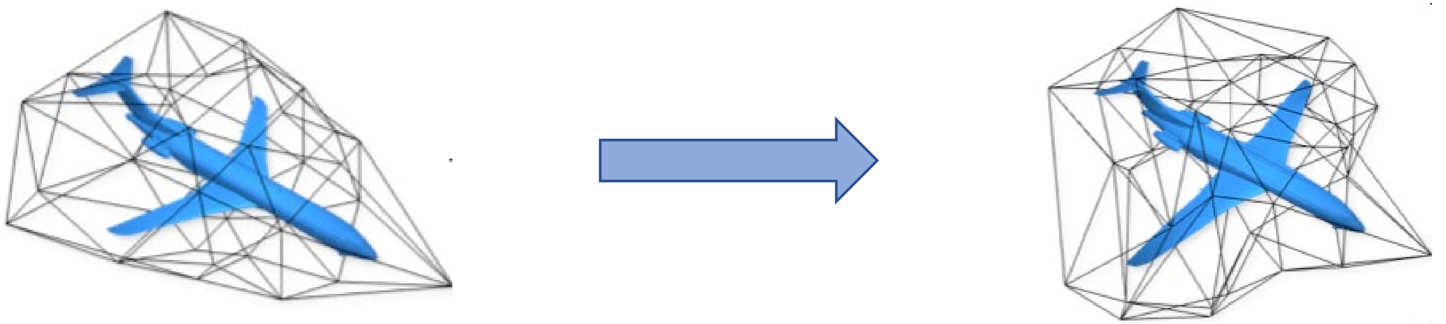
(CVPR 2020)

Key Idea

- Warp the source shape to match the general structure of the target while preserving surface details of the source

- Source shape is enclosed by a coarse control mesh → **cage**

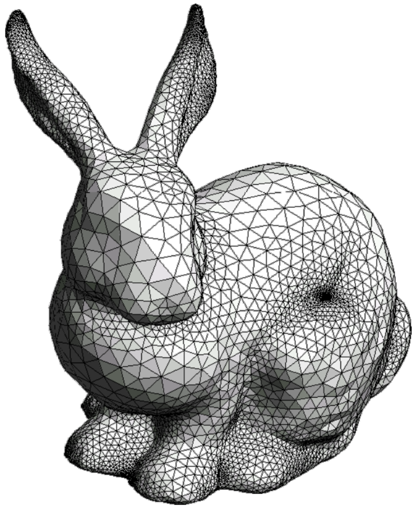
- Neural network learns to optimize both the position of the cage around the source and the deformation of the cage to match the target



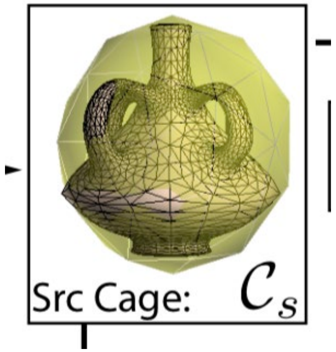
Deformation

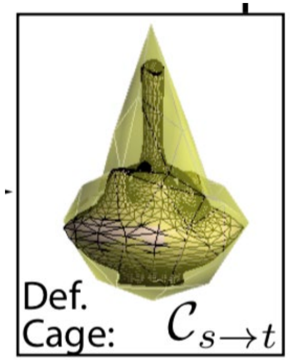
- Competing objectives
  1. Alignment with the target
  2. Quality: minimize distortion, preservation of geometric features

- Cage-based deformation enforces constraints to preserve local geometric features



Cage-based Deformation (CBD)



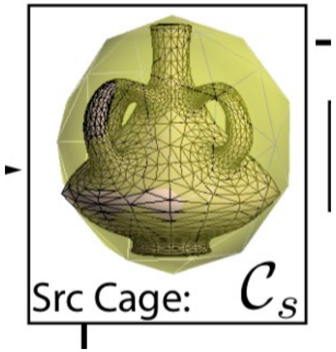


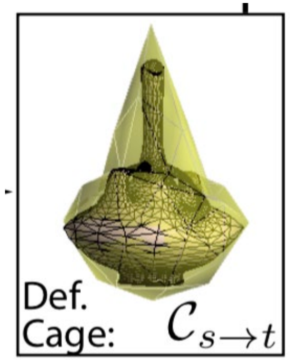
- CBD controls the warping of source S by enclosing it with a coarse triangular mesh C (cage), and warping this cage instead
- Any point $\boldsymbol{p}$ in ambient space is encoded via a generalized barycentric coordinate: weight average of cage vertices:

$$\mathbf{p} = \sum \phi_j^{\mathcal{C}}(\mathbf{p})\,\mathbf{v}_j$$

- Weight functions $\{\phi_j^{\mathcal{C}}\}$ depend on the relative position of p wrt the cage vertices

Cage-based Deformation (CBD)





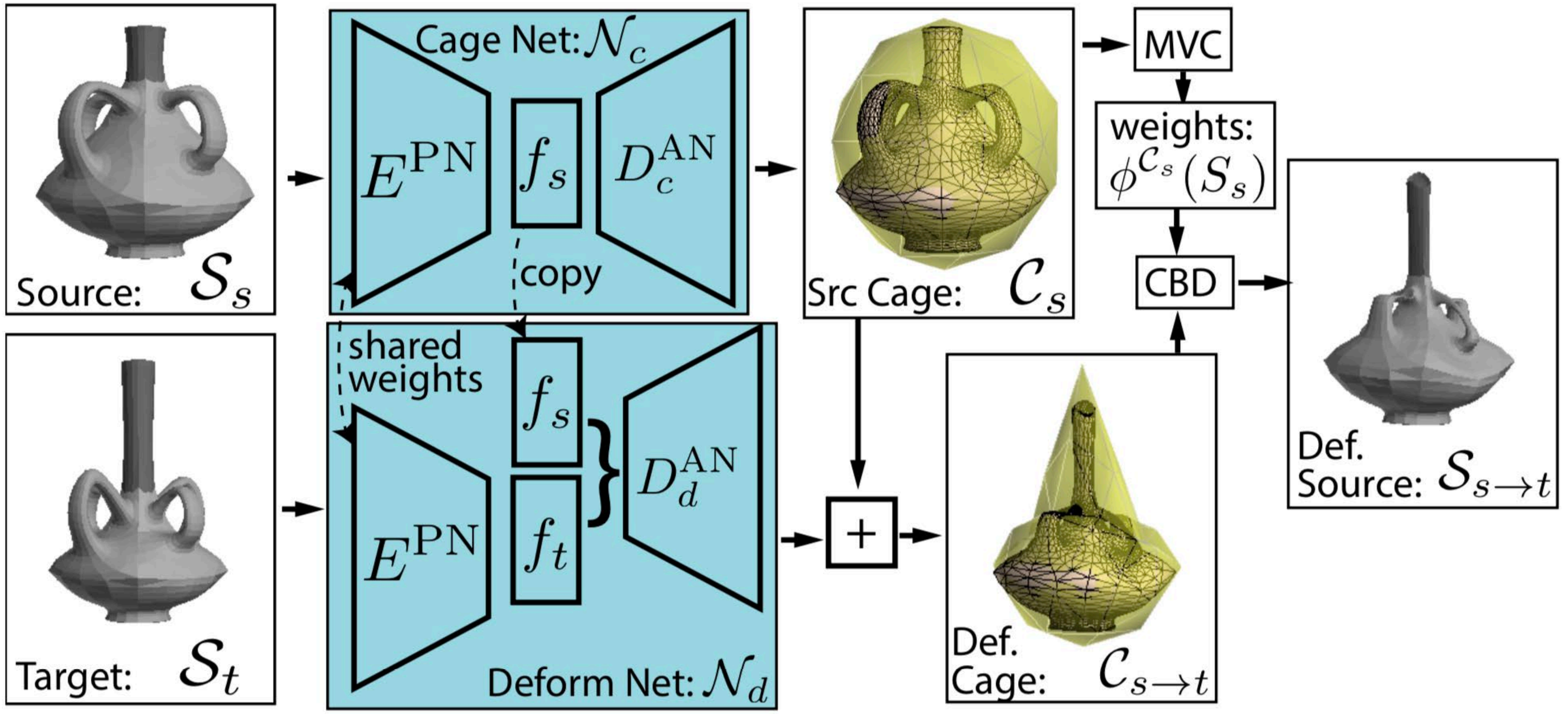
- We then only deform the cage vertices and use the pre-computed weights $\{\phi_j^{\mathcal{C}}\}$
- The deformation of any point in ambient space is then given by

$$\boxed{\mathbf{p}' = \sum_{0\le j<|V_{\mathcal{C}}|} \phi_j^{\mathcal{C}}(\mathbf{p})\,\mathbf{v}_j'}$$

