CreativeAI

3D (Geometric) Domain

Niloy Mitra
UCL

Iasonas Kokkinos
UCL

Paul Guerrero
UCL

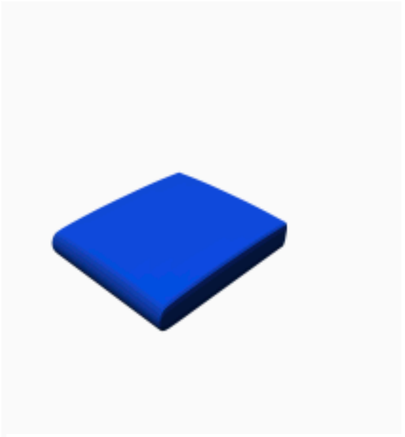
Nils Thuerey
TUM

Tobias Ritschel
UCL

Timetable

			Niloy	Paul	Nils
Theory + Basics	Introduction	2:15 pm	X	X	X
	Machine Learning Basics	~ 2:25 pm	X		
	Neural Network Basics	~ 2:55 pm			X
	Feature Visualization	~ 3:25 pm		X	
	Alternatives to Direct Supervision	~ 3:35 pm		X	
	15 min. break				
State of the Art	Image Domains	4:15 pm		X	
	3D Domains	~ 4:40 pm	X		
	Motion and Physics	~ 5:05 pm			X
	Discussion	~ 5:30 pm	X	X	X

Application #1: 3D Modeling



Deep neural network predicts the **next best part** to add and its **position** to enable non-expert users to create novel shapes.

[Sung et al. 2017]

Application #1: 3D Modeling

Deep neural network predicts the **next best part** to add and its **position** to enable non-expert users to create novel shapes.

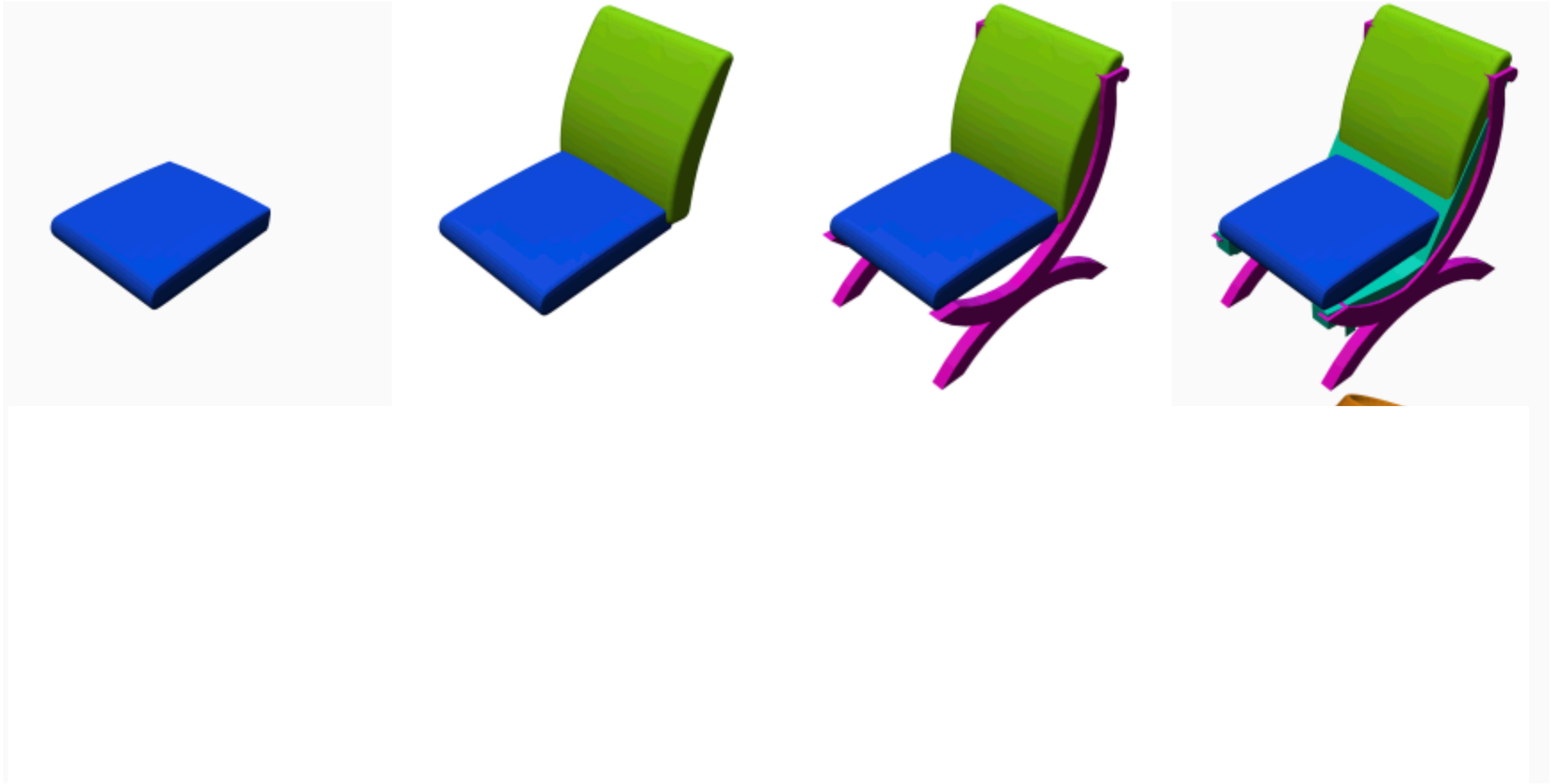
[Sung et al. 2017]

Application #1: 3D Modeling

Deep neural network predicts the **next best part** to add and its **position** to enable non-expert users to create novel shapes.

[Sung et al. 2017]

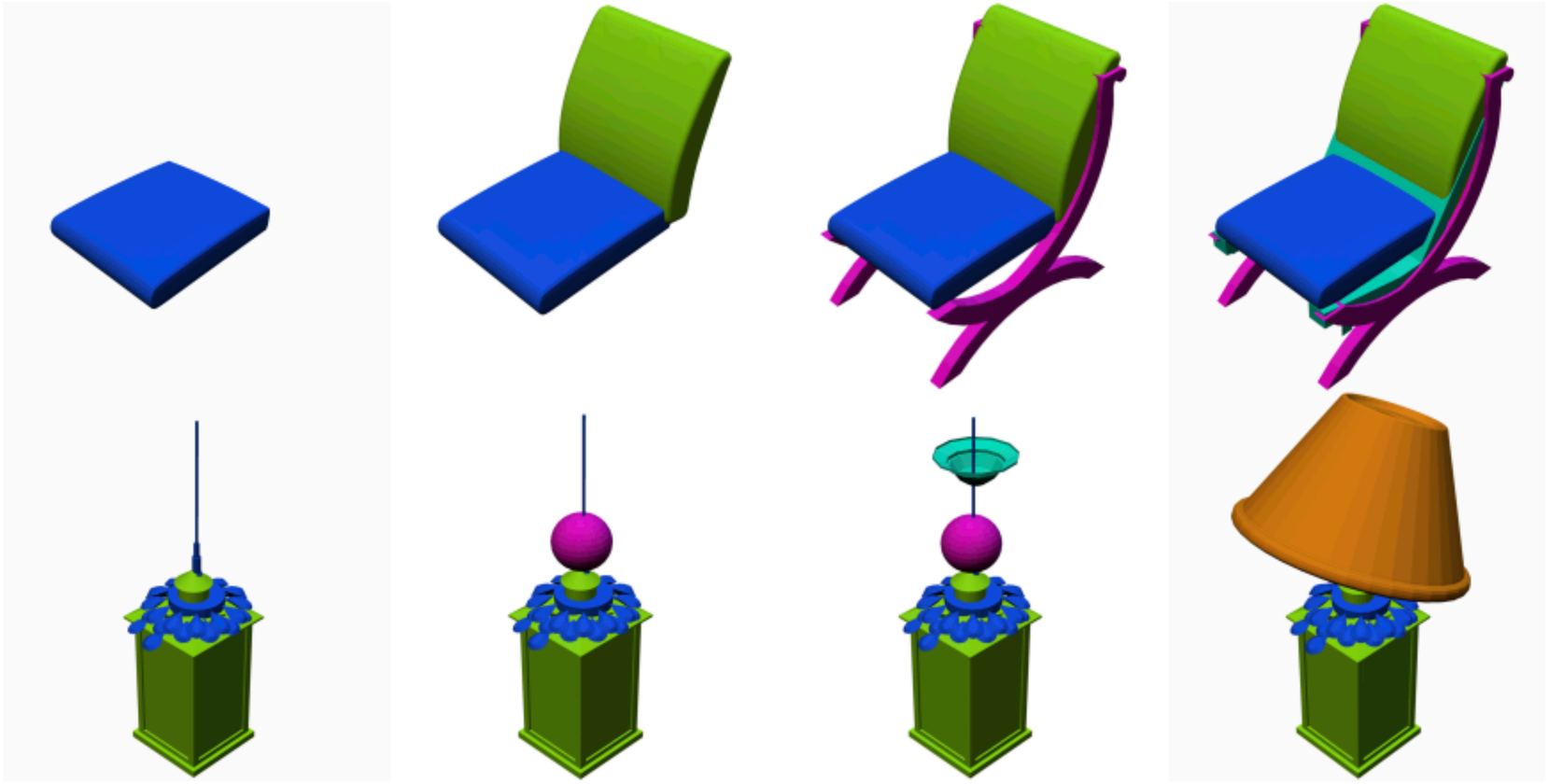
Application #1: 3D Modeling



Deep neural network predicts the **next best part** to add and its **position** to enable non-expert users to create novel shapes.

