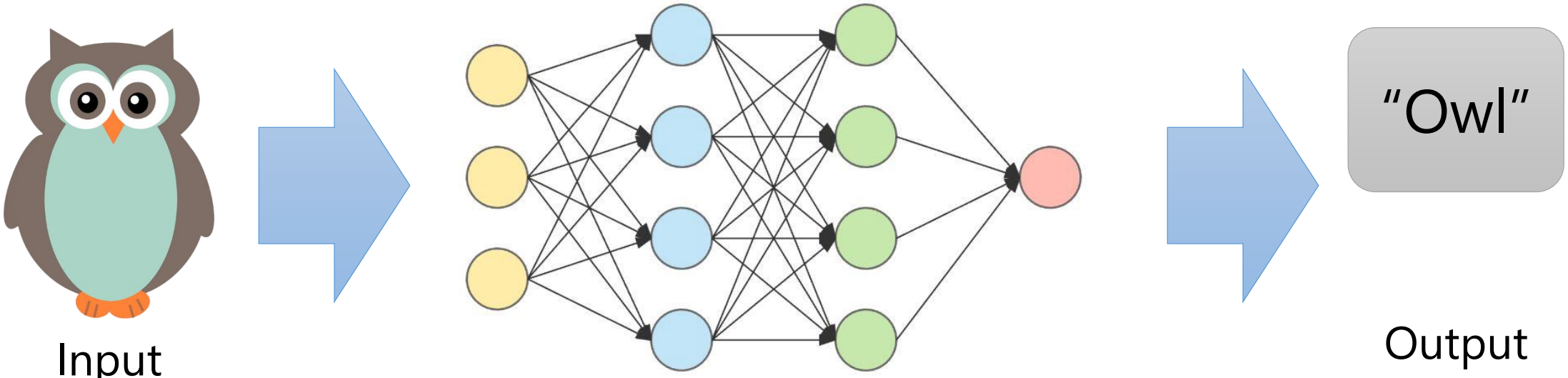
Reconstructing and Generating Scenes

Prof. Angela Dai

Brief Recap

Deep Learning



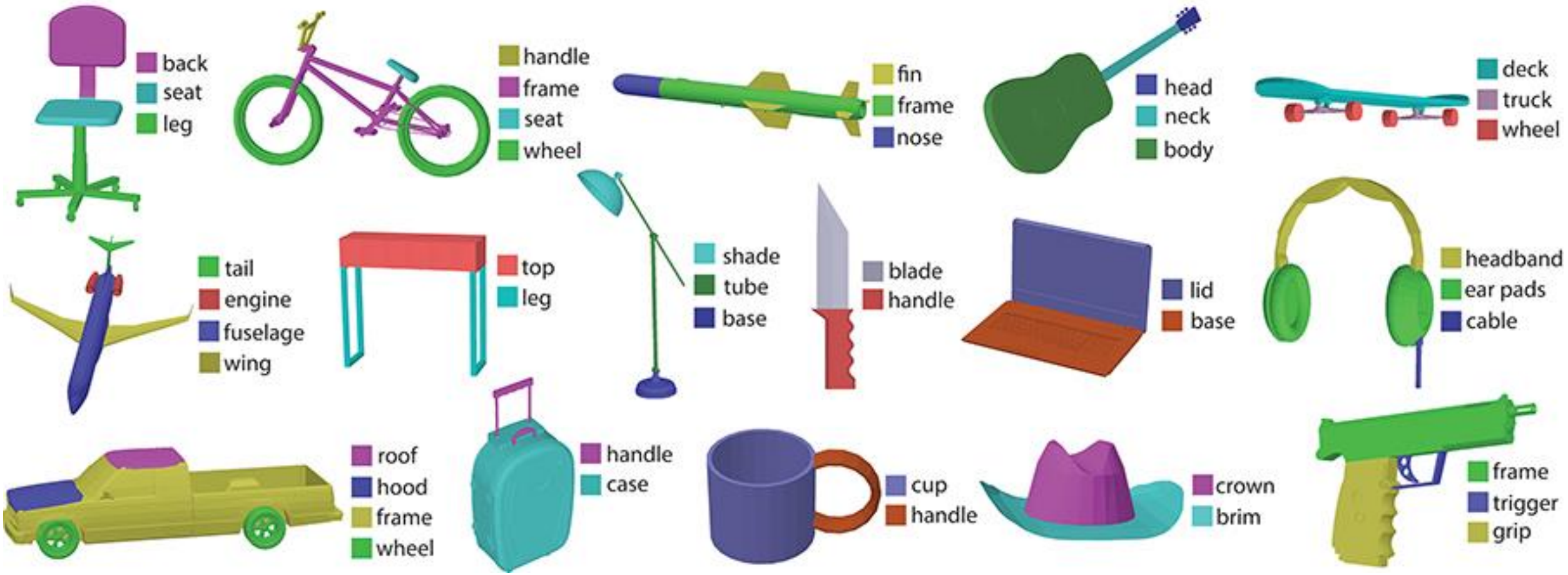
Input

Learned Feature Extraction
and Classification

Output

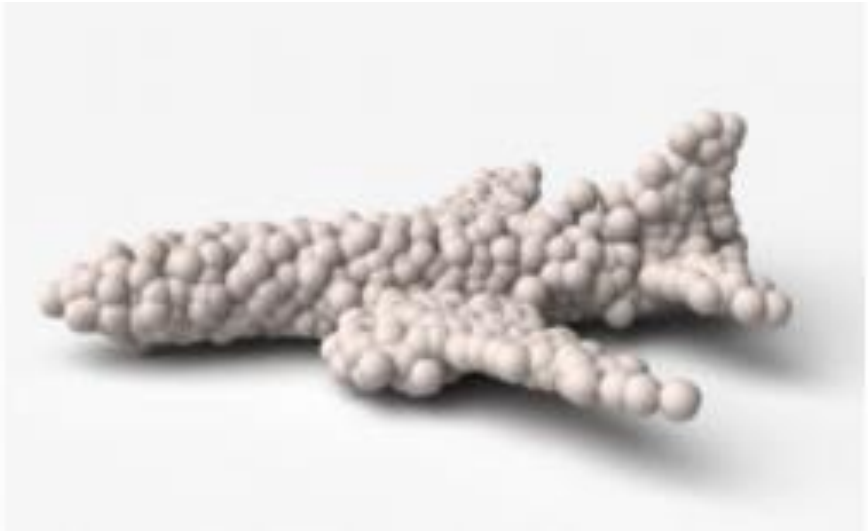
Want to automatically learn good feature representations for the task

Shape segmentation into parts

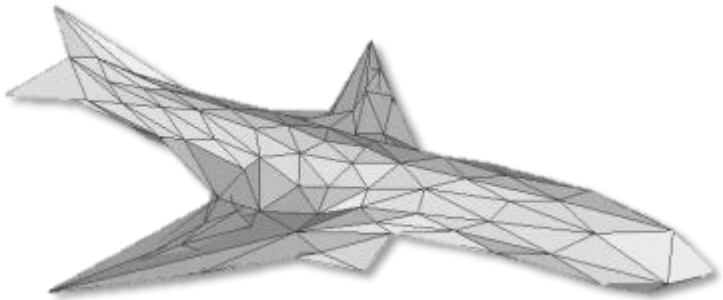


Generating Shapes

Signed Distance Fields

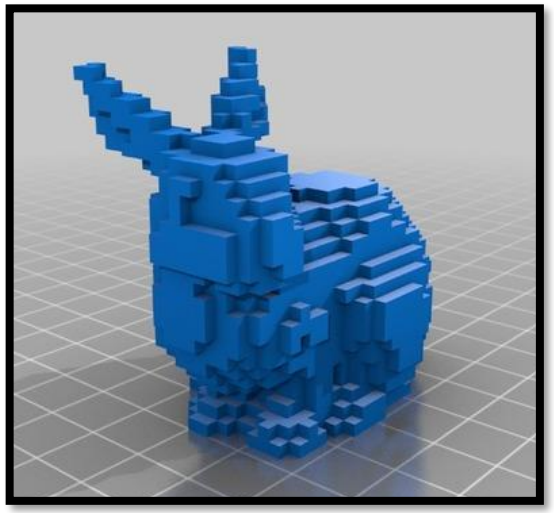


Point Clouds

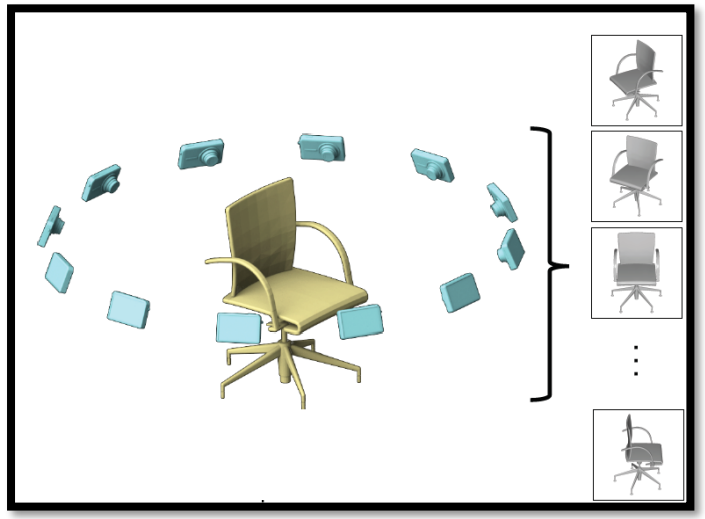


Meshes

3D Deep Learning by Representations

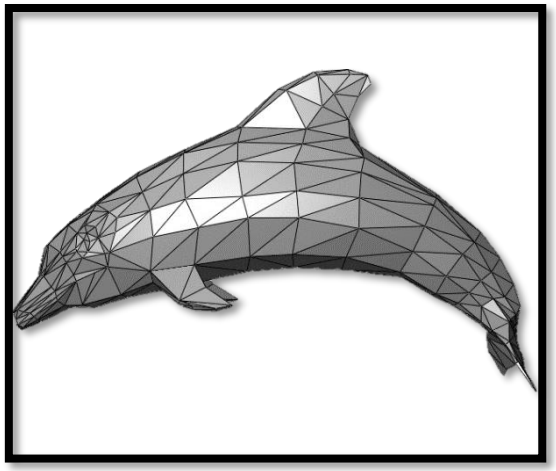


Volumetric
3D CNNs: Dense, Hierarchical, Sparse



Multi-View
(also: multi-view + volumetric/point/mesh)

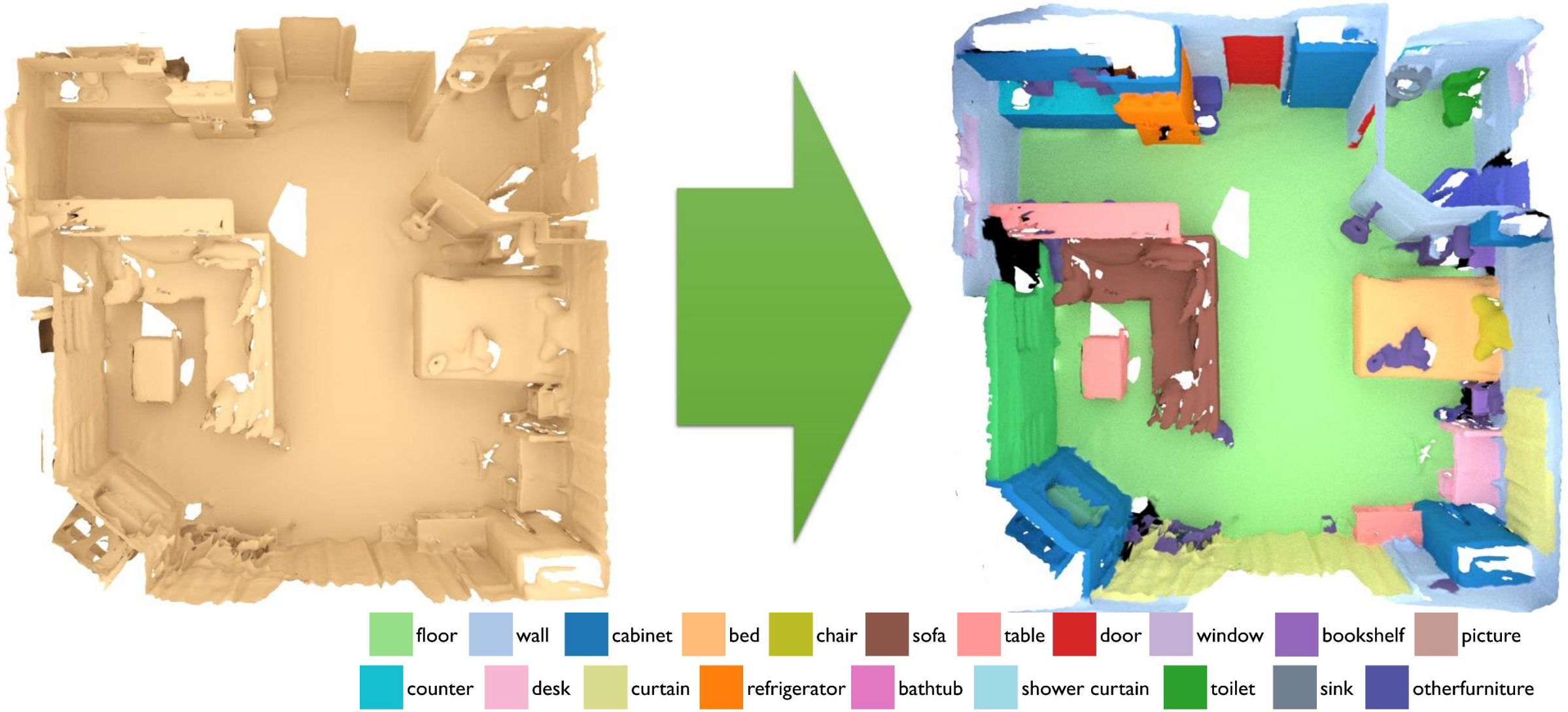
Point Cloud



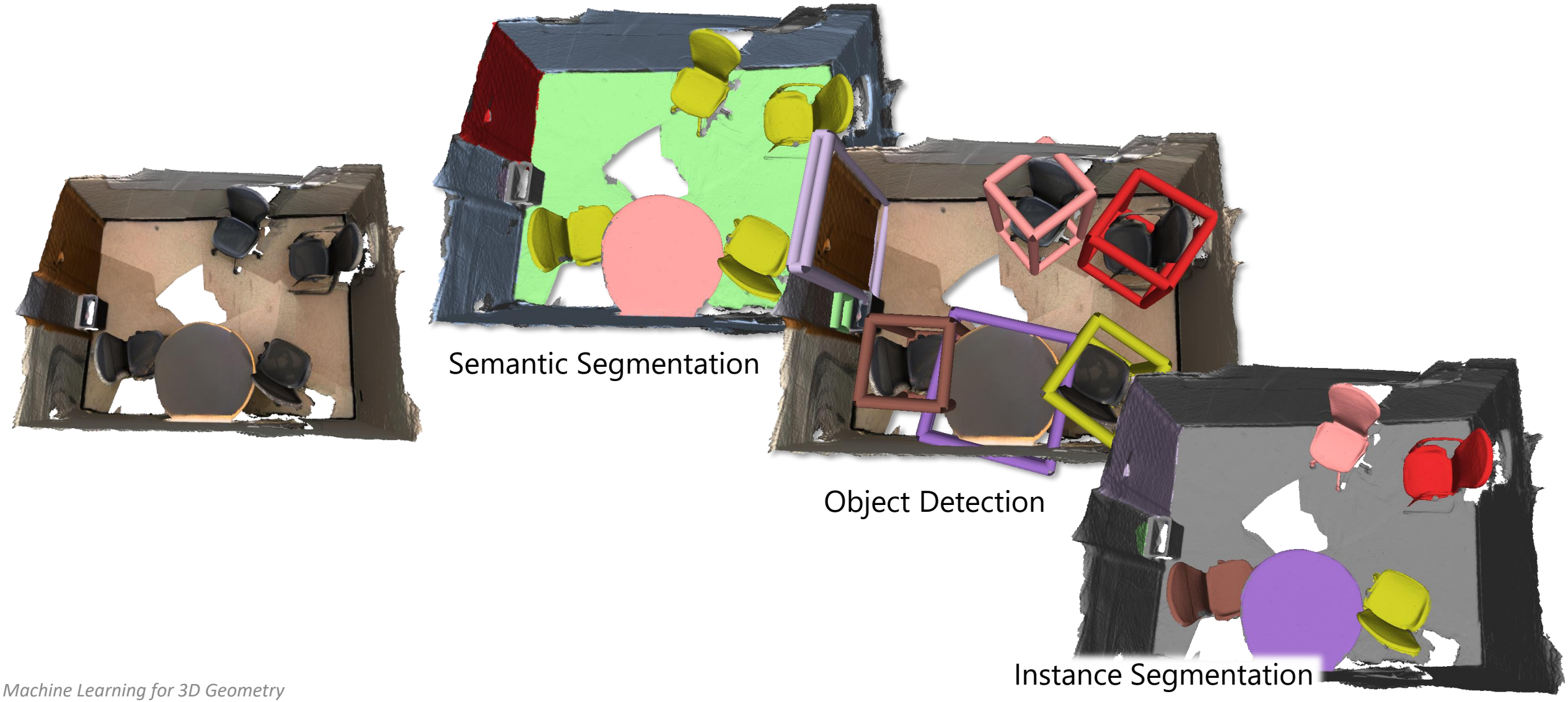
Mesh
Graph Neural Networks

and more!

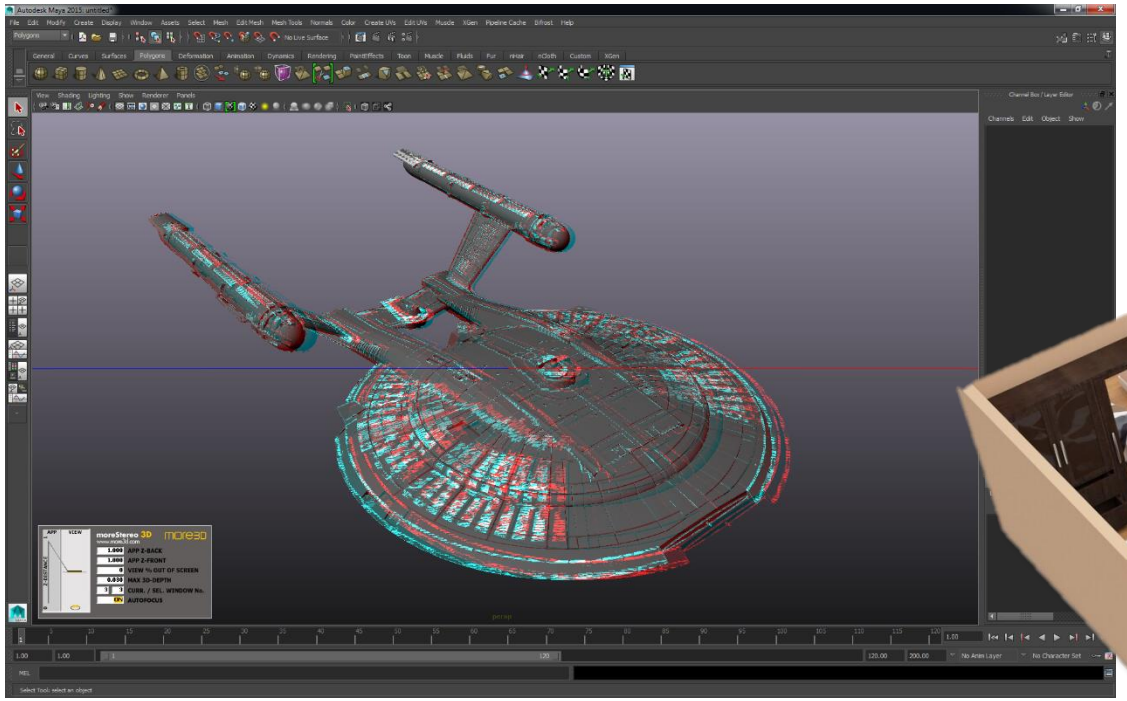
3D Semantic Segmentation



Understanding object-ness



Generating 3D Scenes



Content Creation

Capturing 3D photos

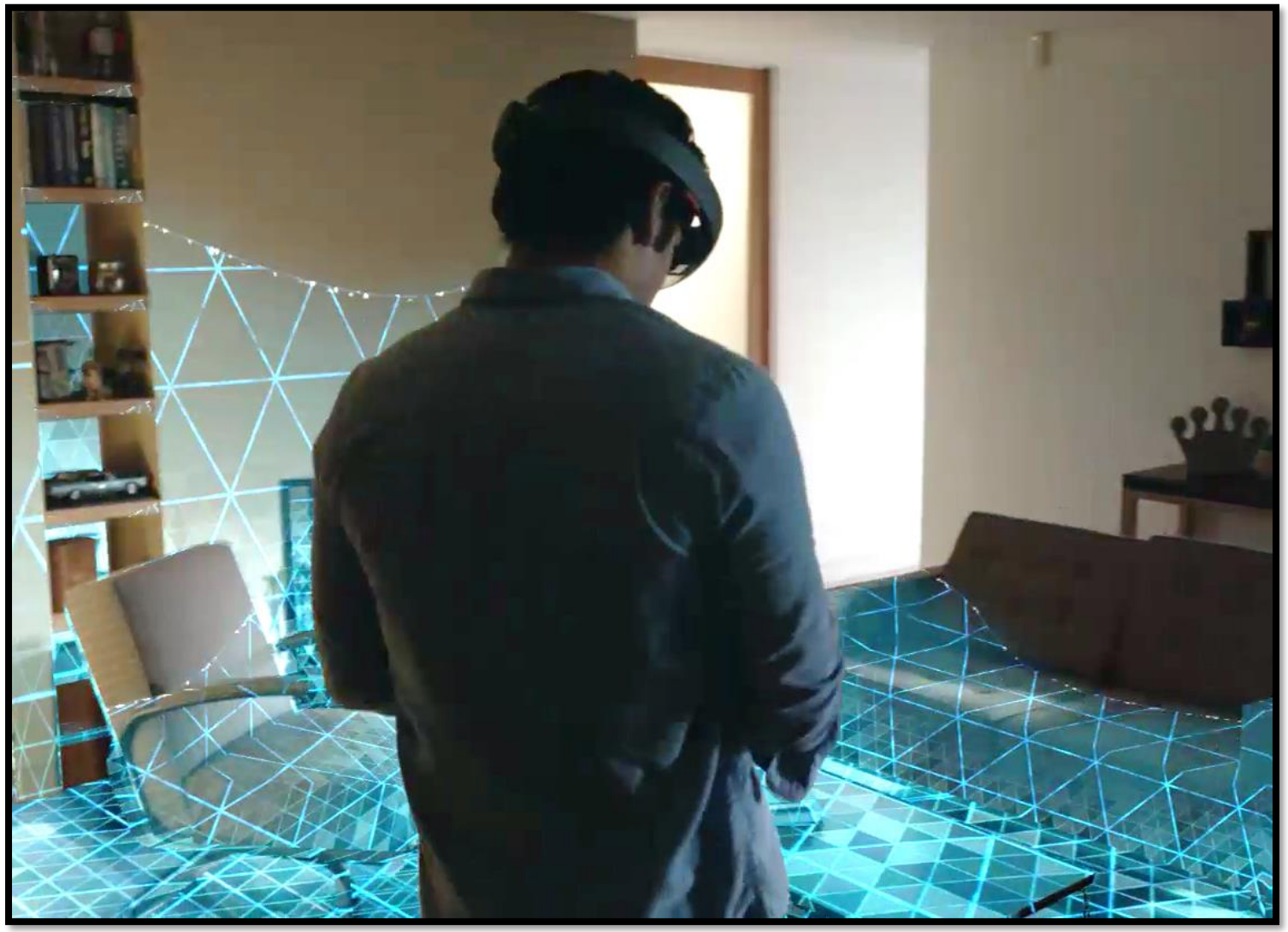
https://tech.fb.com/3d-photos-how-they-work-and-how-anyone-can-take-them/

Capturing 3D photos

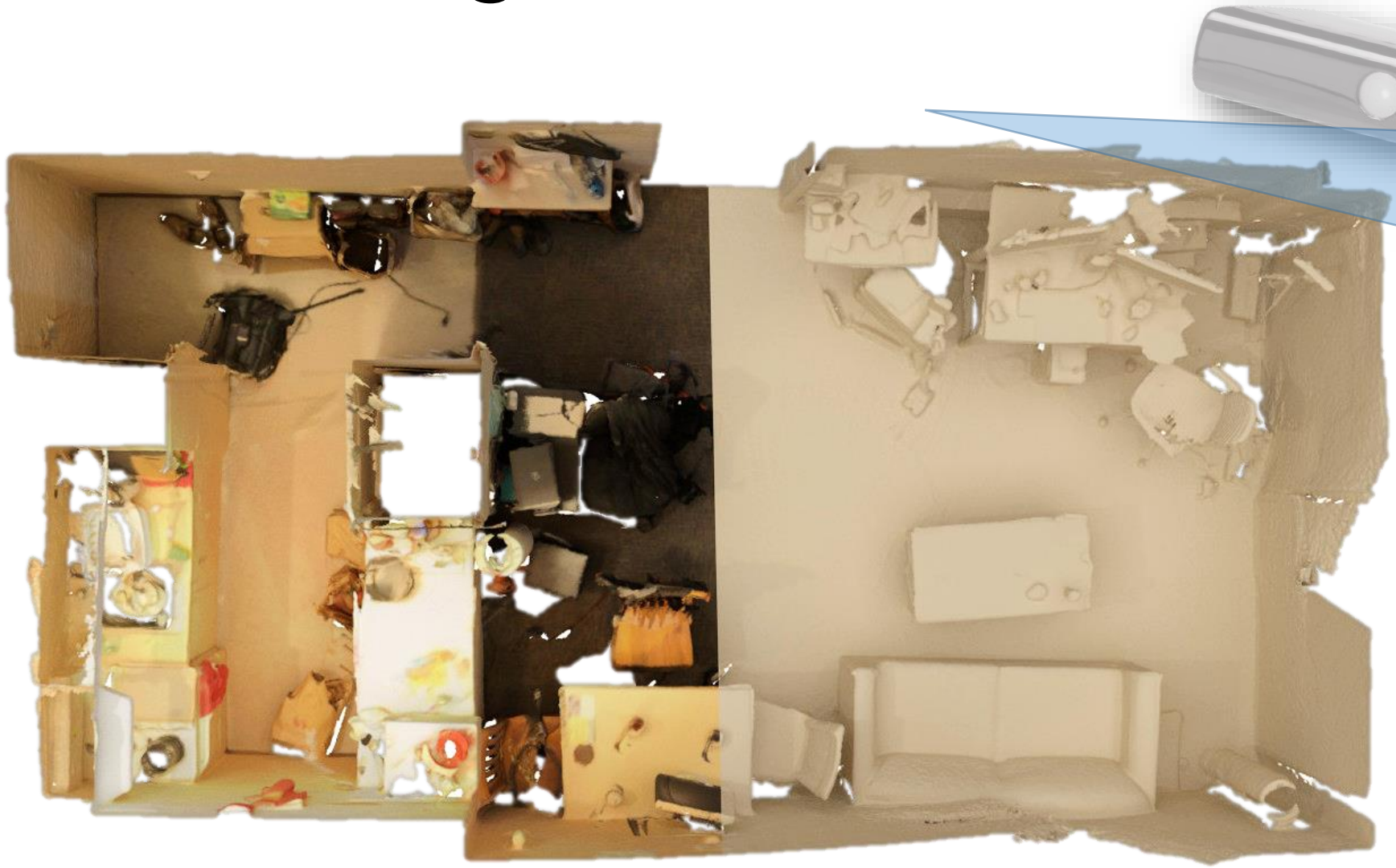
[Mildenhall et al. ’20] NERF

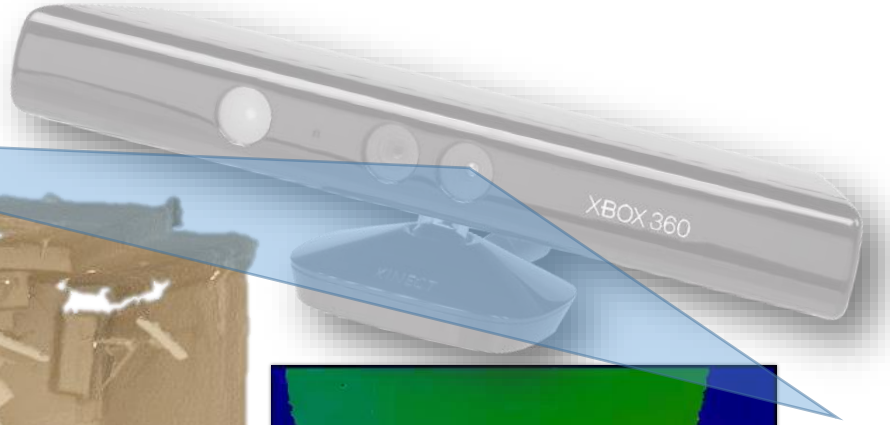
https://tech.fb.com/3d-photos-how-they-work-and-how-anyone-can-take-them/

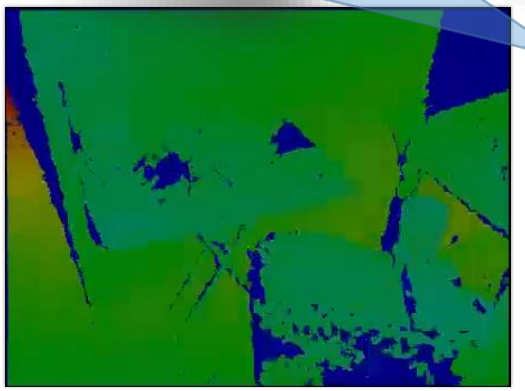
Reconstructing real-world scenes

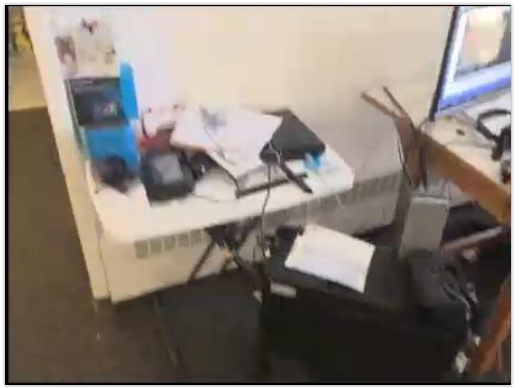


3D scanning for real-world reconstruction



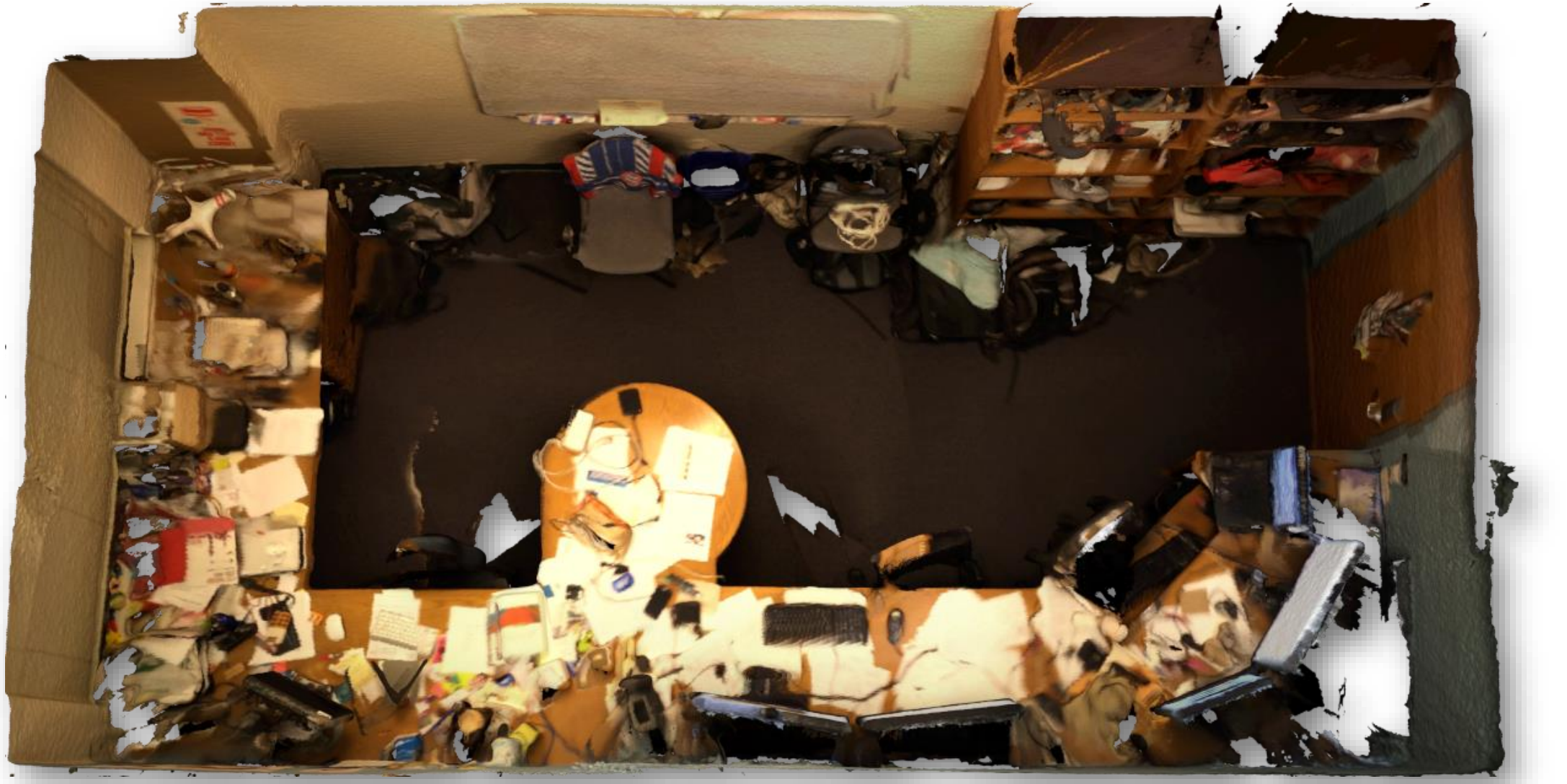






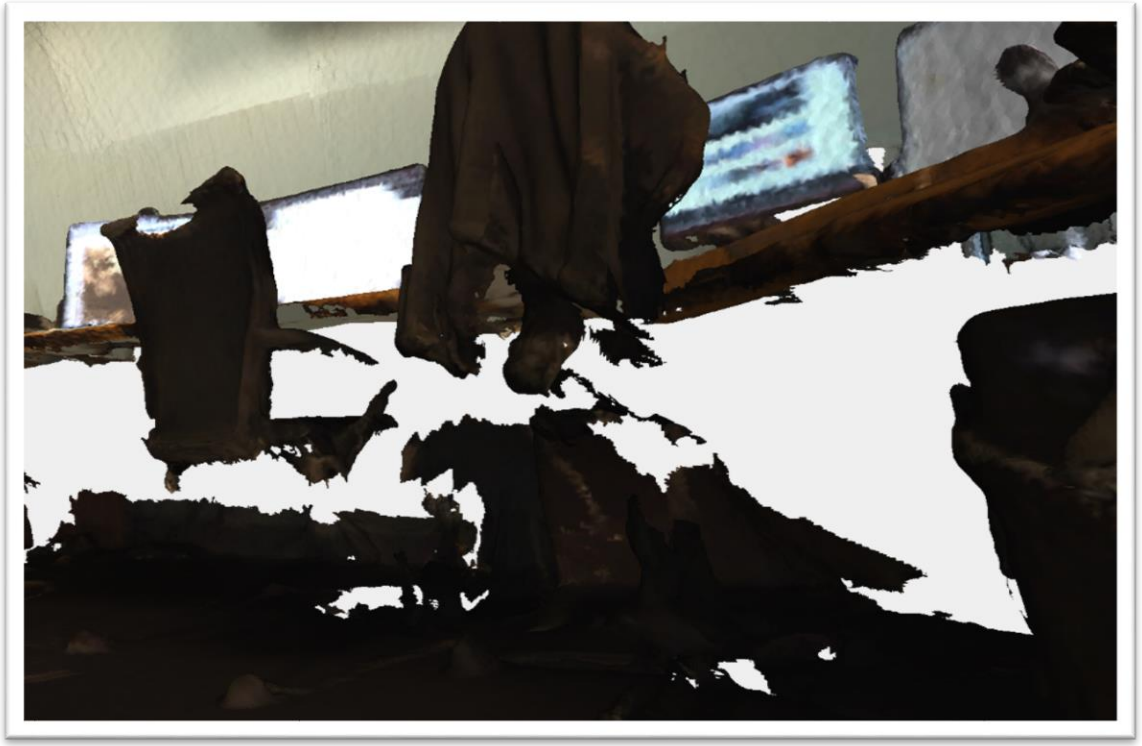
(4x)