- Use mean value coordinates (MVC) [1] to obtain weight functions $\{\phi_j^{\mathcal{C}}\}$: simple and differentiable

[3] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. TOG 2004

Learning cage-based deformation

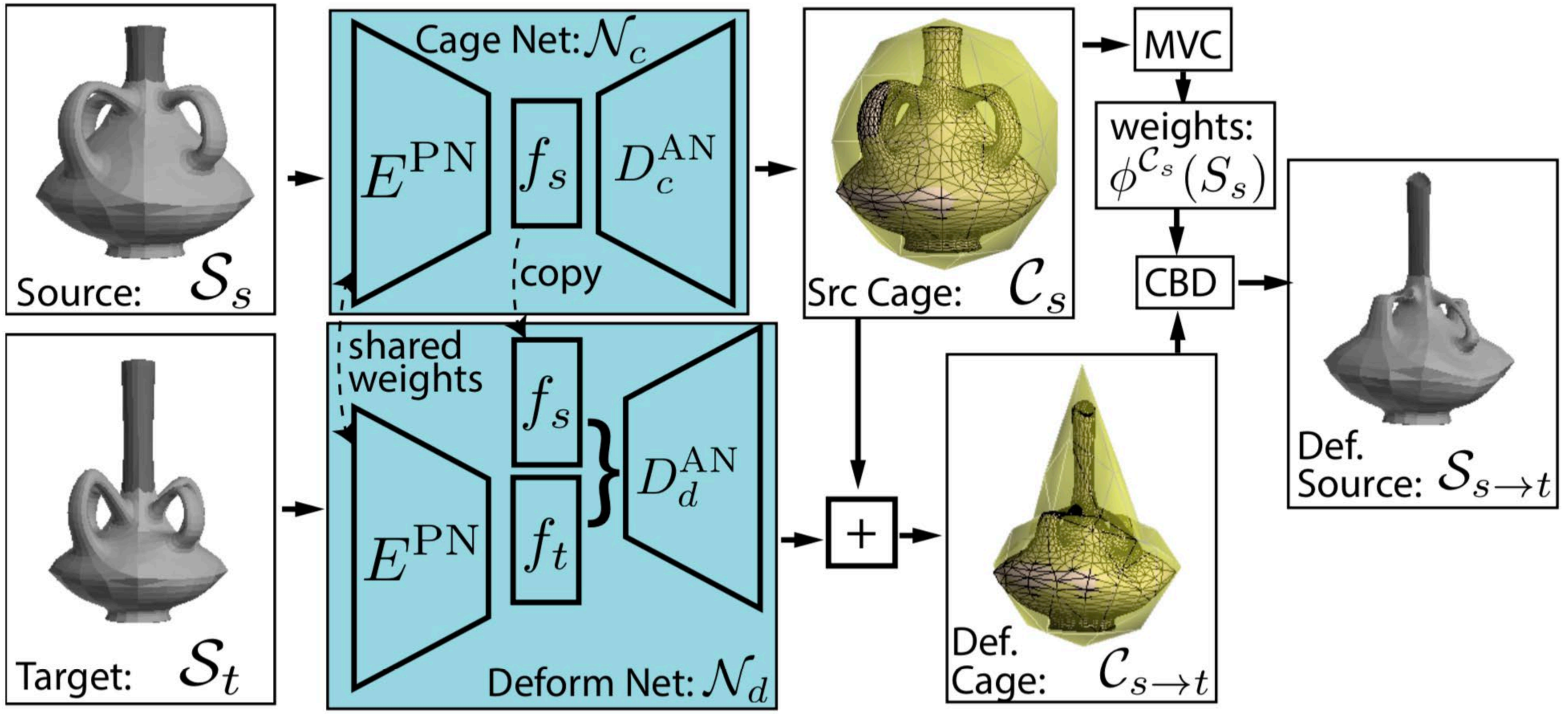


- Train a network to predict both source and target cages

$$\mathcal{C}_s = \mathcal{N}_c\left(\mathcal{S}_s\right) + \mathcal{C}_0, \qquad \mathcal{C}_{s\to t} = \mathcal{N}_d\left(\mathcal{S}_t, \mathcal{S}_s\right) + \mathcal{C}_s$$

- Initial template cage is a 42-vertex sphere C_0

Learning cage-based deformation



- Both branches of encoder and decoder only predict the cage; they don't rely on detailed geometric features of the input
- Network does not require high resolution

Losses

- Cage loss:
  - Encourage positive MVC (weights of the cage)
  - Negative weights occur when the source cage is highly concave, self-overlapping or when points lie outside the cage

- Alignment loss:
  - Chamfer distance / L2 distance or deformed source to target

Losses

- Shape Loss
  - Basically to ensure good quality of the output shape
  - Modified Laplacian:

$$\mathcal{L}_{\text{p2f}} = \frac{1}{|\mathcal{S}_s|} \sum_{i=1}^{|\mathcal{S}_s|} \|d_i - d_i'\|^2$$

  - PCA normals:

$$\mathcal{L}_{\text{normal}} = \frac{1}{|\mathcal{S}_s|} \sum_{i}^{|\mathcal{S}_s|} (1 - \mathbf{n}_i^T \mathbf{n}_i')$$

  - Symmetry loss
  - Total Shape loss: $\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}} + \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{symm}}$

- Total loss: $\boxed{\mathcal{L} = \alpha_{\text{MVC}} \mathcal{L}_{\text{MVC}} + \mathcal{L}_{\text{align}} + \alpha_{\text{shape}} \mathcal{L}_{\text{shape}}}$

Experiments

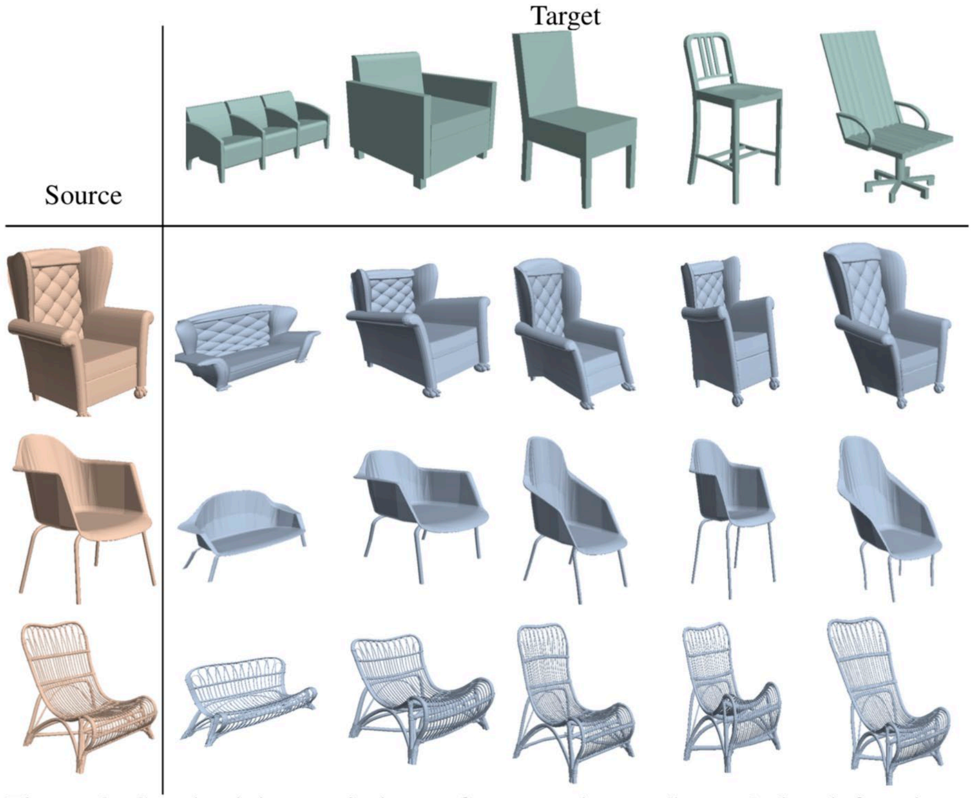


Figure 3: Synthesizing variations of source shapes (brown), by deforming them to match targets (green).