[Sung et al. 2017]

Application #1: 3D Modeling



Deep neural network predicts the **next best part** to add and its **position** to enable non-expert users to create novel shapes.

[Sung et al. 2017]

Application #2: Image Understanding

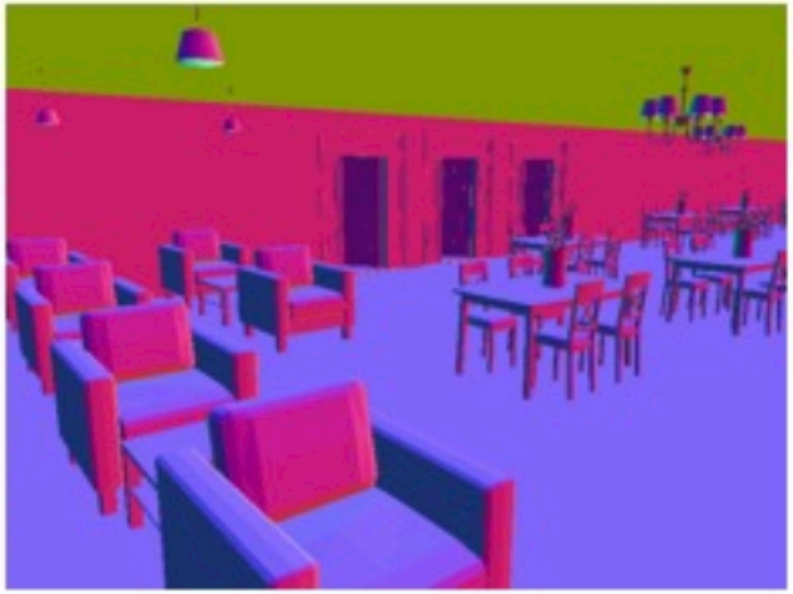
understanding 3D shapes can benefit image understanding

[Zhang et al. 2017]

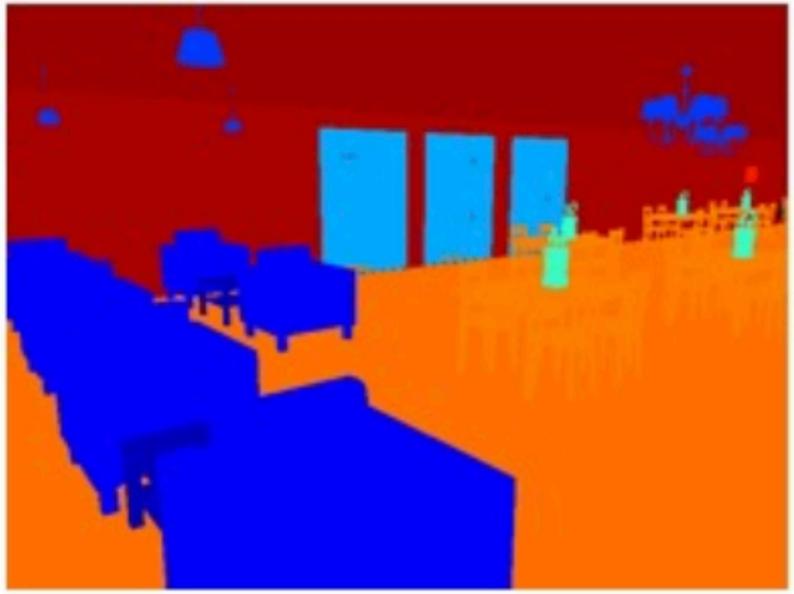
Application #2: Image Understanding

understanding 3D shapes can benefit image understanding

Physically based Rendering



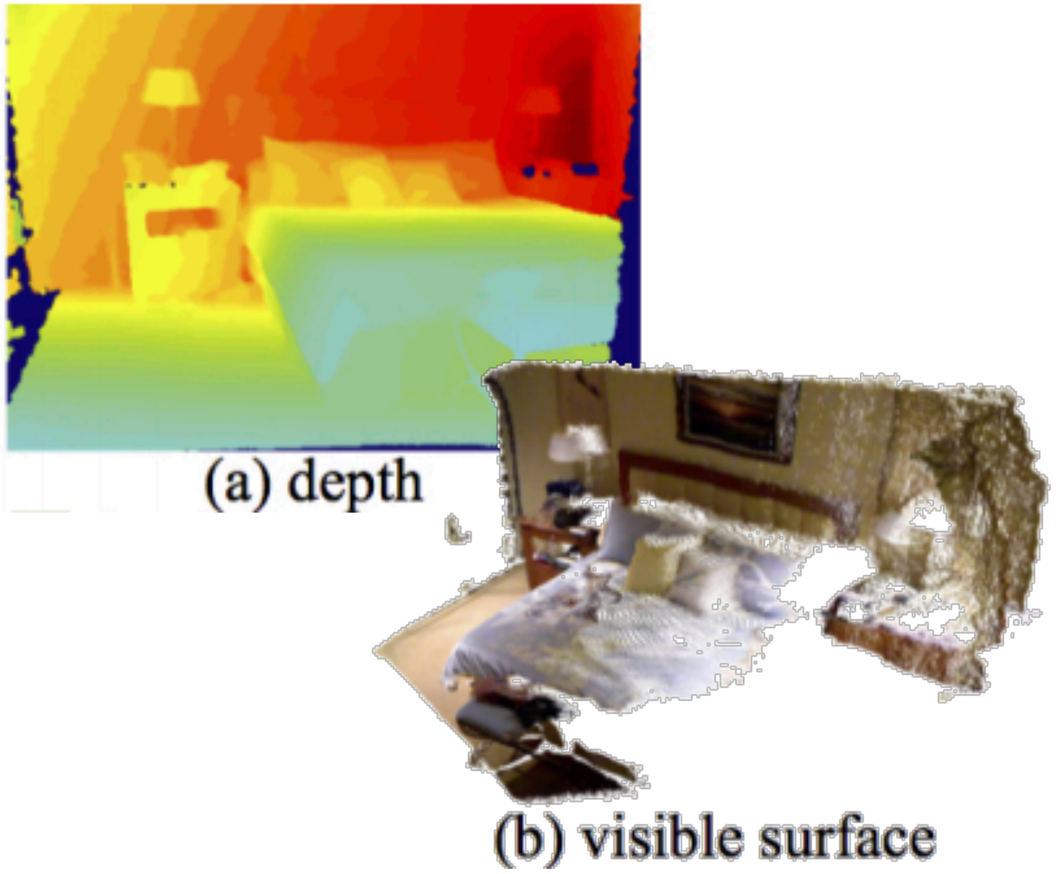
Surface Normal

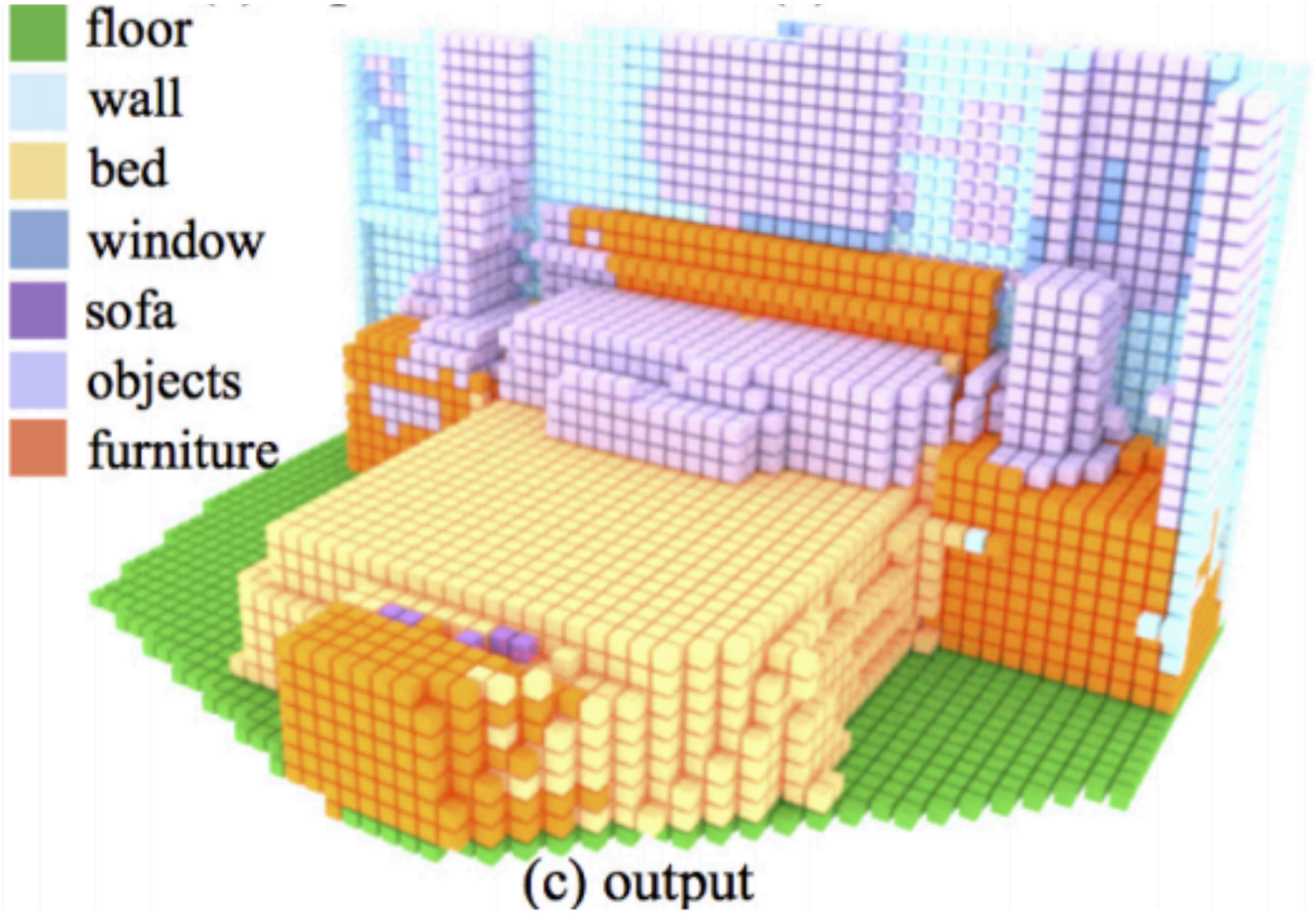


Semantic Segmentation

[Zhang et al. 2017]