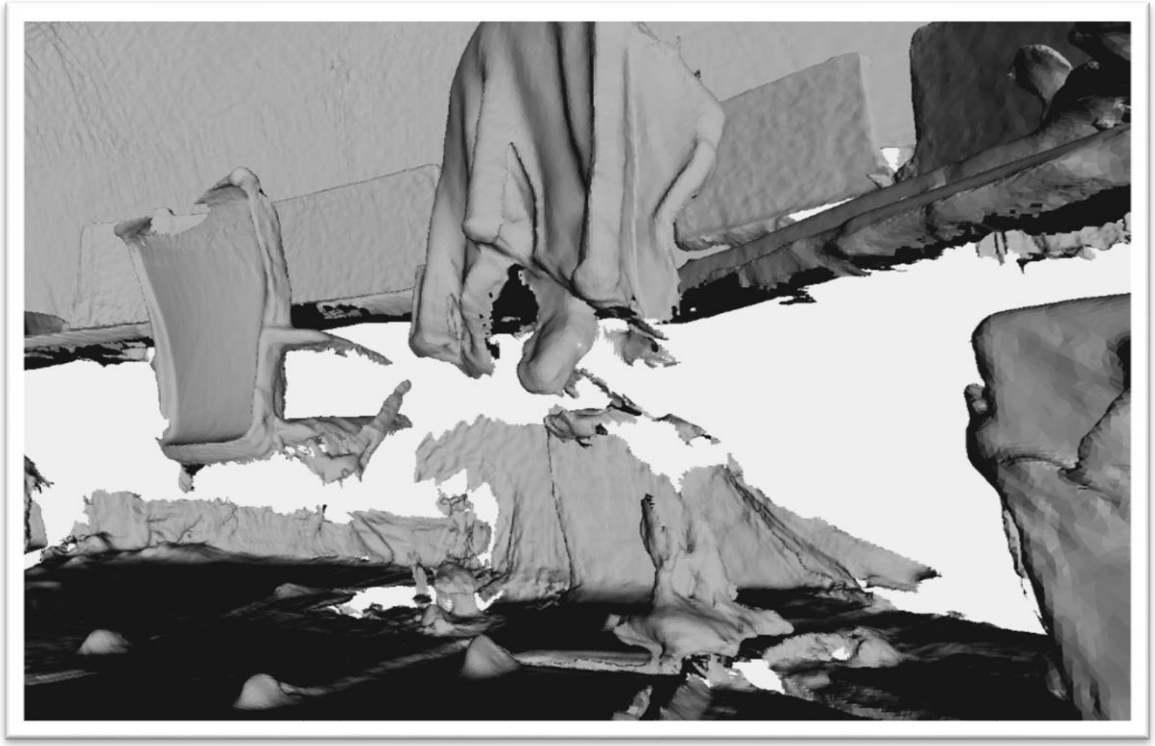
State-of-the-art 3D Reconstructions



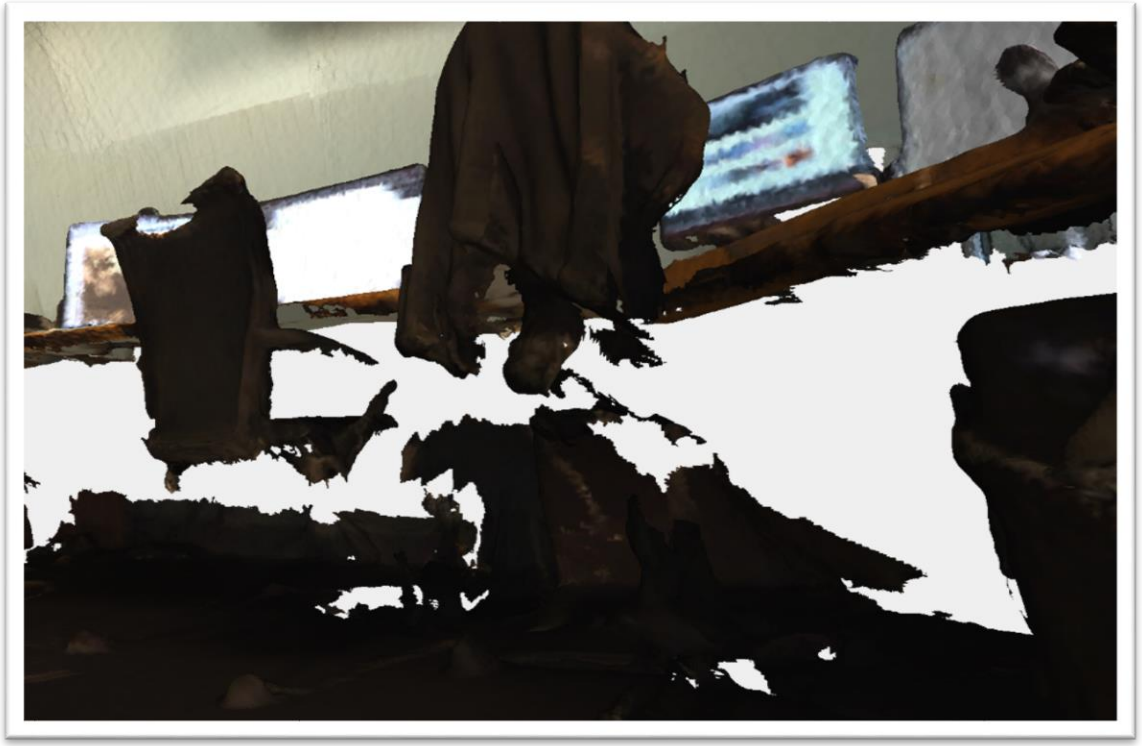
[Dai et al. ‘17]

Problem: Missing Geometry

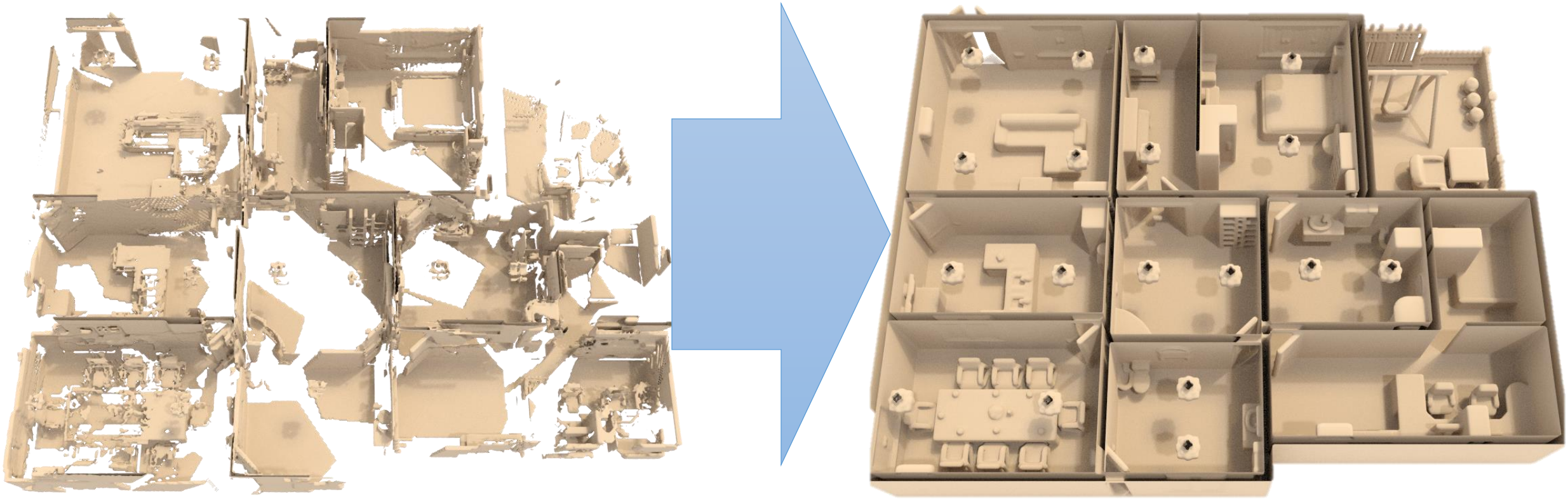




Problem: Missing Geometry

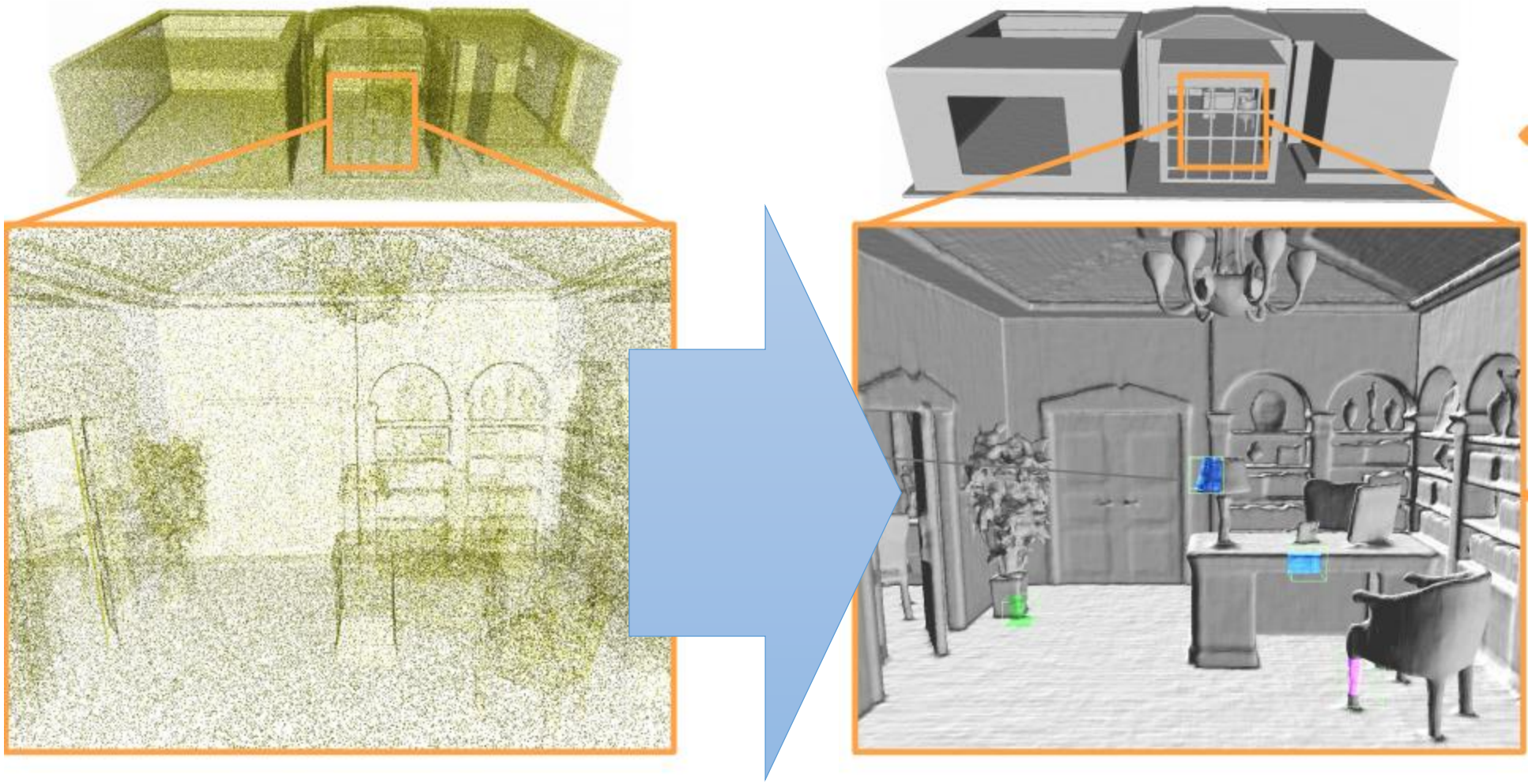


Generative Tasks: Scan Completion



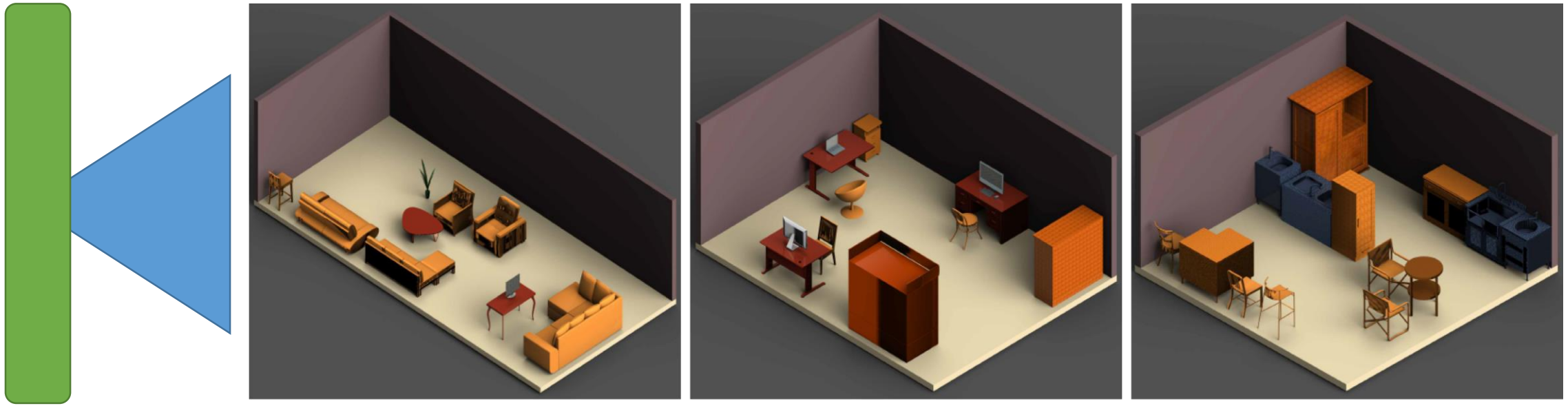
[Dai et al. ‘18]

Generative Tasks: Surface Reconstruction



[Jiang et al. ’20]

Generative Tasks: Scene Generation



Sampled latent code

[Li et al. ’18]

2D Image Generation

- Simple generator

Reconstruction loss, e.g., ℓ_1, ℓ_2

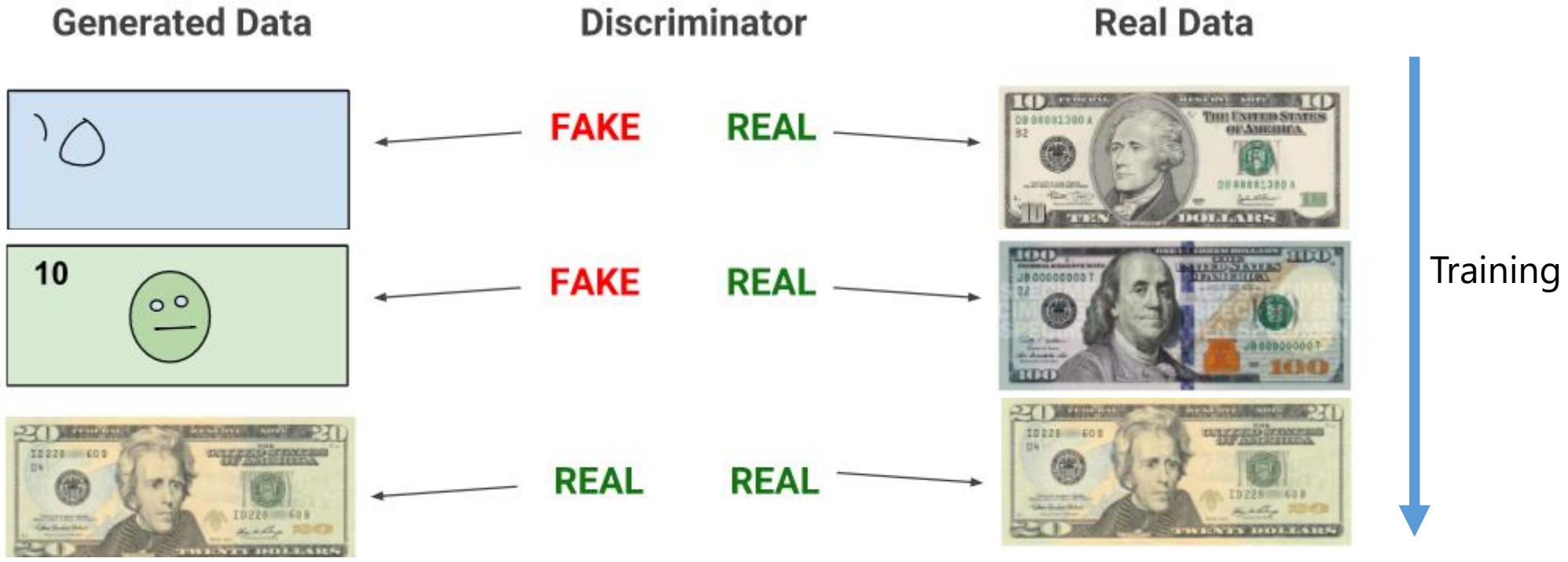
Successes in 2D Generation

- Generative adversarial networks [Goodfellow et al. ’14]

[Karras et al. ’20] StyleGAN

Successes in 2D Generation

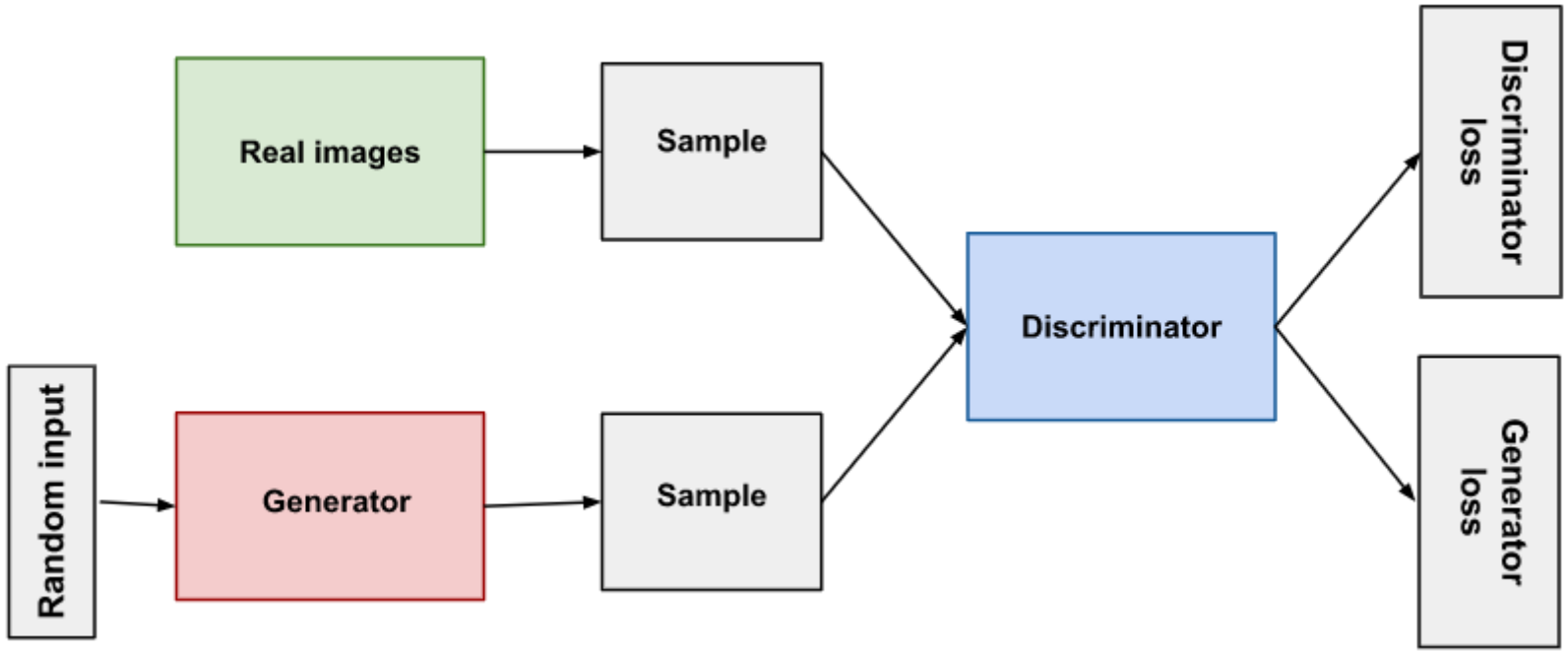
- Generative adversarial networks [Goodfellow et al. ’14]



https://developers.google.com/machine-learning/gan/gan_structure

Successes in 2D Generation

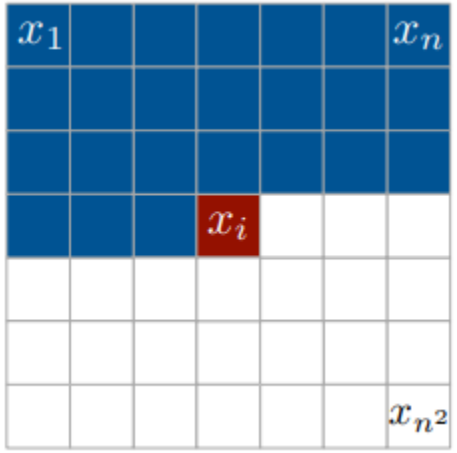
- Generative adversarial networks [Goodfellow et al. ’14]



https://developers.google.com/machine-learning/gan/gan_structure

Successes in 2D Generation

- Autoregressive models
- Generate image pixel by pixel



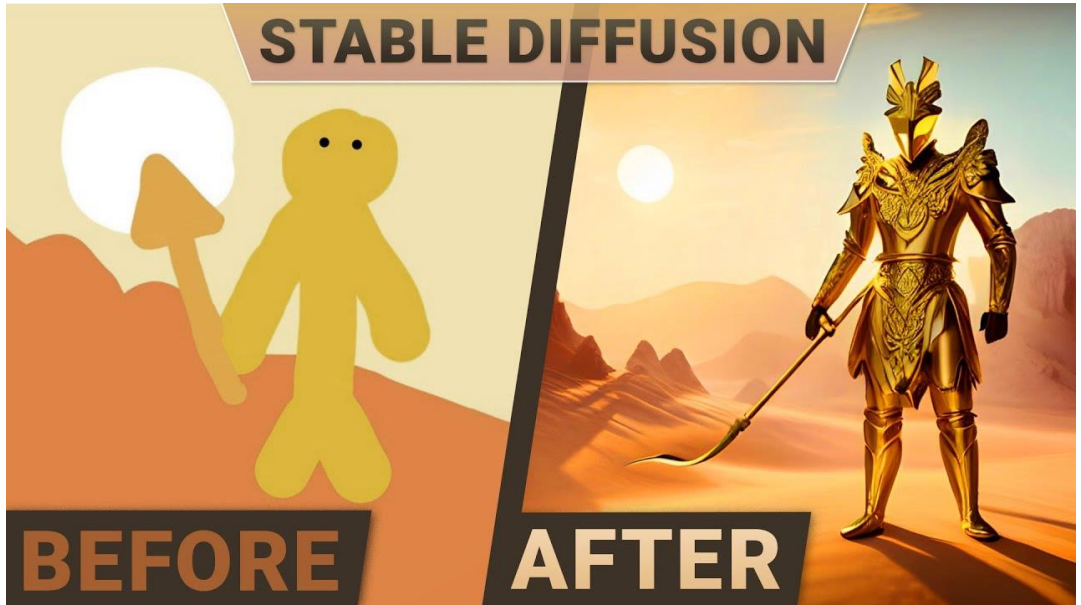
PixelRNN, PixelCNN

VQ-VAE, VQ-VAE2

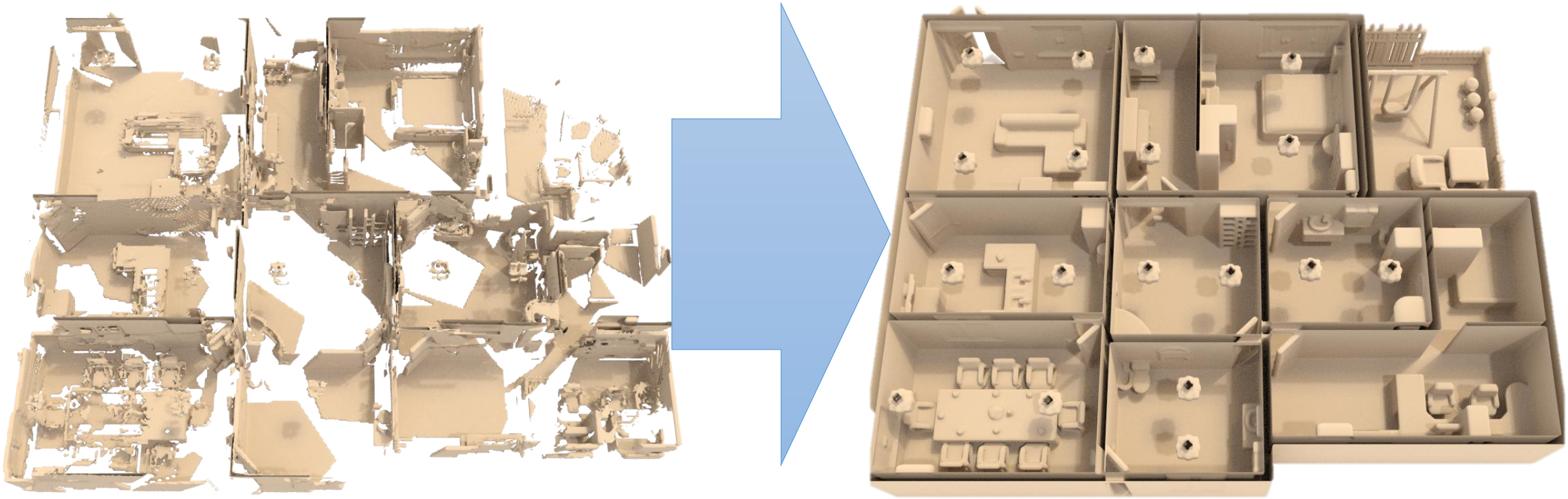
[Van den Oord et al. '16; Van den Oord et al. '18]

Successes in 2D Generation

- Diffusion models
- Model noising process



Generative Tasks: Scan Completion



[Dai et al. ‘18]

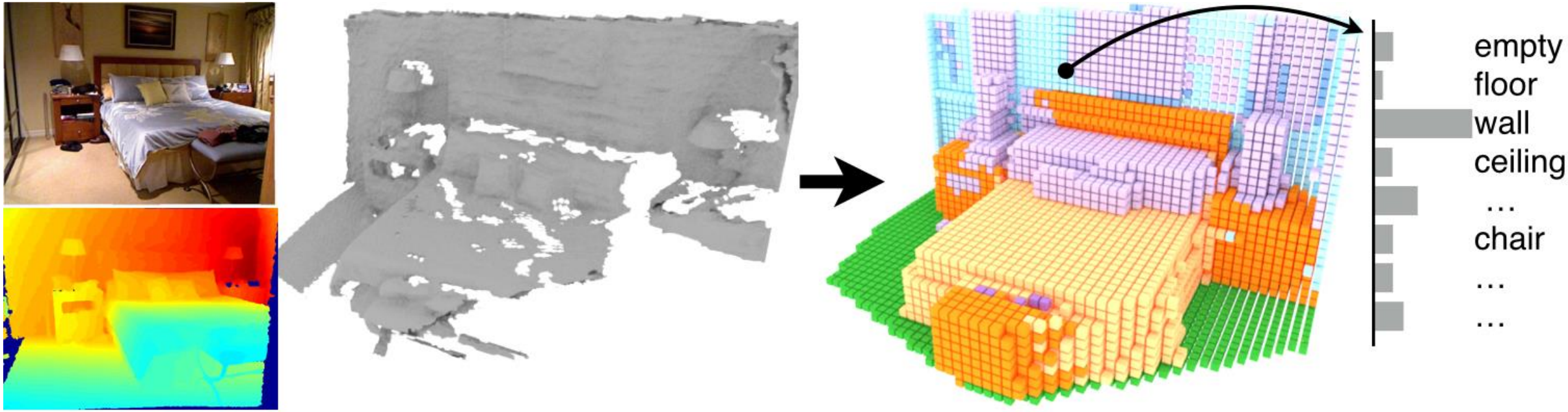
Complete Ground Truth Data?

- Synthetic 3D Datasets:
- 3D-FRONT: 3D Furnished Rooms with layOuts and semantics [Fu et al. ’20]
  - https://tianchi.aliyun.com/specials/promotion/alibaba-3d-scene-dataset
- ICL-NUIM [Handa et al. ’14]
  - https://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html

- Self-supervision on incomplete real data?

SSCNet

- Input: RGB-D image
- Output: $2.26\mathrm{m}^3$ geometry occupancy + semantic labels

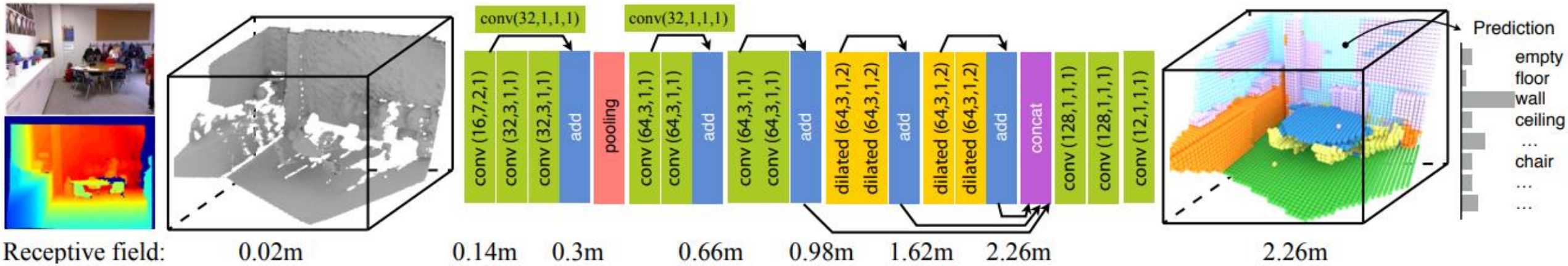


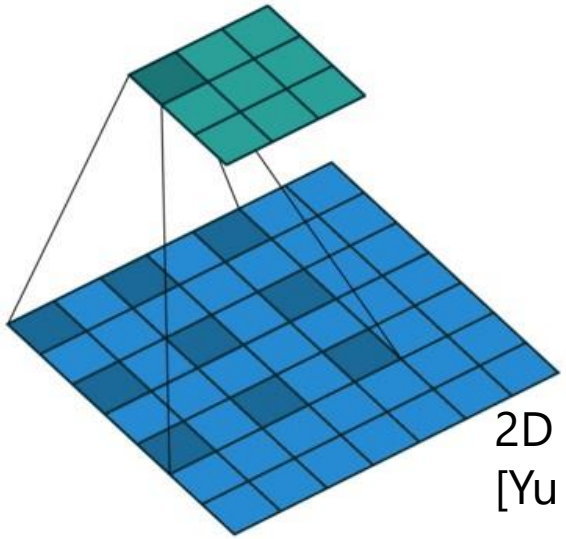
Input: single depth map

Output: semantic scene completion

[Song et al. '17]

SSCNet

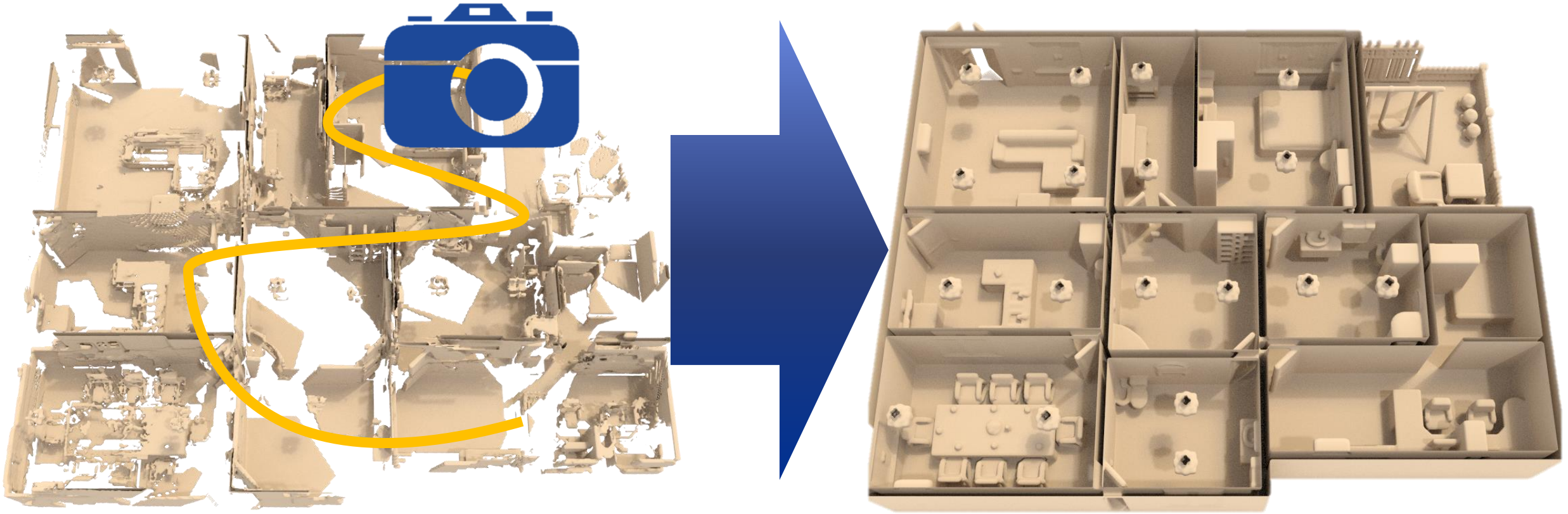




2D dilated convolution
[Yu and Koltun '16]

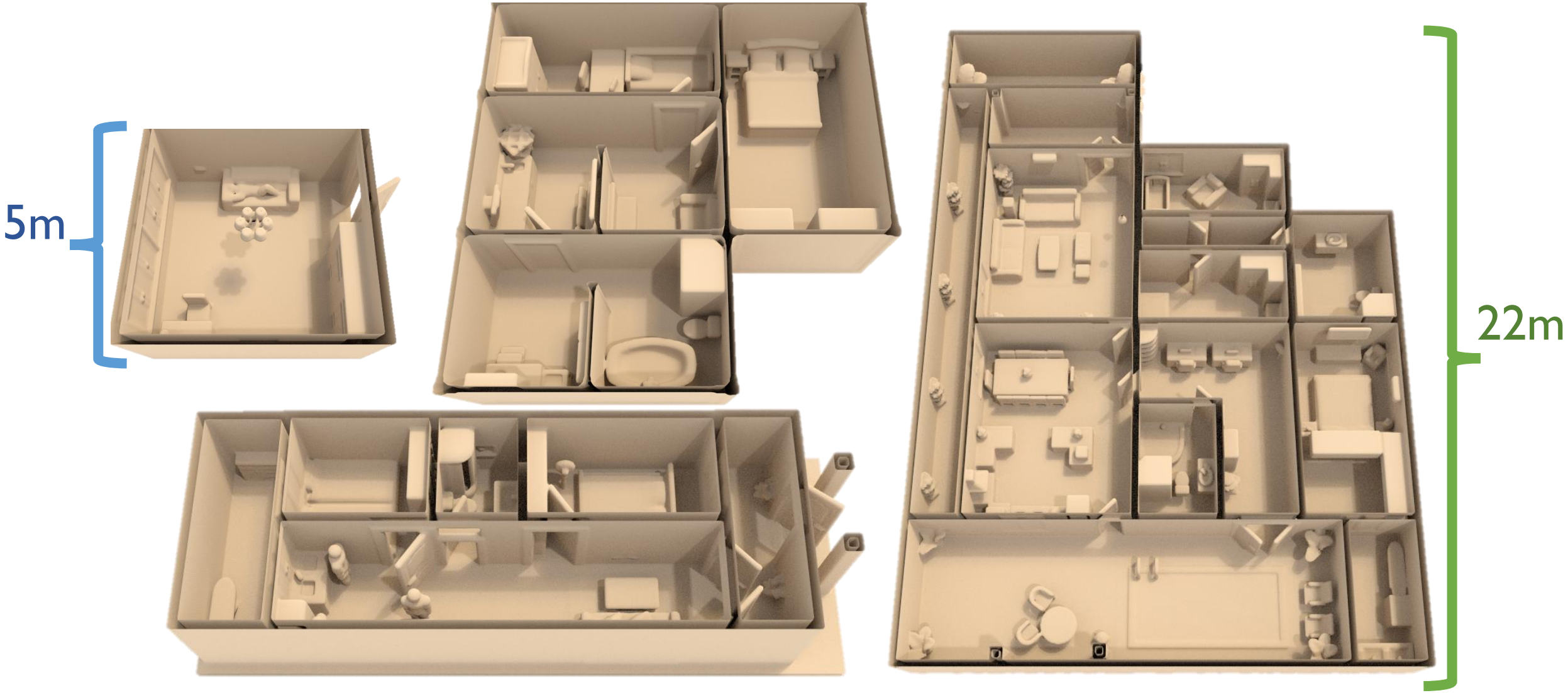
[Song et al. '17]

ScanComplete



[Dai et al. ’18]