Figure 4: Comparison of our method with other non-homogeneous deformation methods. Our method achieves superior detail preservation of the source shape in comparison to optimization-based [10] and learning-based [6, 9, 28] techniques, while still aligning aligning output to the target.

Experiments

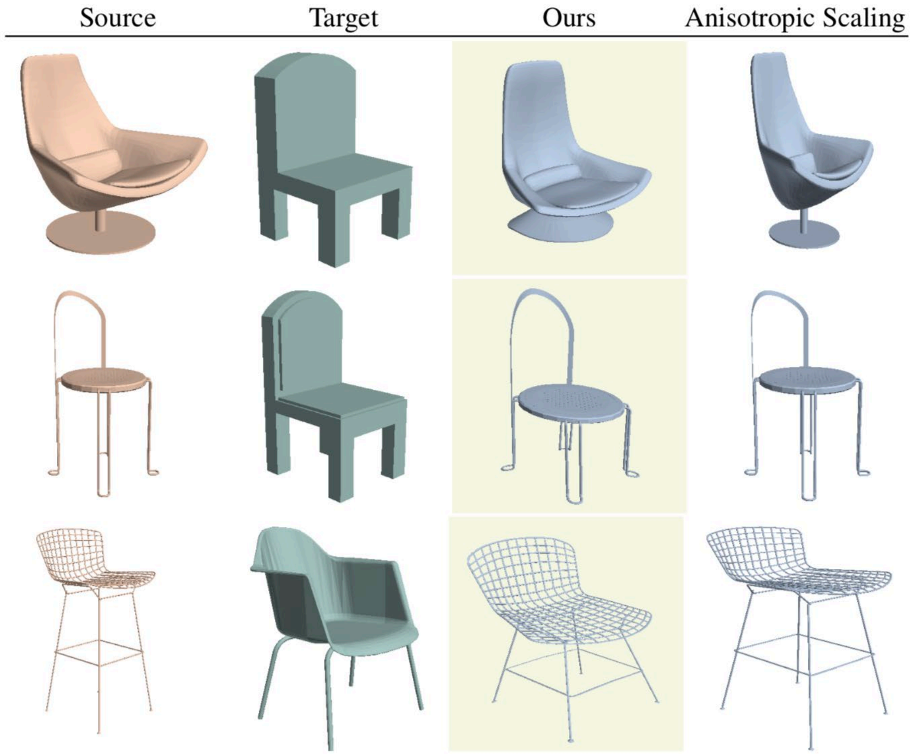


Figure 5: Comparison of our method with anisotropic scaling. Our method better matches corresponding semantic parts.

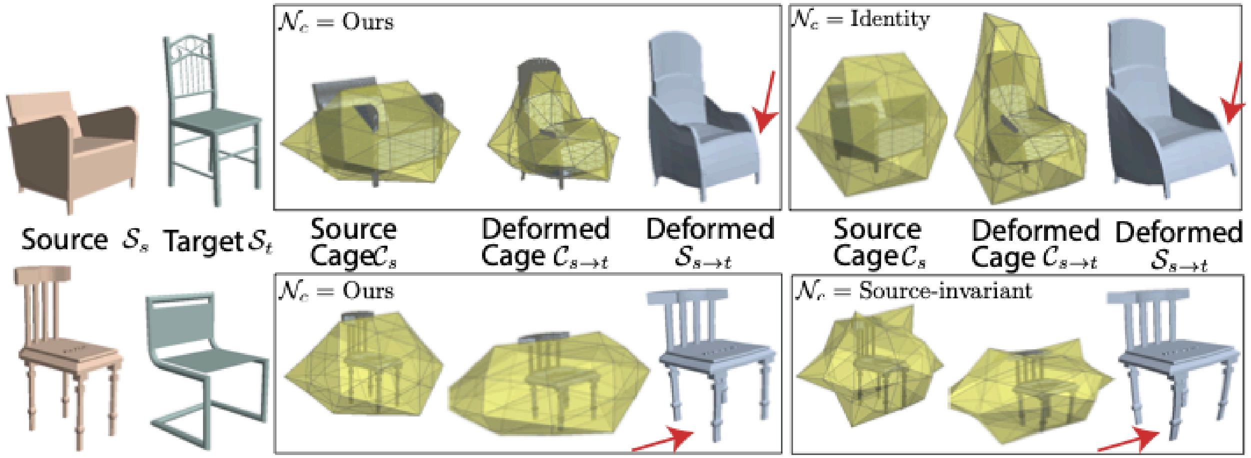


Figure 14: The effect of source-cage prediction. We compare our per-instance prediction of $\mathcal{N}_c$ with (1) a static spherical cage (top right) and (2) a single optimized cage prediction over the entire training set (bottom right). Our approach achieves better alignment with the target shape.

DeepMetaHandles: Learning Deformation Meta-Handles of 3D Meshes with Biharmonic Coordinates

Minghua Liu, Minhyuk Sung, Radomir Mech, Hao Su

(CVPR 2021)

Key Idea

- Conditional generative model based on mesh deformation
- Able to generate variations of a shape without a specific target shape (as in Neural Cages)

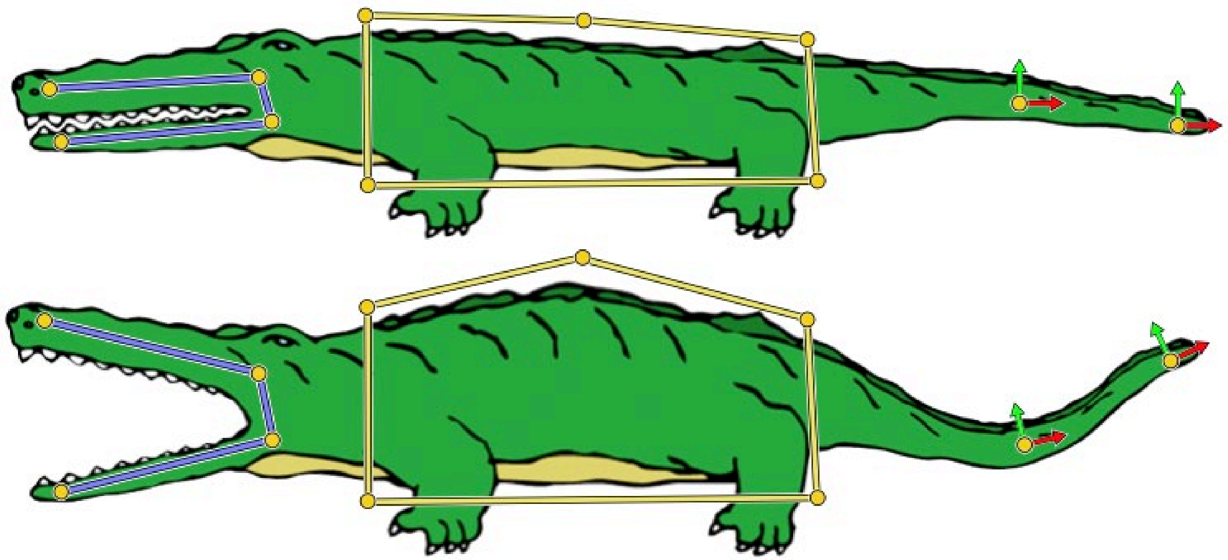
- Deformations represented as a combination of given handles