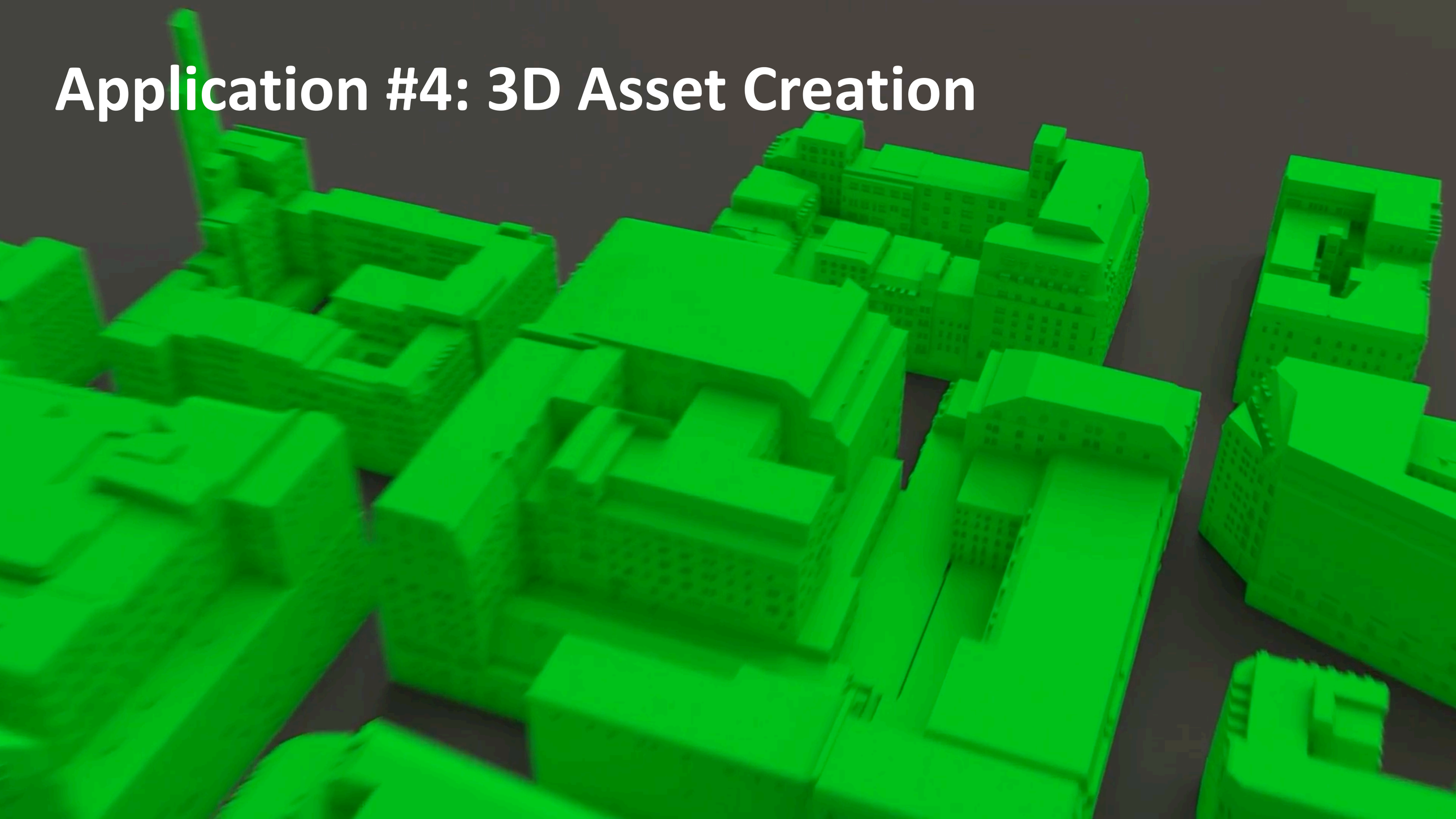
Application #3: Semantic Scene Understanding



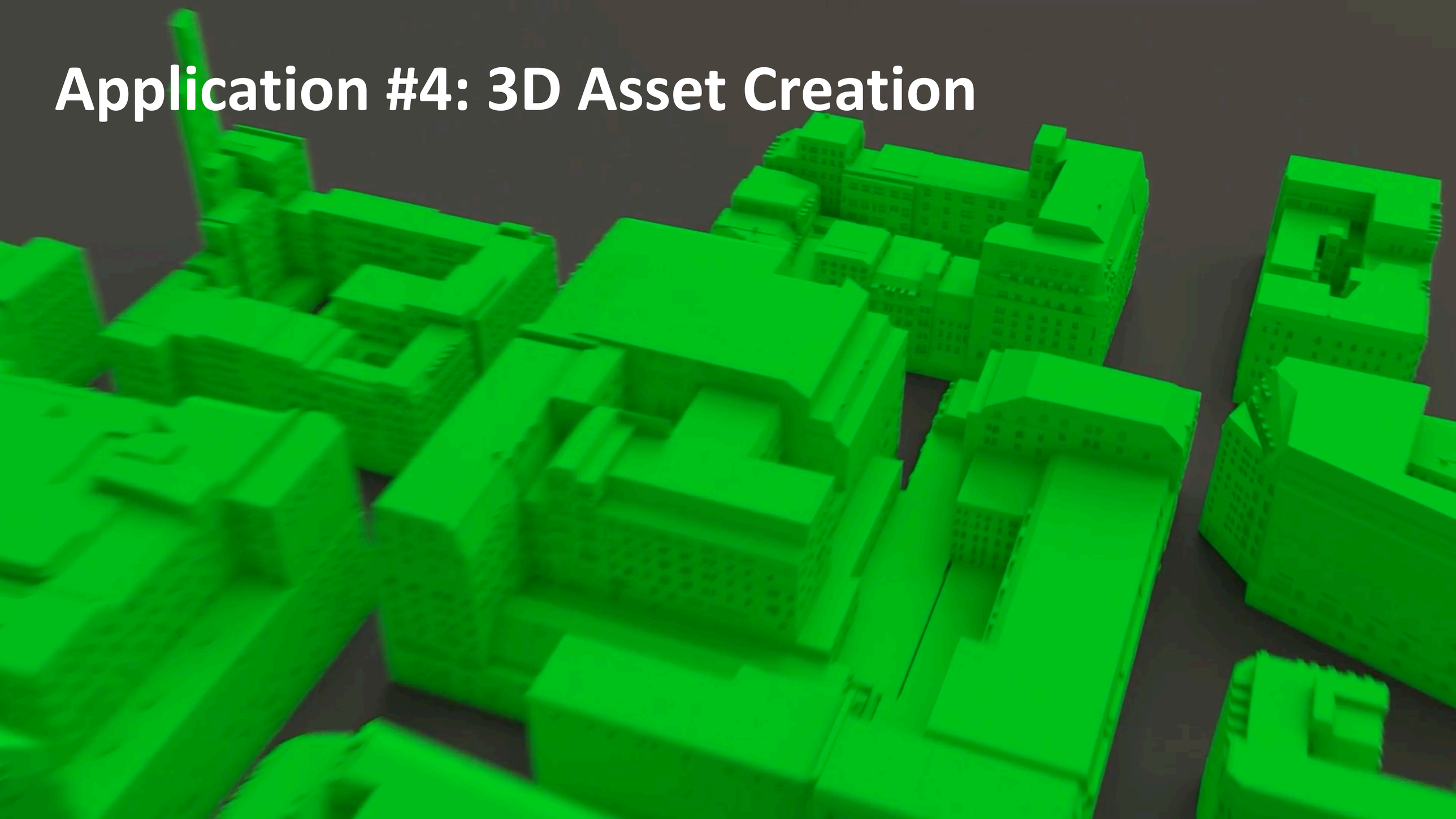


[Song et al. 2017]