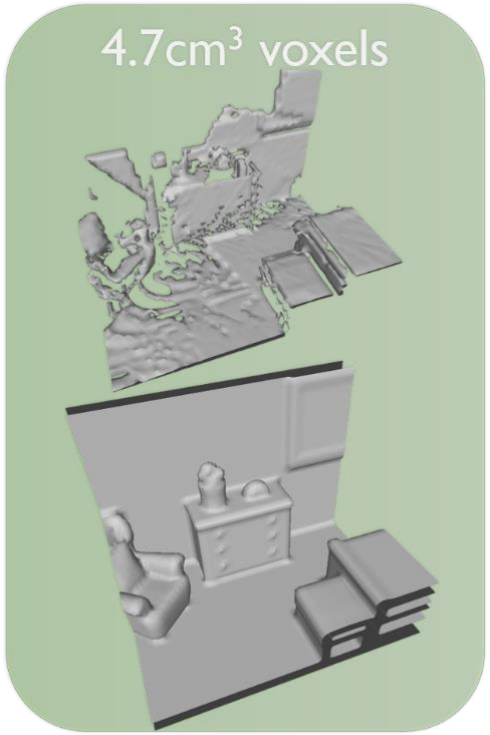
In 3D: large, varying-sized scenes



ScanComplete

- Fully-convolutional approach to apply to varying-sized scenes

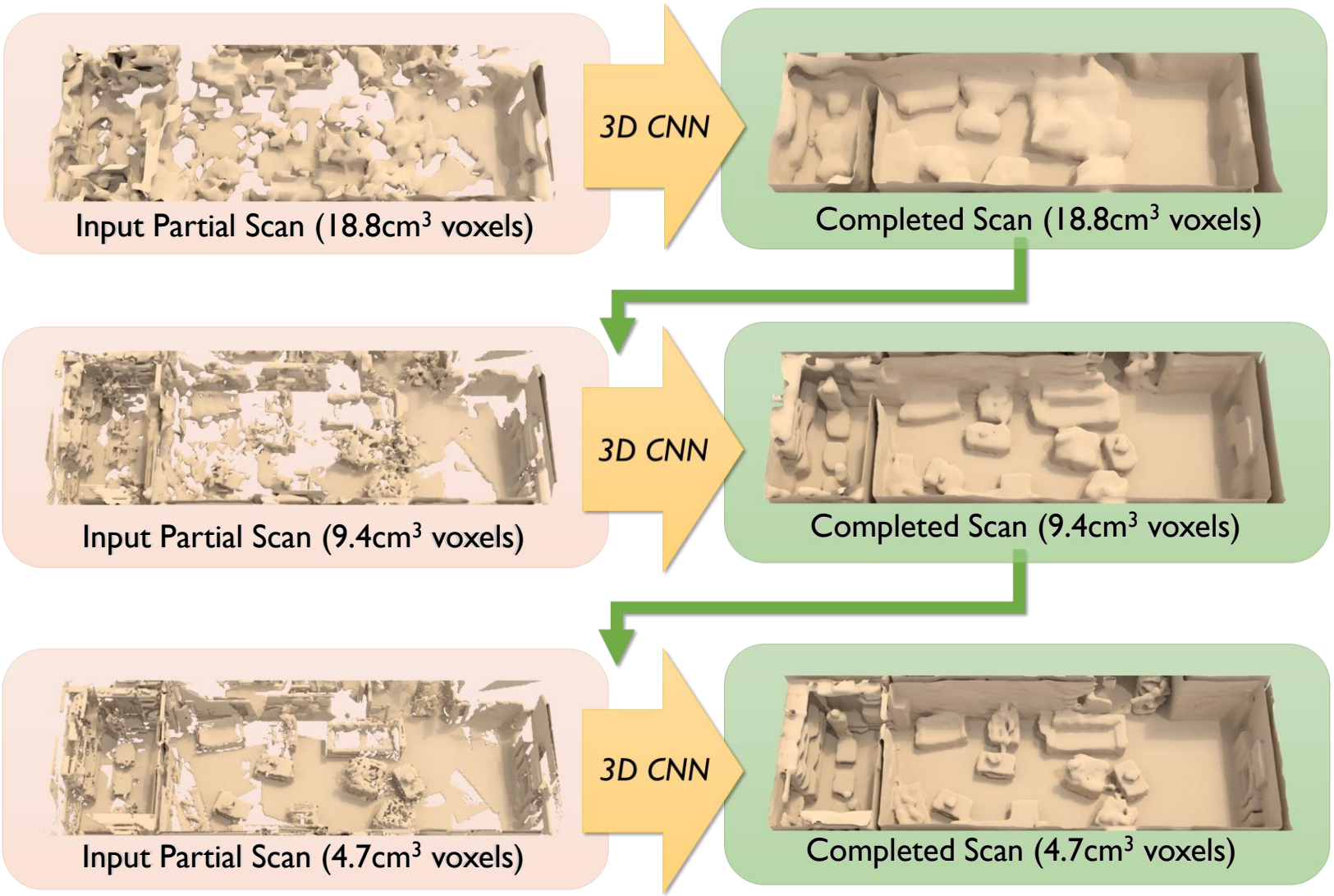
Train on crops of scenes



Test on entire scenes

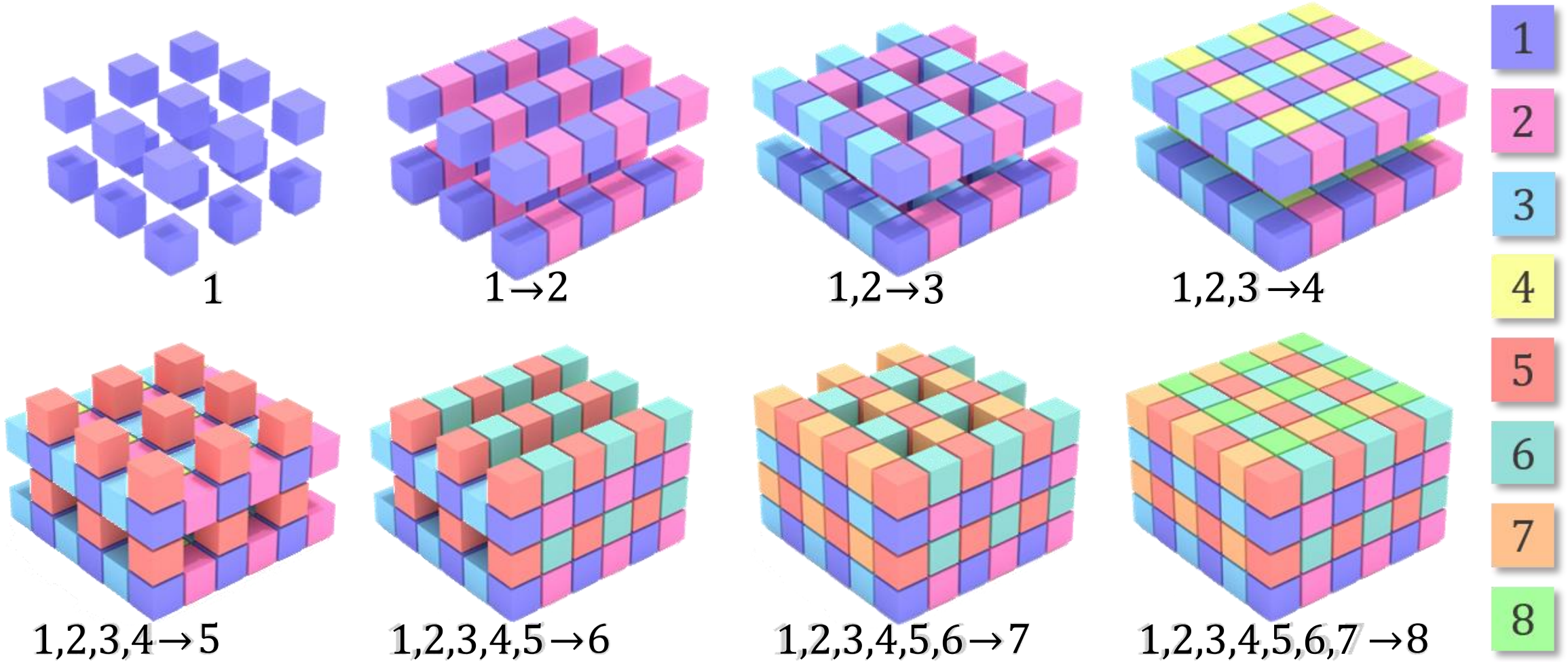
[Dai et al. '18]

ScanComplete



[Dai et al. '18]

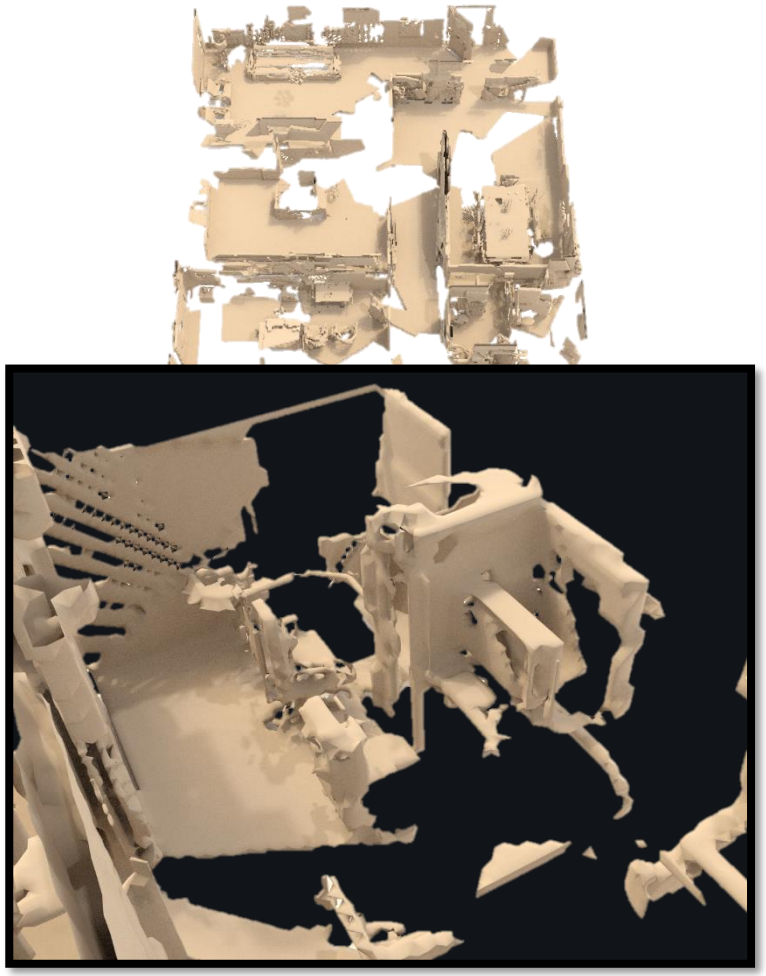
ScanComplete



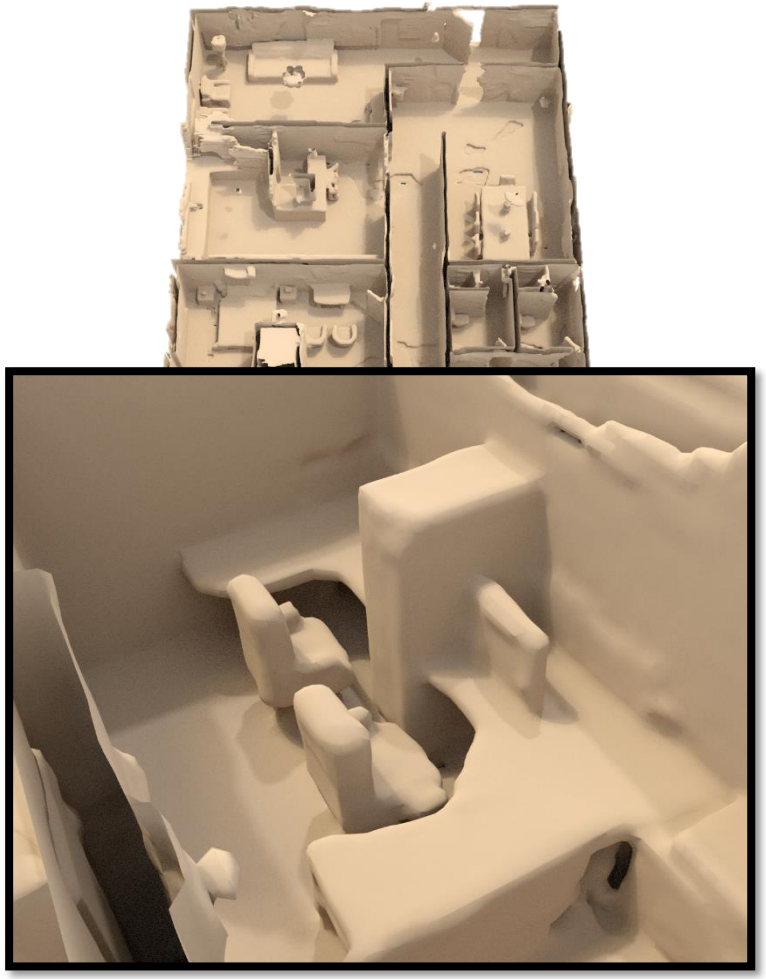
[Dai et al. '18]

ScanComplete

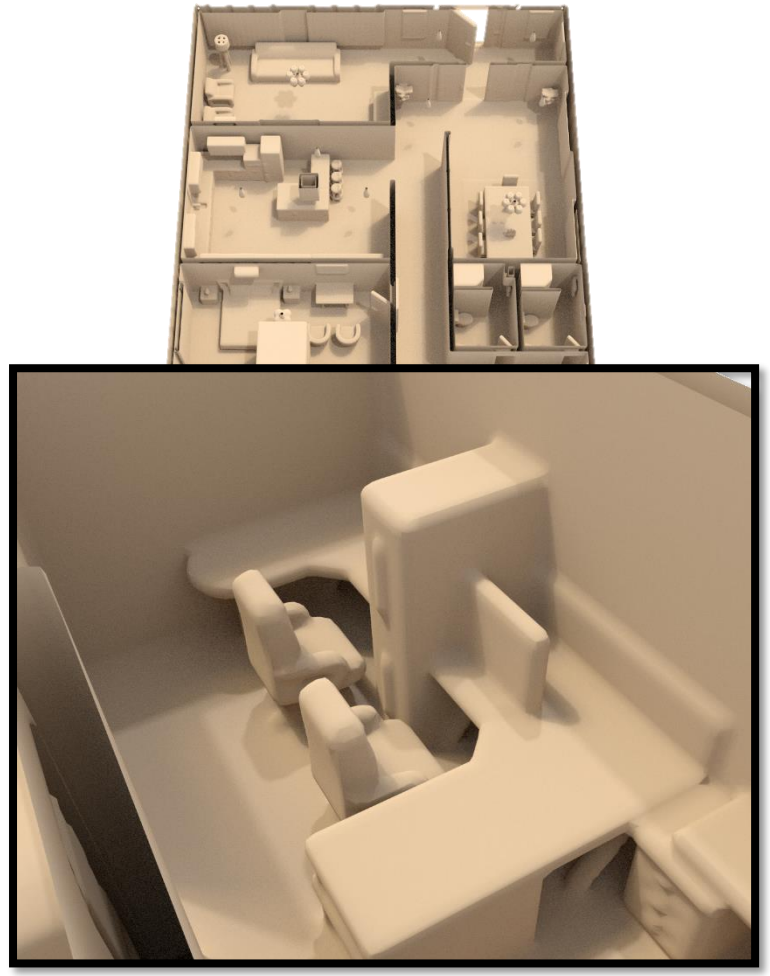
Input



Completion



Ground Truth



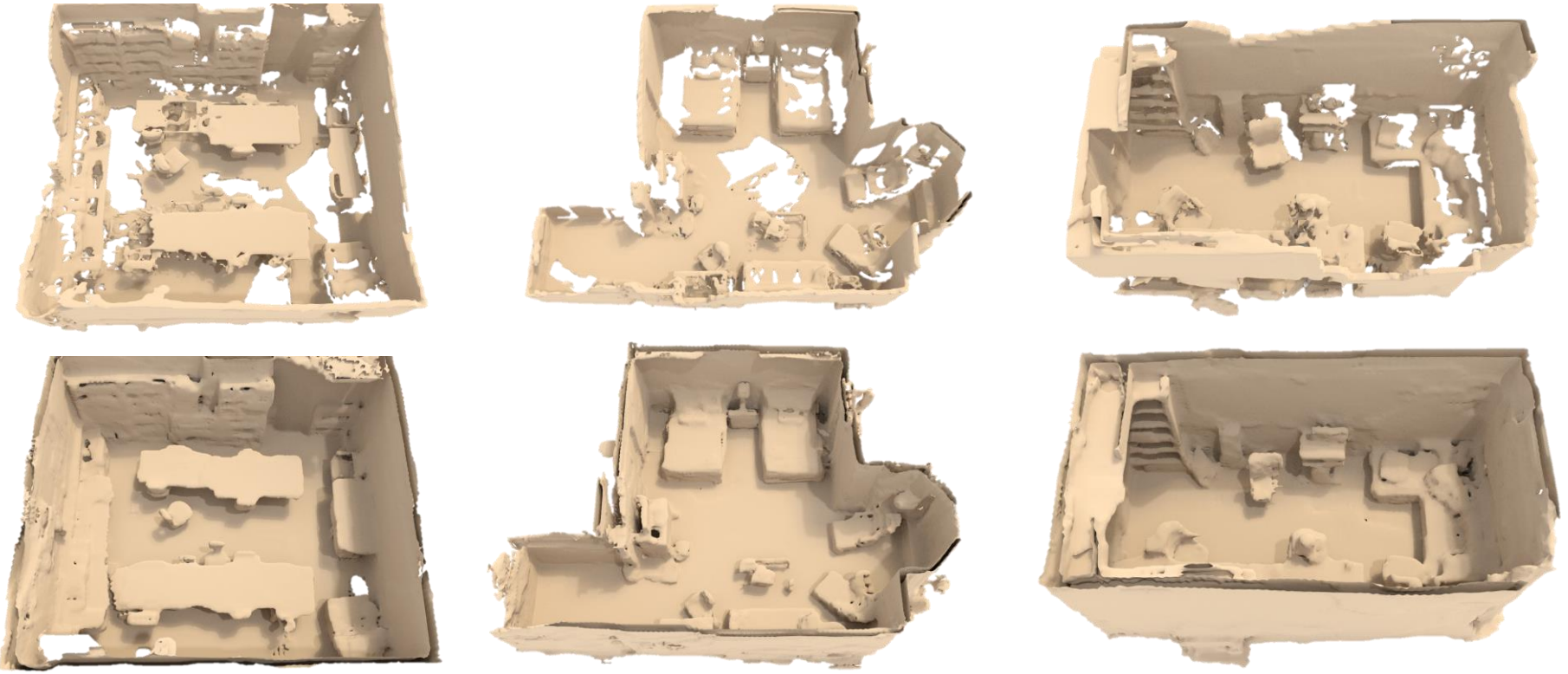
[Dai et al. '18]

ScanComplete

- On real-world scan data

Input

Completion



[Dai et al. ’18]

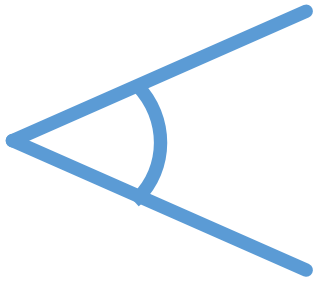
ScanComplete

- On real-world scan data

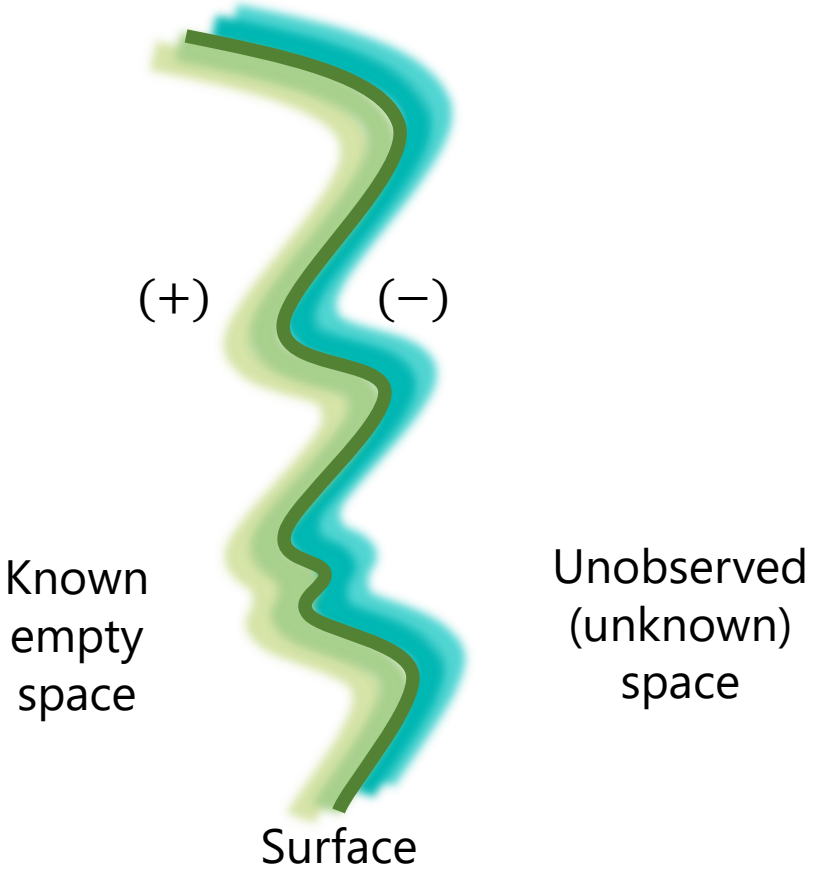
[Dai et al. '18]

SG-NN: Self-supervised Scan Completion

- Recall: signed distance field representation
- In 3D scanning:

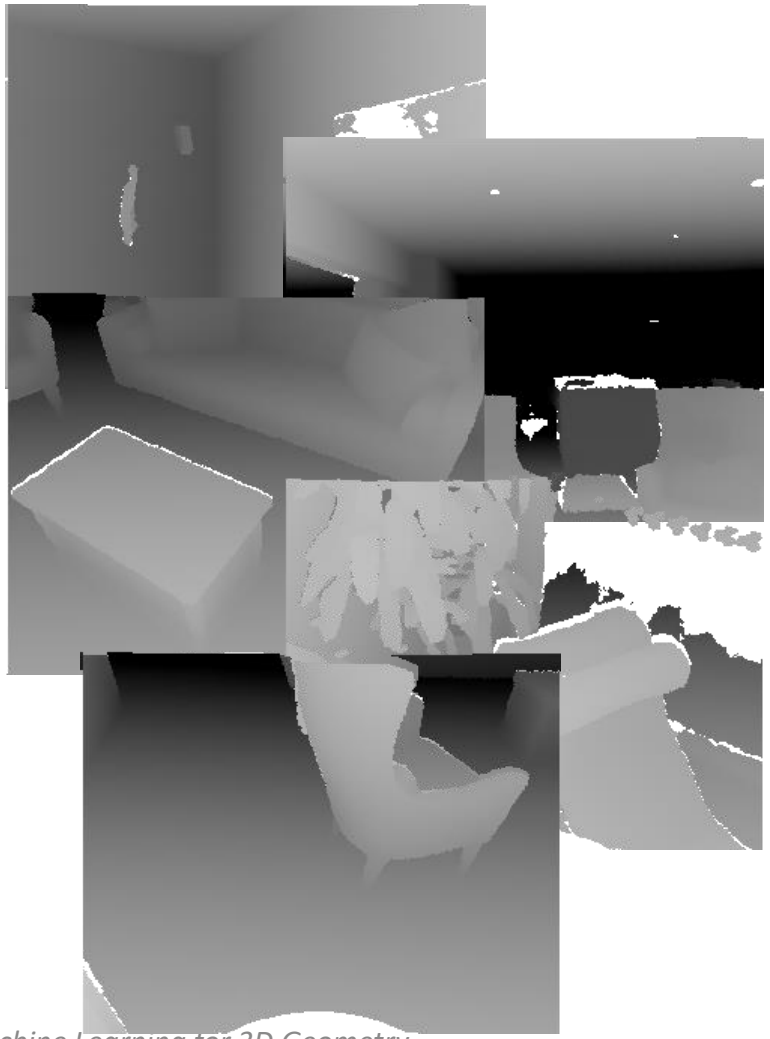


Camera



SG-NN: Self-supervised Scan Completion

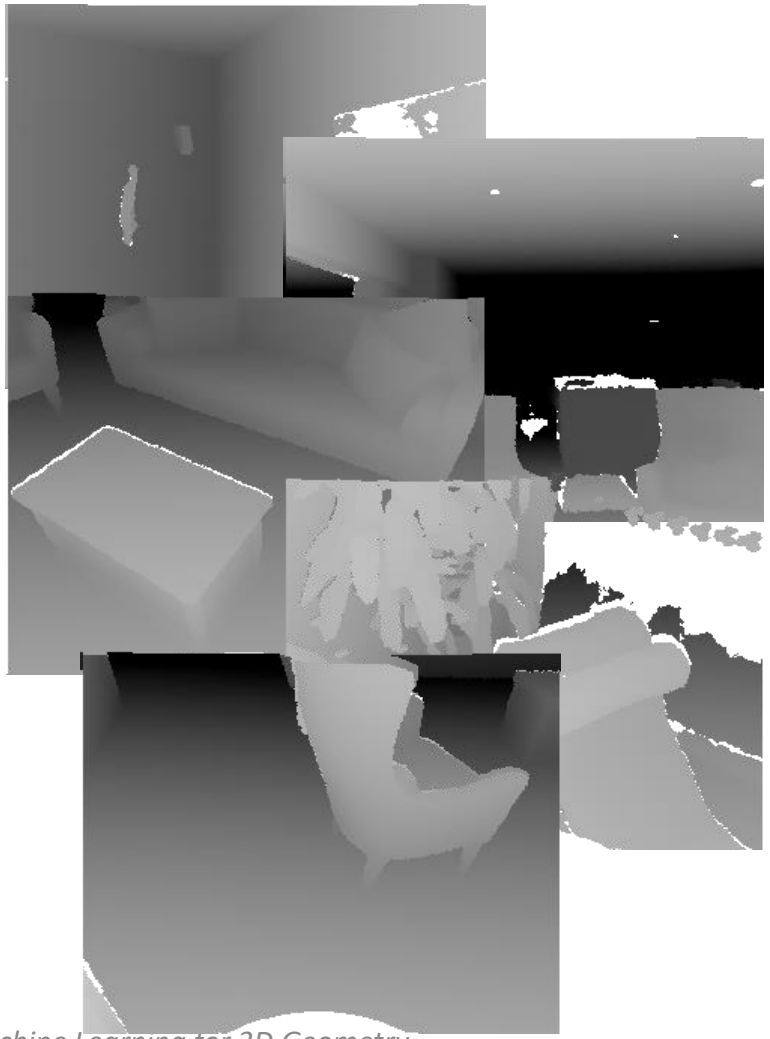
Depth Frames



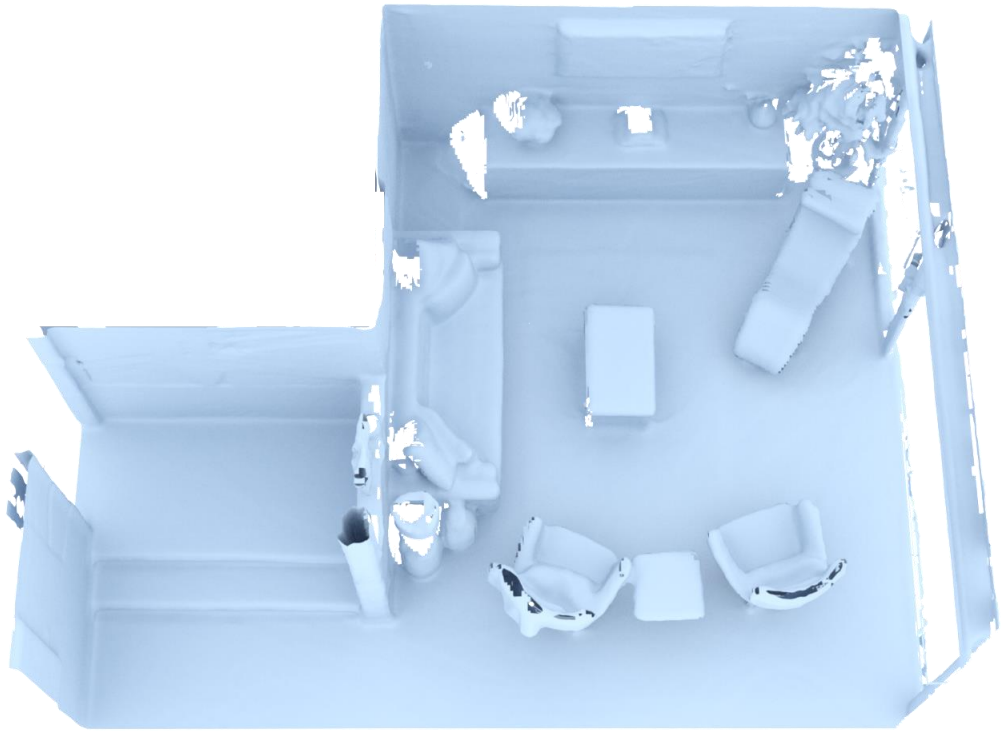
[Dai et al. '20]

SG-NN: Self-supervised Scan Completion

Depth Frames



Target Scan

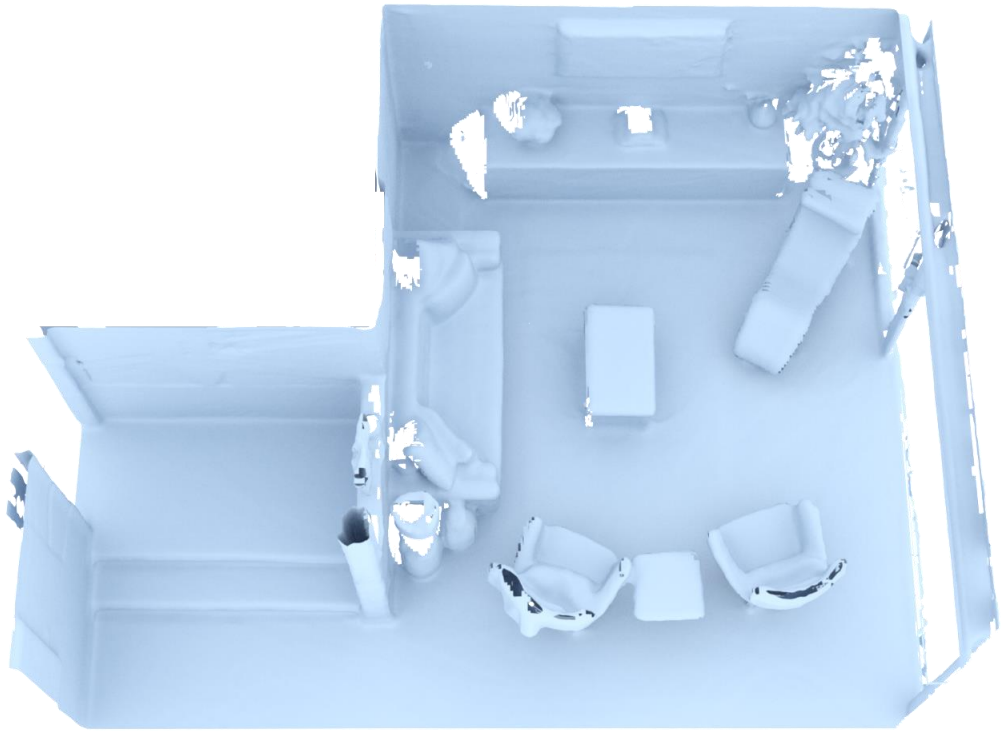


[Dai et al. '20]

SG-NN: Self-supervised Scan Completion

Depth Frames

Target Scan



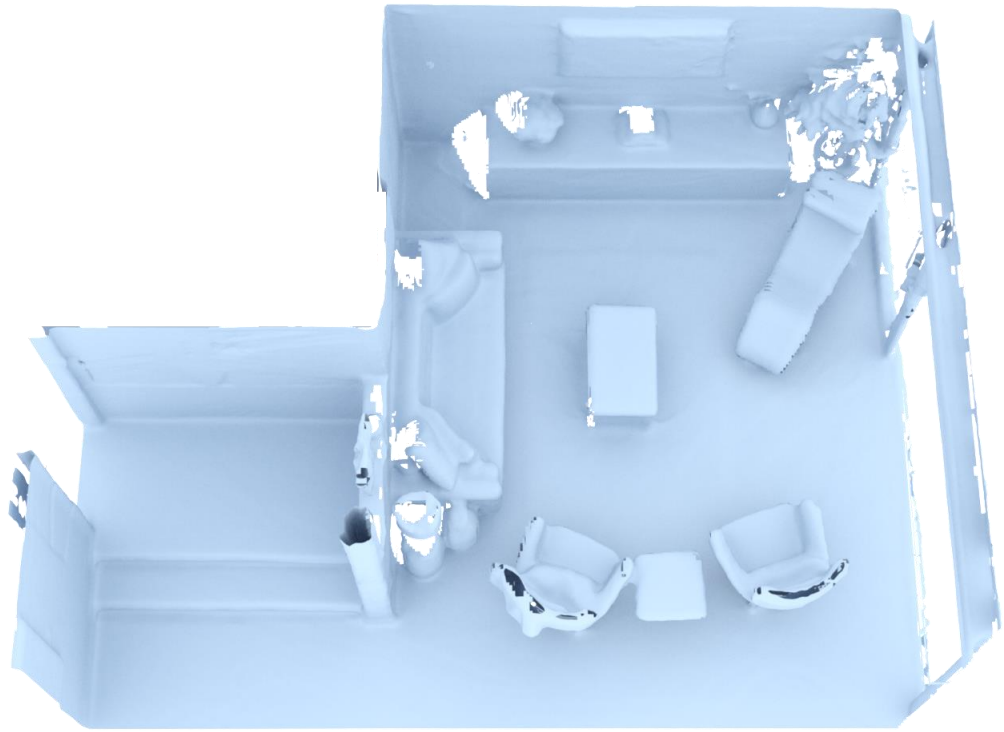
[Dai et al. '20]

SG-NN: Self-supervised Scan Completion

Depth Frames

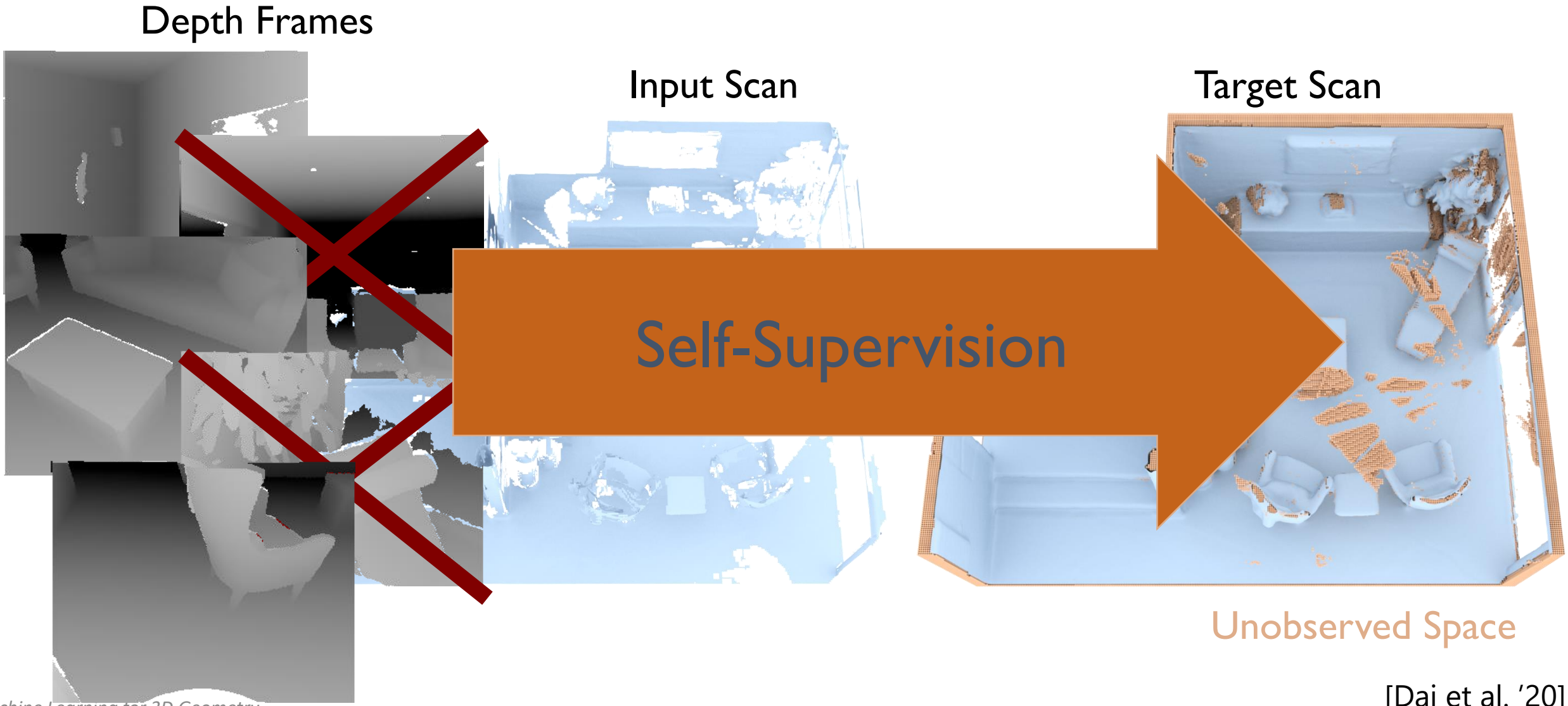
Input Scan

Target Scan



[Dai et al. '20]

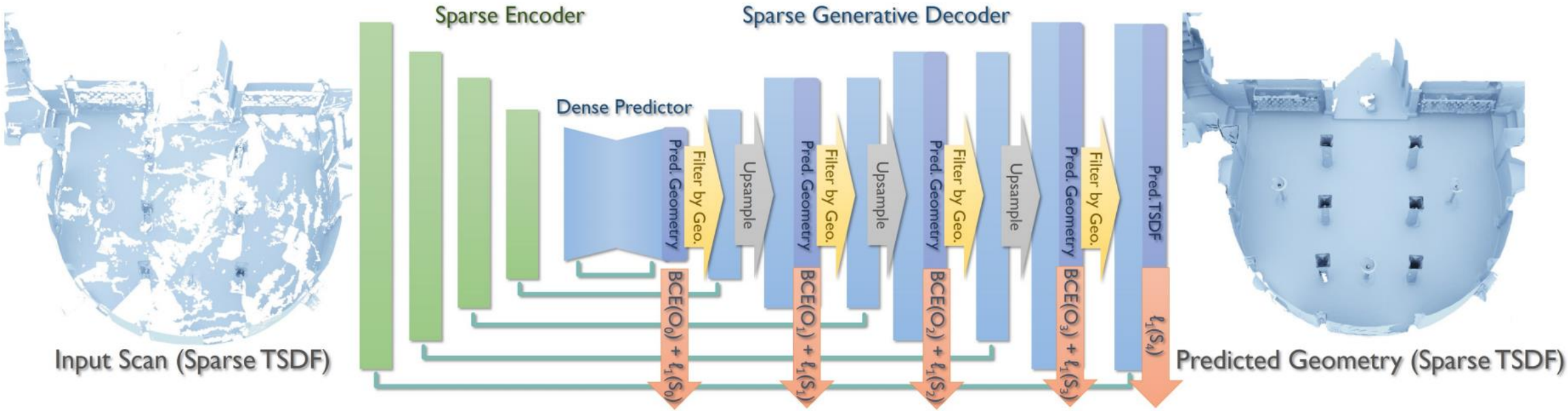
SG-NN: Self-supervised Scan Completion



[Dai et al. '20]