Control points / handles

- Bounded biharmonic weights

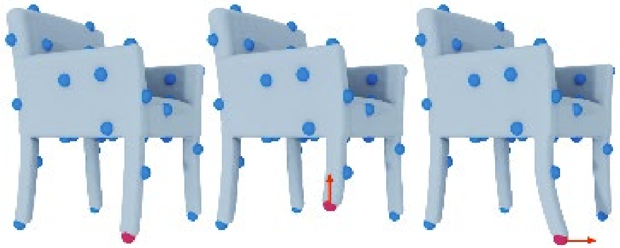
$$\boxed{\mathbf{p}' = \sum_{j=1}^{m} w_j(\mathbf{p})\, T_j \mathbf{p},}$$

$$\begin{aligned}
&\underset{w_j,\ j=1,\ldots,m}{\arg\min} \sum_{j=1}^{m} \frac{1}{2} \int_{\Omega} \|\Delta w_j\|^2 dV && && (2)\\
&\text{subject to:}\ \ w_j|_{H_k} = \delta_{jk} && && (3)\\
&\qquad w_j|_F \ \text{ is linear} && \forall F \in \mathcal{F}_{\mathcal{C}} && (4)\\
&\qquad \sum_{j=1}^{m} w_j(\mathbf{p}) = 1 && \forall \mathbf{p} \in \Omega && (5)\\
&\qquad 0 \le w_j(\mathbf{p}) \le 1,\ \ j = 1,\ldots,m, && \forall \mathbf{p} \in \Omega, && (6)
\end{aligned}$$



[4] Alec Jacobson, et al. Bounded biharmonic weights for real-time deformation. TOG 2011

Metahandles



- Given:
  - Mesh vertices: $\mathbf{V} \in \mathbb{R}^{n \times 3}$
  - Control points: $\mathbf{C} \in \mathbb{R}^{c \times 3}$
  - Linear map: $\mathbf{W} \in \mathbb{R}^{n \times c}$, pre-computed "biharmonic coordinates"

$$\mathbf{V} = \mathbf{W}\mathbf{C}$$

  - Naive deformation: $f : \mathbb{R}^{c \times 3} \rightarrow \mathbb{R}^{n \times 3}$, $\boxed{f(\mathbf{C}) = \mathbf{W}\mathbf{C}}$ has 3c DoF!

- A metahandle is represented as a set of offsets over the *c* control points

$$\boxed{\mathbf{M}_i = [\vec{t}_{i1}, \cdots, \vec{t}_{ic}]^T}$$

- Deformation function is defined as a linear combination of metahandles

$$\boxed{g\left(\mathbf{a}; \{\mathbf{M}_i\}_{i=1\cdots m}\right) = \mathbf{W}(\mathbf{C}_0 + \sum_{i=1}^{m} a_i \mathbf{M}_i),}$$

Network Architecture

- Predict set of metahandles and ranges

c : num control points
m : num metahandles

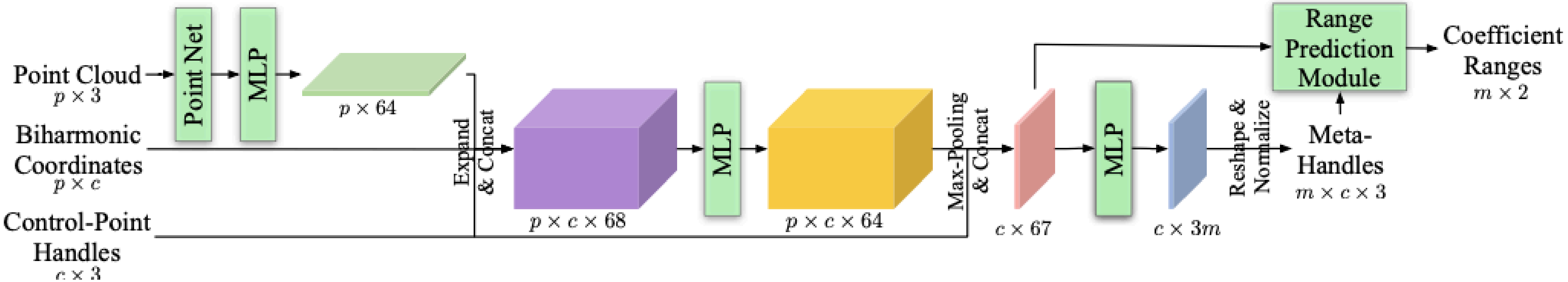


Figure 3: Architecture of MetaHandleNet: it incorporates the information from the shape (point cloud), control-point handles, and biharmonic coordinates by building a 3D tensor, and predicts a set of meta-handles with the corresponding coefficient ranges for the shape.

Network Architecture

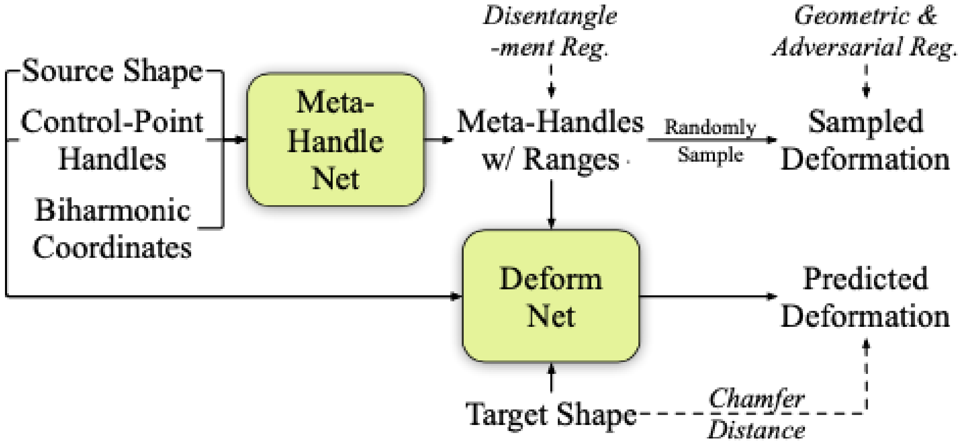


Figure 4: Overview of our method. We learn the meta-handles in an unsupervised fashion.

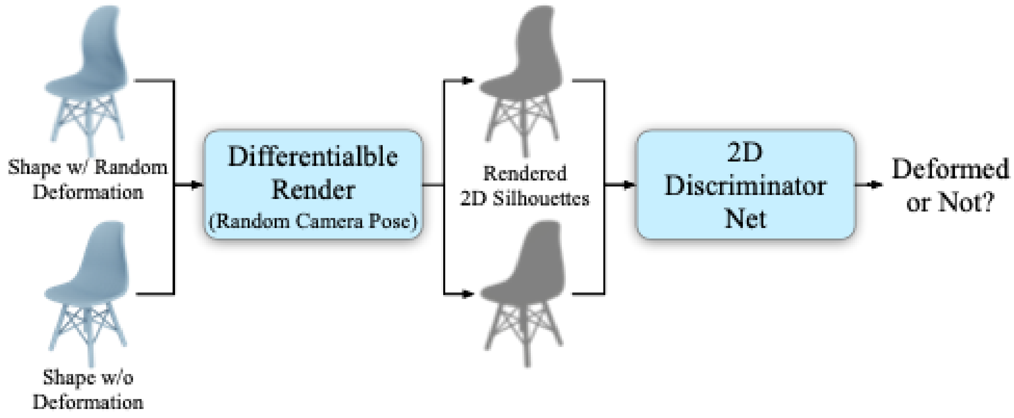


Figure 5: We utilize a soft rasterizer [24] and a 2D discriminator network to penalize unrealistic deformations.

$$\boxed{\mathcal{L} = \mathcal{L}_{fit} + \mathcal{L}_{geo} + \mathcal{L}_{adv} + \mathcal{L}_{disen}.}$$