Application #4: 3D Asset Creation



Application #4: 3D Asset Creation



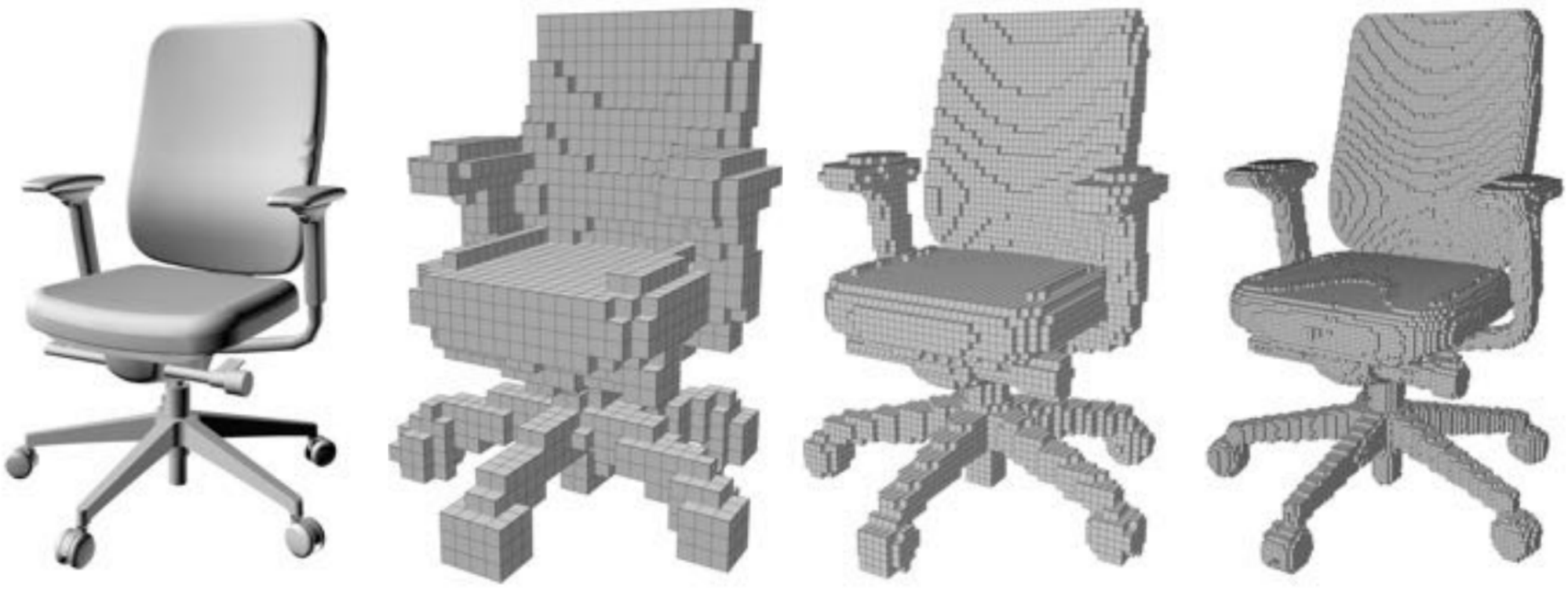
What’s Different in 3D?

What's Different in 3D?

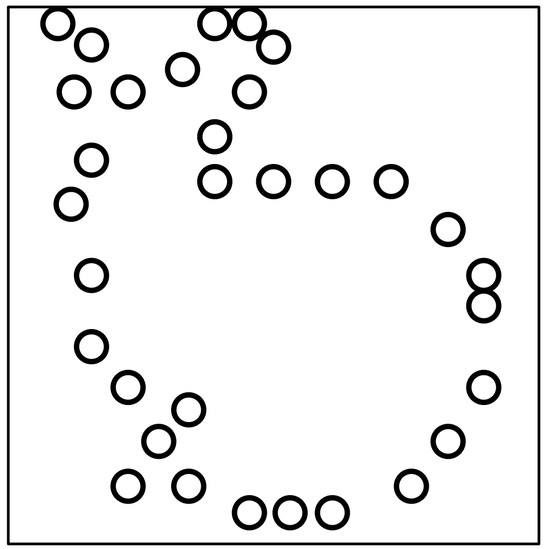
- Number of Voxels grows as $O(n^3)$ versus occupied **surface** $O(n^2)$

What's Different in 3D?

- Number of Voxels grows as $O(n^3)$ versus occupied **surface** $O(n^2)$

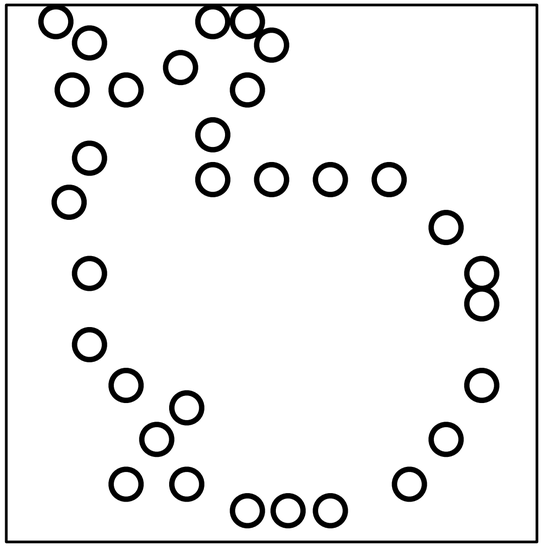


Data Representation .. Many Possibilities!

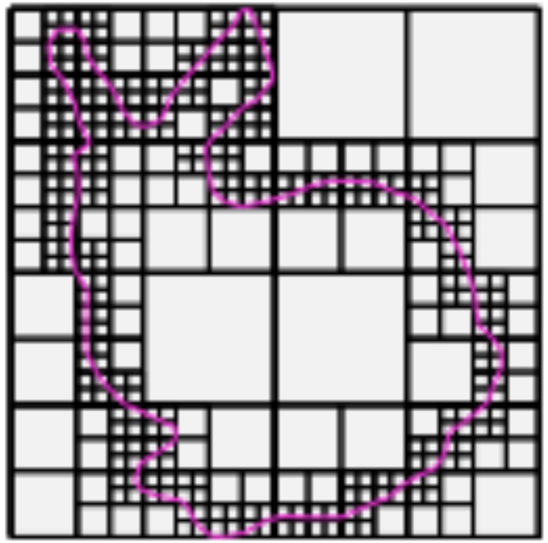


points

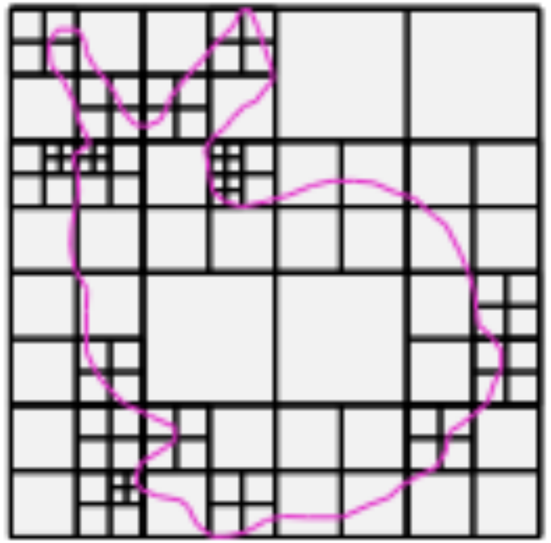
Data Representation .. Many Possibilities!



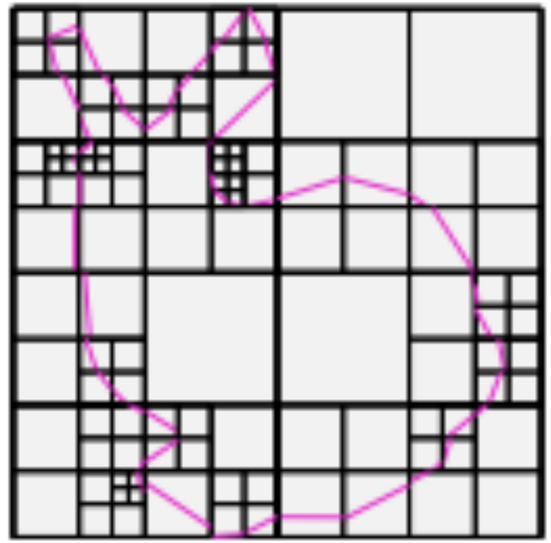
points



voxels



cells



patches

Challenges

Challenges

1. Representation

Challenges

1. Representation

2. **Neighborhood** information
   - who are the neighbouring elements
   - how are the elements ordered

Challenges

1. Representation

2. **Neighborhood** information
   - who are the neighbouring elements
   - how are the elements ordered

3. **Extrinsic** versus **intrinsic** representation

Challenges

1. Representation

2. **Neighborhood** information
   - who are the neighbouring elements
   - how are the elements ordered

3. **Extrinsic** versus **intrinsic** representation

4. Simplicity versus memory/runtime tradeoff

Representation for 3D

- Image-based
- Volumetric
- Surface-based
- Point-based

Representation for 3D

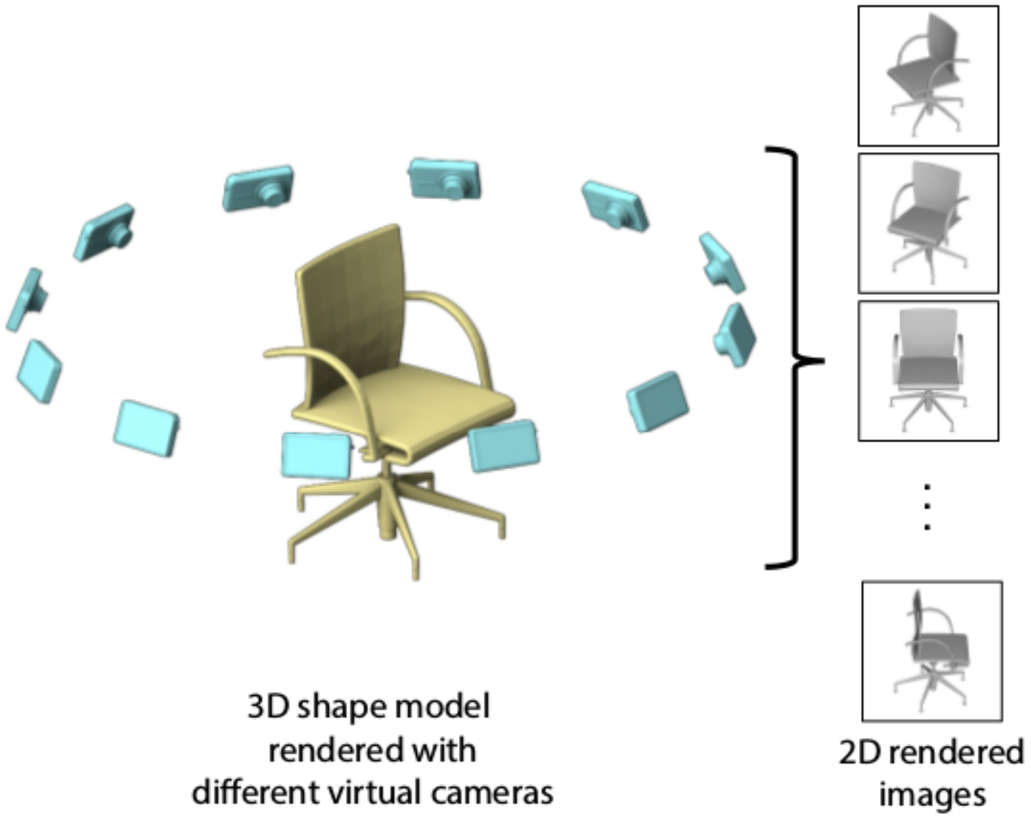
- **Image-based**
- Volumetric
- Surface-based
- Point-based