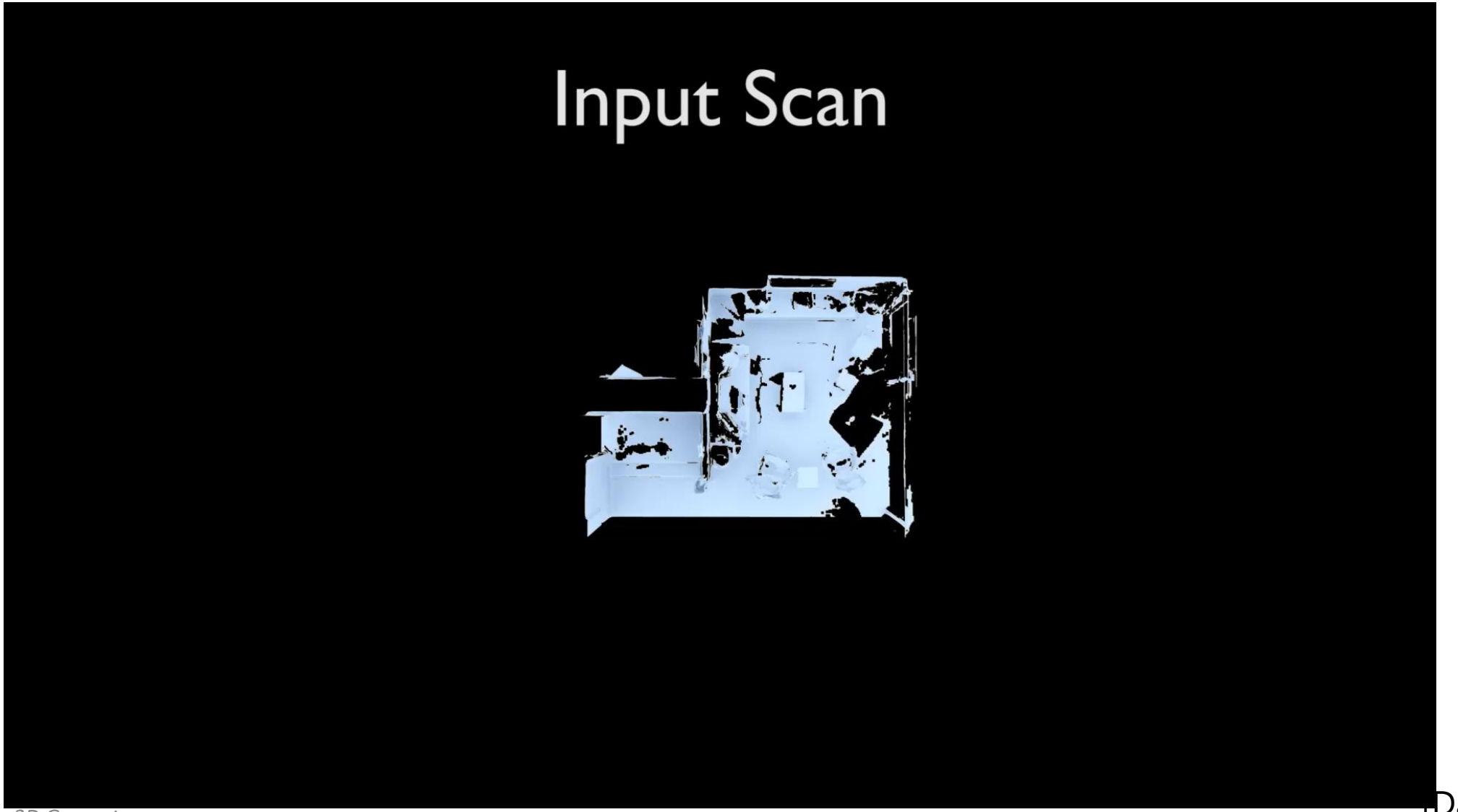
SG-NN: Self-supervised Scan Completion

- Generating high-resolution geometry



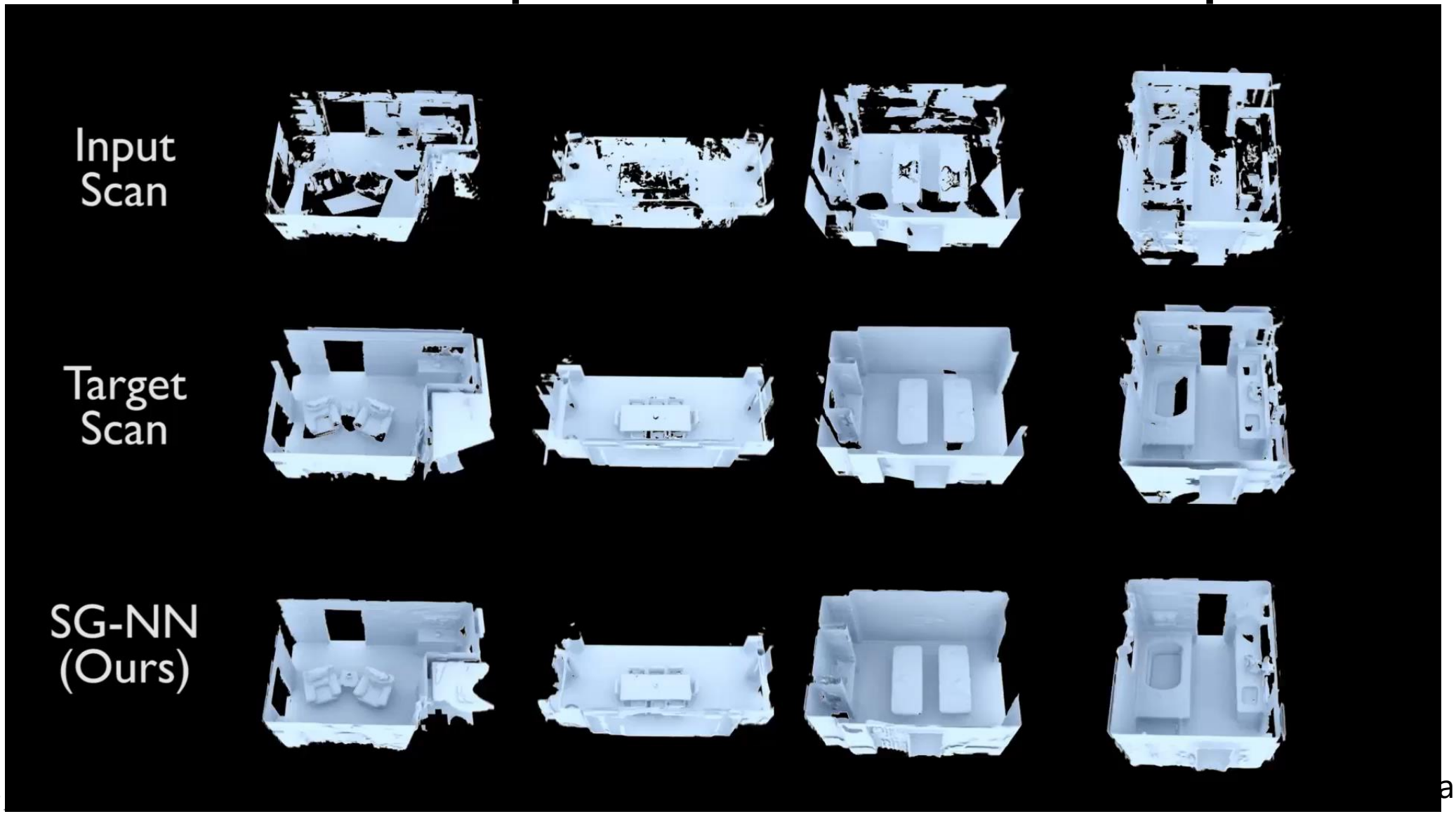
[Dai et al. '20]

SG-NN: Self-supervised Scan Completion



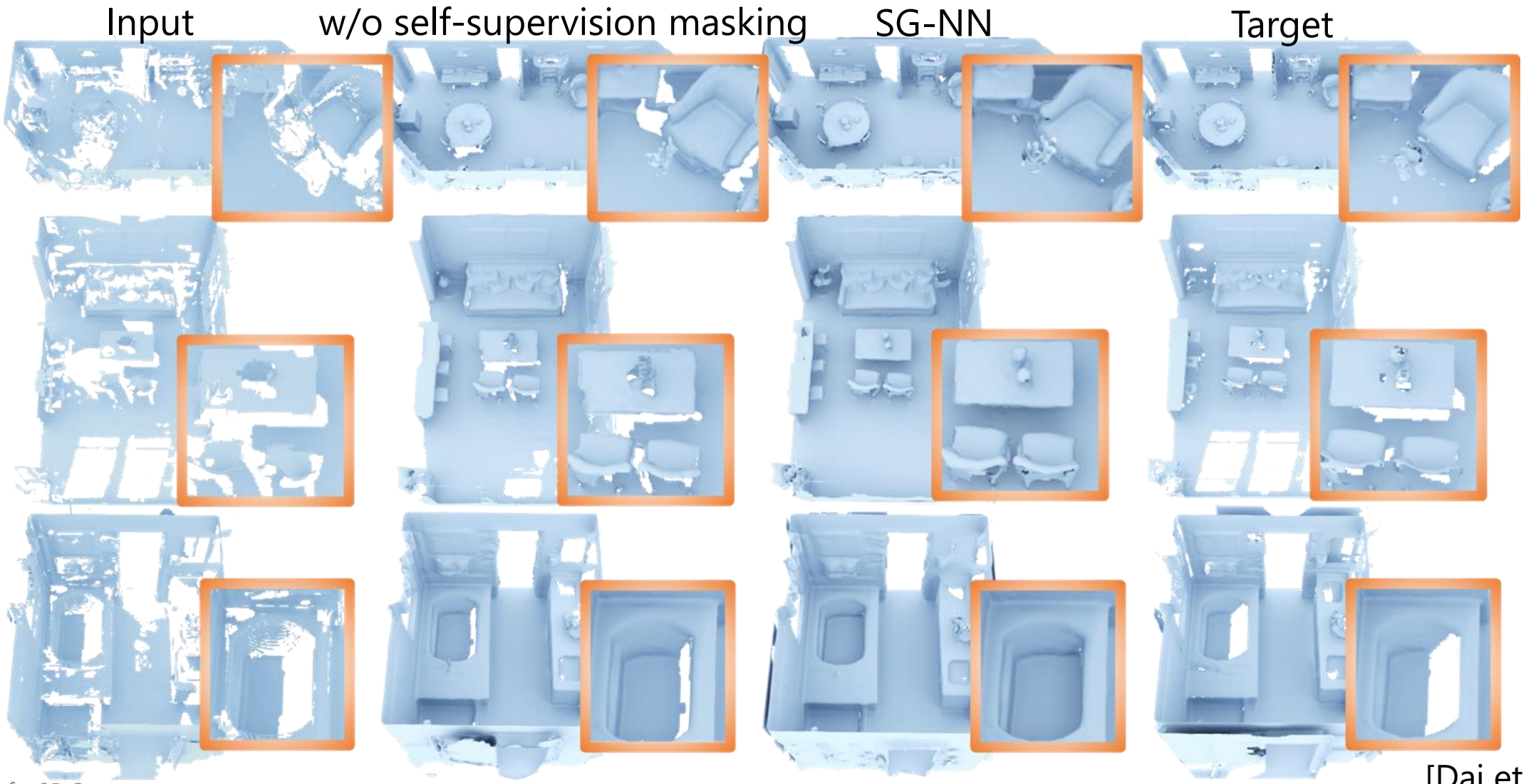
[Dai et al. '20]

SG-NN: Self-supervised Scan Completion



ai et al. ‘20]

SG-NN: Self-supervised Scan Completion



[Dai et al. '20]

SPSG: Self-supervised color generation

Input Scan

Predicted 3D Scene

[Dai et al. ’21]

SPSG: Self-supervised color generation

- Just add color?

Not enough!

[Dai et al. '21]

SPSG: Self-supervised color generation

Input Scan

Predicted 3D Scene

View-Guided Synthesis

→ 2D Reconstruction Loss

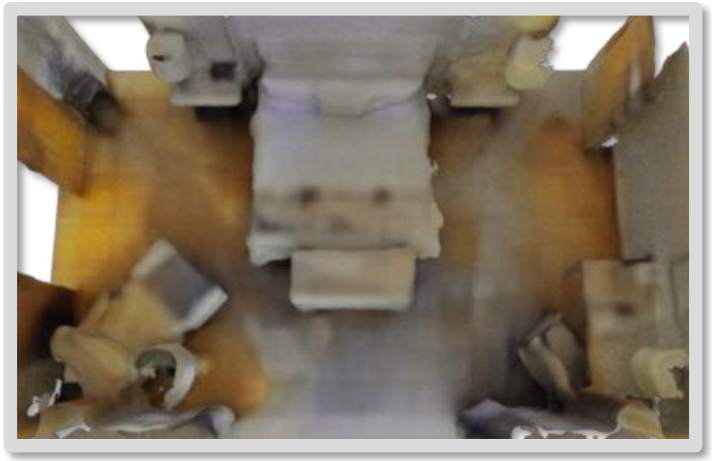
→ 2D Perceptual Loss

→ 2D Adversarial Loss

[Dai et al. '21]

SPSG: Self-supervised color generation

Input Scan



ℓ_1 only

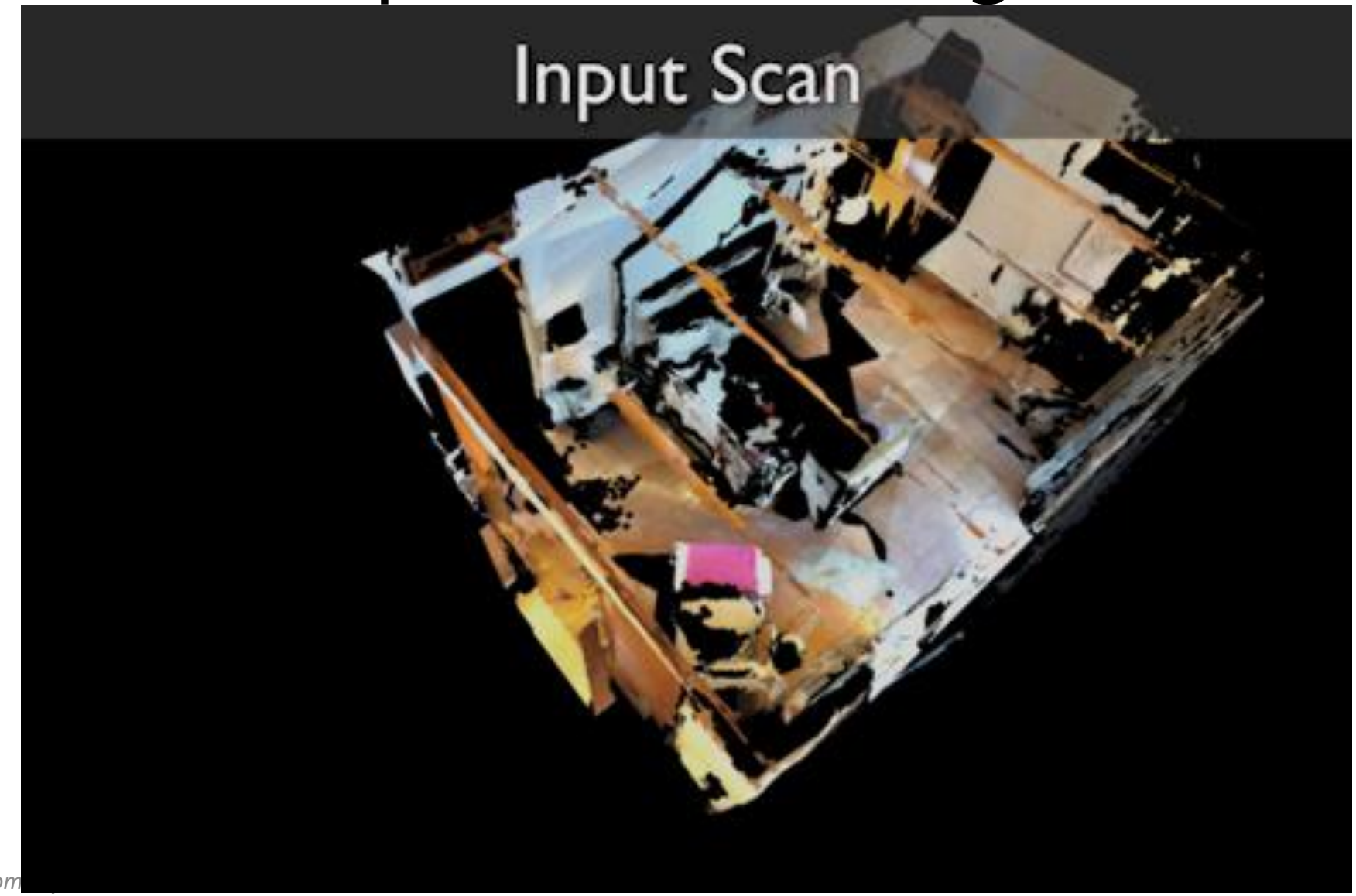
ℓ_1 + adversarial

ℓ_1 + perceptual

ℓ_1 + perceptual + adversarial

[Dai et al. '21]

SPSG: Self-supervised color generation



[Dai et al. ’21]

Leverage implicit reconstruction networks?

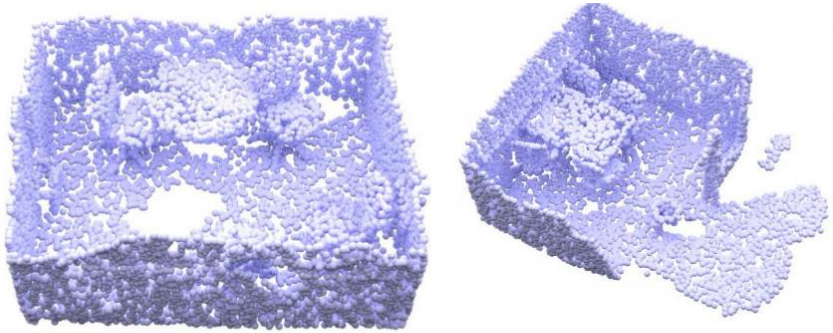
- Impressive performance on 3D shapes – what about scenes?

Occupancy Networks on shapes:

[Mescheder et al. ’19]

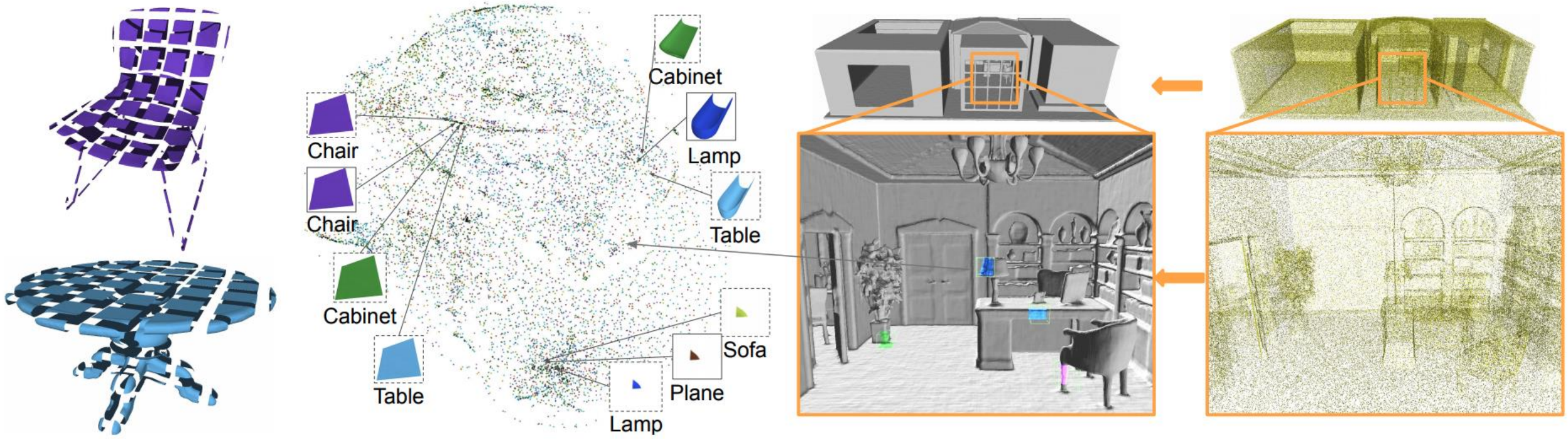
Occupancy Networks on scenes:

Input Point Cloud



ONet Reconstruction

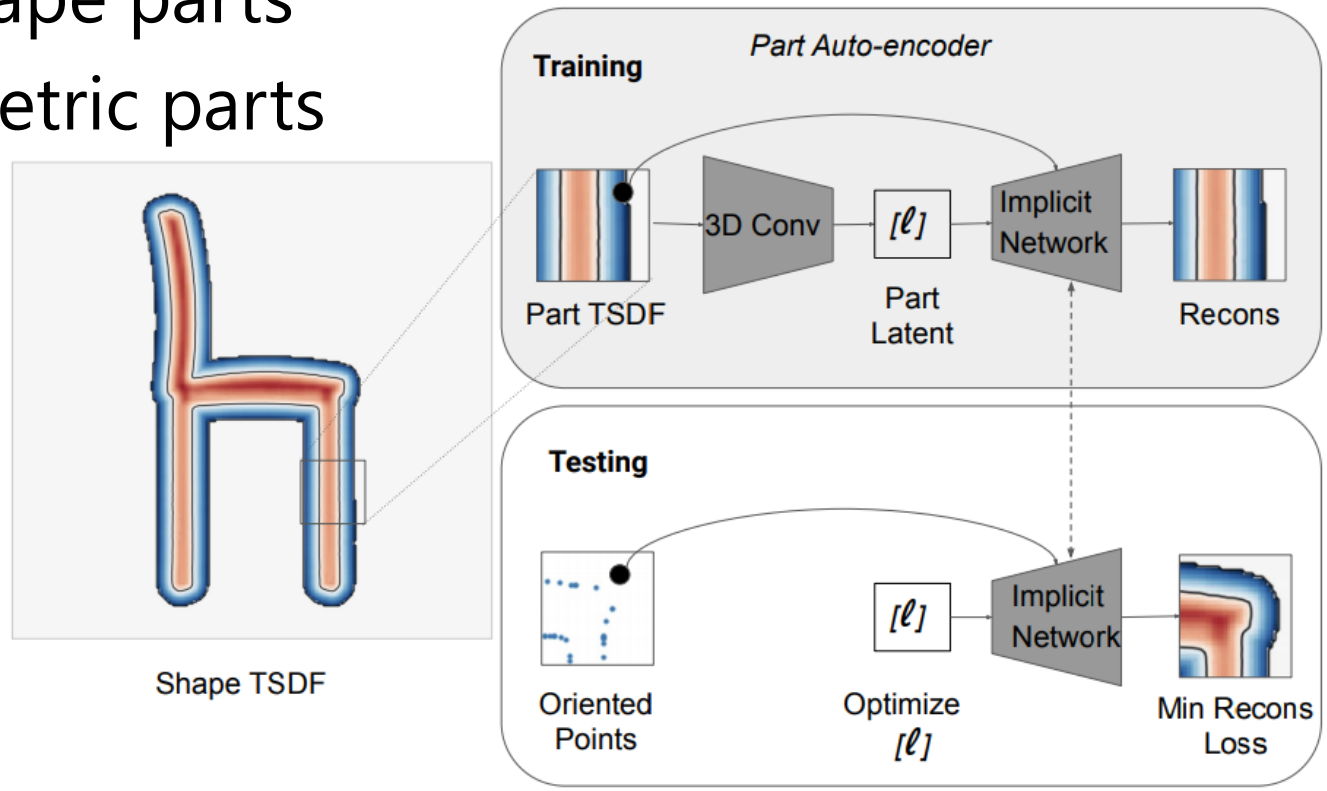
Local Implicit Grids



[Jiang et al. '20]

Local Implicit Grids

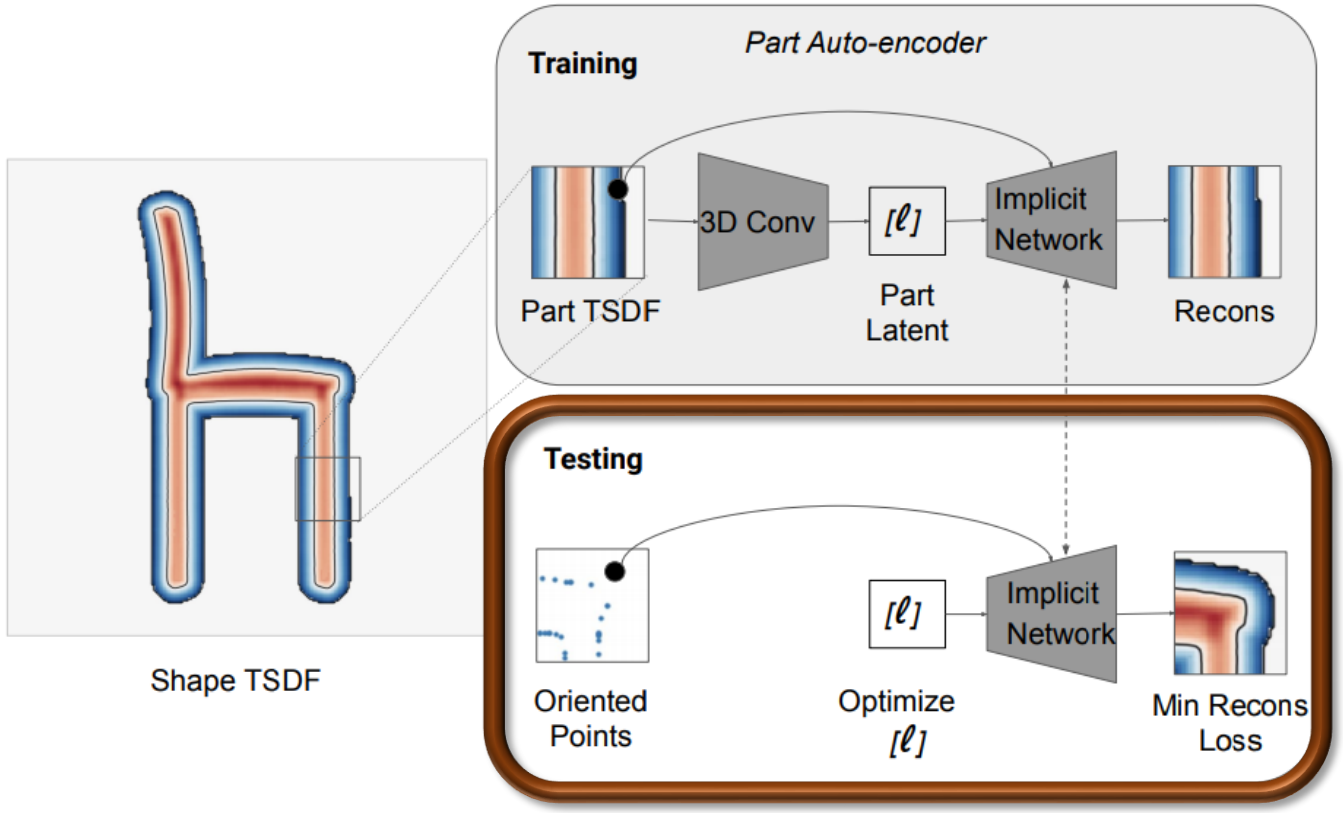
- Learn geometric priors from shapes
- Construct embedding of shape parts
- Exploit similar shared geometric parts across shapes and scenes



[Jiang et al. '20]

Local Implicit Grids

- Test time: sparse point cloud input
- Decompose into a coarse grid, each grid cell has set of points
- Use trained decoder (fixed), and optimize for latent code which best fits the observed points

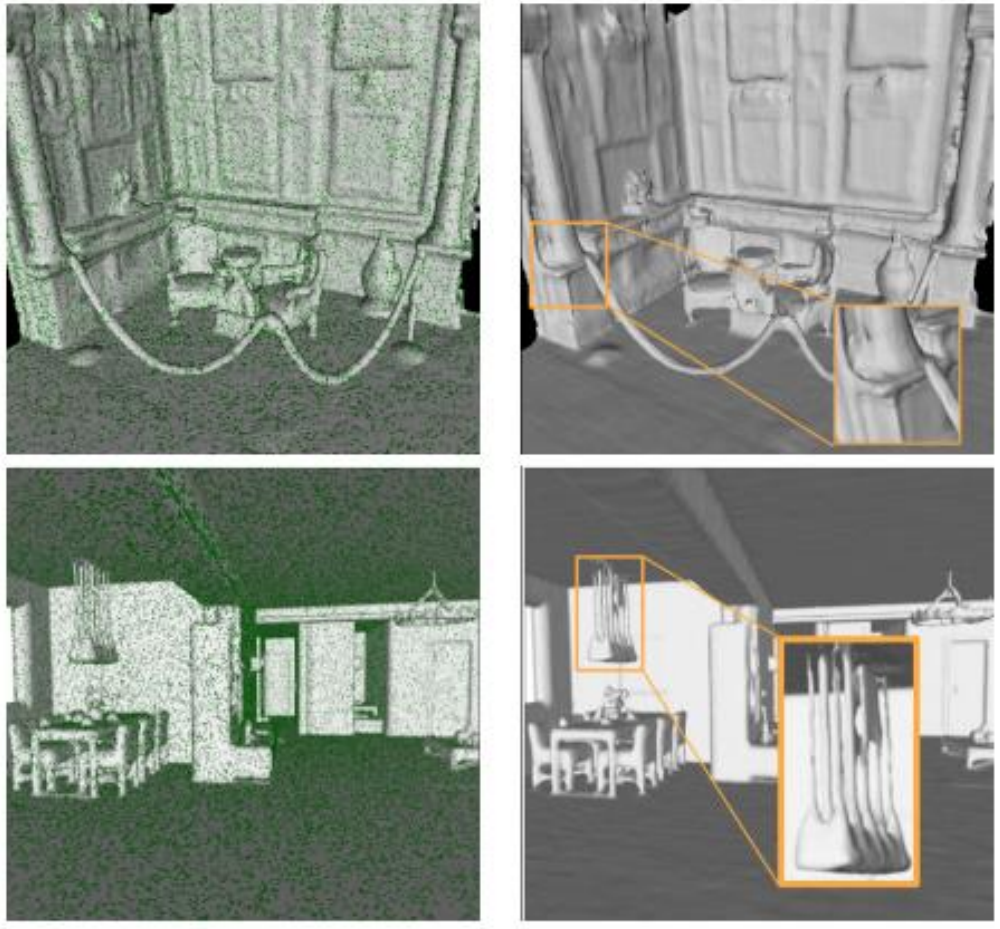


[Jiang et al. '20]

Local Implicit Grids

- Scene reconstruction from points

Input points (overlaid with ground truth scene geometry)

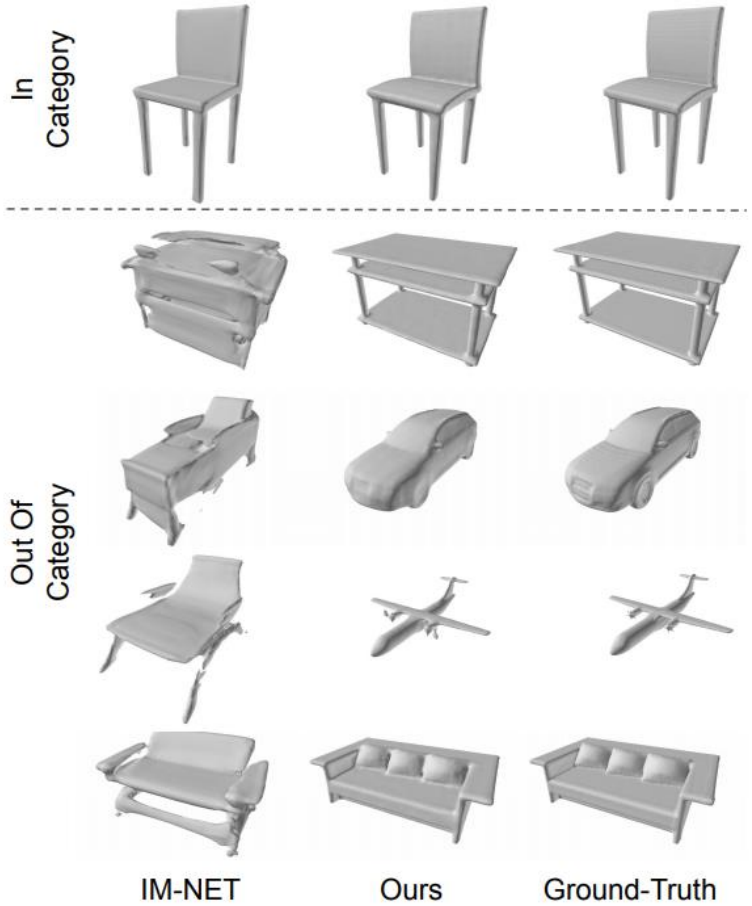


Local implicit grid reconstruction

[Jiang et al. ‘20]

Local Implicit Grids

- Part geometric priors can enable broader generalizability



[Jiang et al. '20]

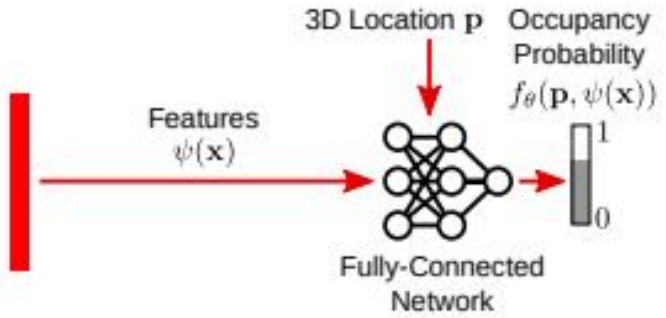
Convolutional Occupancy Networks

- Hybrid convolutional + deep implicit 3D representation
- Leverage translational invariance of convolutions
- Leverage object detail representation of deep implicit representations

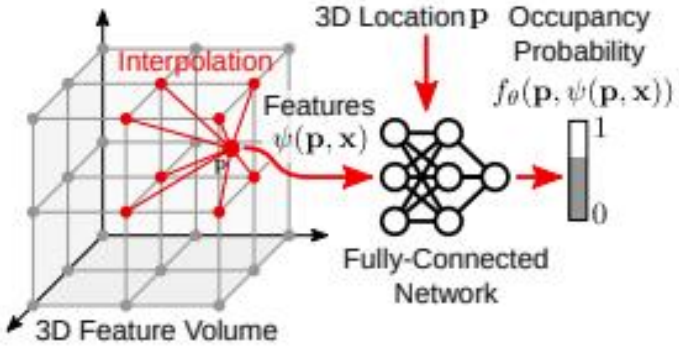
[Peng et al. ’20]

Convolutional Occupancy Networks

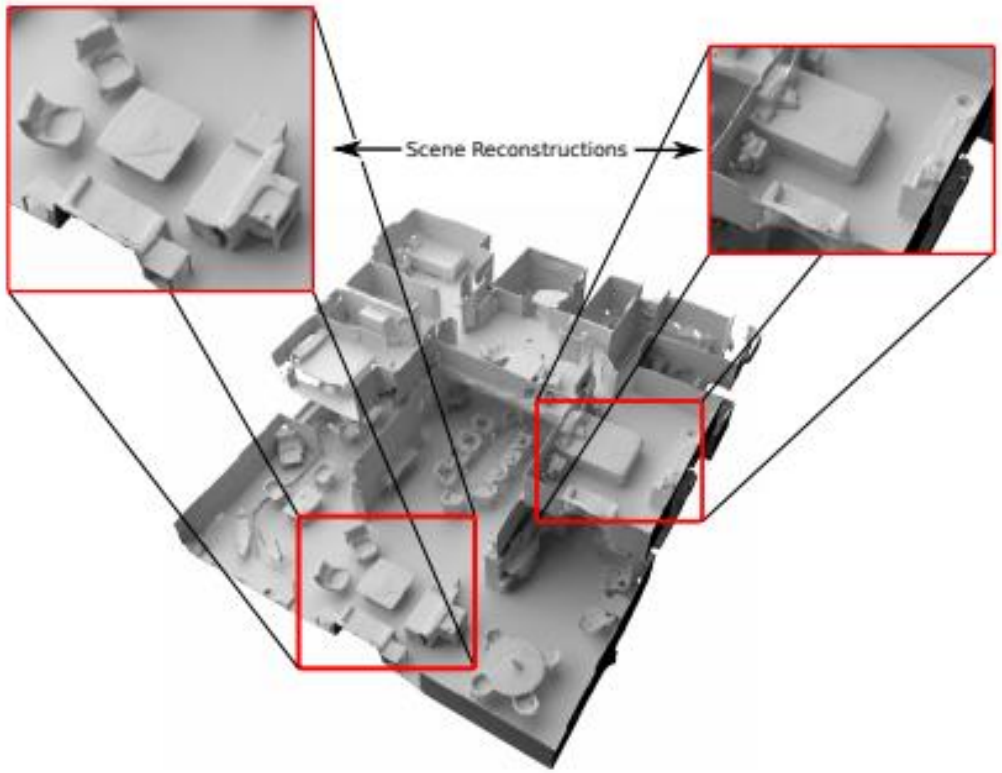
- Hybrid convolutional + deep implicit 3D representation



(a) Occupancy Network [26]



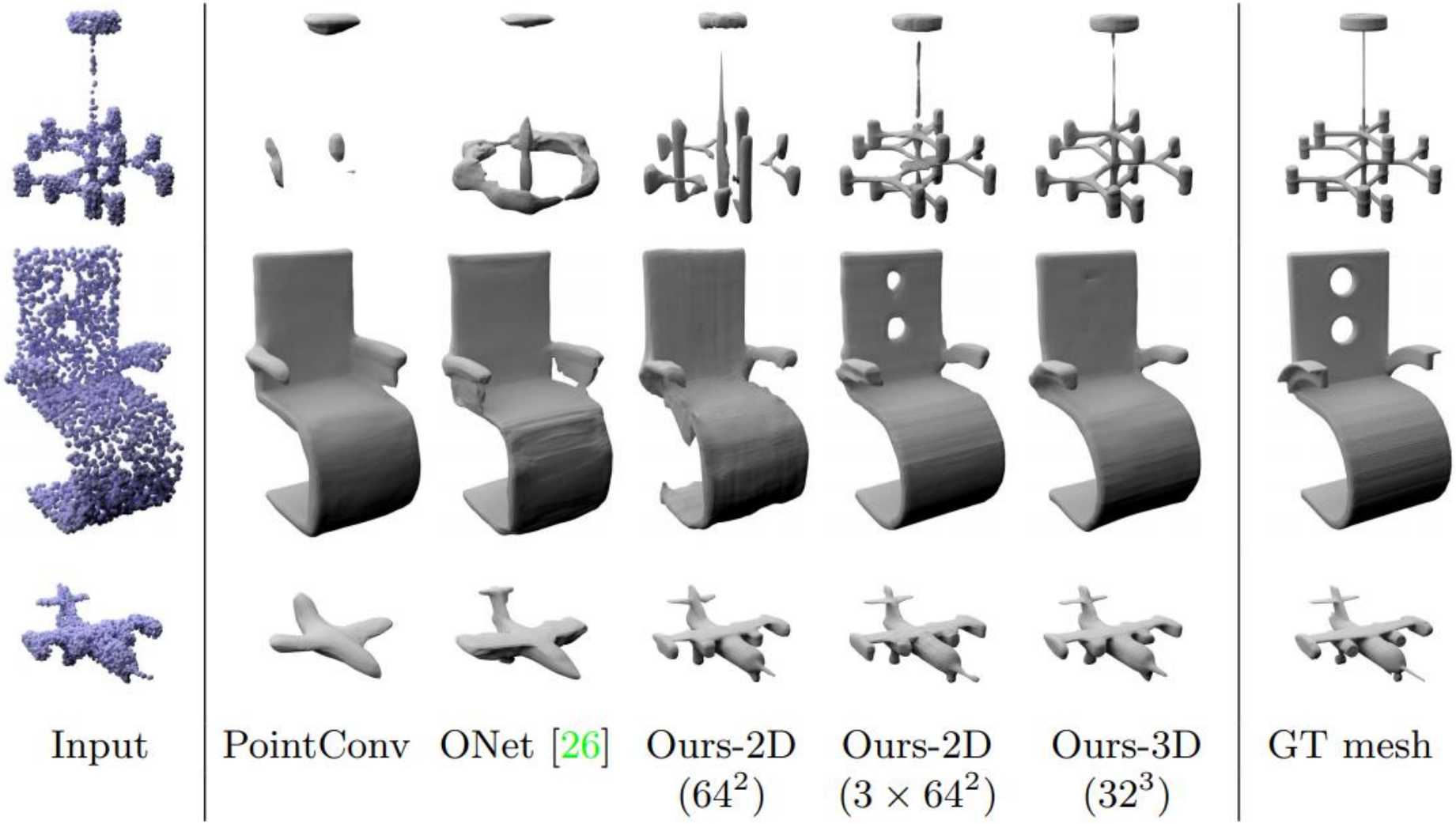
(b) Conv. Occupancy Network



(c) Reconstruction on Matterport3D [1]