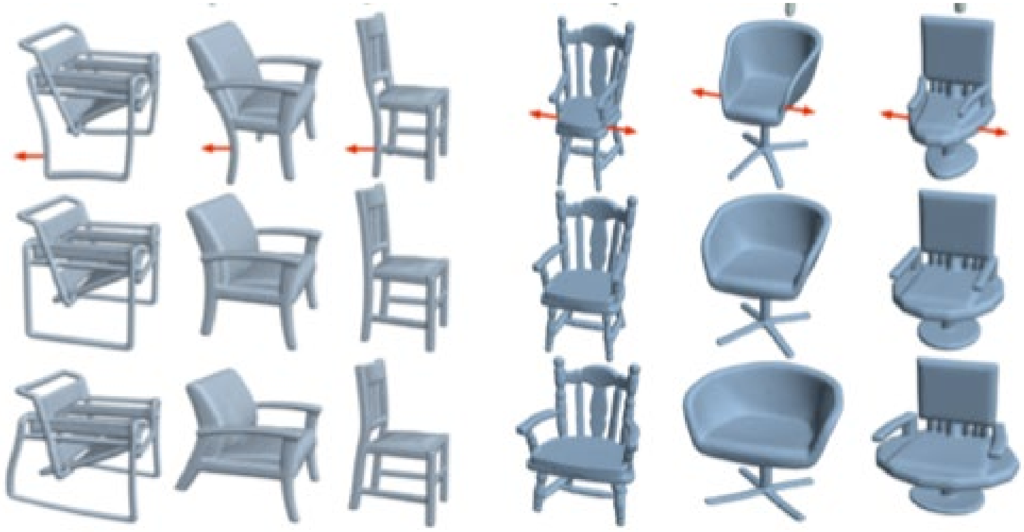
$$\mathcal{L}_{geo} = \mathcal{L}_{symm} + \mathcal{L}_{nor} + \mathcal{L}_{Lap},$$

$$\mathcal{L}_{disen} = \mathcal{L}_{sp} + \mathcal{L}_{cov} + \mathcal{L}_{ortho} + \mathcal{L}_{SVD}.$$

Experiments

Figure 6: Qualitative comparison of our method with other deformation methods [13, 39, 10, 46]. Our method allows flexible deformation and fine-grained detail preservation. Our results are also more plausible, especially when the source-target pairs do not share the same structures (see the second and the fourth columns). Please zoom in for details.

Experiments



Deformation-Aware Retrieval

M.Uy, J. Huang, M. Sung, T. Birdal, L. Guibas (ECCV 20)
M. Uy, V. Kim, M. Sung, N. Aigerman, S. Chaudhuri, L. Guibas (CVPR 21)

Problem

Not all shapes are ”deformable” to each other!

Key Idea

Query Model

Retrieve

Closest Model

Ours Retrieved

Deform

Ours Deformed

Chamfer Distance:

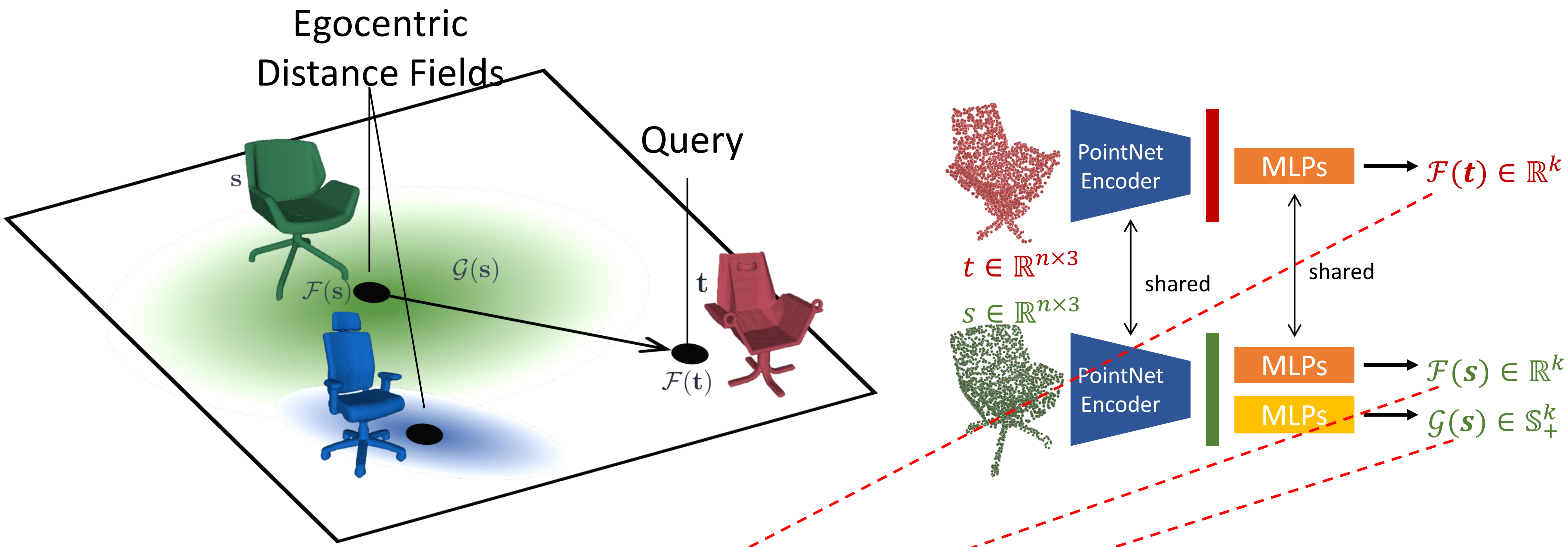
7.09×10^{-2} ↑

4.45×10^{-2}

Chamfer Distance:

1.71×10^{-2} ↓

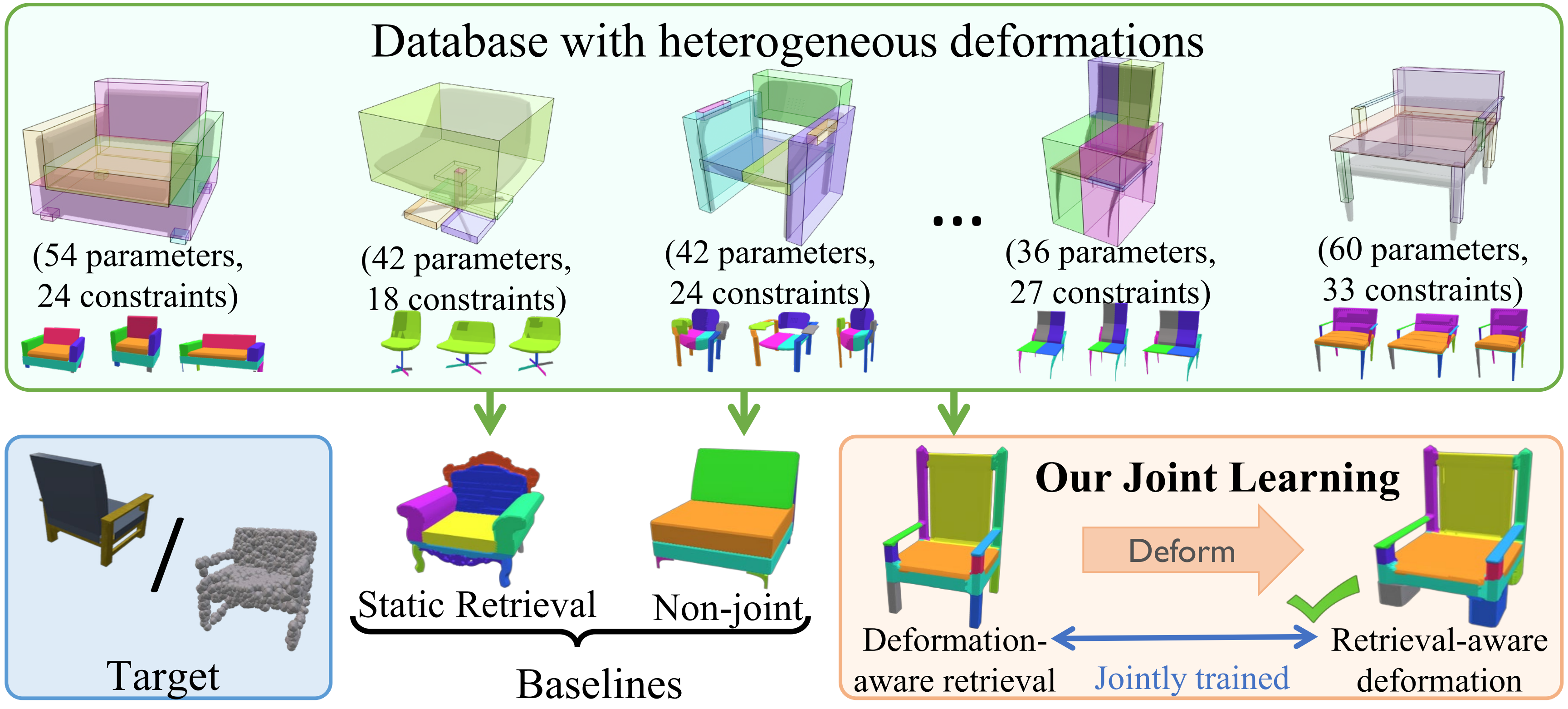
Deformation-Aware Embedding



$$e_{\mathcal{D}}(\mathbf{s},\mathbf{t}) \sim \sqrt{(\mathcal{F}(\mathbf{t}) - \mathcal{F}(\mathbf{s}))^T \mathcal{G}(\mathbf{s})(\mathcal{F}(\mathbf{t}) - \mathcal{F}(\mathbf{s}))}$$

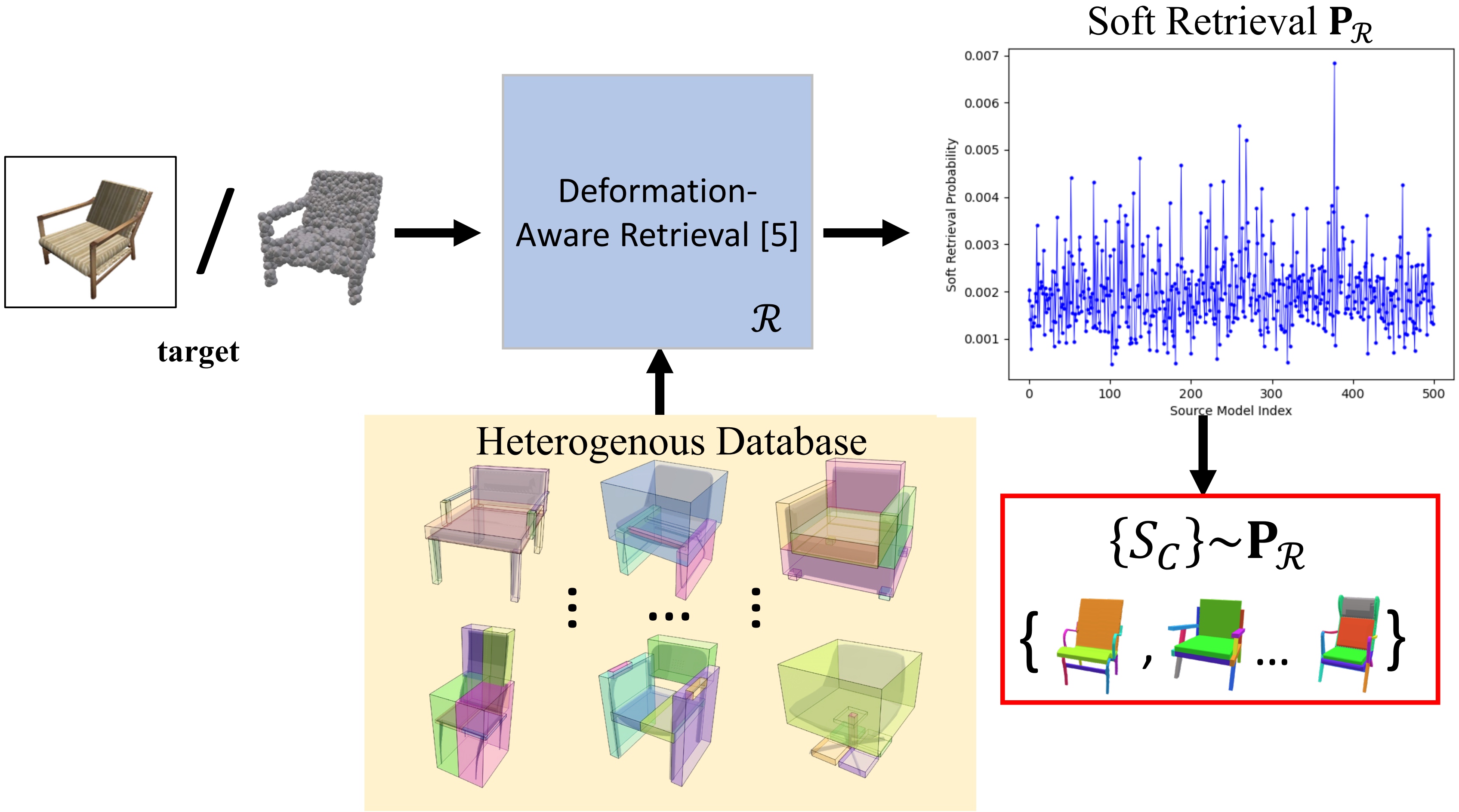
[M. Uy, J. Huang, M. Sung, T. Birdal, L. Guibas, ECCV 2020]

Joint Learning – Retrieval-Aware Deformation



[M. Uy, V. Kim, M. Sung, N. Aigerman, S. Chaudhuri L. Guibas, CVPR 2021]