Representation for 3D: Multi-view CNN

3D shape model
rendered with
different virtual cameras

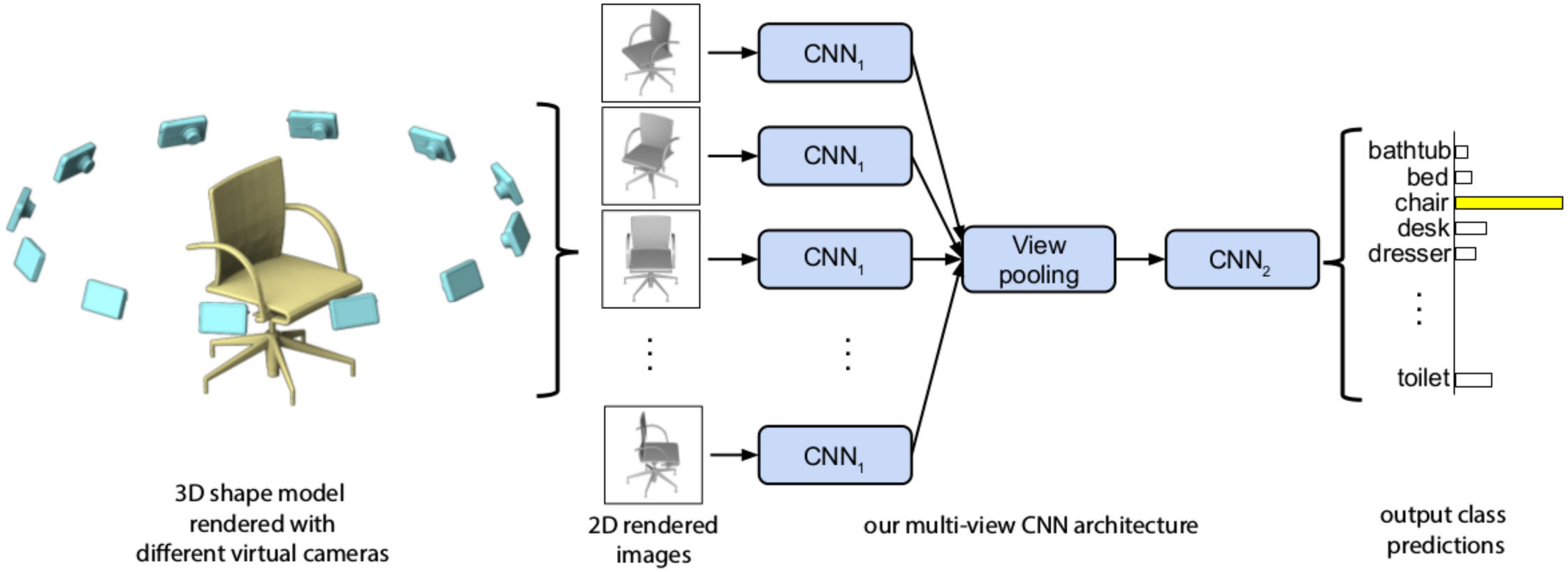
[Kalogerakis et al. 2015]

Representation for 3D: Multi-view CNN



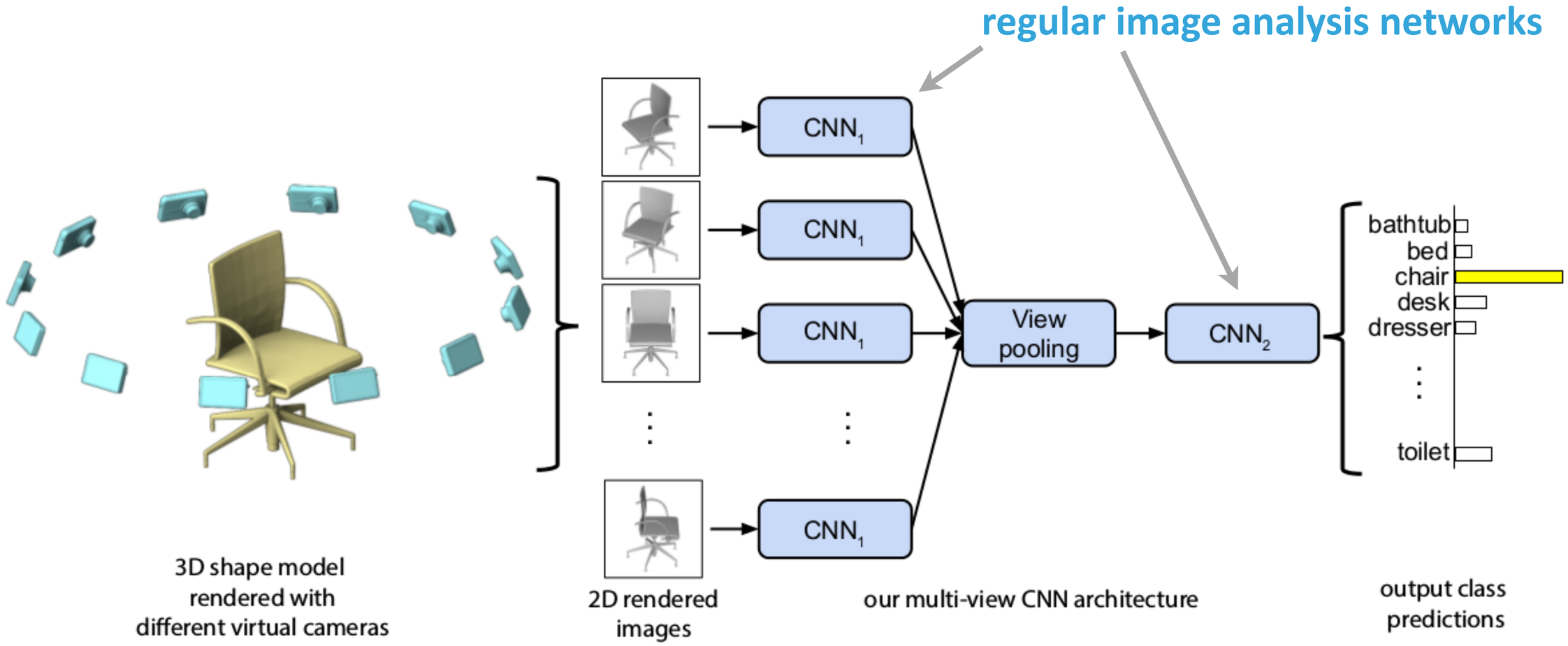
[Kalogerakis et al. 2015]

Representation for 3D: Multi-view CNN



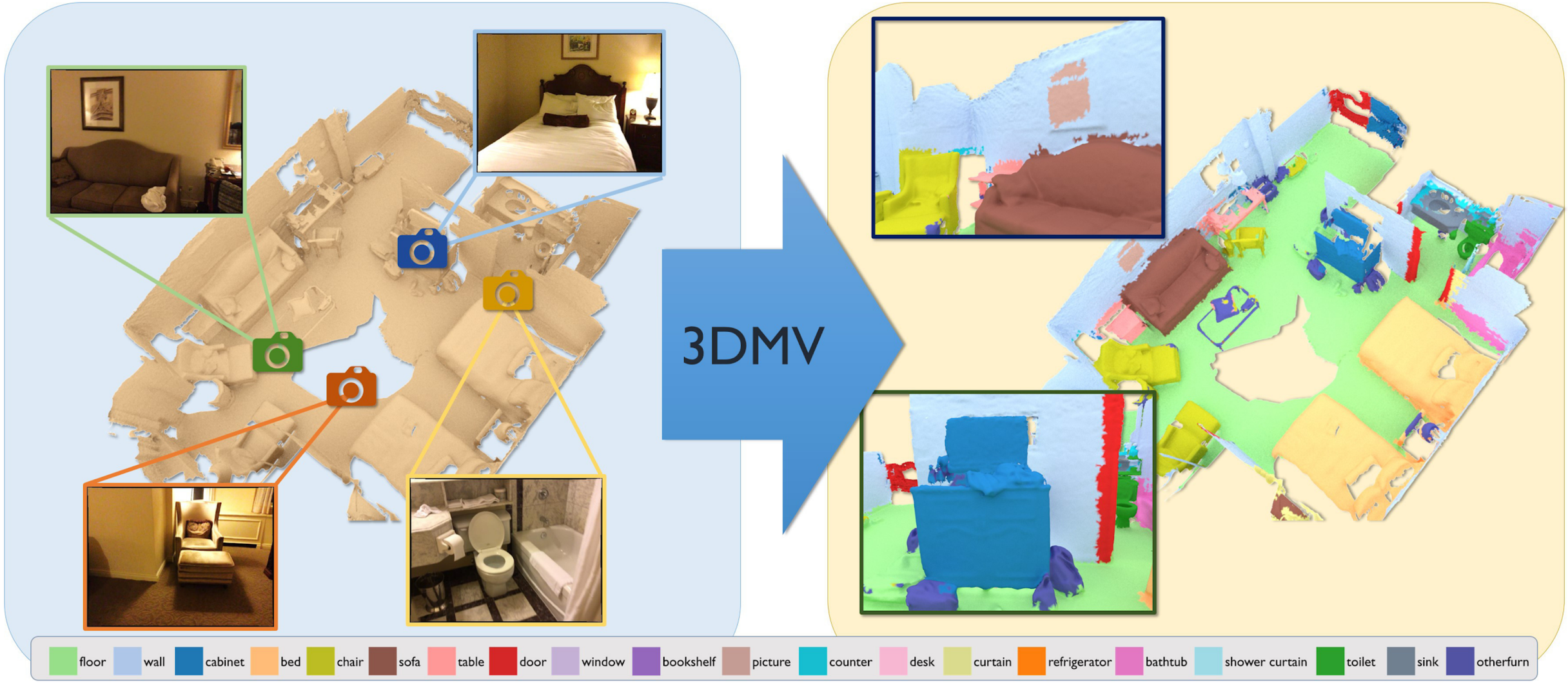
[Kalogerakis et al. 2015]