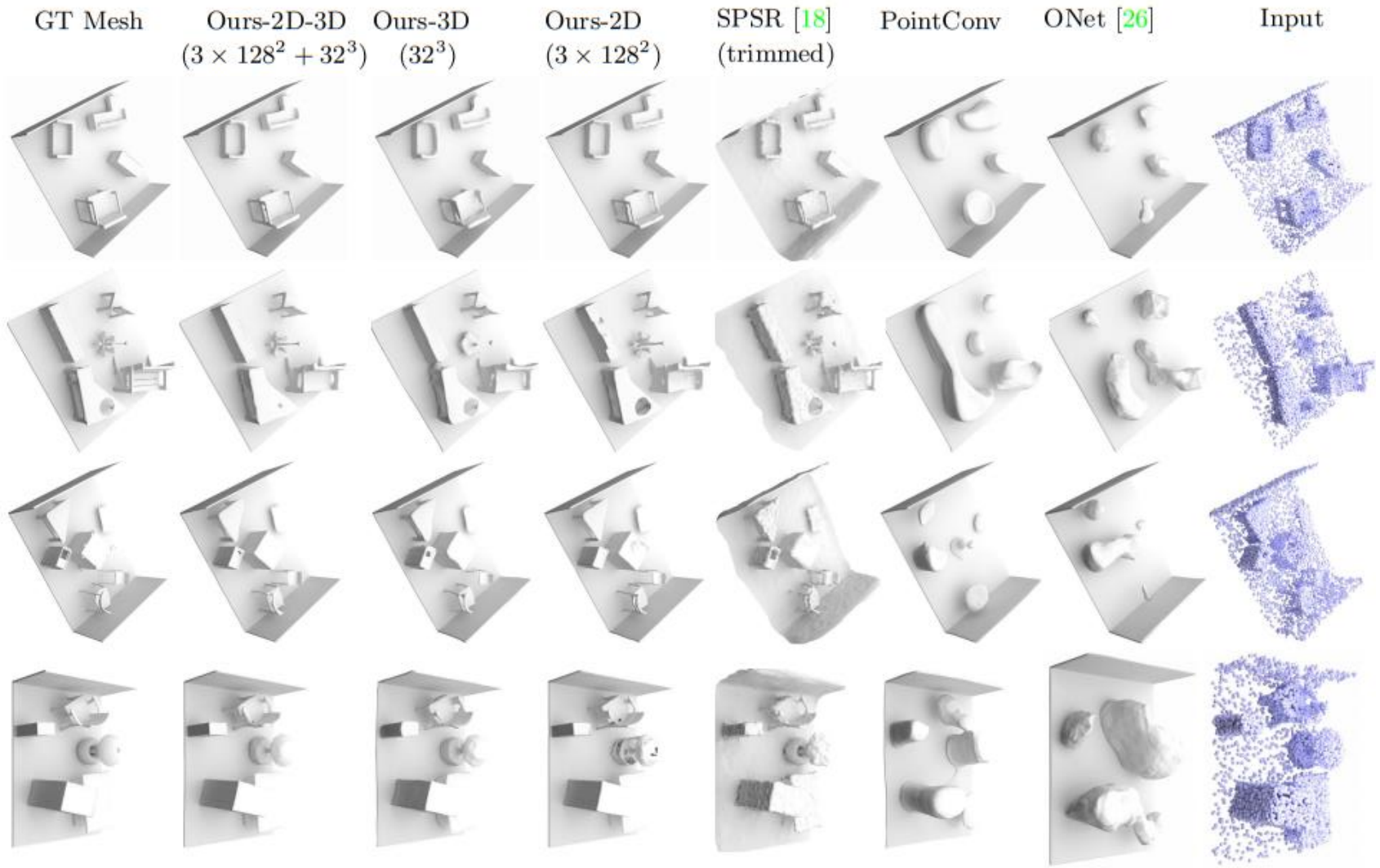
[Peng et al. '20]

Convolutional Occupancy Networks



[Peng et al. '20]

Convolutional Occupancy Networks



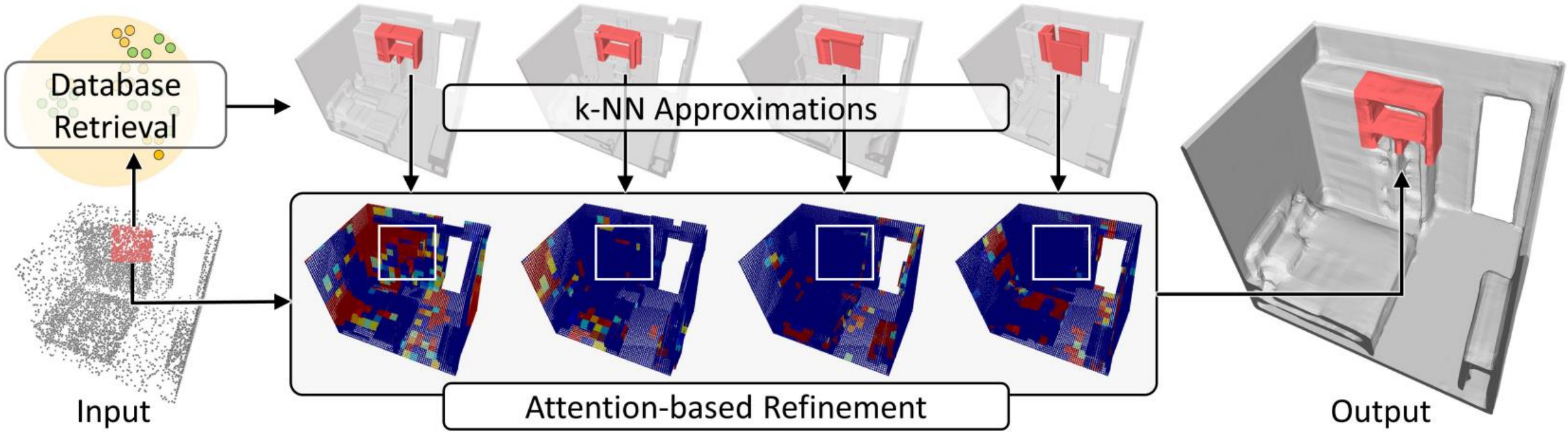
[Peng et al. '20]

RetrievalFuse

- Learn to exploit dictionary of training data *at test time*
- Most generative 3d approaches: learn to encode information from train data during training; discard train data at test time
- Instead: create initial reconstruction estimate as a composite of train geometry chunks

[Siddiqui et al. ’21]

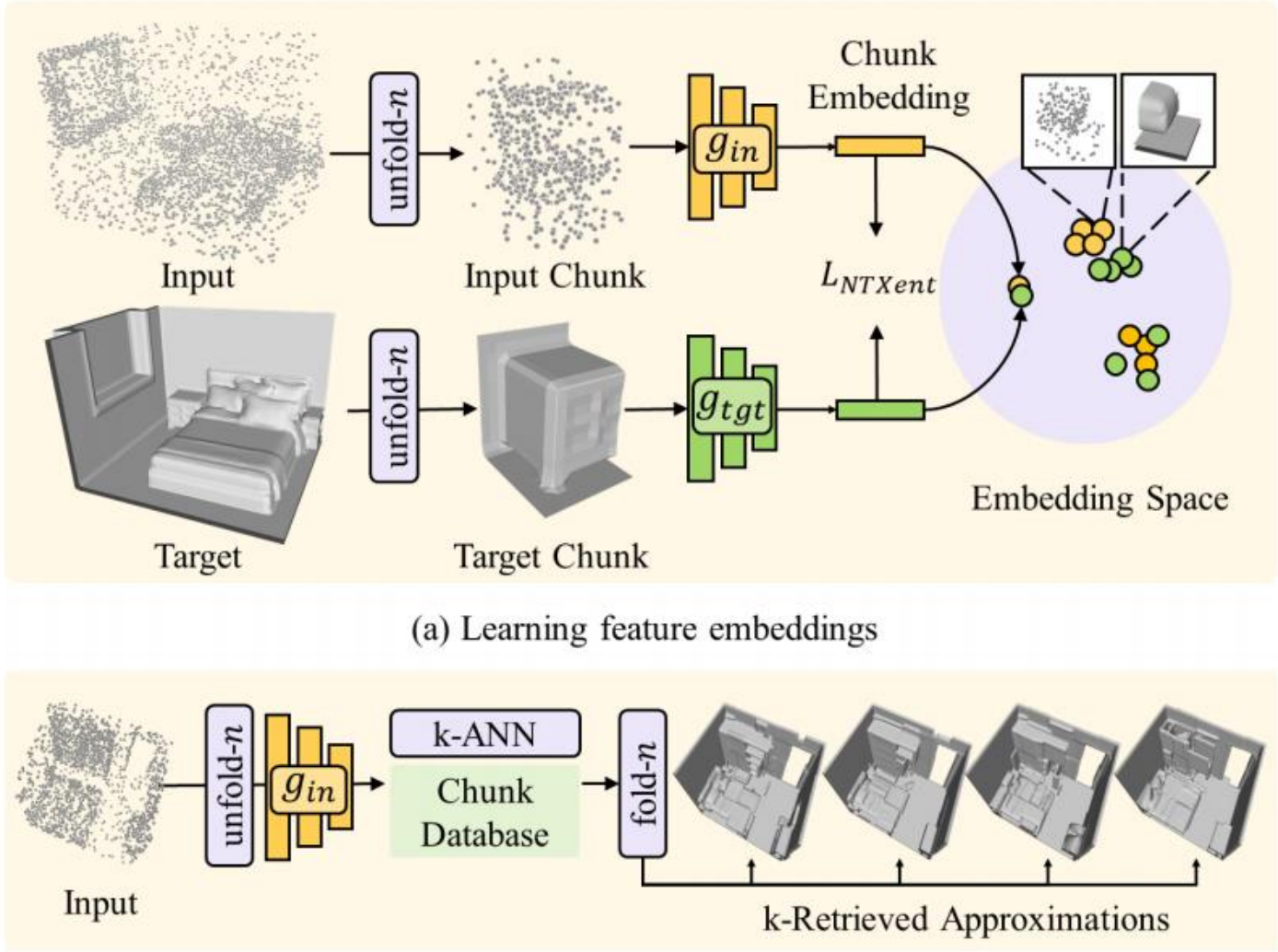
RetrievalFuse



[Siddiqui et al. '21]

RetrievalFuse

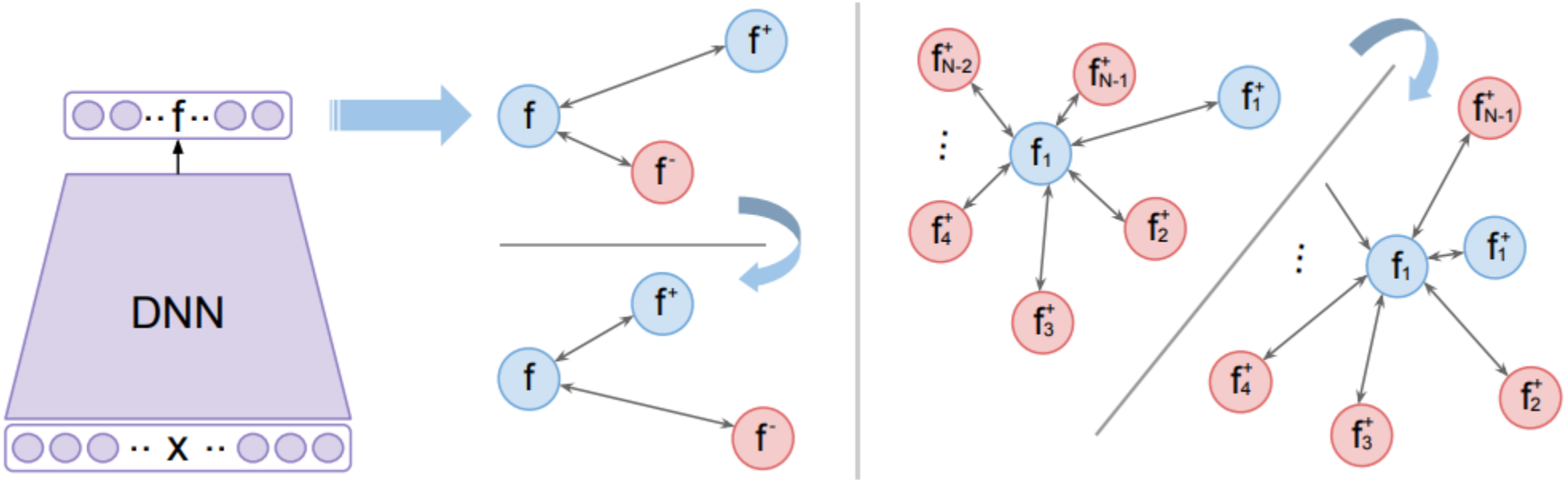
Train chunk-based reconstructions



[Siddiqui et al. '21]

Deep Metric Learning

- Embedding space construction with deep metric learning



$$L = -\log \frac{\exp\left(\frac{\mathrm{sim}(z_i, z_j)}{\tau}\right)}{\sum_{k=1}^{N} \mathbb{I}_{[k \neq i]} \exp\left(\frac{\mathrm{sim}(z_i, z_k)}{\tau}\right)}$$

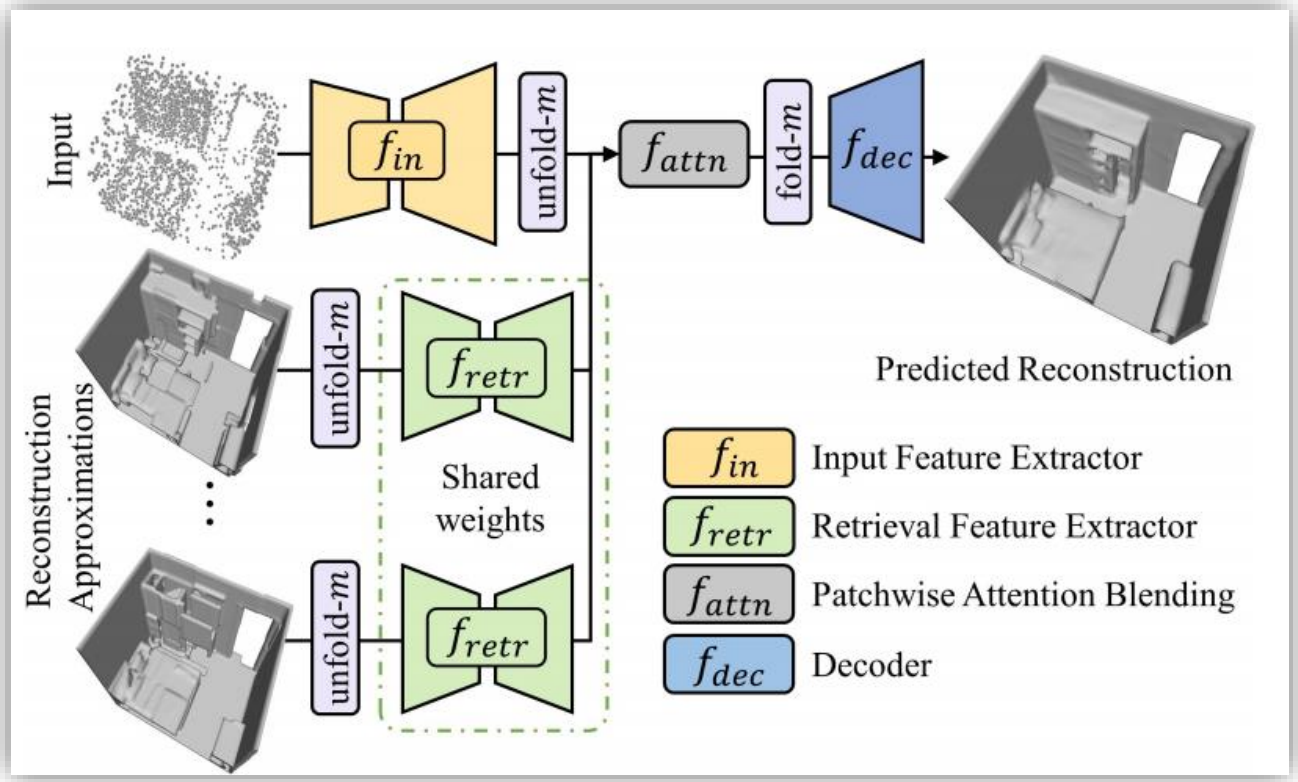
$\mathrm{sim}(u, v)$: dot product similarity

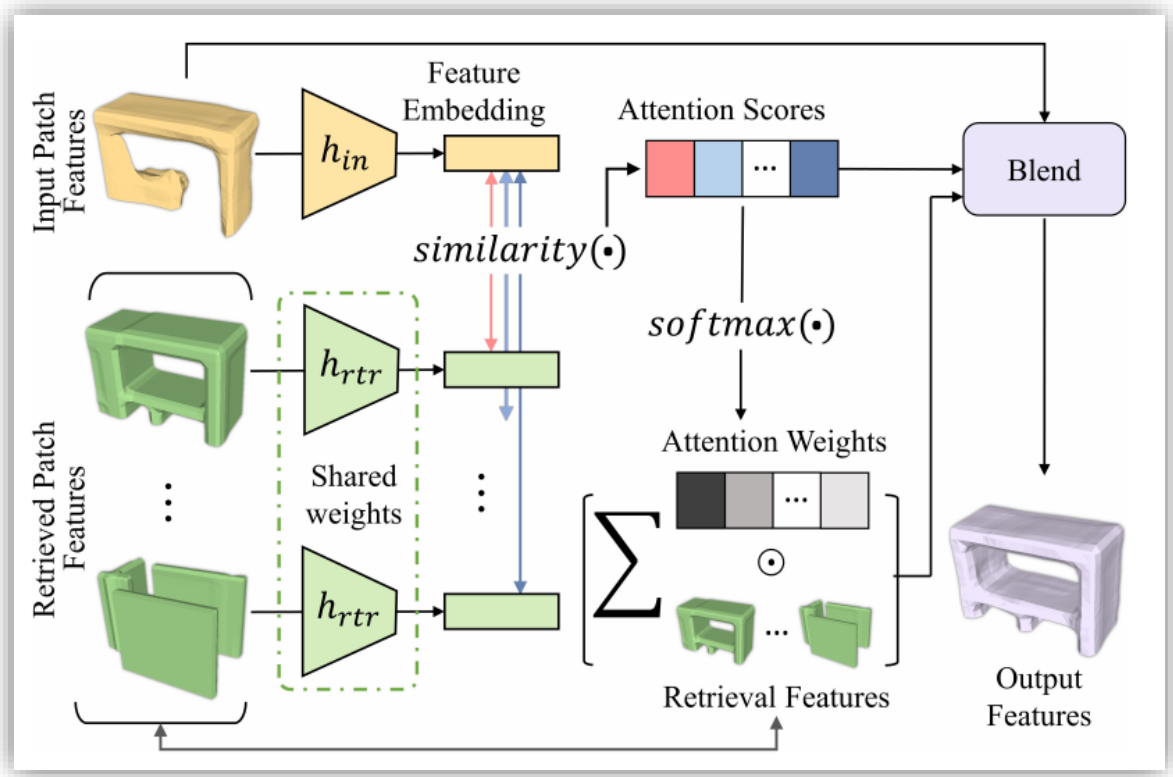
$\mathbb{I}_{[k \neq i]} \in \{0,1\}$: 1 iff $k \neq i$

τ: temperature

[Sohn et al. '16]

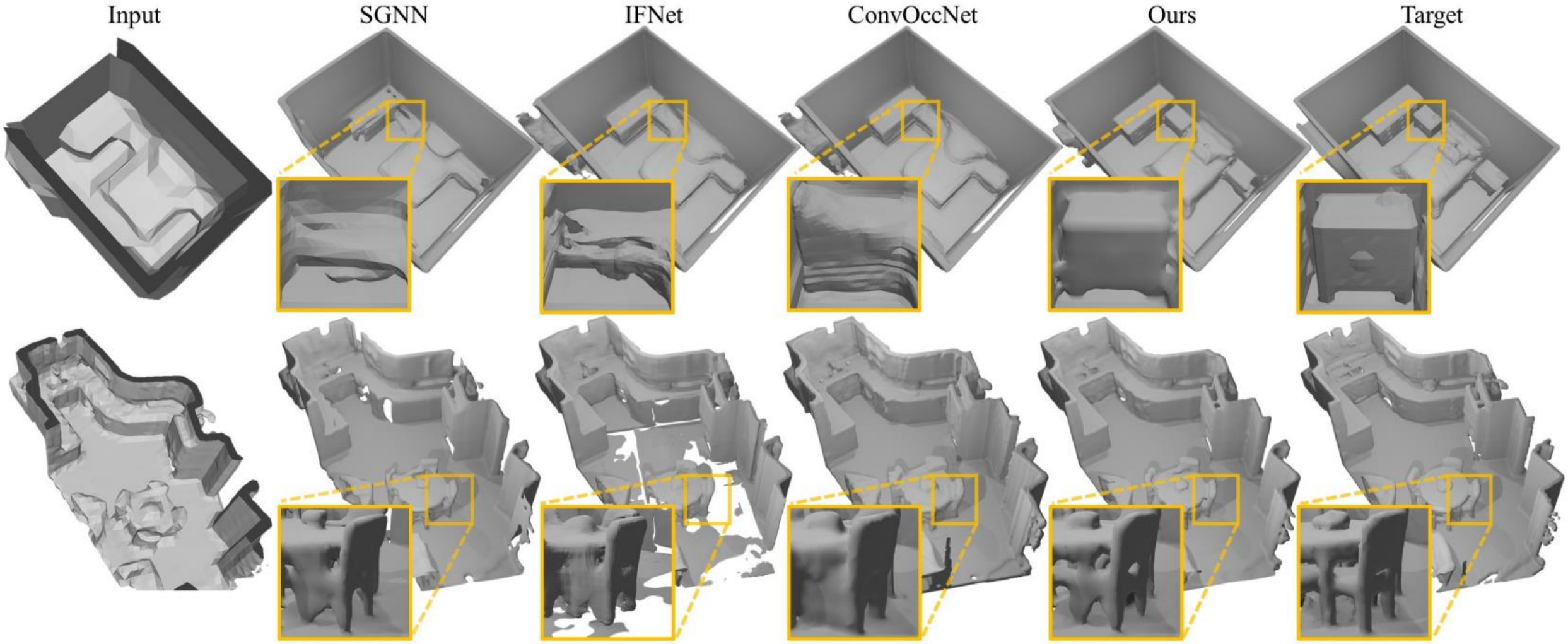
RetrievalFuse





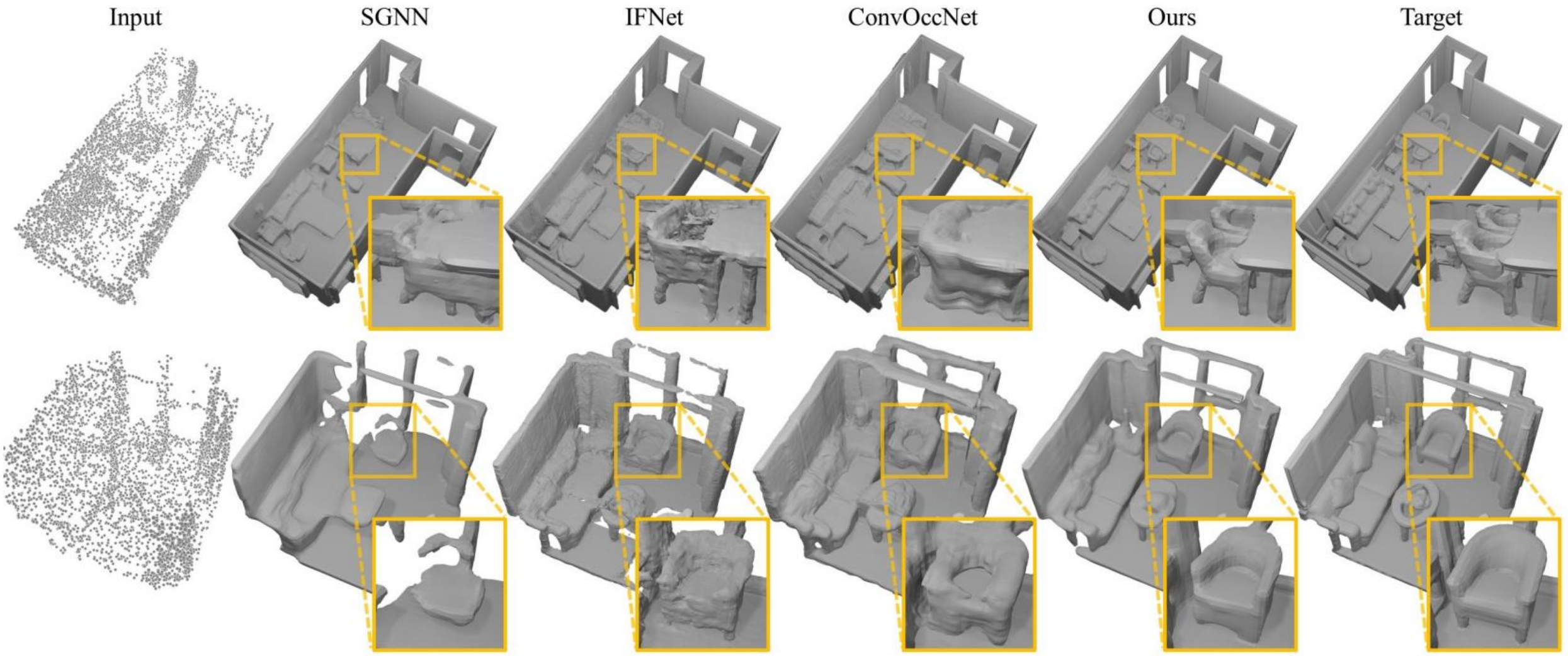
[Siddiqui et al. '21]

RetrievalFuse



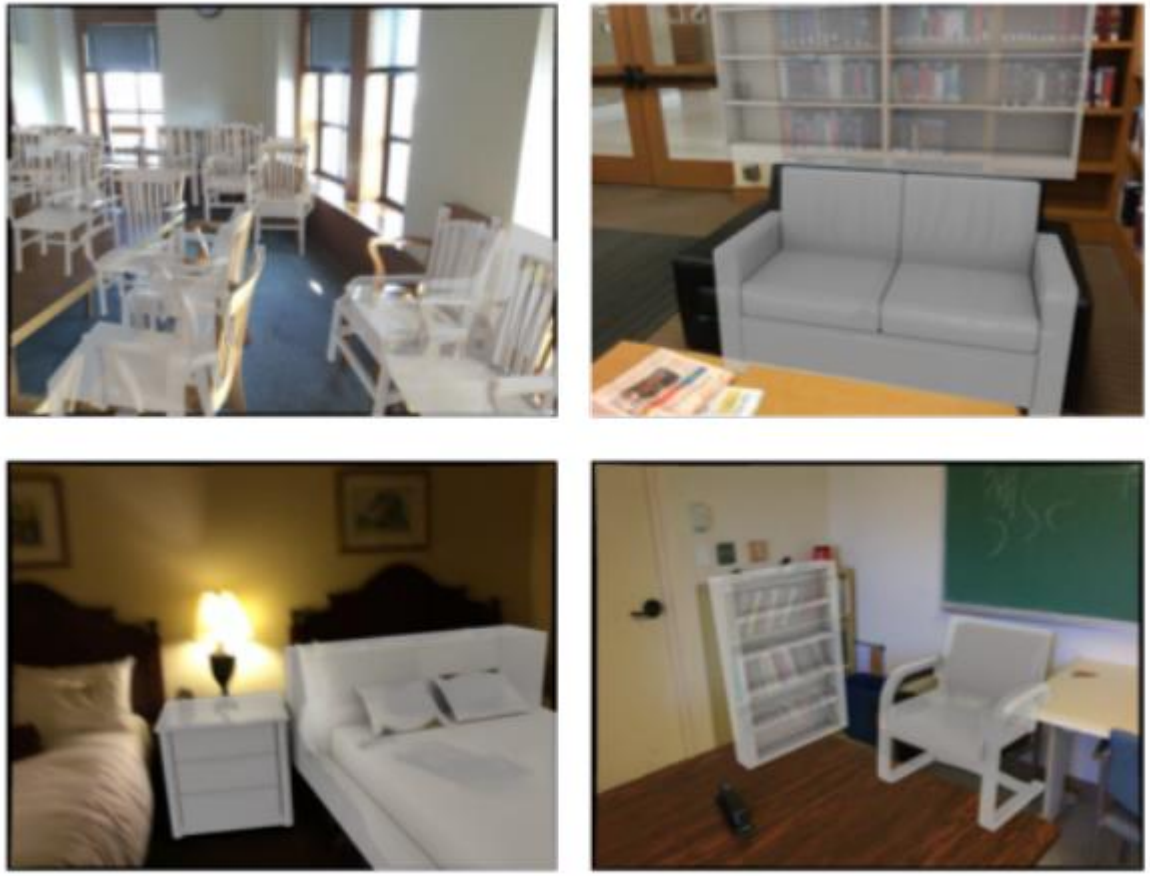
[Siddiqui et al. '21]

RetrievalFuse

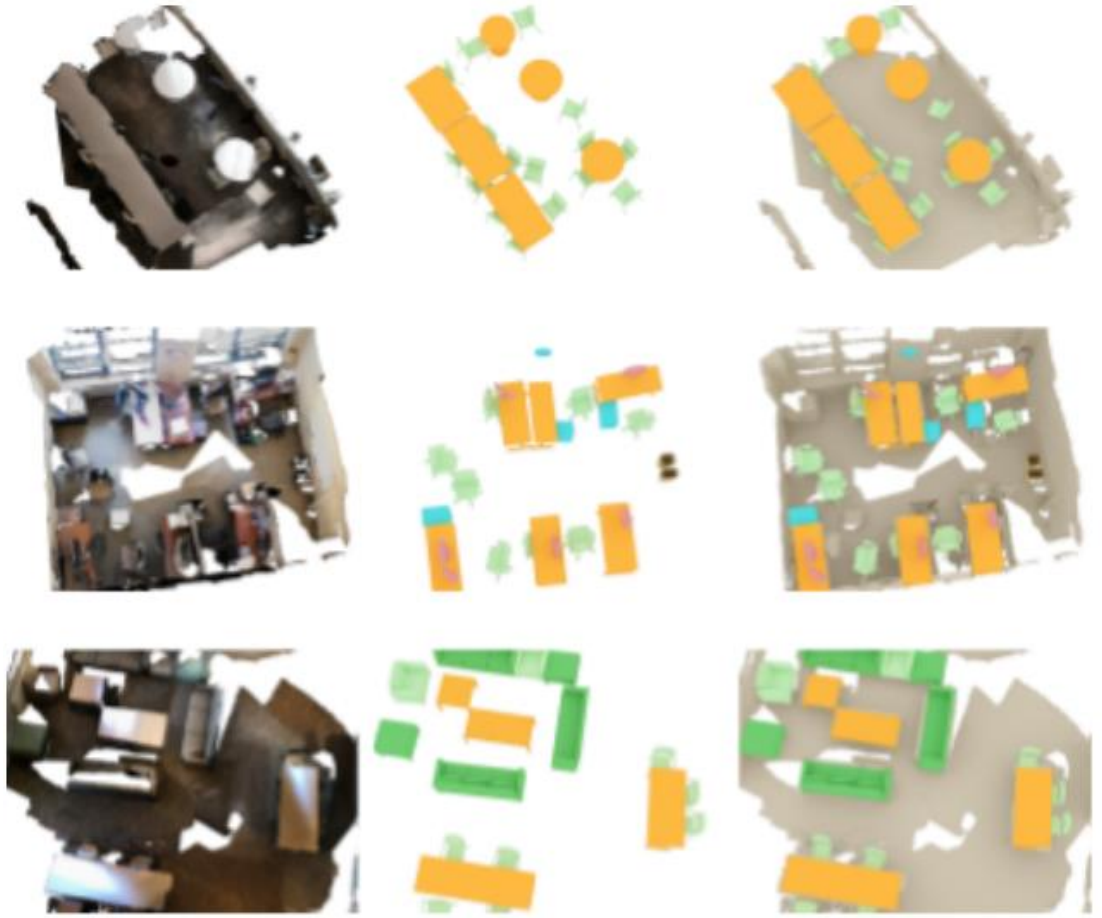


[Siddiqui et al. '21]

Retrieval as Reconstruction?



[Kuo et al. ‘20]



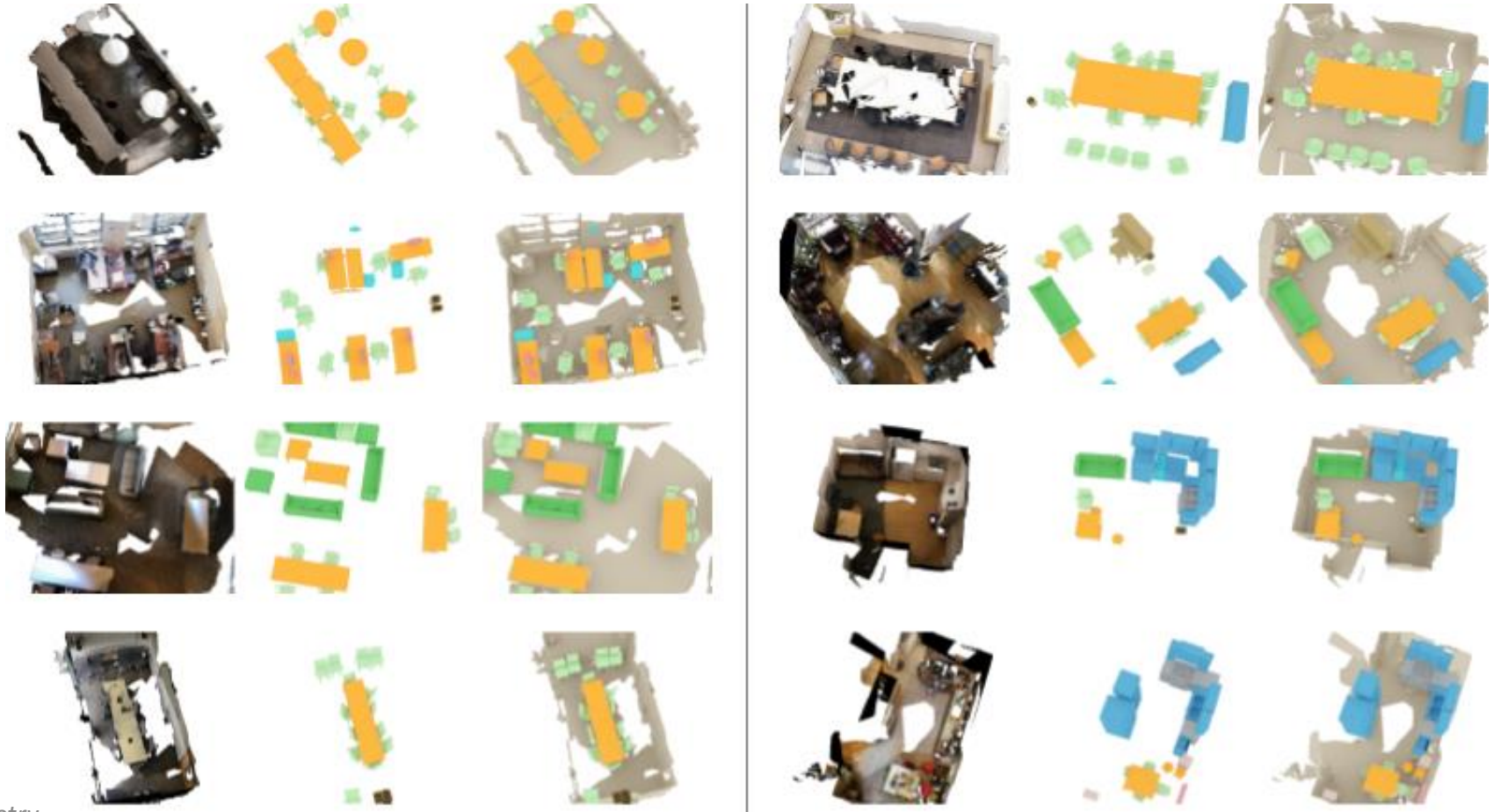
[Avetisyan et al. ‘19]

Retrieval as Reconstruction?