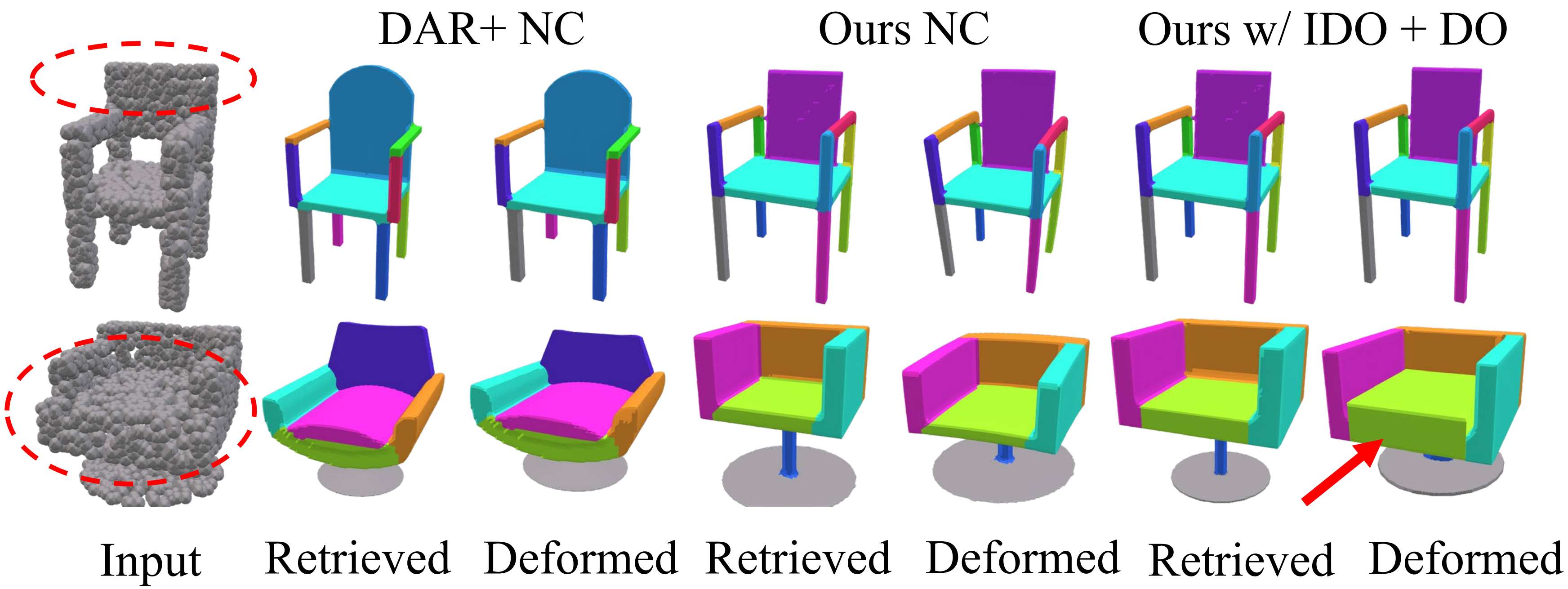
Joint Approach – Soft Retrieval



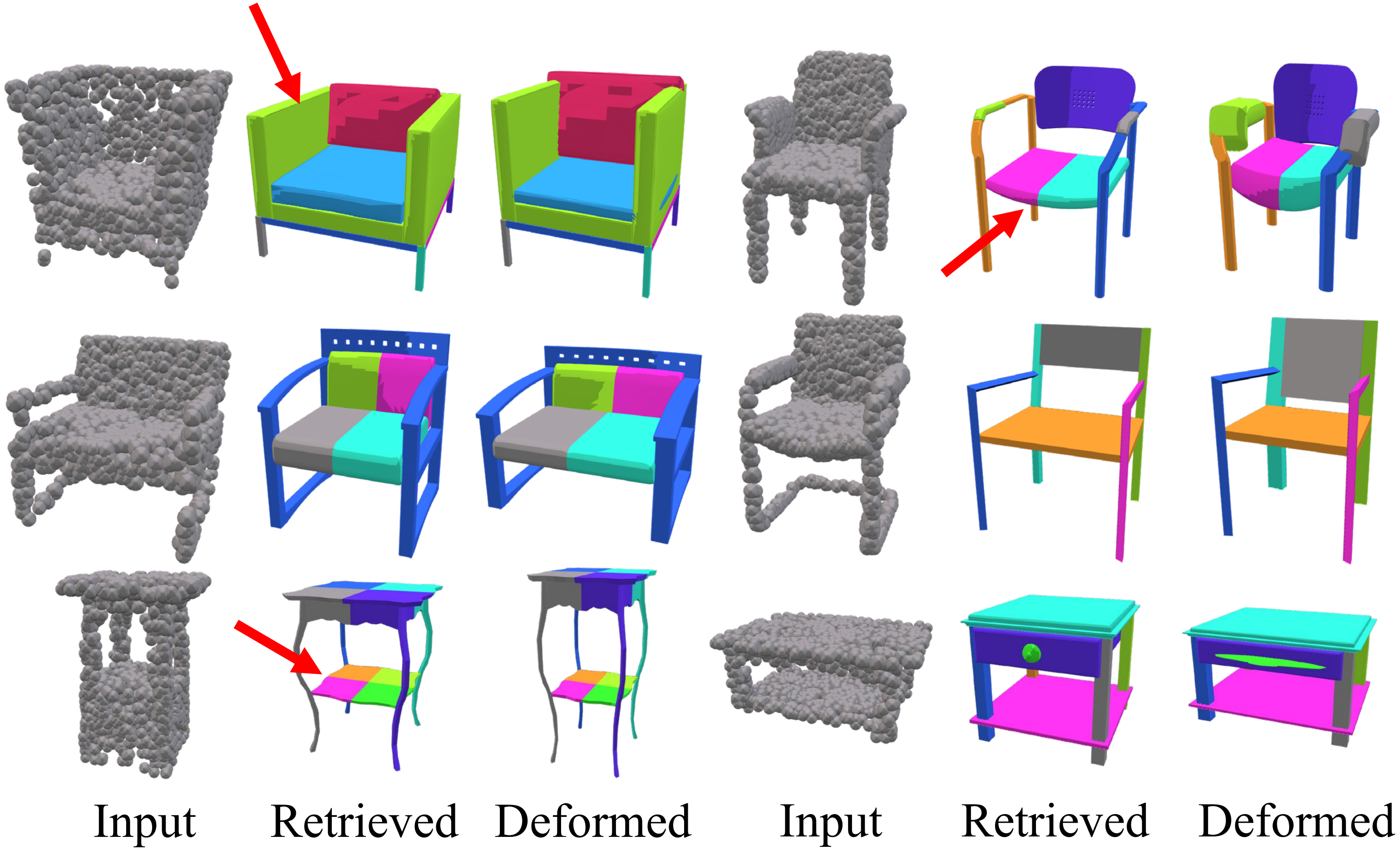
$$\mathcal{P}_{\mathcal{R}}(\mathbf{s},\mathbf{t}) = \frac{\exp(-\frac{\mathrm{d}_{\mathcal{R}}(\mathbf{s},\mathbf{t})}{\sigma_0})}{\sum_{\mathbf{s}'}\exp(\mathrm{d}_{\mathcal{R}}(\mathbf{s}',\mathbf{t})/\sigma_0)}$$

Results

- Our joint approach on Neural Cages:

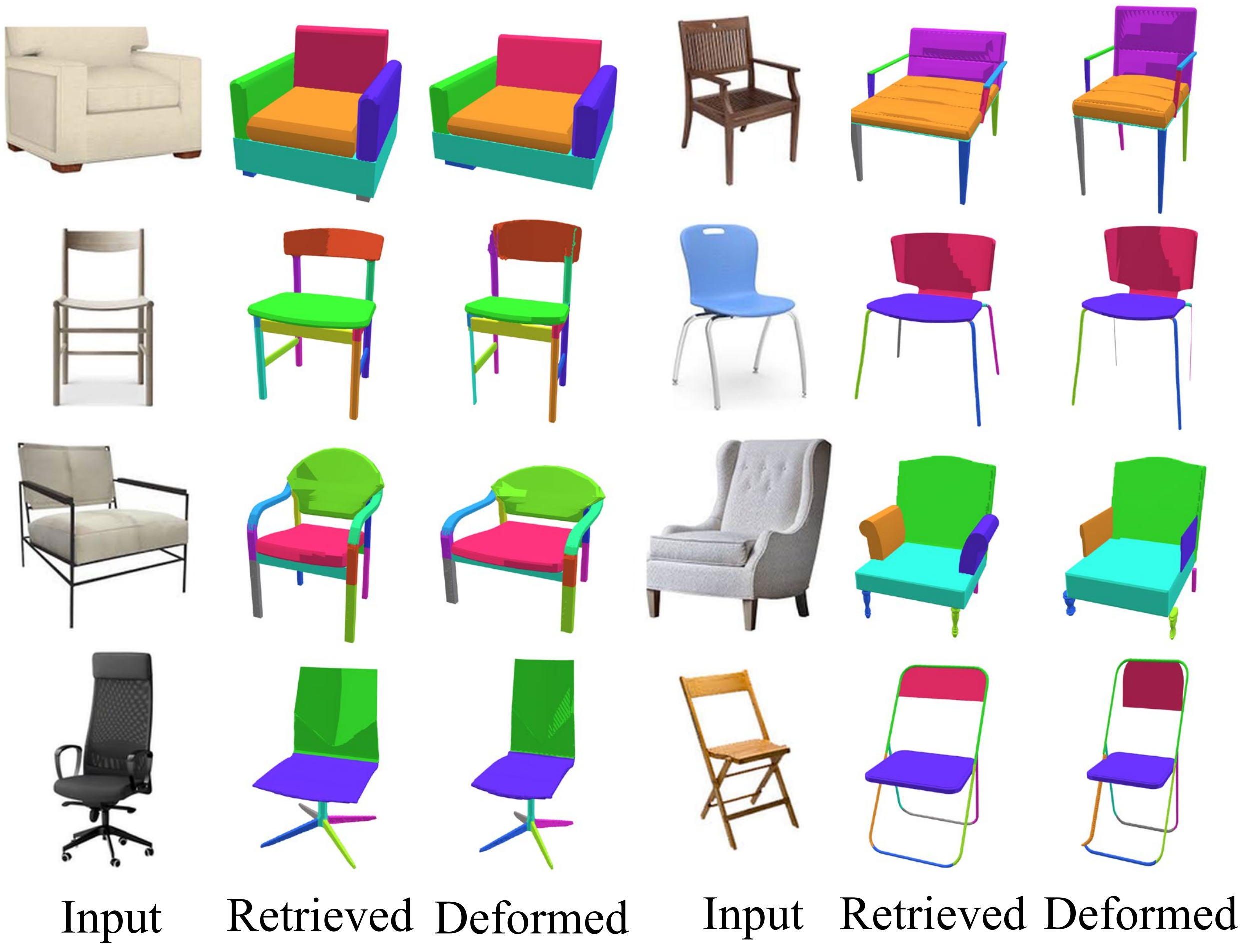


Results – autosegmented dataset

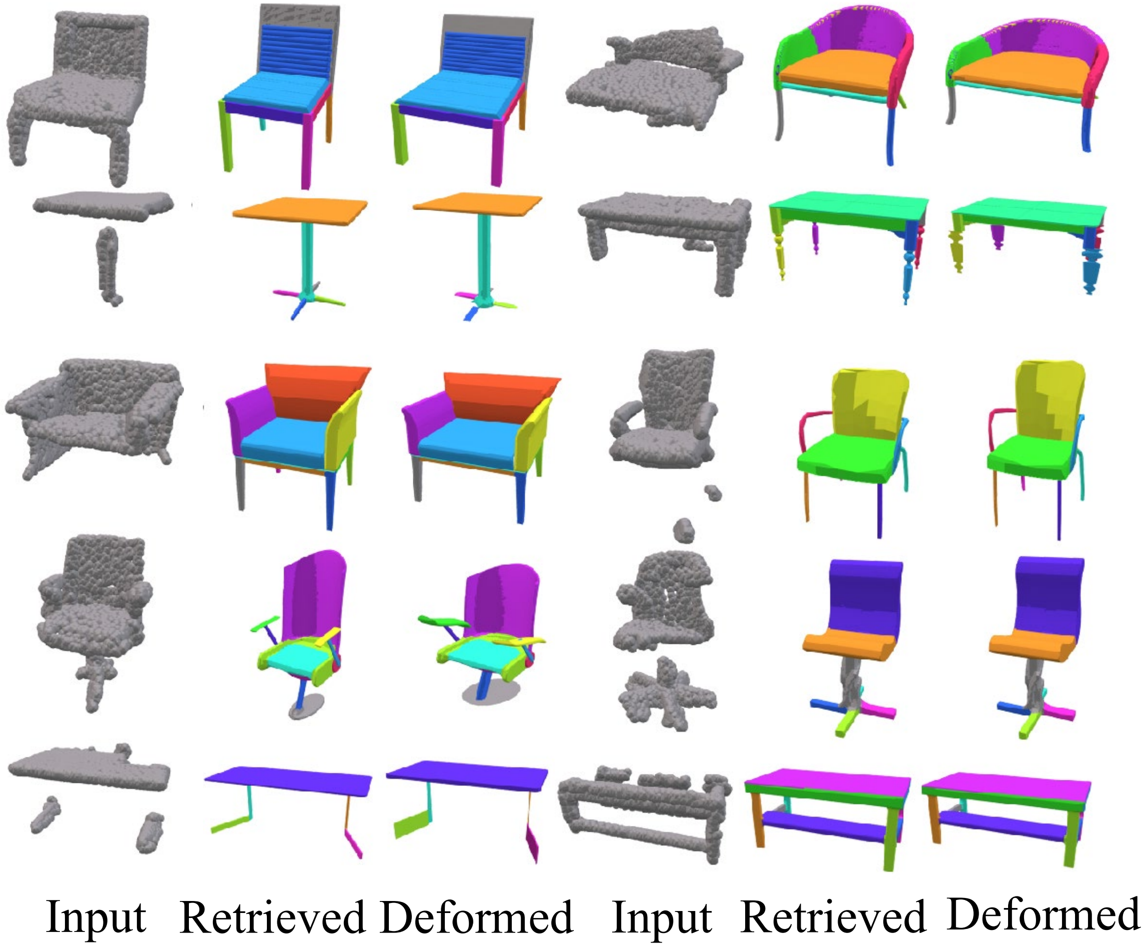


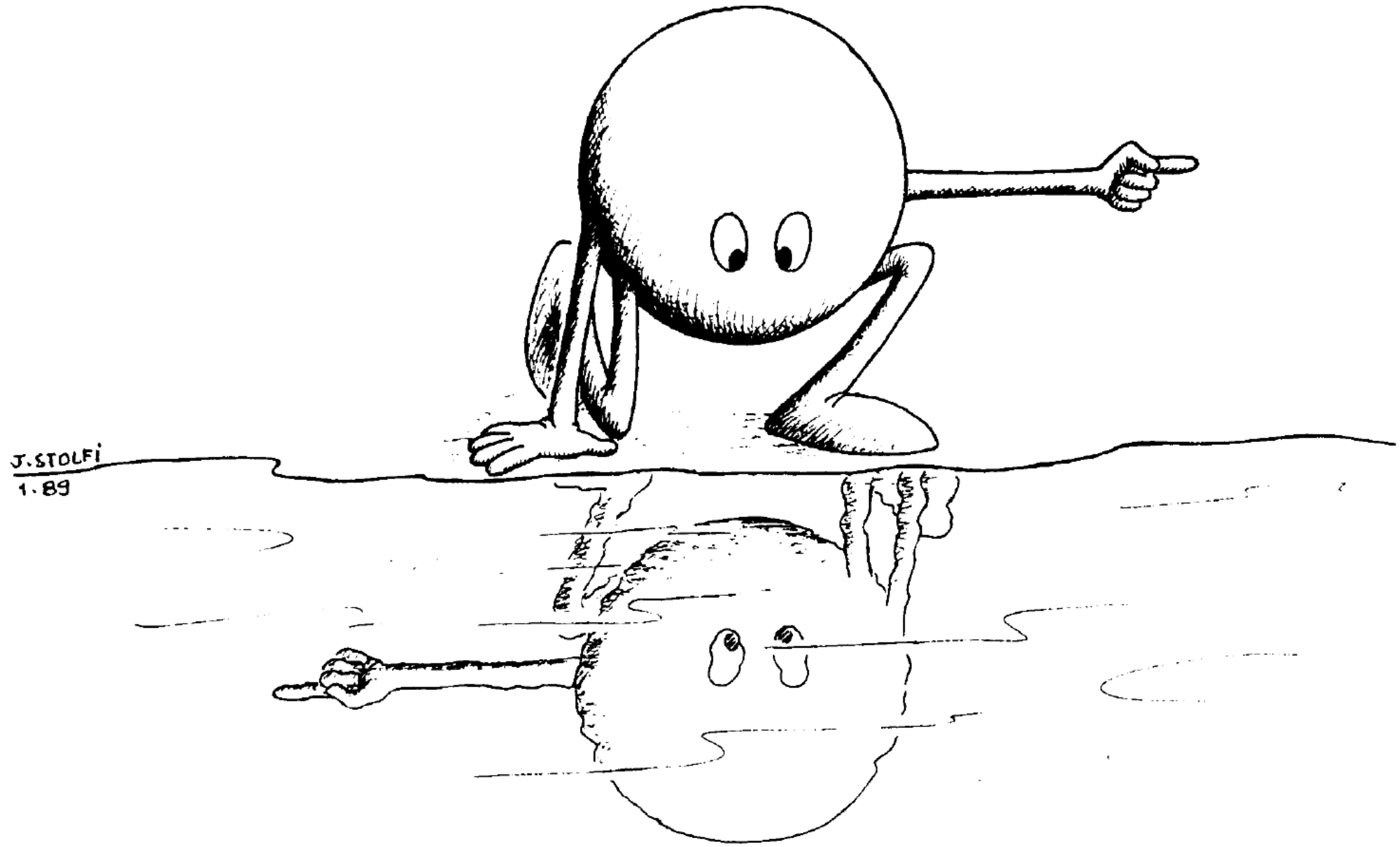
Applications

Product Images from Google search:



Real Scans:




J.STOLFI
1.89