Representation for 3D: Multi-view CNN

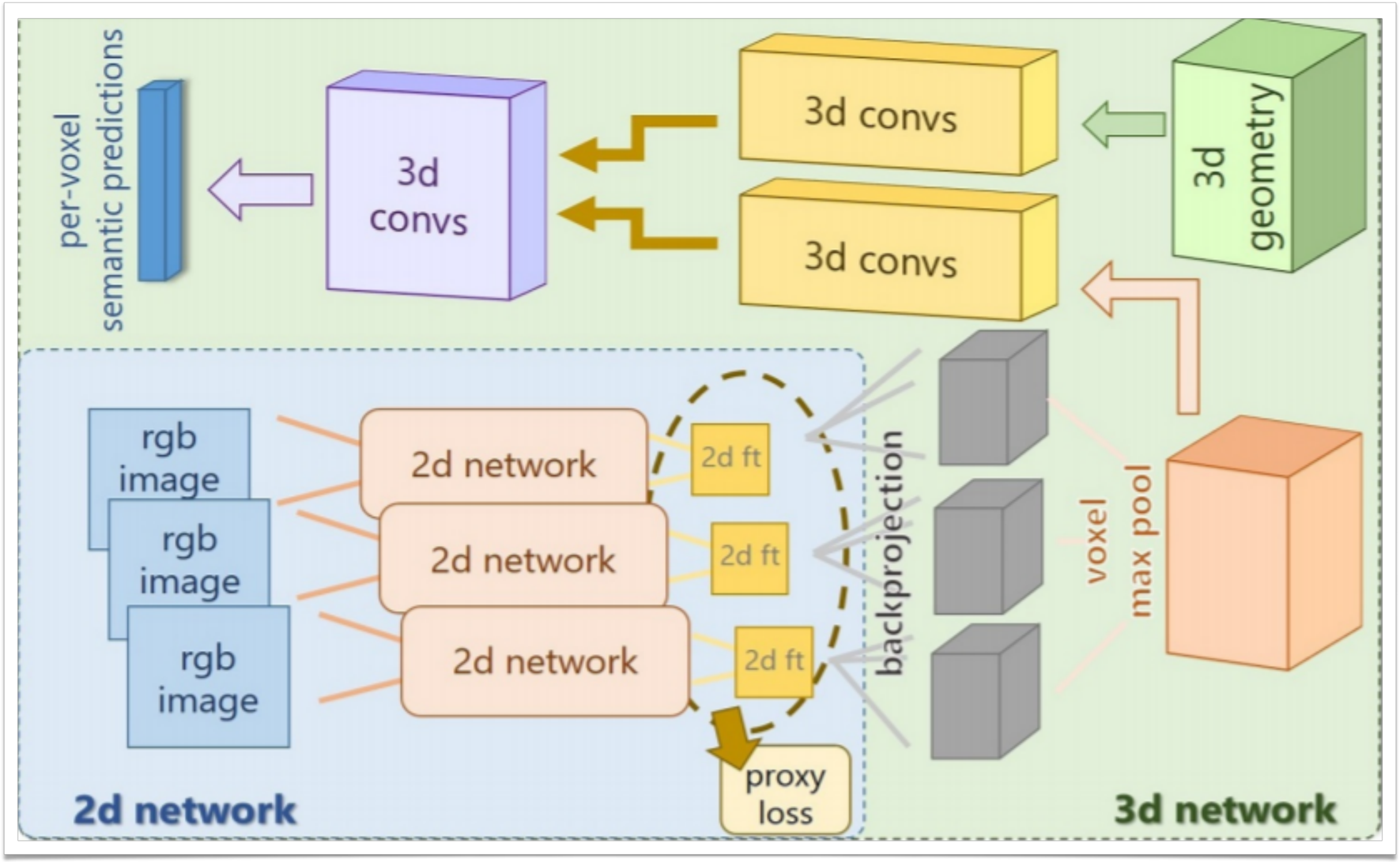


[Kalogerakis et al. 2015]

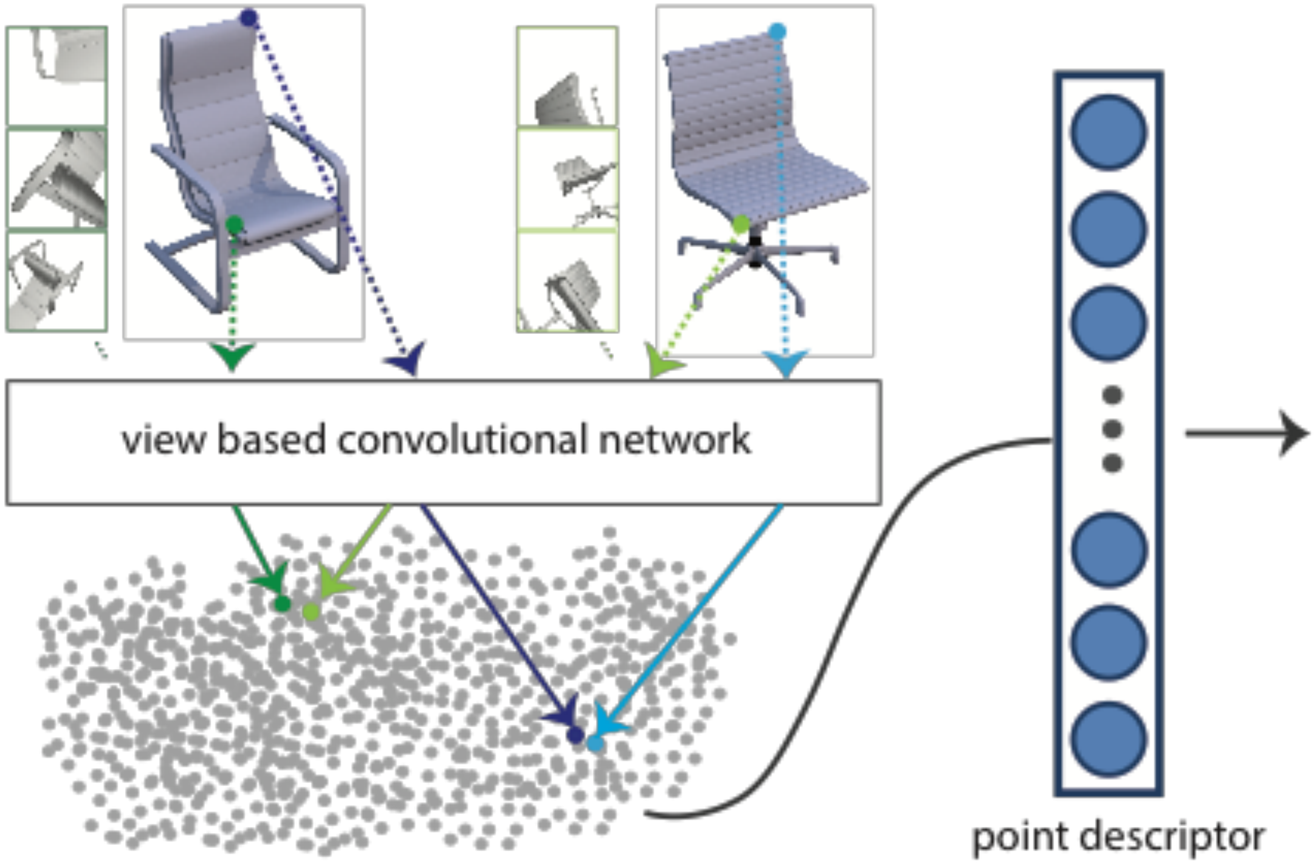
3DMV: Joint 3D-Multi-View Prediction for 3D Semantic Scene Segmentation