- Object primitives: if database objects are all physically representative, can guarantee for all object retrievals for reconstruction
- Potential to leverage database information for each retrieved object (e.g., material annotations, part segmentations, etc)
- In practice: no exact geometric matches

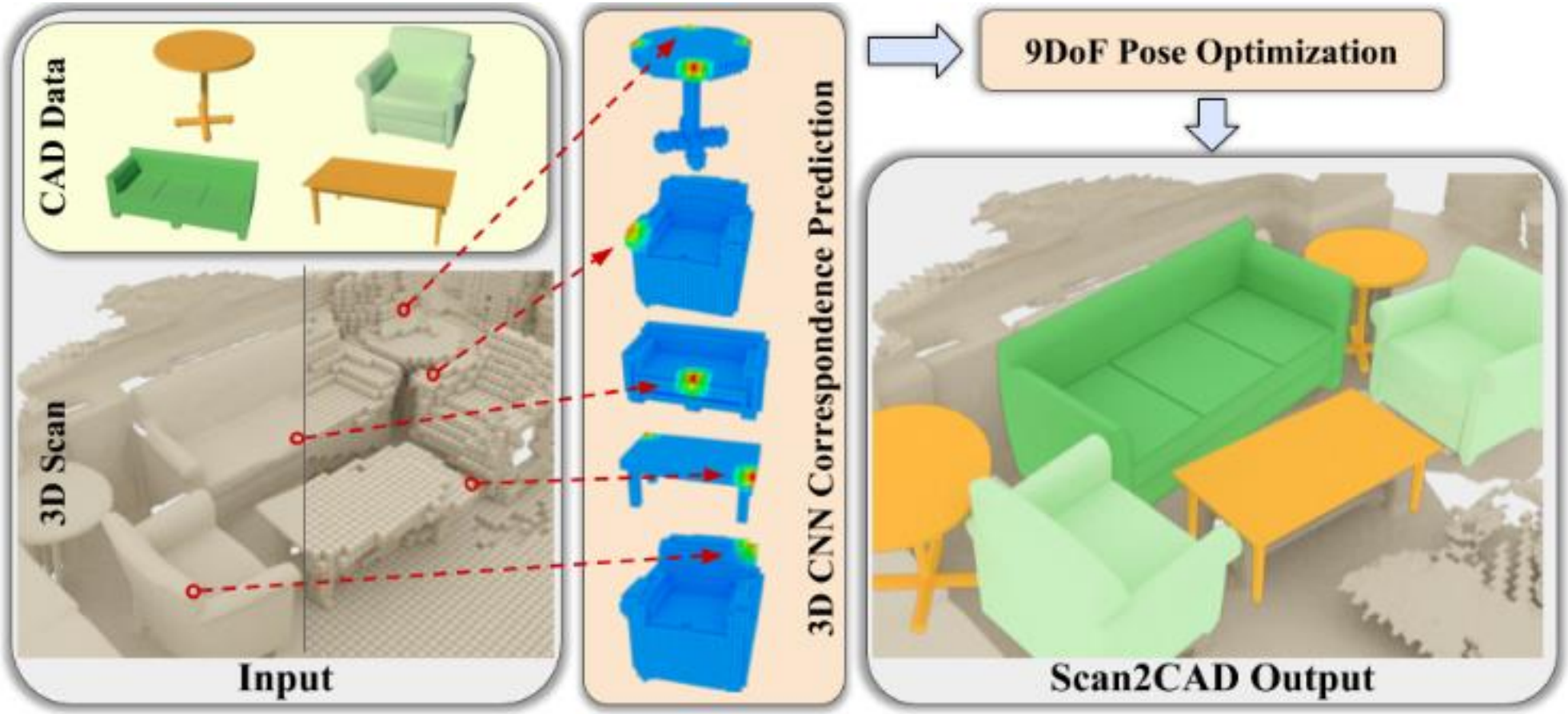
Scan2CAD

- Dataset of CAD models (ShapeNet) aligned to real scans (ScanNet)



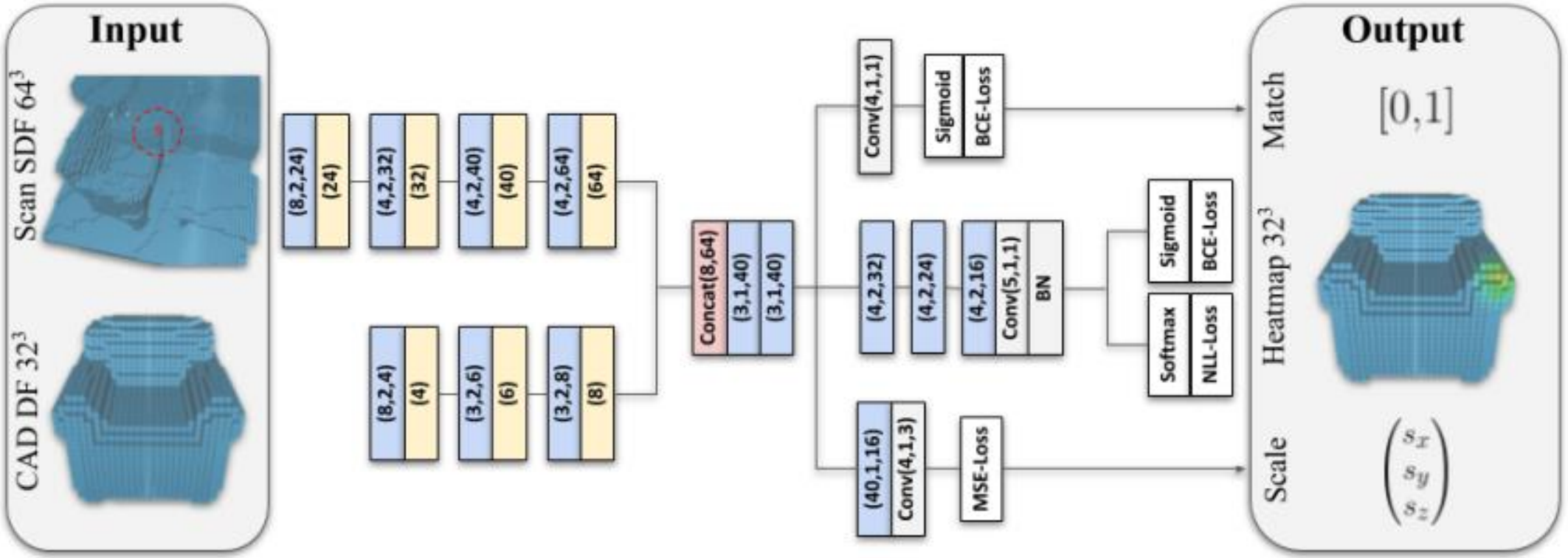
[Avetisyan et al. ’19]

Scan2CAD



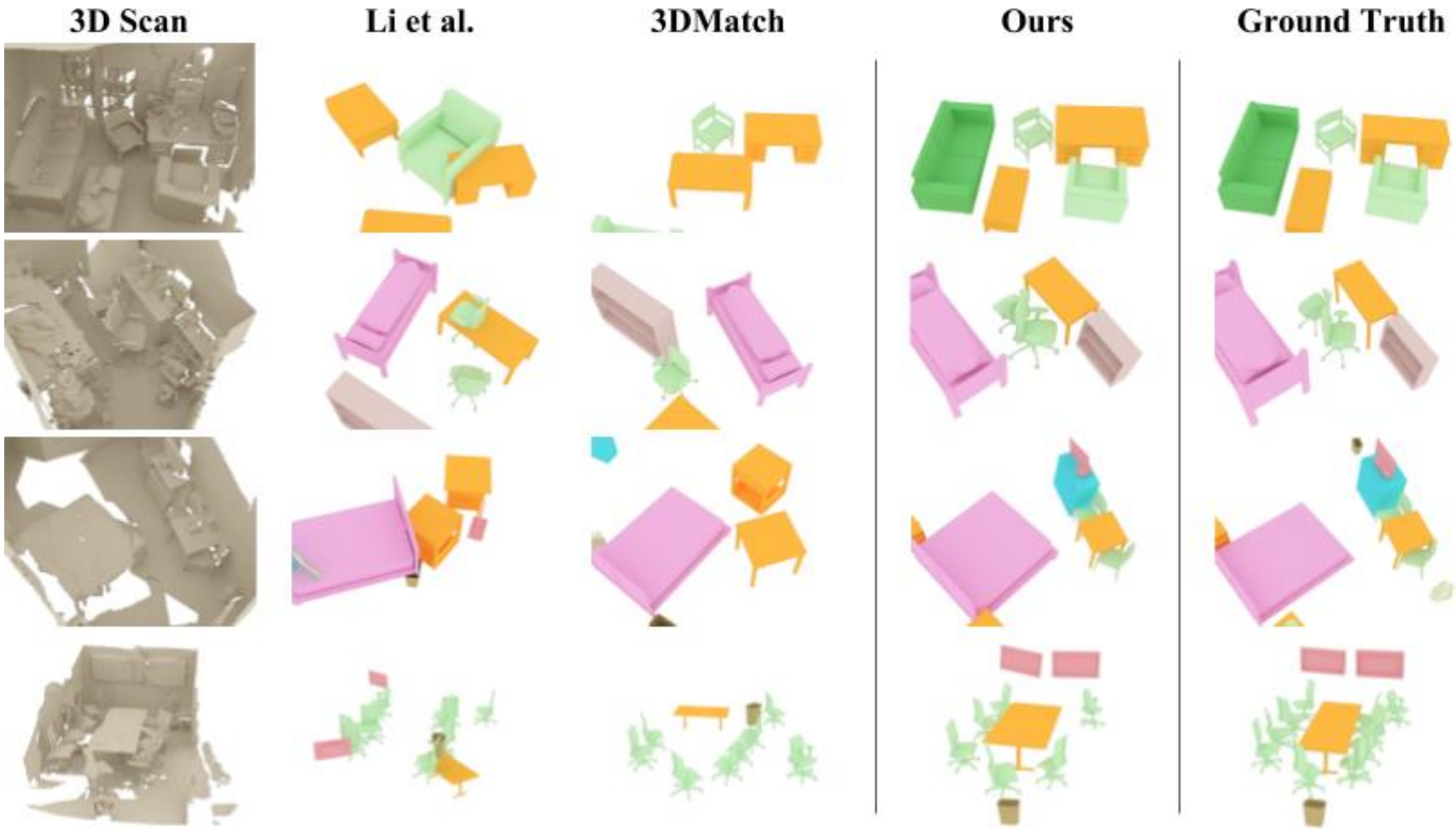
[Avetisyan et al. '19]

Scan2CAD



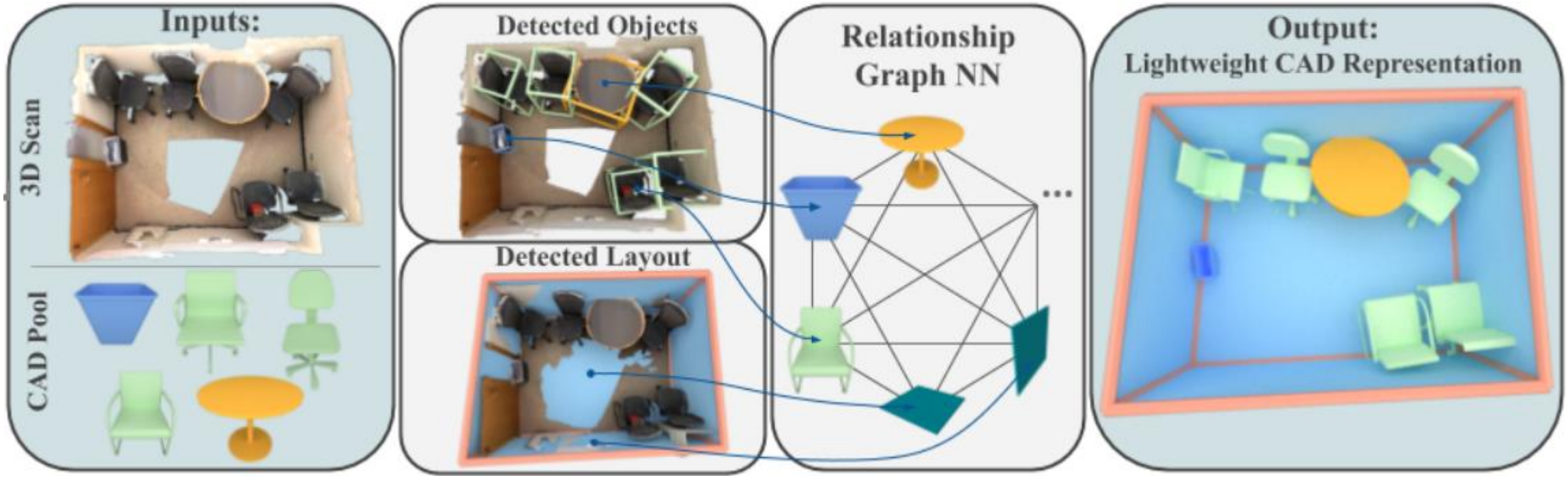
[Avetisyan et al. '19]

Scan2CAD



[Avetisyan et al. ’19]

SceneCAD



[Avetisyan et al. '20]

Scene Synthesis

- Create a scene by iteratively adding synthetic objects to a room

[Paschalidou et al. ’21]

Scene Synthesis

- Create a scene by generating a set of objects on a room layout

[Paschalidou et al. ’21]

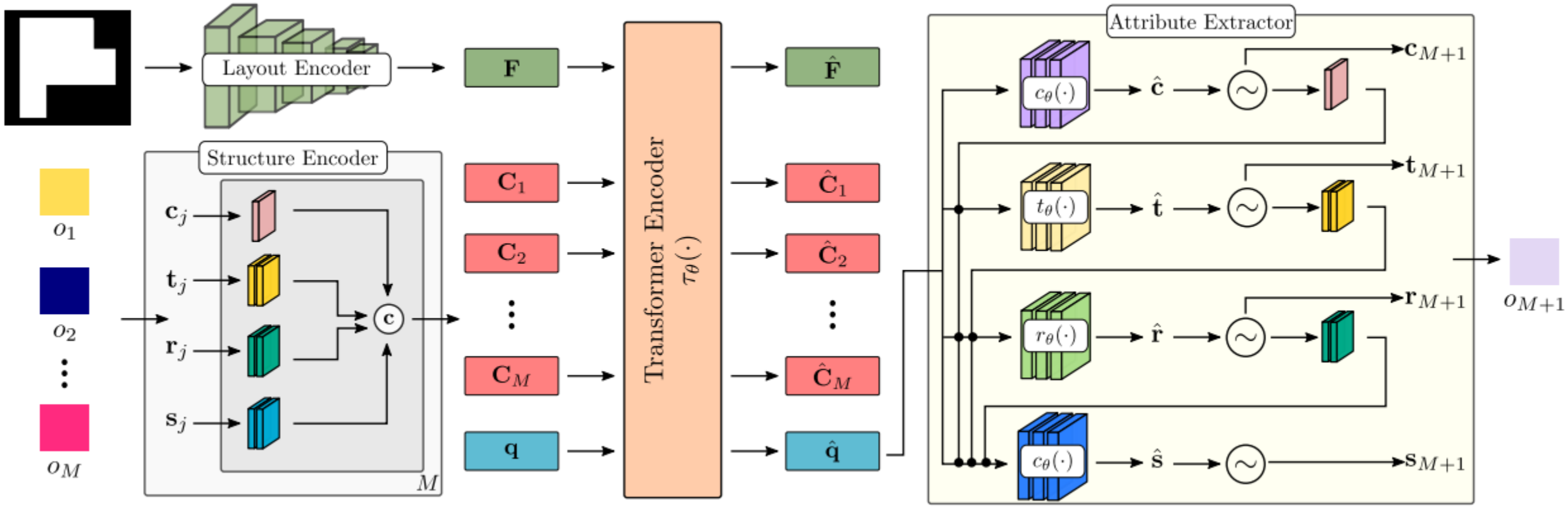
Scene Synthesis

- Scene elements:
  - Floor plan as 2D mask
  - Unordered set of objects $O = \{o_i\}, |O| = N$
    - $o_i = \{c_i, s_i, r_i, t_i\}$: object category, size, orientation, and location

- ATISS: autoregressive transformers for indoor scene synthesis
  - Generate sequence, but shouldn't be order dependent
  - Consider probability of generating set O in any order

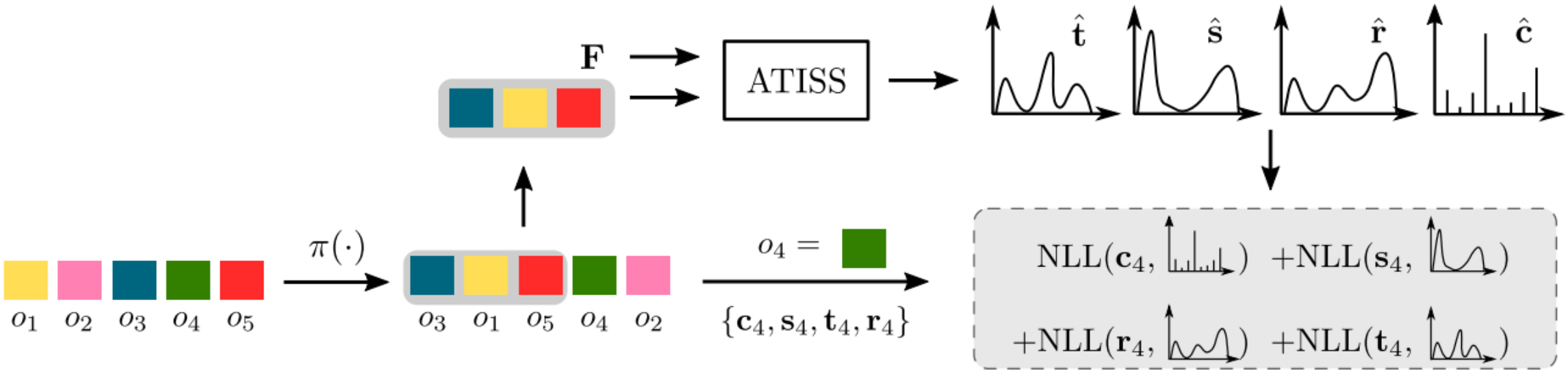
[Paschalidou et al. '21]

Autoregressive Scene Synthesis



[Paschalidou et al. '21]

Train to Encourage Permutation Invariance



[Paschalidou et al. '21]

Autoregressive Scene Synthesis

[Paschalidou et al. '21]

How to quantitatively evaluate?

- Unconditional generation -> no 1:1 correspondences with ground truth scenes
- Measure similarity between two sets of data (real, generated)

- FID (Frechet Inception Distance) Score (Heusel et al. ’17)
  - Use SOTA classification model (pretrained) to compute features for real and generated samples
  - Summarize feature activations as multivariate Gaussian, compute Frechet distance
- Classification accuracy
  - classifier trained to distinguish real from generated
- Note: sensitive to sample sizes

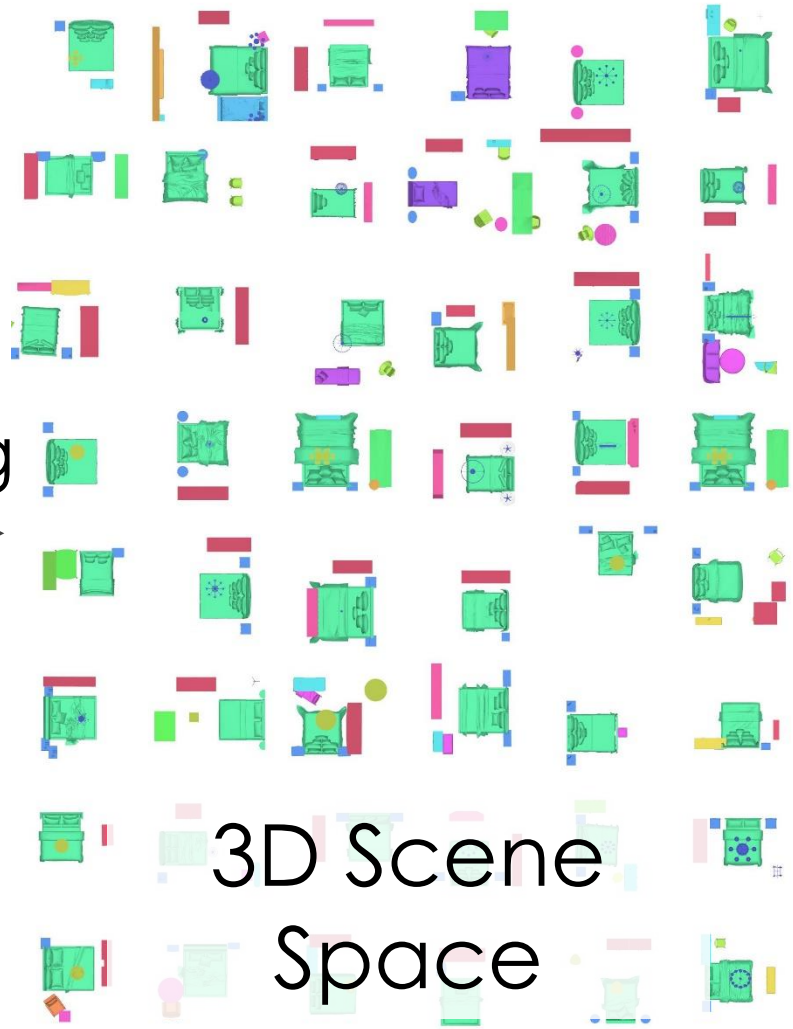
Learn to Generate 3D Scenes from 2D?

2D multi-view data

Learning

Decoding

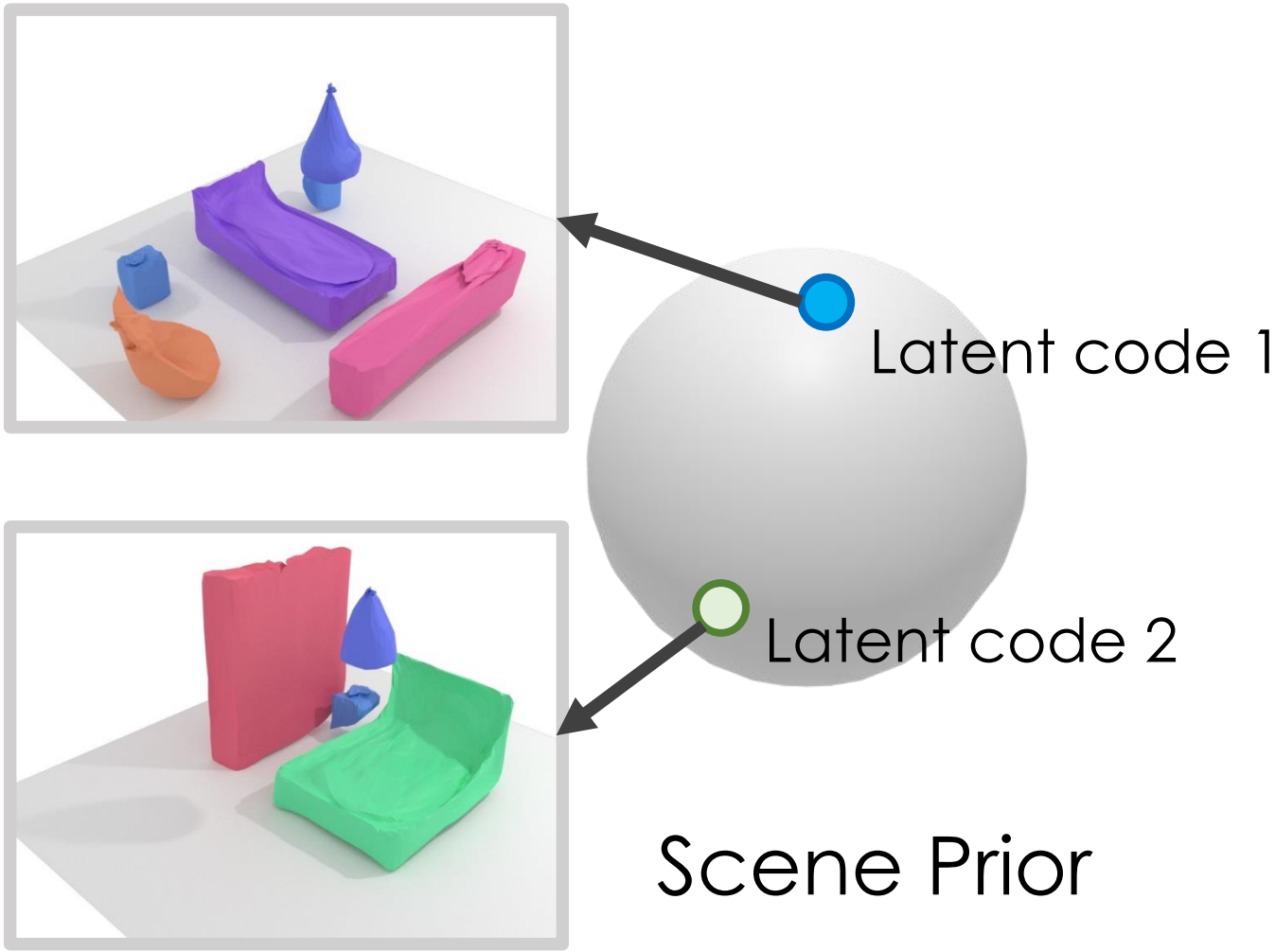
Latent Space (Scene Prior)



3D Scene Space

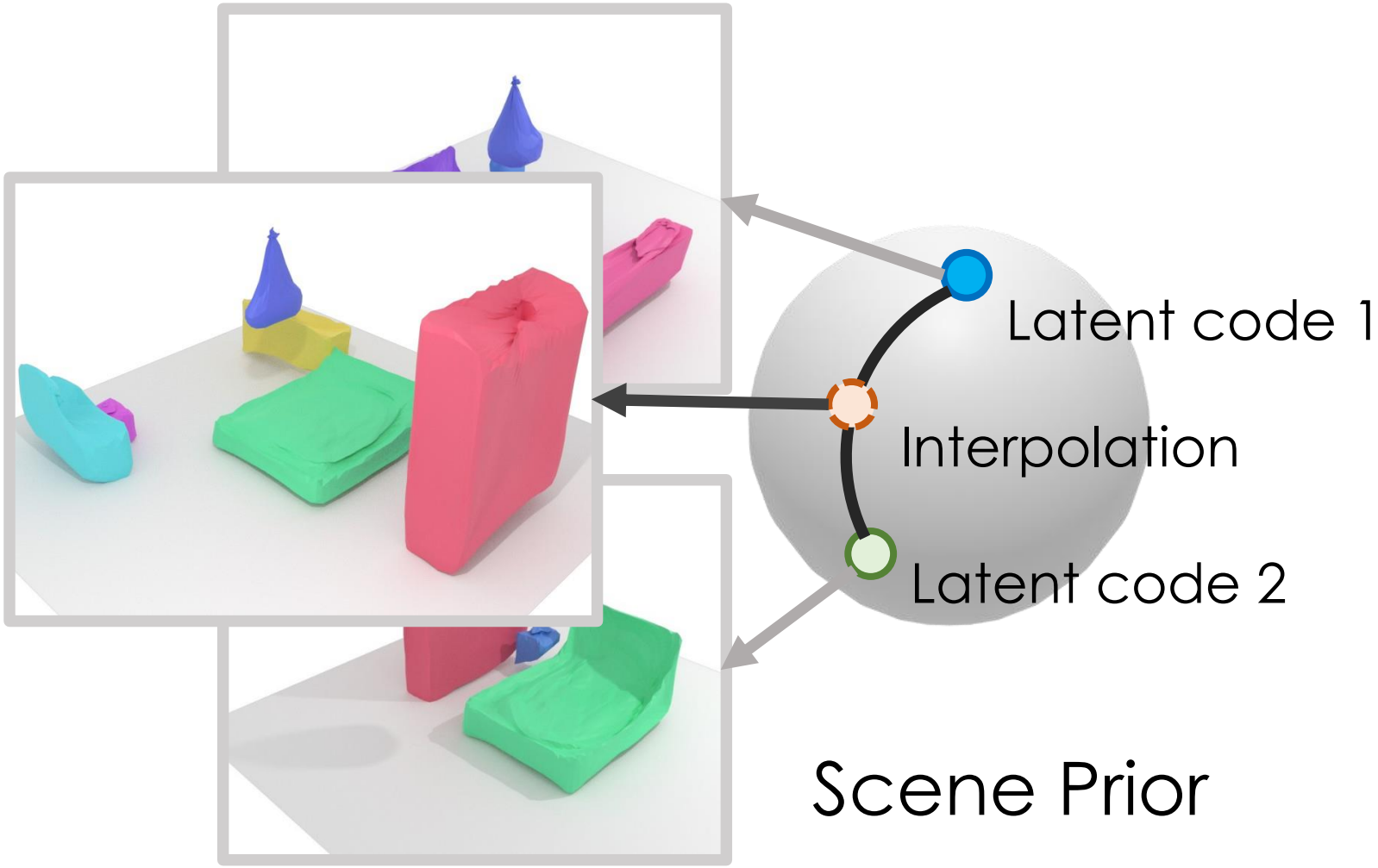
[Nie et al. '23]

Manifold of 3D Scenes



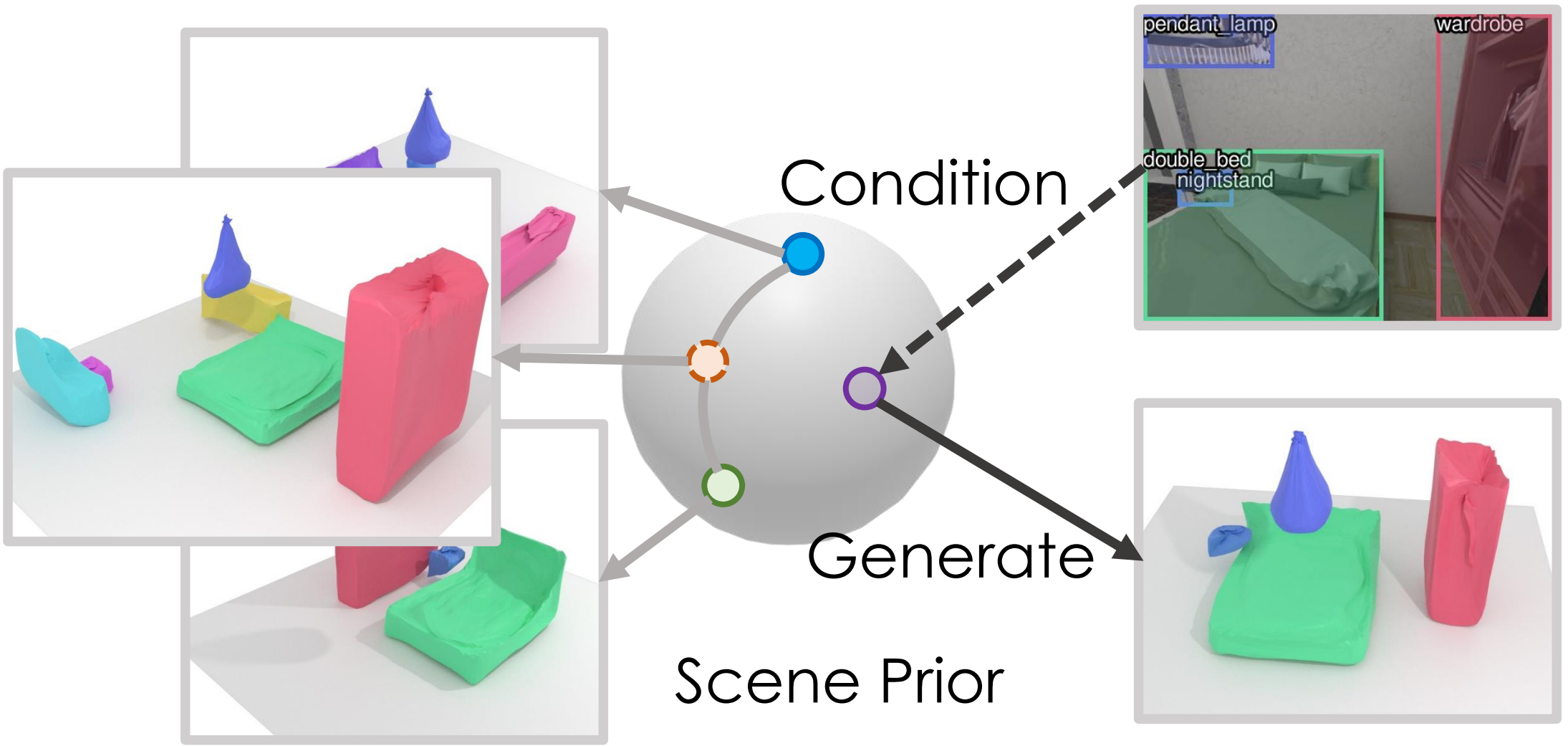
[Nie et al. '23]

Manifold of 3D Scenes



[Nie et al. '23]

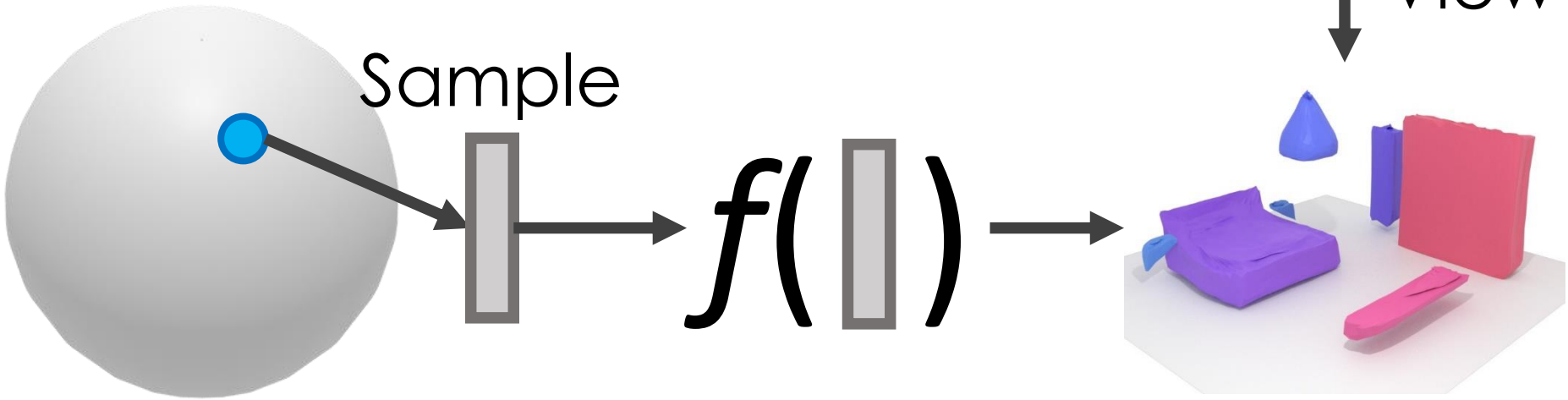
Manifold of 3D Scenes



[Nie et al. '23]

Learn Manifold of 3D Scenes from 2D

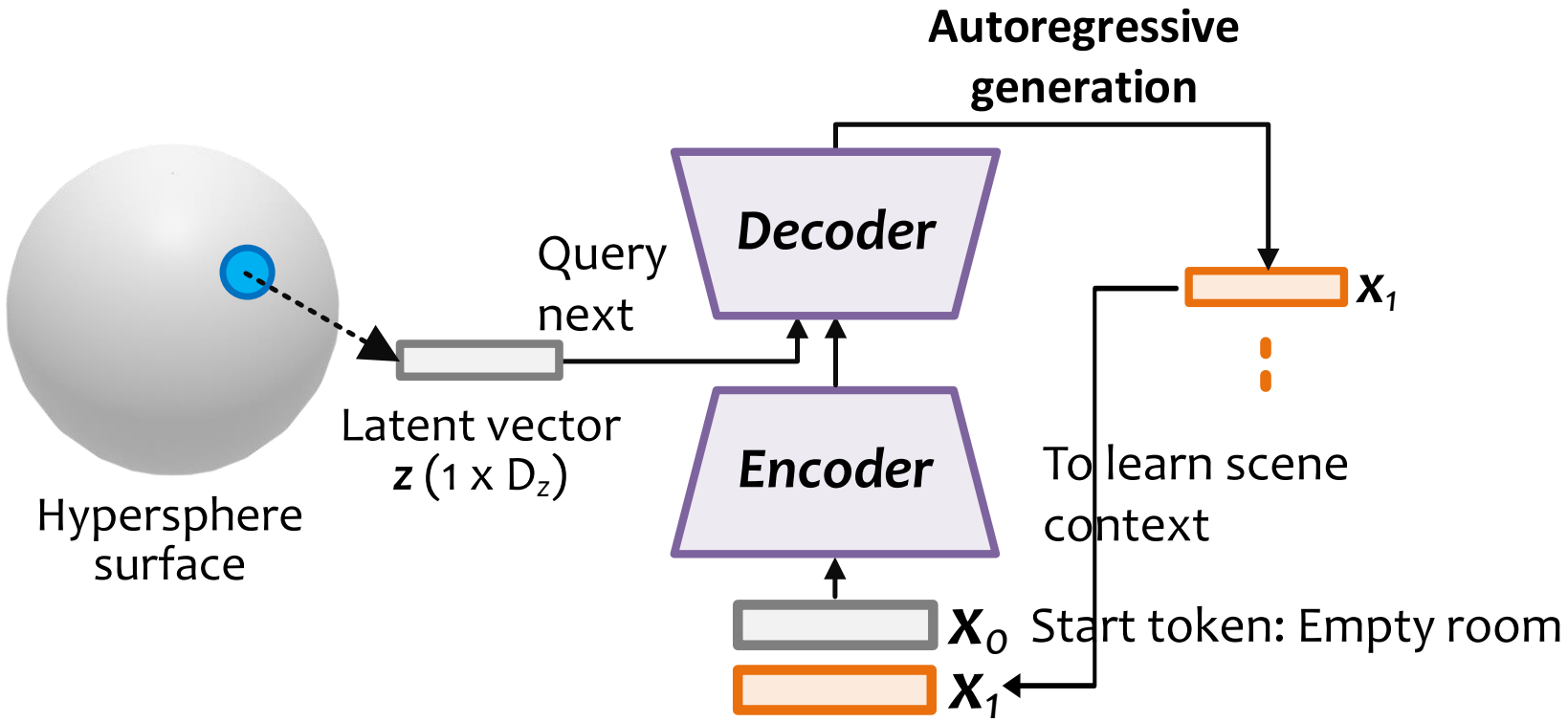
View loss



Latent Space Decoder Scene Space

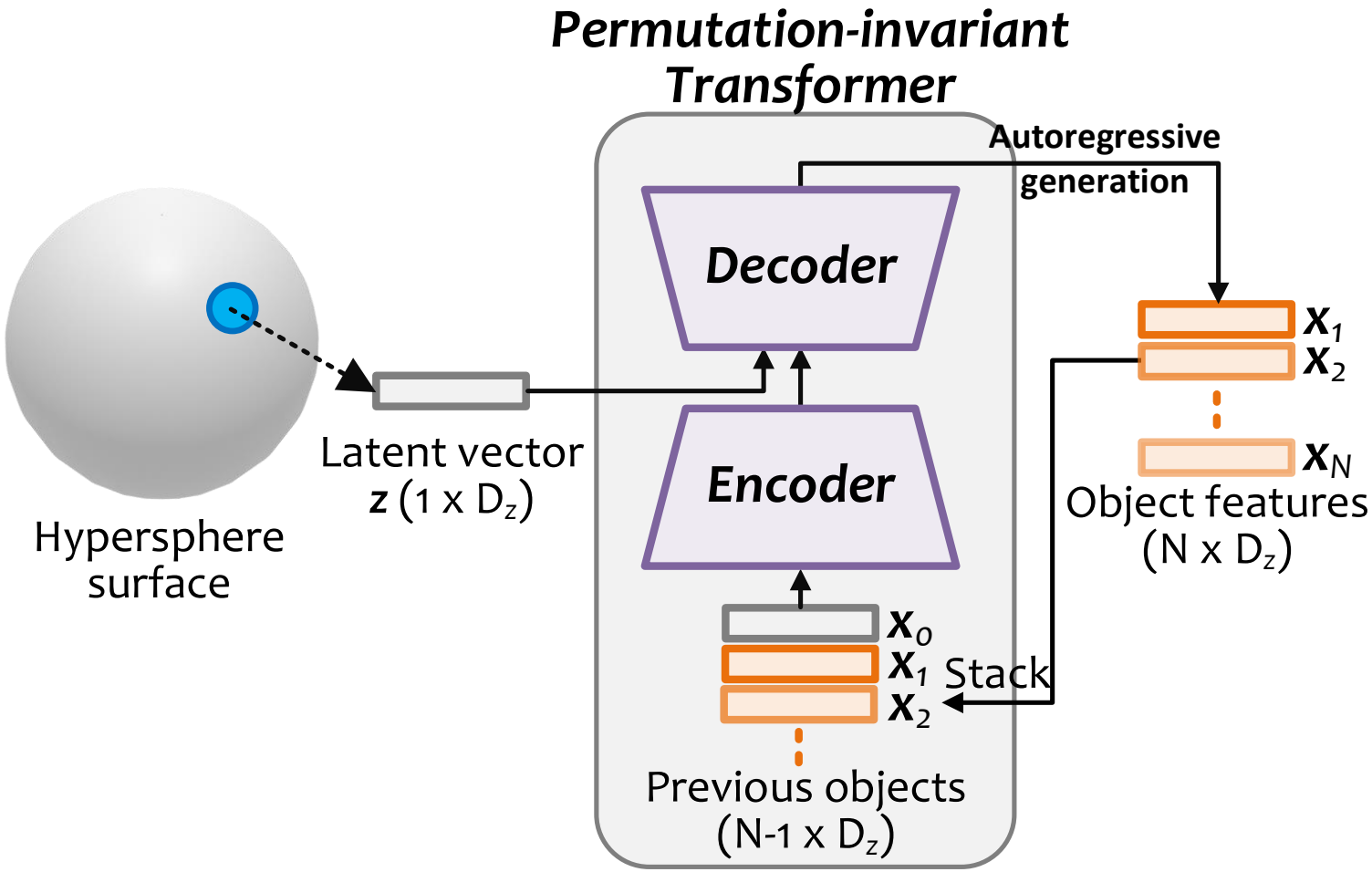
[Nie et al. '23]

Learn Manifold of 3D Scenes from 2D



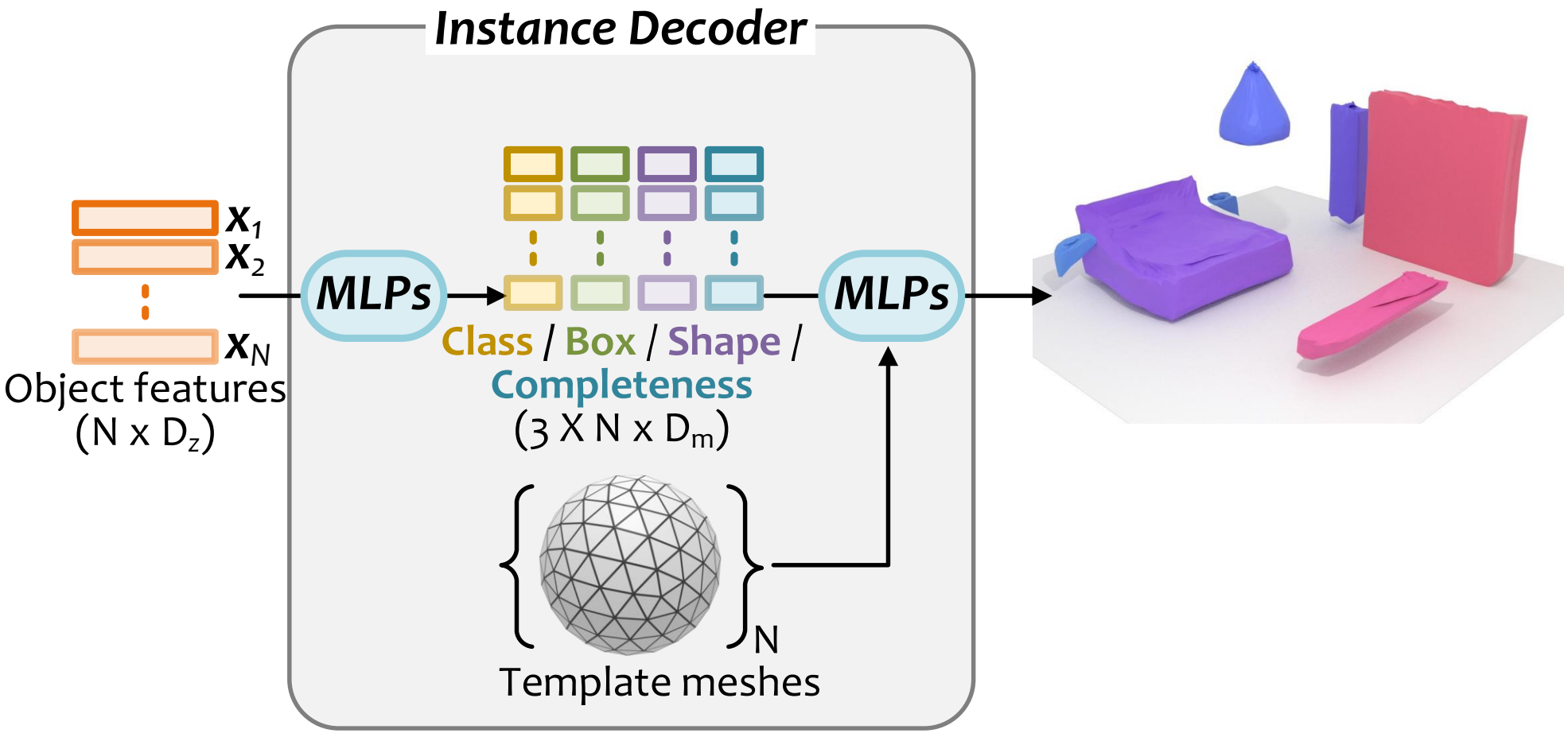
[Nie et al. '23]

Learn Manifold of 3D Scenes from 2D



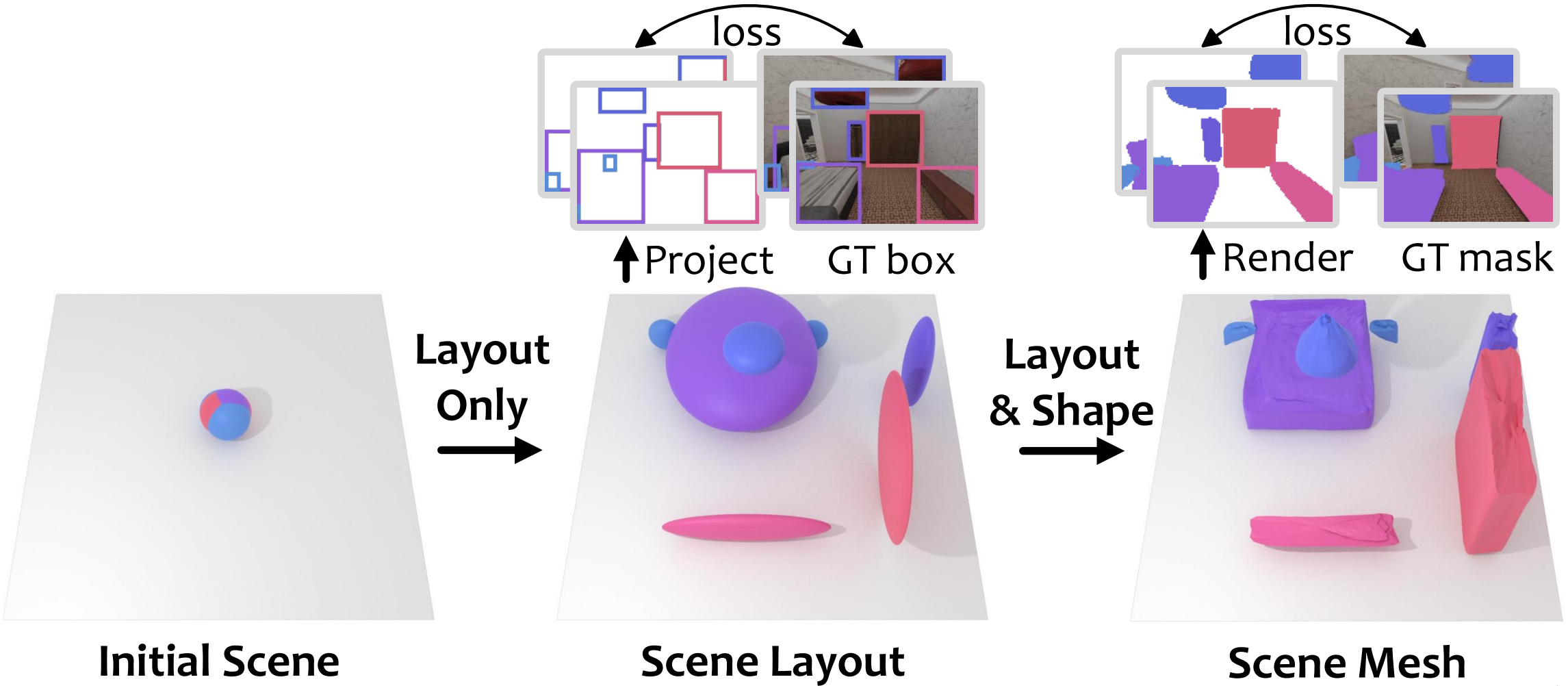
[Nie et al. '23]

Learn Manifold of 3D Scenes from 2D



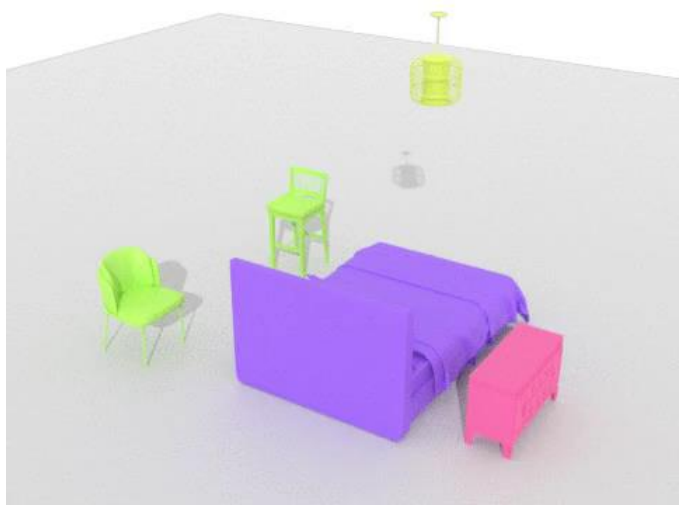
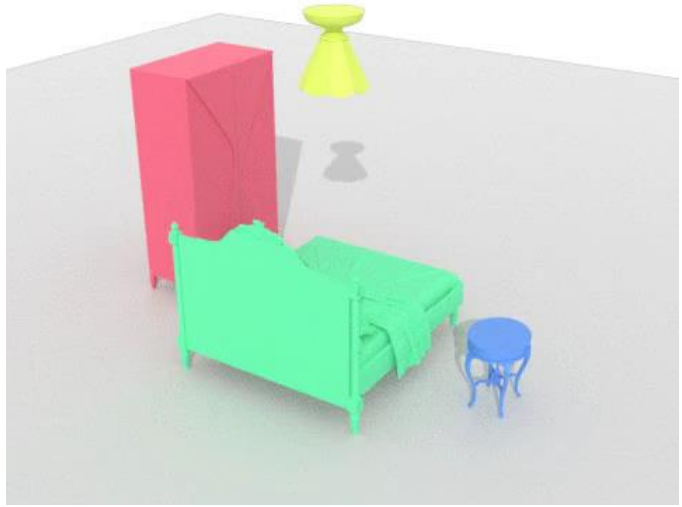
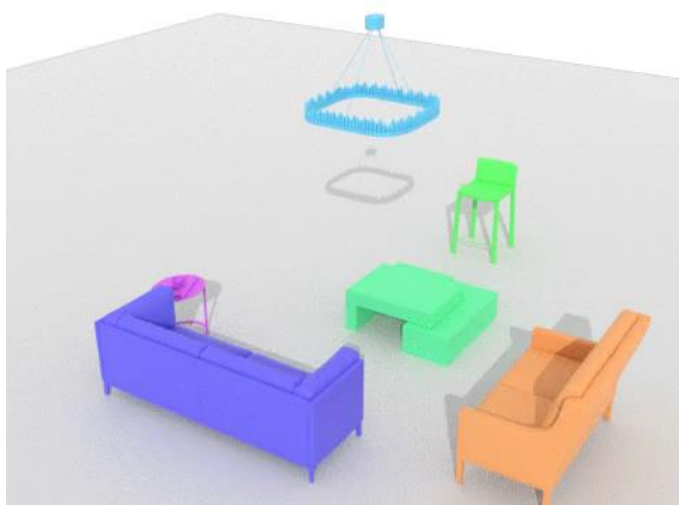
[Nie et al. '23]

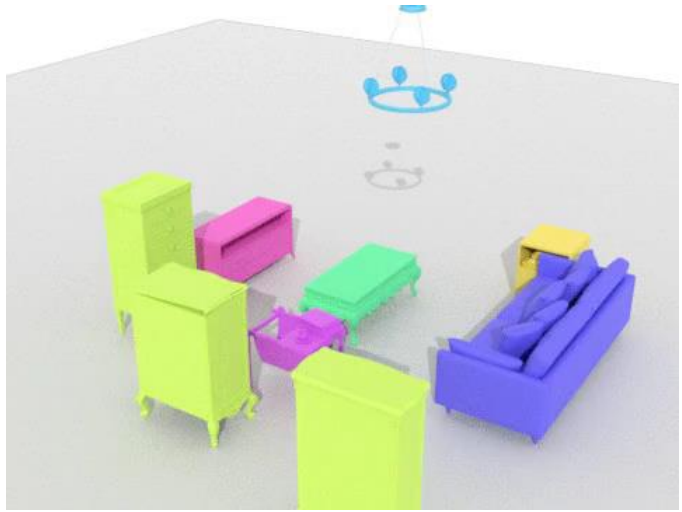
Learn Manifold of 3D Scenes from 2D



[Nie et al. '23]

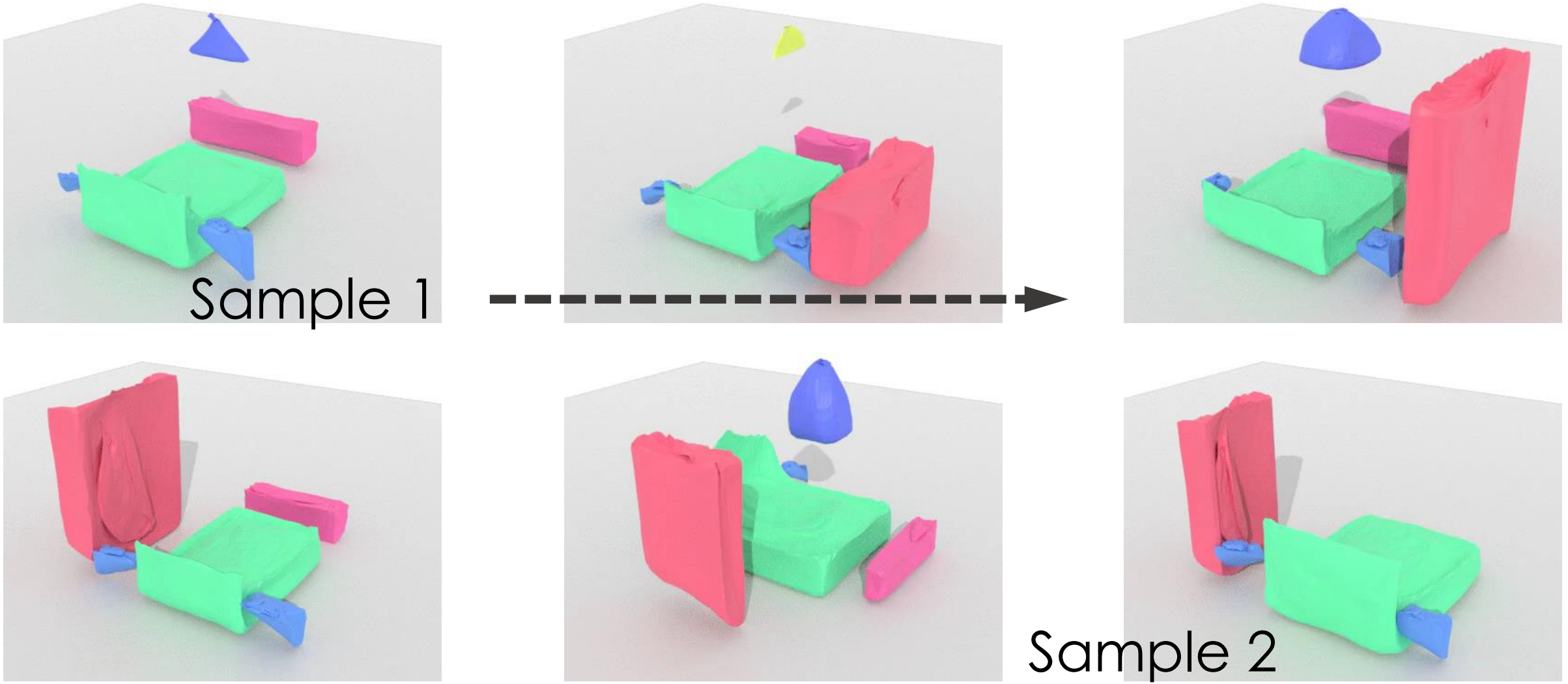
Sampling 3D Scenes



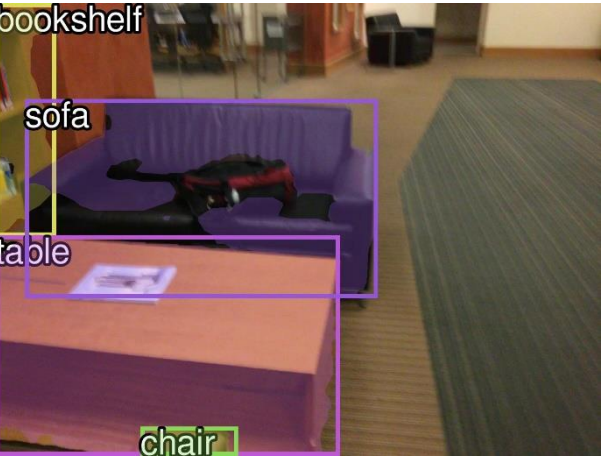

[Nie et al. '23]

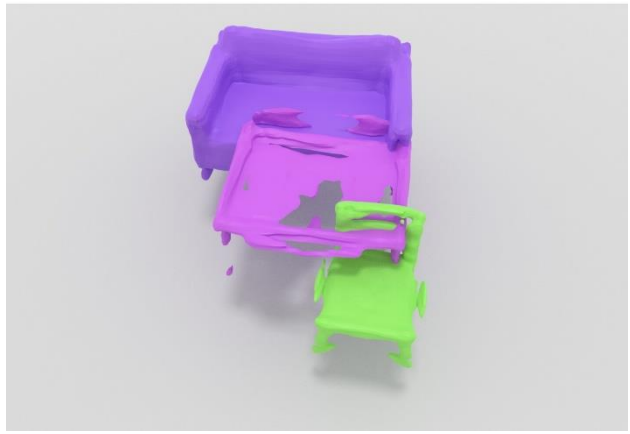
Interpolating 3D Scenes

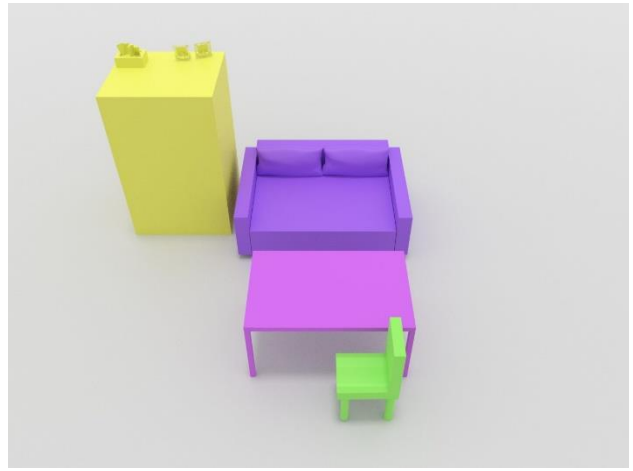


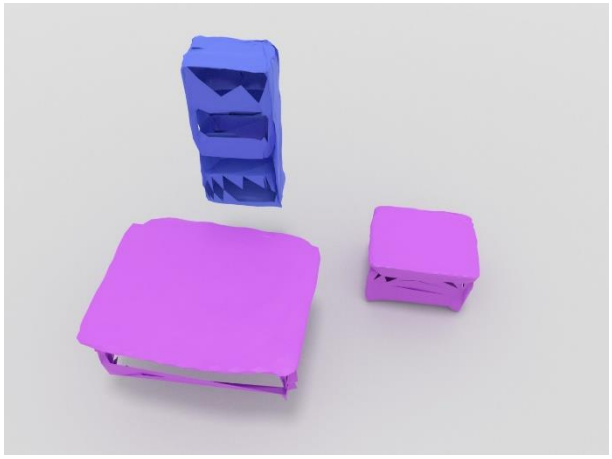
[Nie et al. '23]

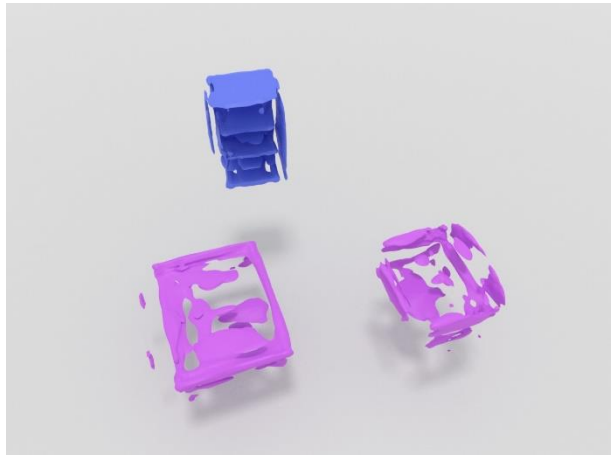
Conditional 3D Scene Generation

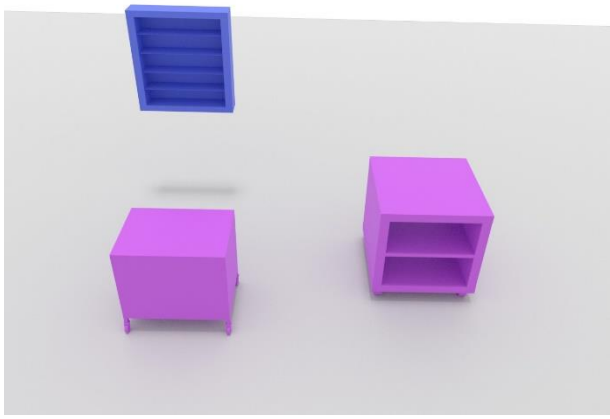












Input

Total3D
[Nie et al. ‘20]

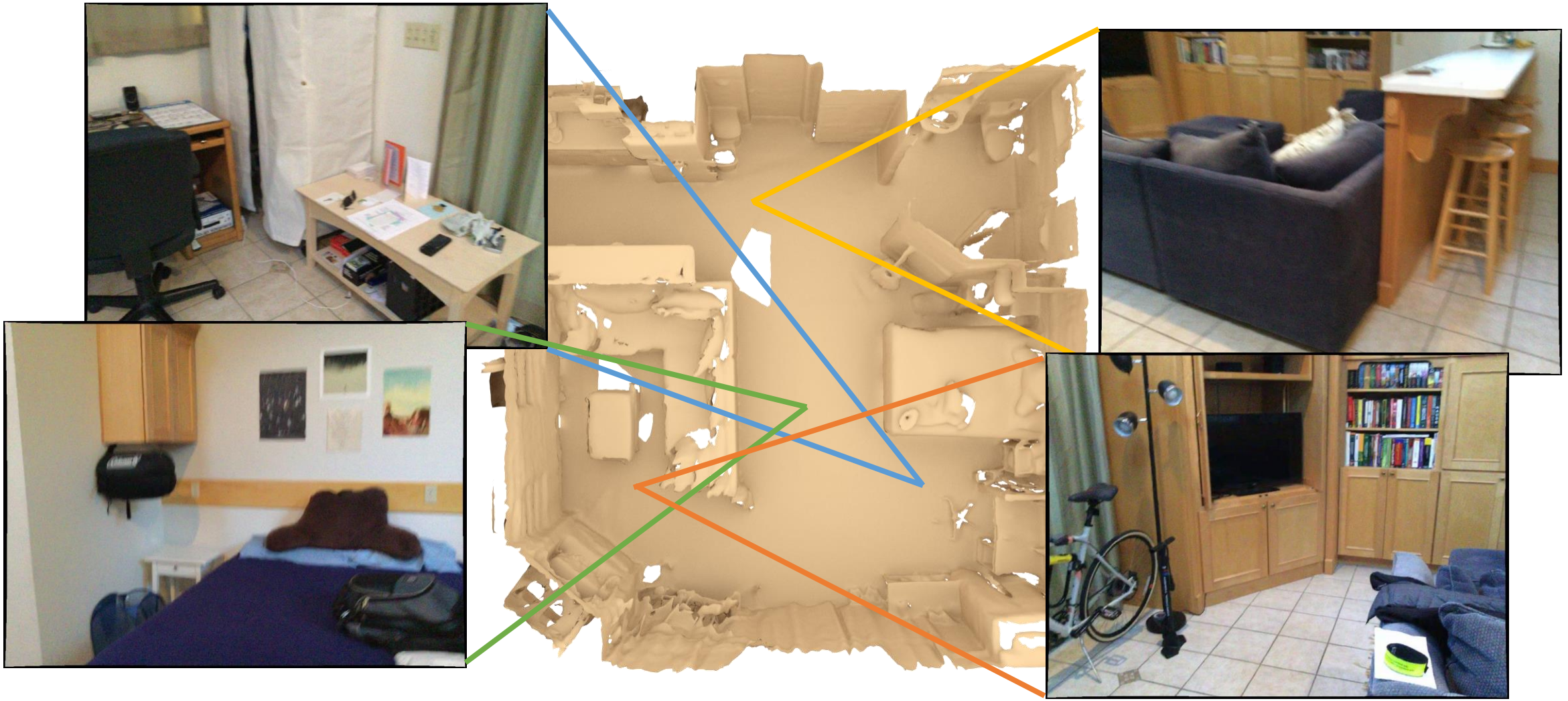
Im3D
[Yang et al. ‘21]

Ours
[Nie et al. ’23]

Textured Scene Generation

- Real-world scenes have varying appearance properties, not just geometry
- For content creation, visualization: need to generate materials, textures, lighting

Texture Optimization for RGB-D Scans

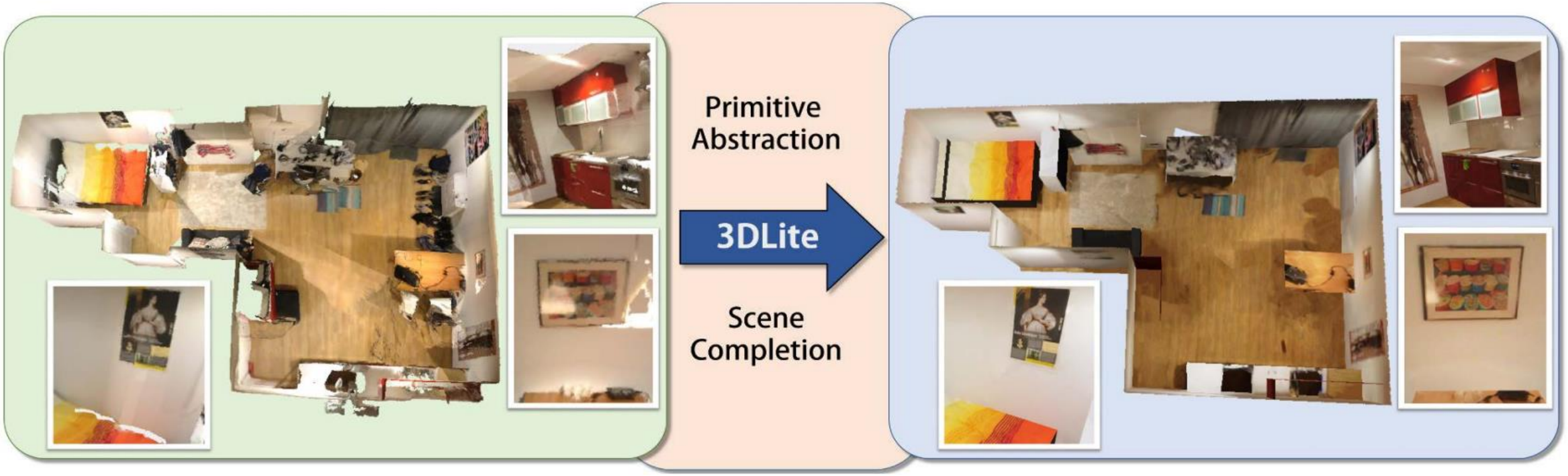


Texture Optimization for RGB-D Scans

- Challenges:
- Camera pose estimation errors from the 3D reconstruction/tracking process
- Motion blur and distortion artifacts from the color camera
- View-dependency for many materials, lighting

3DLite

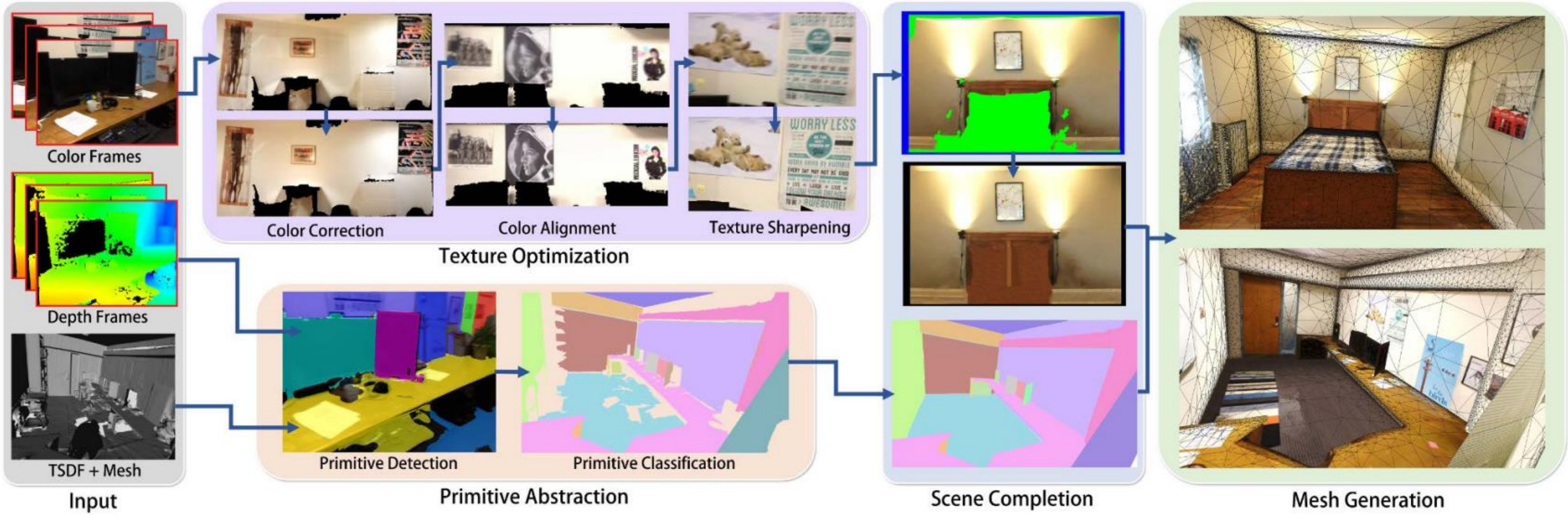
- Simple geometric primitives to enable high-resolution texturing



[Huang et al. '17]

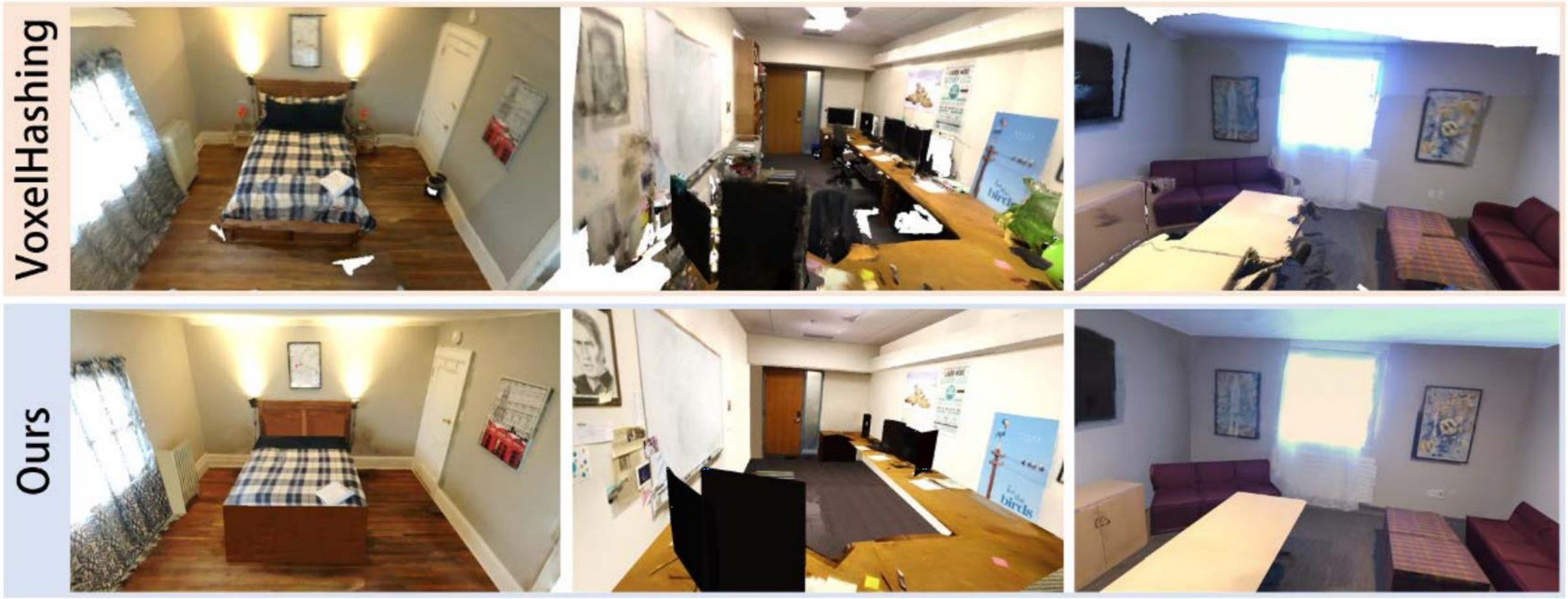
3DLite

- Simple geometric primitives to enable high-resolution texturing



[Huang et al. '17]

3DLite



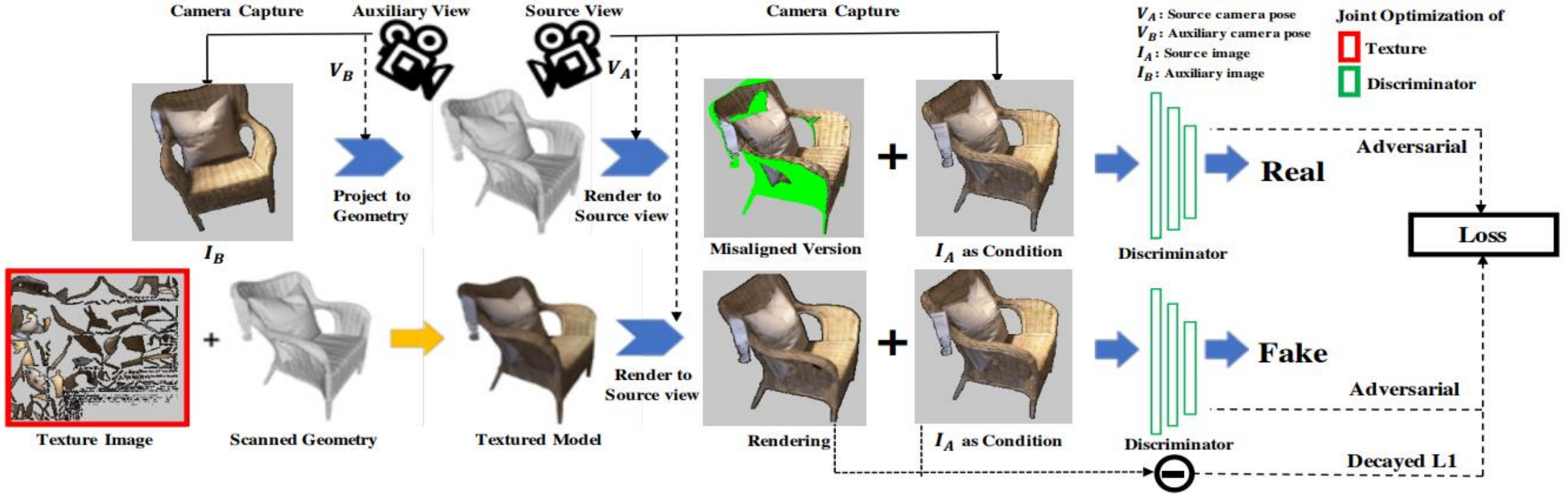
[Huang et al. ‘17]

Adversarial Texture Optimization

- Formulate texture optimization energy by a learned (adversarial) objective function
- Learned adversarial objective designed to be robust to camera misalignments

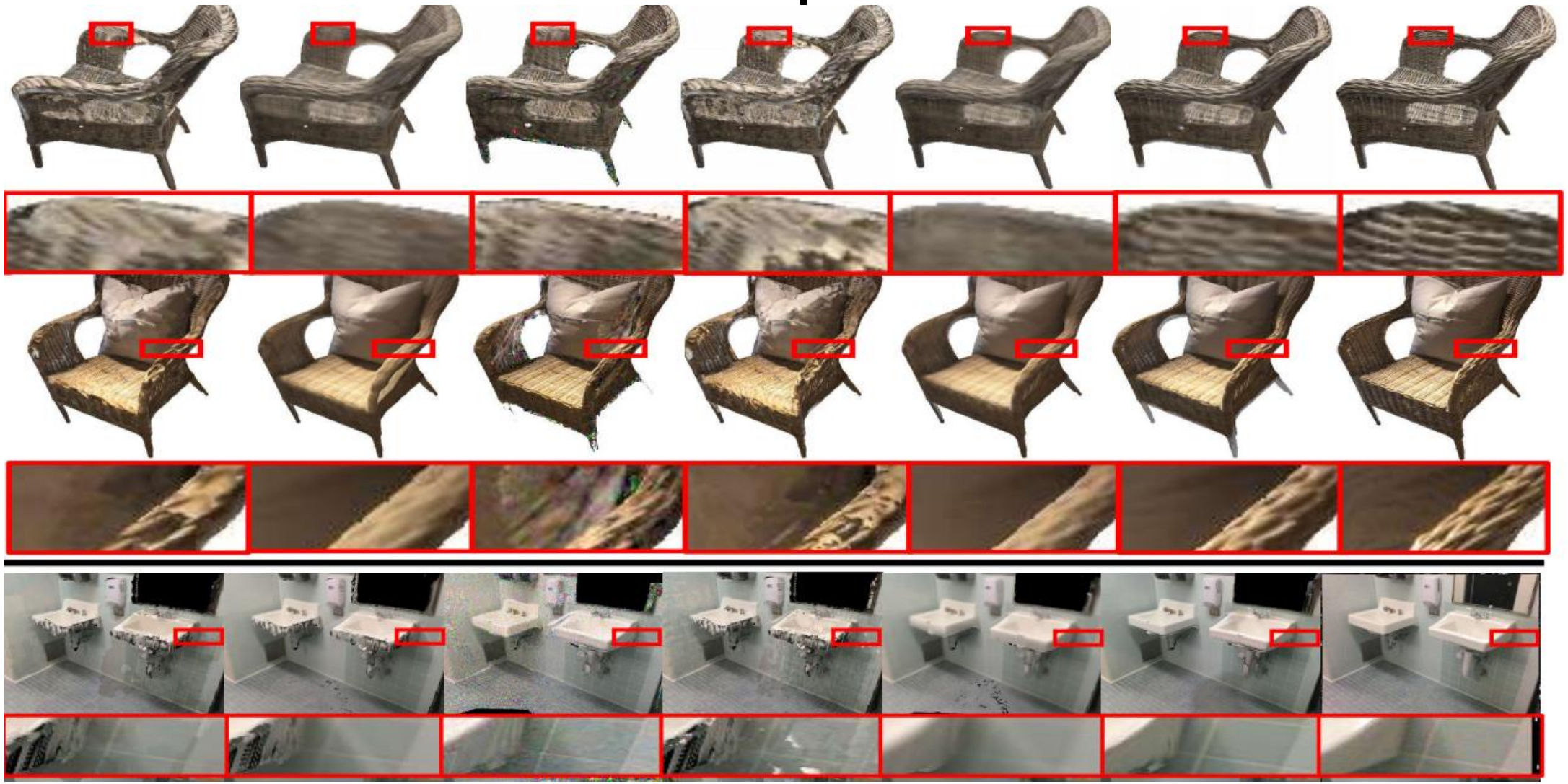
[Huang et al. '20]

Adversarial Texture Optimization



[Huang et al. '20]

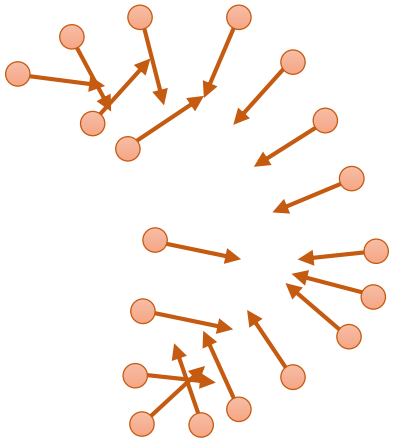
Adversarial Texture Optimization



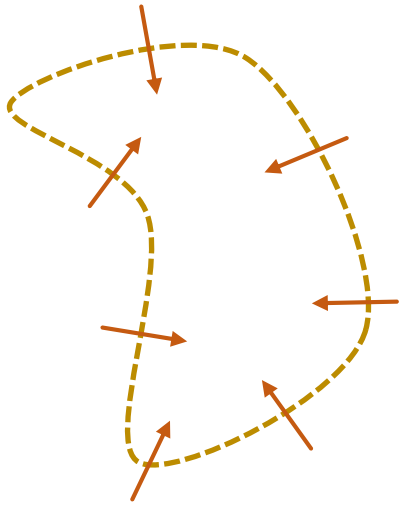
[Huang et al. ’20]

Optimizing Coordinate Fields for Scenes?