Integrating View Information

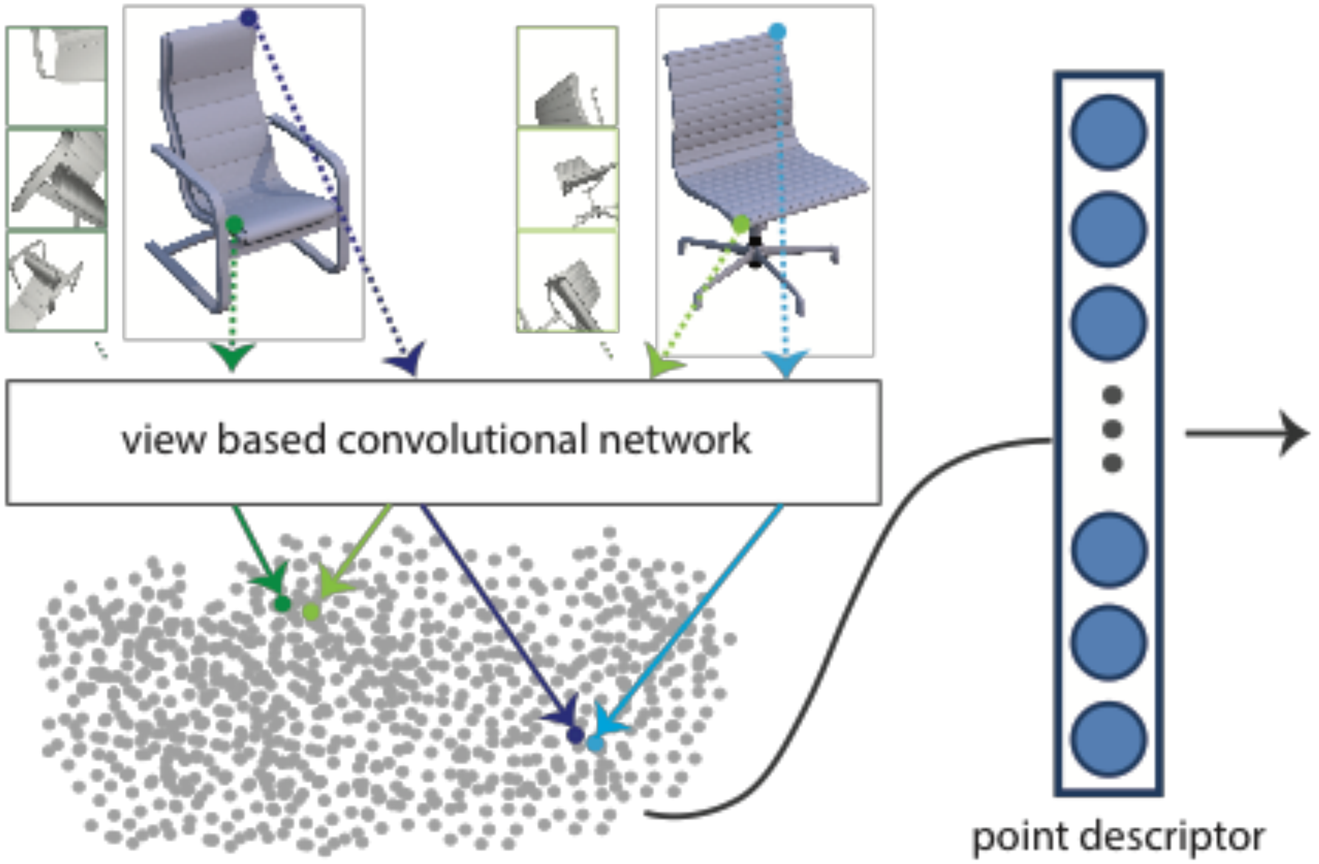


Representation for 3D: Local Multi-view CNN



[Huang et al. 2018]

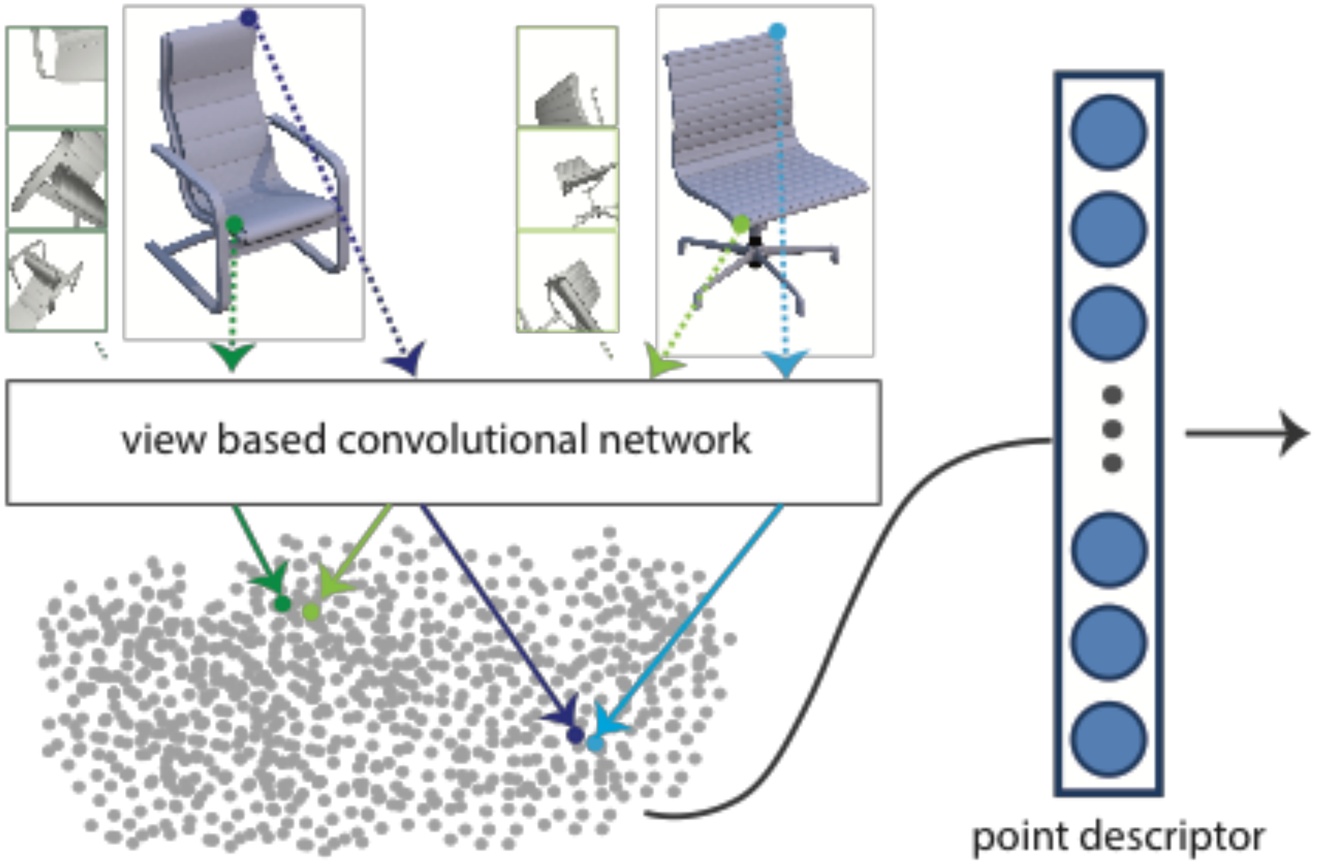
Representation for 3D: Local Multi-view CNN



Segmentation
Correspondence
Feature matching
Predicting semantic functions

[Huang et al. 2018]

Representation for 3D: Local Multi-view CNN

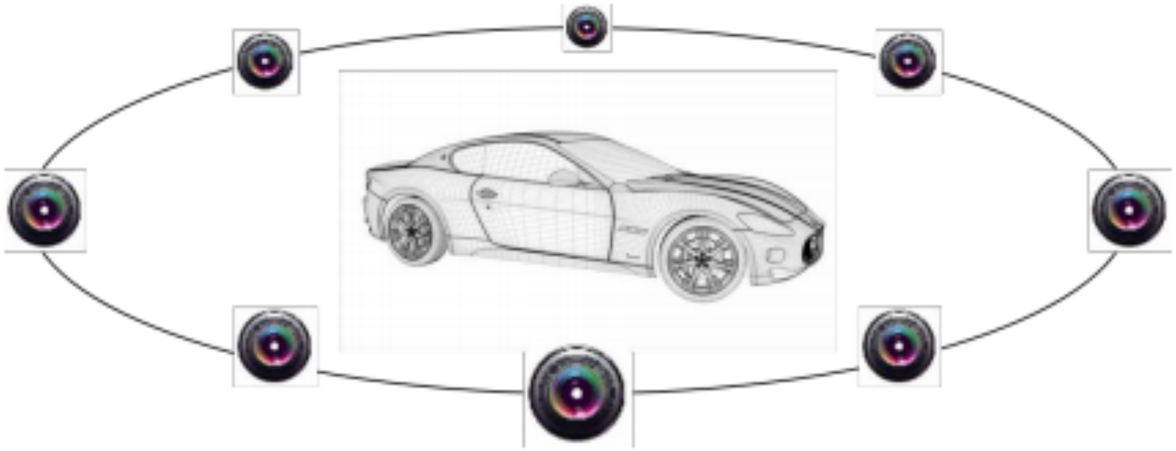


Segmentation
Correspondence
Feature matching
Predicting semantic functions

localized renderings for point-wise features

[Huang et al. 2018]

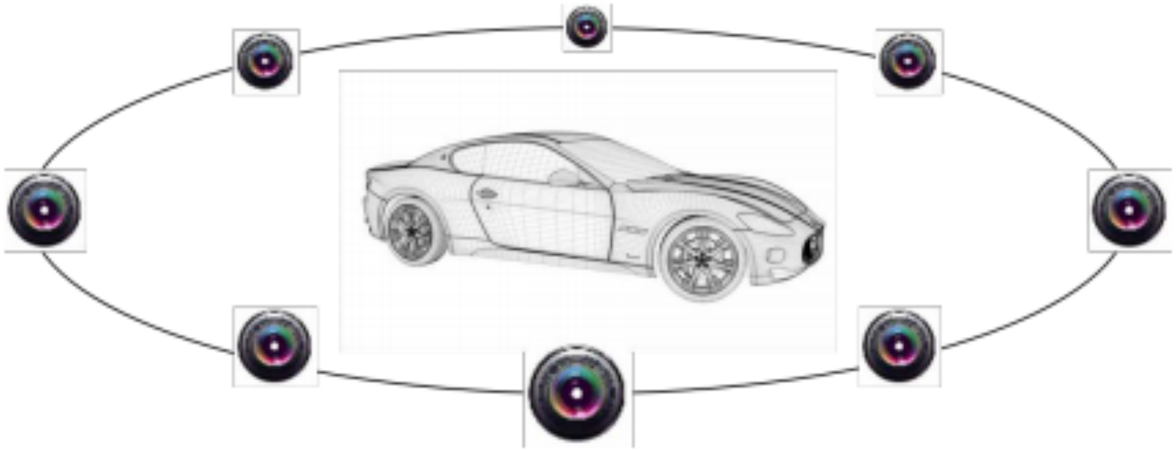
Tangent Convolutions



loses information due to occlusion

[Tatarchenko et al. 2018]