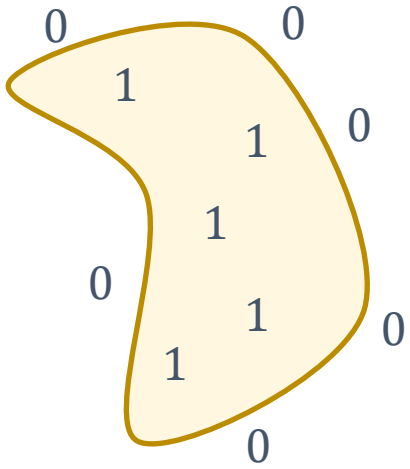
Recall: Poisson Surface Reconstruction



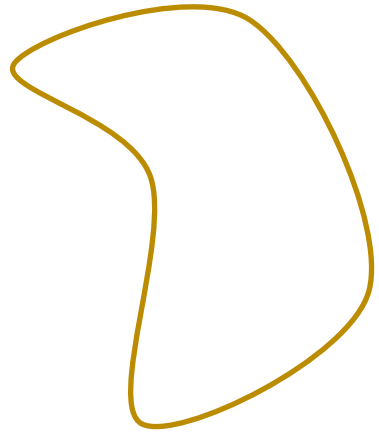
Oriented Points



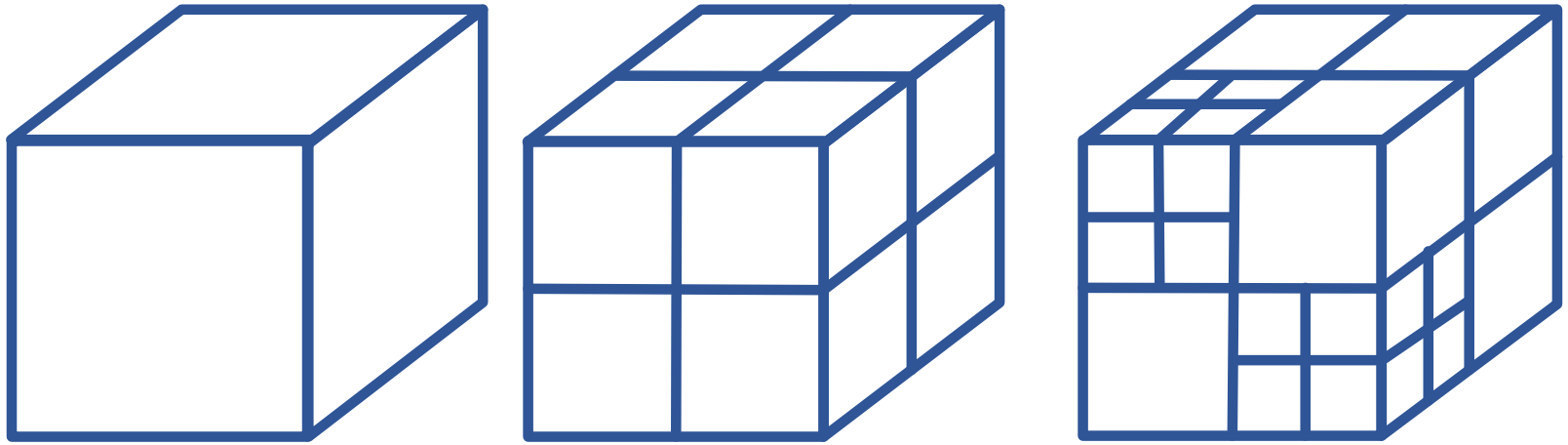
Indicator Gradient $\nabla\chi$



Indicator Function χ

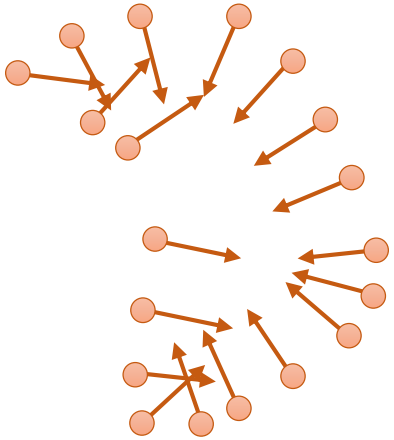


Surface $\delta\Omega$

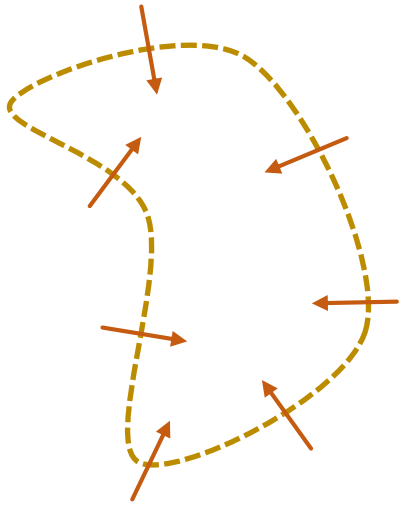


[Kazhdan et al. 06, Kazhdan et al. 13]

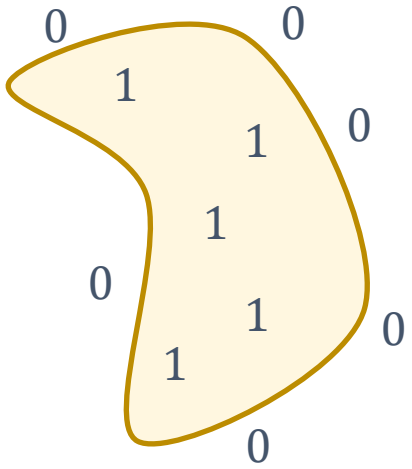
Neural Poisson



Oriented Points



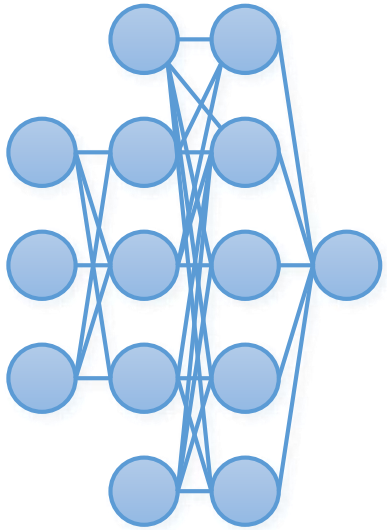
Indicator Gradient $\nabla\chi$



Indicator Function χ

Surface $\delta\Omega$

$x \in \mathbb{R}^3$

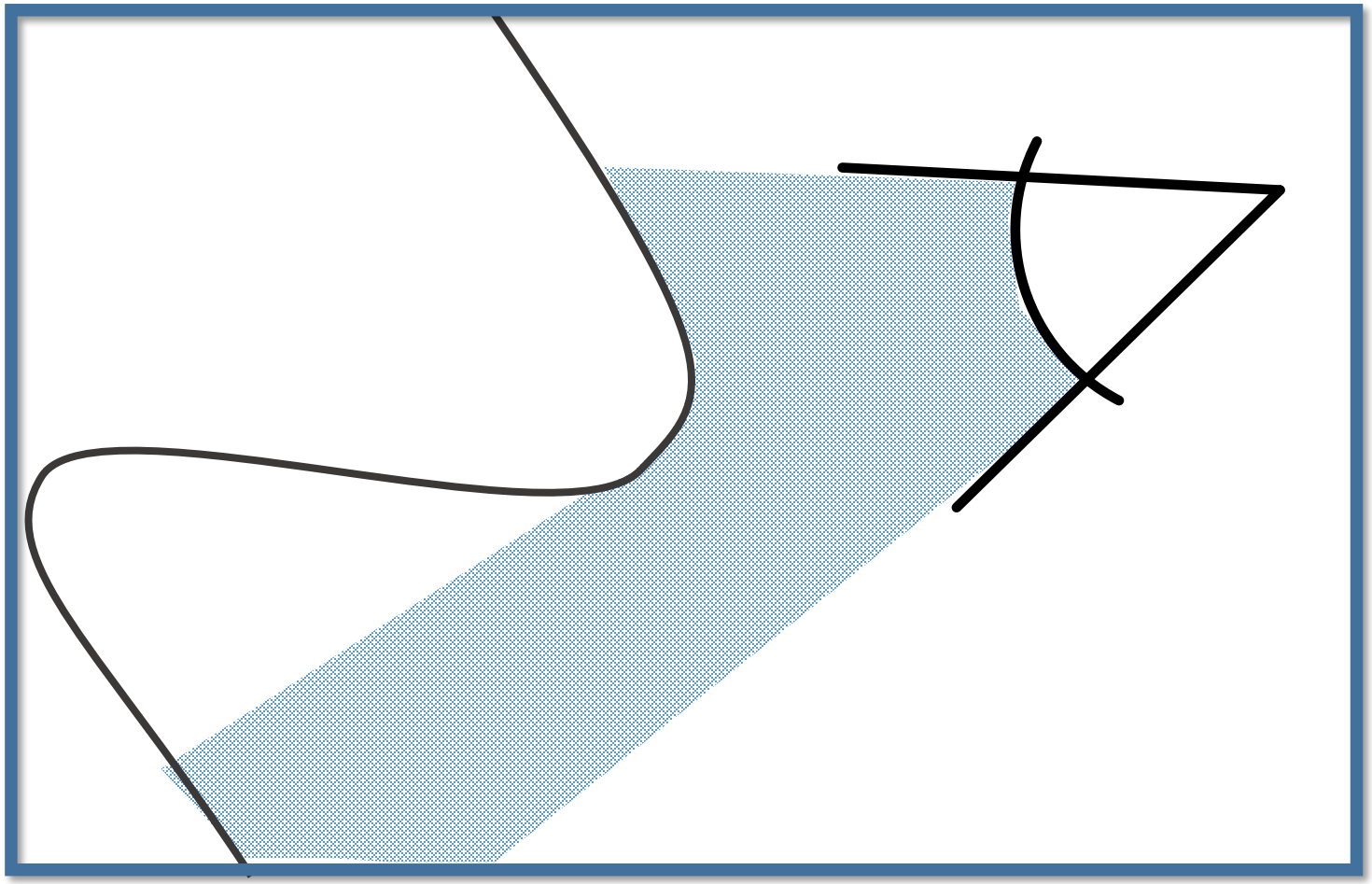


$$\mathbf{1}_\Omega(x) = \begin{cases} 1 \text{ if } x \in \Omega \\ 0 \text{ if } x \notin \Omega \end{cases}$$

[Dai et al. '22]

Leverage Empty Space from Scanning

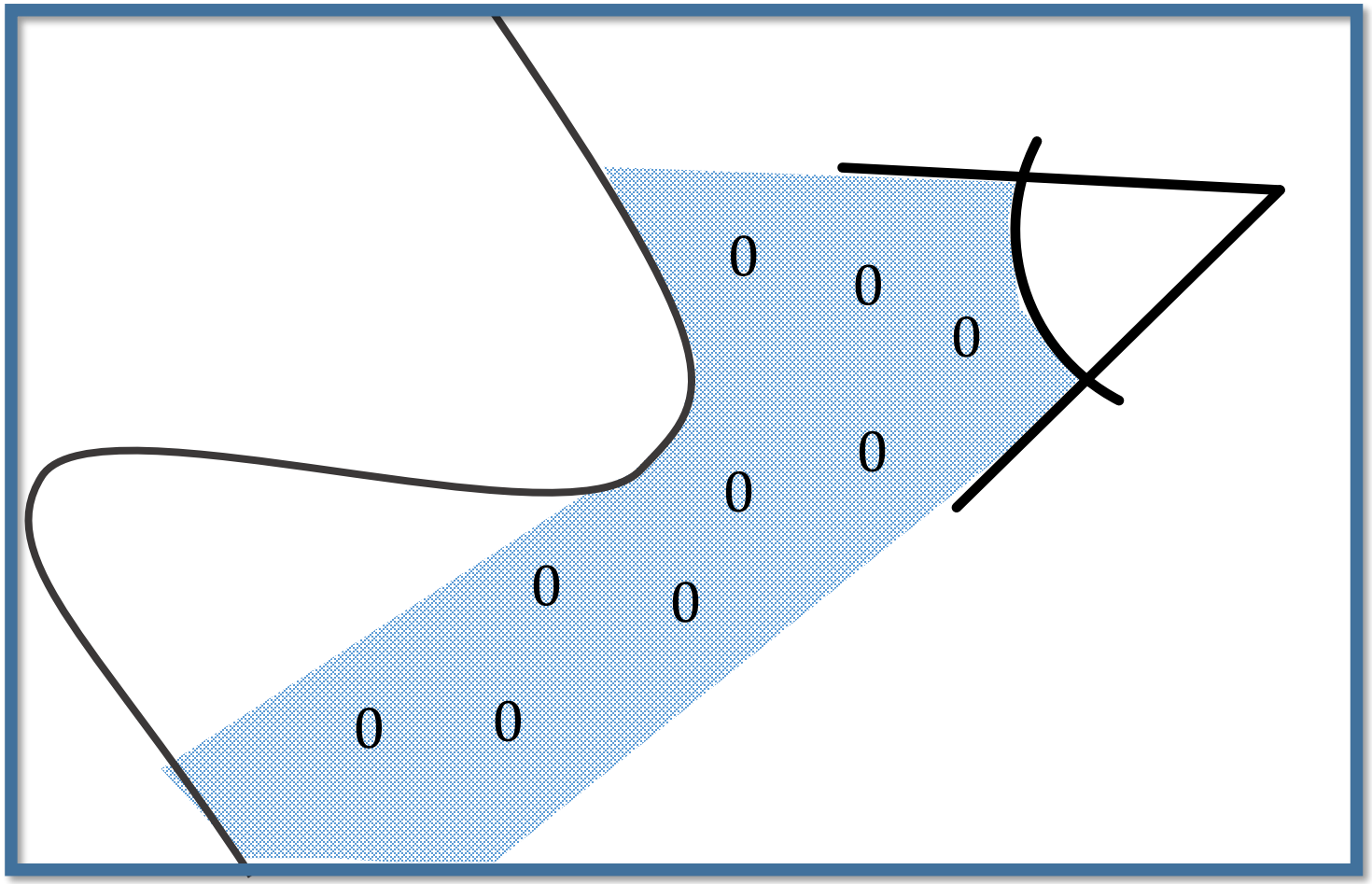
$$\mathbf{1}_{\Omega}(x) = \begin{cases} 1 \text{ if } x \in \Omega \\ 0 \text{ if } x \notin \Omega \end{cases}$$



[Dai et al. '22]

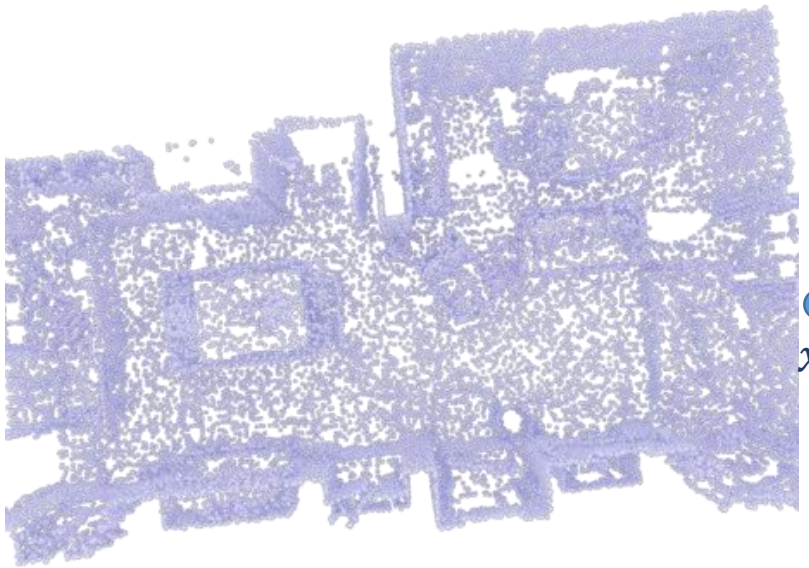
Leverage Empty Space from Scanning

$$\mathbf{1}_{\Omega}(x) = \begin{cases} 1 \text{ if } x \in \Omega \\ 0 \text{ if } x \notin \Omega \end{cases}$$



[Dai et al. '22]

Leverage Empty Space from Scanning



Input Scan: (P_s, P_e, N)

$P_s = \{p \in \mathbb{R}^3\}, P_e = \{p \in \mathbb{R}^3\}, N = \{\nabla p\}$

$x \in \mathbb{R}^3$

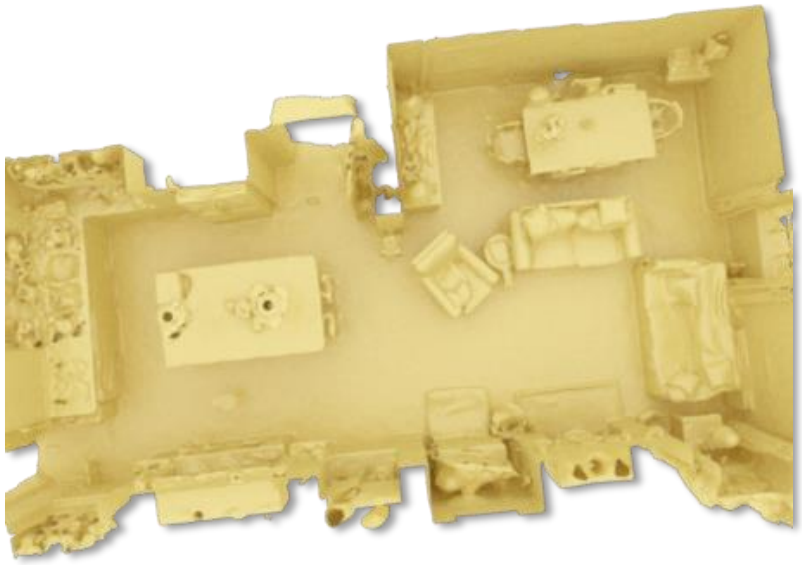
MLP

$\chi_\Omega(x) = \begin{cases} 1 \text{ if } x \in \Omega \\ 0 \text{ if } x \notin \Omega \end{cases}$

Gradient Loss: $\|\nabla\chi - \vec{V}\|_2$

Surface Loss: $\chi(P_s) = 0.5$

Empty Loss: $\chi(P_e) = 0$



[Dai et al. '22]

Neural Poisson

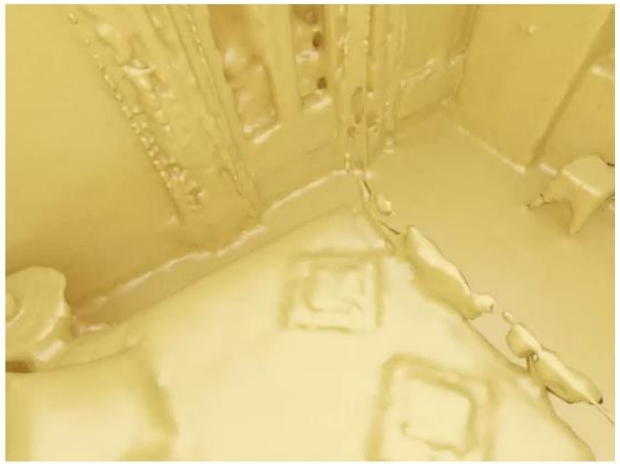
```
def gradient(y, x, grad_output):
    grad = torch.autograd(y, [x], grad_outputs=grad_output,
                          create_graph=True)[0]
# indicator: 0: empty, 1: surf, 2: normal
# [0,1] shifted to [-0.5, 0.5] in practice
def compute_loss(input, pred, gt, indicator):
    grad = gradient(pred, input)
    gradient_loss = torch.where(indicator != 0, (grad – gt).pow(2).sum(-1),
                                torch.zeros_like(grad[…,0]))
    surf_loss = torch.where(indicator == 1, pred, torch.zeros_like(pred))
    empty_loss = torch.where(indicator == 0, (pred + 0.5).pow(2).sum(-1),
                             torch.zeros_like(pred))
    return { ‘gradient_loss’: gradient_loss.mean(),
             ‘surf_loss’: surf_loss.mean(),
             ‘empty_loss’: empty_loss.mean() }
```

[Dai et al. ’22]

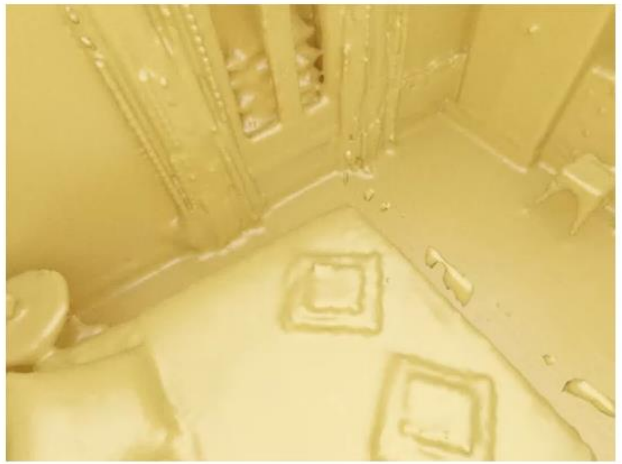
Neural Poisson

Input

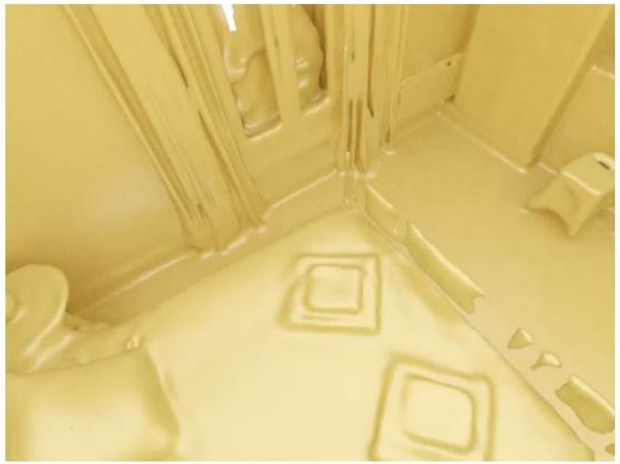
(50% points shown for visualization purposes)



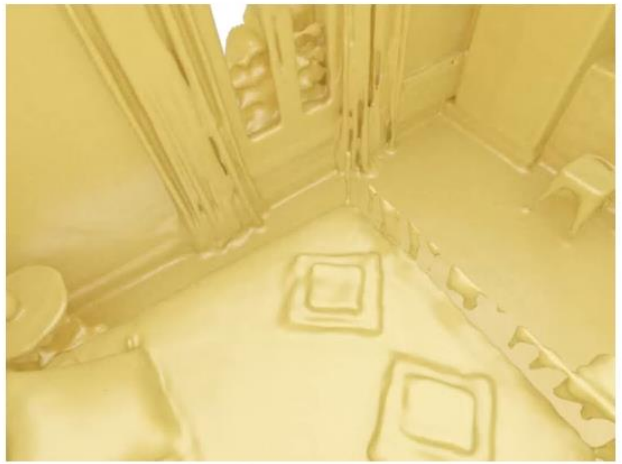
SSD



SPSR



Siren



Ours

[Dai et al. '22]

Radiance Fields + Reconstruction

- Recall: Neural radiance fields optimize for photometric losses
- Under-constrained scene representation – scene density distributed as “floaters”
- Can we better constrain the geometry?
- Can we directly optimize for geometry?

Image formation process

- What scene characteristics inform the color observation for a pixel?
- Geometry + Material + Lighting
- NeRF optimization: fit to one capture setting
  - How to modify these attributes separately?
- 3D modeling for content creation:
  - Artists model geometry, material, and lighting separately
  - Can visualize under different settings – particularly lighting, which changes in real settings

Regularize with 2D Diffusion Models

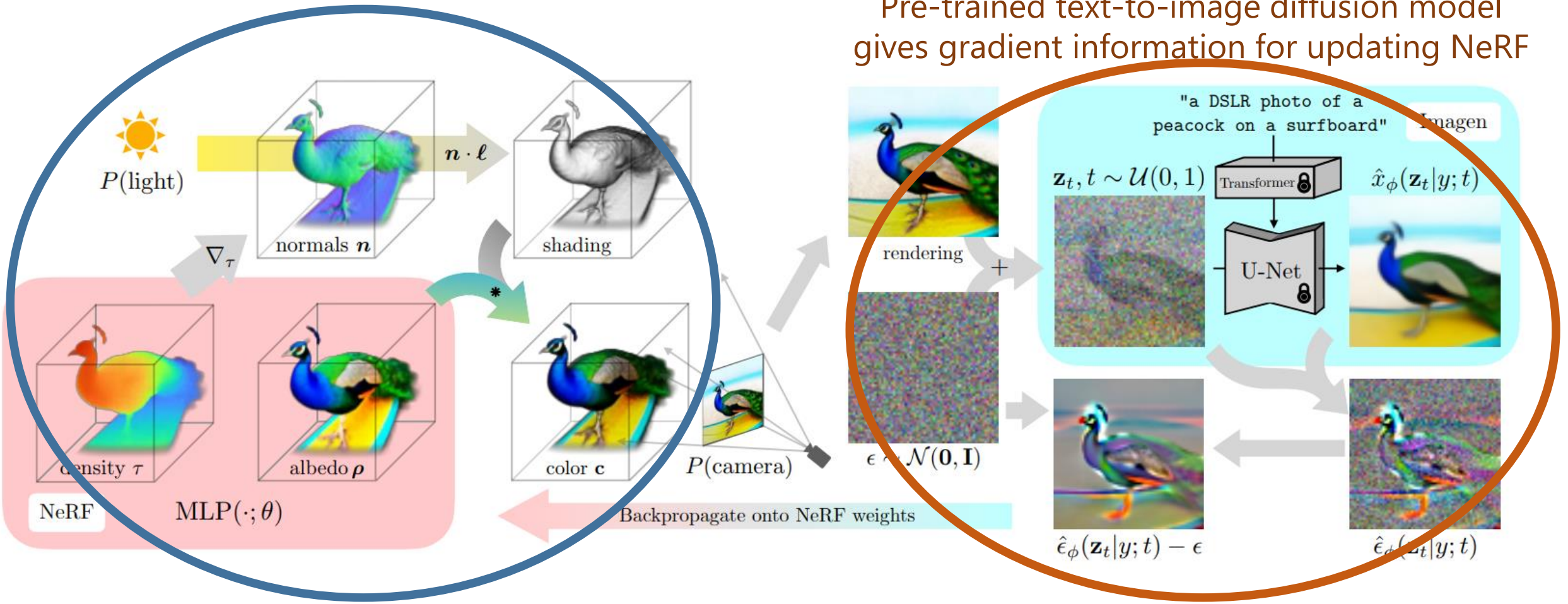
- DreamFusion
- Generate radiance field from text input

an orangutan making a clay bowl on a throwing wheel*

[Poole et al. ‘23]

Regularize with 2D Diffusion Models

Pre-trained text-to-image diffusion model gives gradient information for updating NeRF

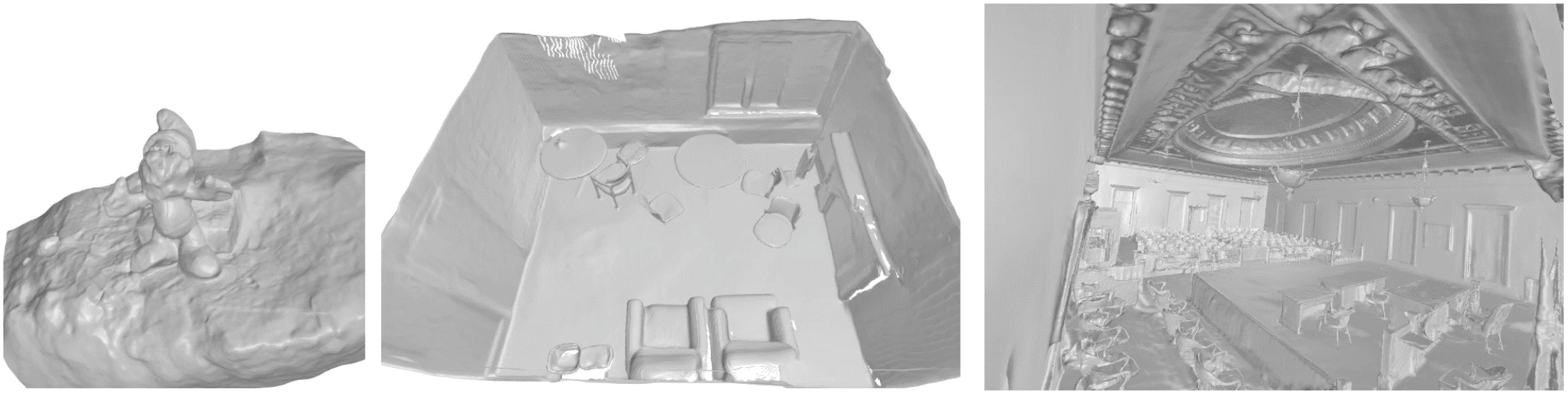


Model color of surface
Light with controlled illumination

[Poole et al. '23]

Volume Rendering for Surface Reconstruction

- Input: set of posed RGB images (images + camera poses)
- Output: 3D surface as SDF



DTU (3 views) ScanNet Tanks and Temples

[Yu et al. '22]

Volume Rendering for Surface Reconstruction

- Input: set of posed RGB images (images + camera poses)
- Output: 3D surface as SDF

- How to relate SDF and density for volume rendering?


Volume Density


Sign Distance Function

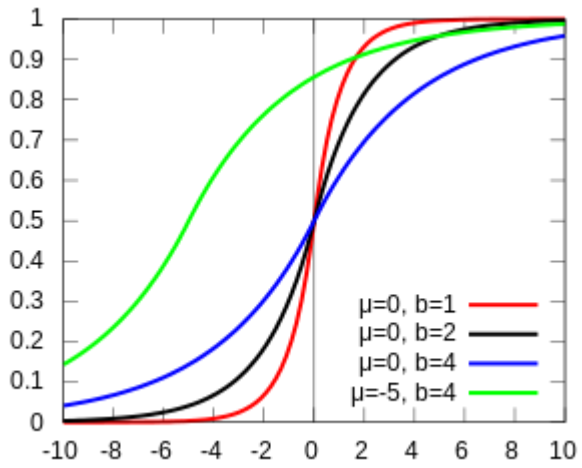
[Yariv et al. '21]

Volume Rendering for Surface Reconstruction

- How to relate SDF d and density σ for volume rendering?
- Approximate relation:

$$\sigma = \alpha \Phi_\beta(-d)$$

  - $\alpha, \beta > 0$ are learnable parameters
  - Φ_β is CDF of Laplace distribution with 0 mean and β scale



Volume Density

Sign Distance Function

[Yariv et al. '21]

Volume Rendering for Surface Reconstruction

- Scene Representation:
- Geometry: $f: \mathbb{R}^3 \rightarrow \mathbb{R}, \mathbb{R}^k, f(x) = (s, z)$
  - s: signed distance
  - z: feature vector
  - n: analytical gradient of SDF function f
- Color: $c = f_c(x, \theta, n, z)$
  - θ: viewing direction

Volume Rendering for Surface Reconstruction

- Compute color $\hat{C}(r)$ for ray r as usual:
  - T_r^i, α_r^i are transmittance and opacity for sample point i along ray r
  - δ_r^i is distance between neighboring sample points

$$\hat{C}(r) = \sum_{i=1}^{M} T_r^i \alpha_r^i c_r^i \qquad T_r^i = \prod_{j=1}^{i-1} (1 - \alpha_r^j) \qquad \alpha_r^i = 1 - \exp(-\sigma_r^i \delta_r^i)$$

- Compute depth $\widehat{D}(r)$ and normal $\widehat{N}(r)$ similarly:

$$\widehat{D}(r) = \sum_{i=1}^{M} T_r^i \alpha_r^i t_r^i \qquad \widehat{N}(r) = \sum_{i=1}^{M} T_r^i \alpha_r^i n_r^i$$

t_r^i: distance along ray

Optimizing for Surface Reconstruction

- RGB Reconstruction $L_{rgb} = \sum_{r \in \mathcal{R}} \left\| \hat{C}(r) - C^{gt}(r) \right\|_1$ $\mathcal{R}$: rays in minibatch
- SDF Regularization (Eikonal Loss)

$$L_{eik} = \sum_{x \in \mathcal{X}} (\|\nabla f(x)\|_2 - 1)^2$$ $\mathcal{X}$: sampled points

- Consistency between depth and predicted depth from RGB

$$L_{depth} = \sum_{r \in \mathbb{R}} \left\| \left(w\widehat{D}(r) + q \right) - D^{rgb}(r) \right\|^2$$ w, q: computed scale/shift for each batch

- Consistency between normal and predicted normal from RGB

$$L_{normal} = \sum_{r \in \mathbb{R}} \left\| \widehat{N}(r) - N^{rgb}(r) \right\|_1 + \left\| 1 - \widehat{N}(r)^T N^{rgb}(r) \right\|_1$$

Radiance Fields vs 3D Surface Representations

- Compare/contrast with 3D meshes, voxels, point clouds, etc.
- Consider: from “perfect” synthetic data, can obtain well-defined GT for 3D surface (up to a resolution for discretized representations)
- Can we do the same with radiance fields?