Tangent Convolutions

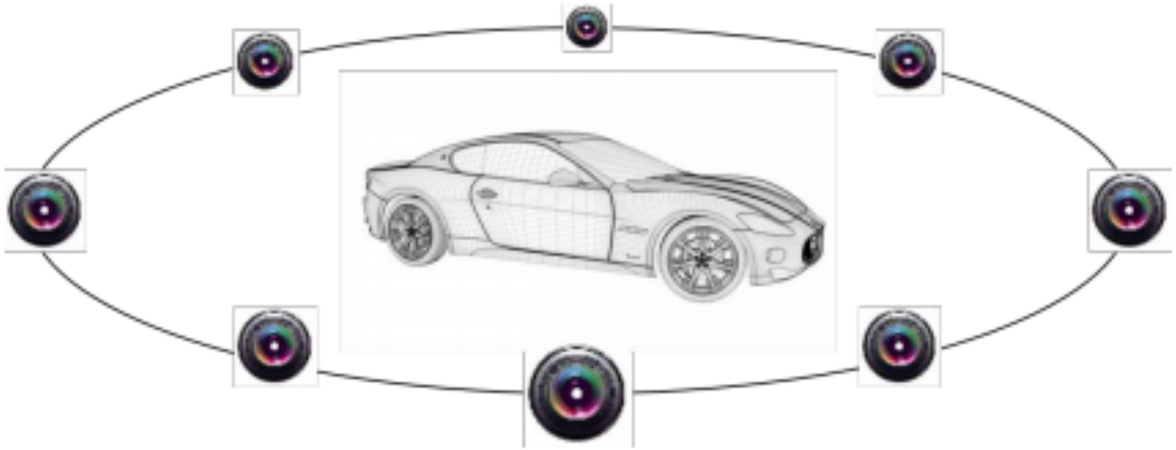


loses information due to occlusion

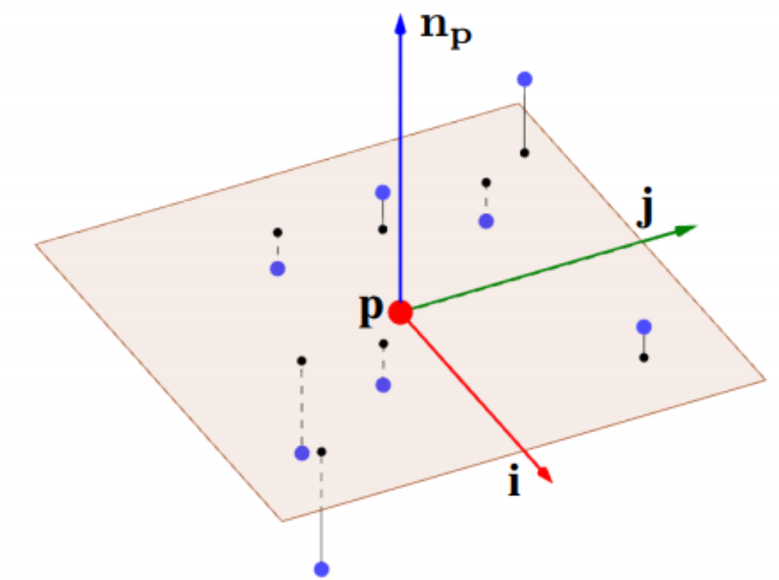
project to local patches
(contrast with PCPNet construction)

[Tatarchenko et al. 2018]

Tangent Convolutions



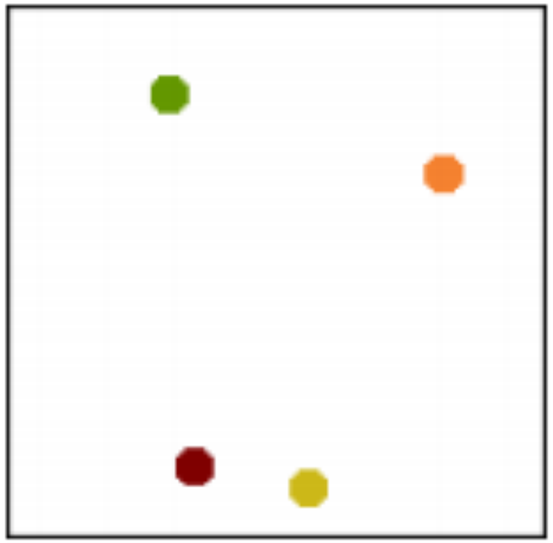
loses information due to occlusion



project to local patches
(contrast with PCPNet construction)

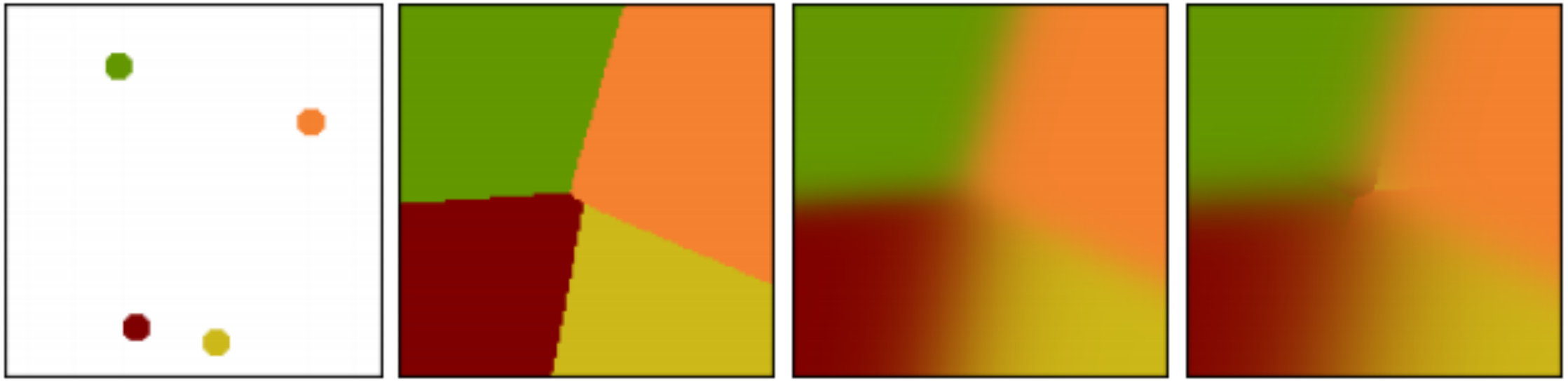
[Tatarchenko et al. 2018]

Dealing with Sparse Points



(a)

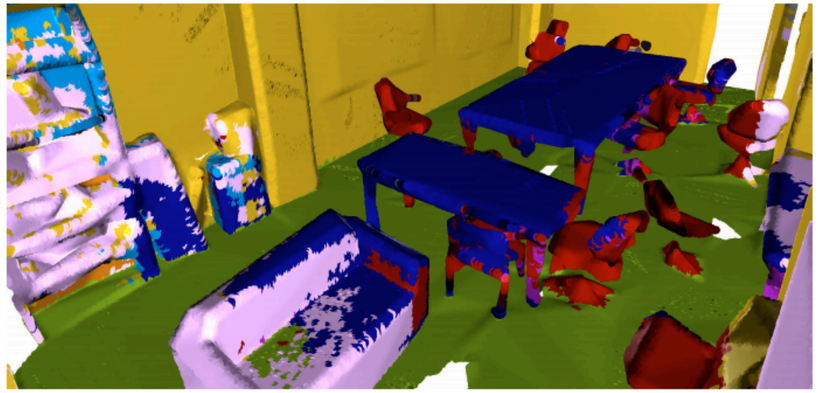
Dealing with Sparse Points



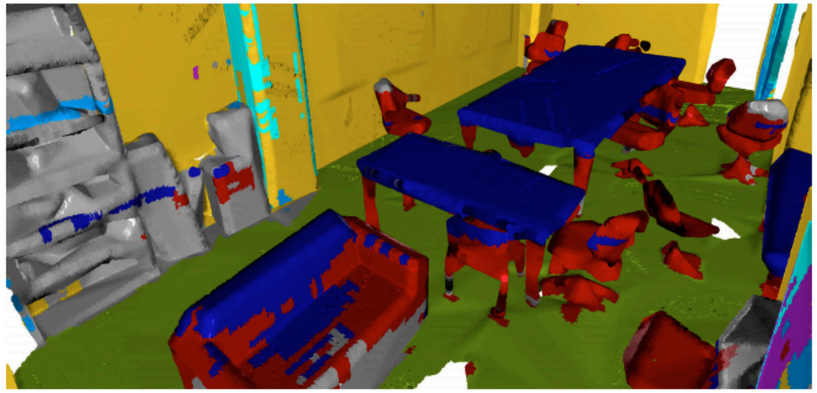
(a) (b) (c) (d)

Improved Performance

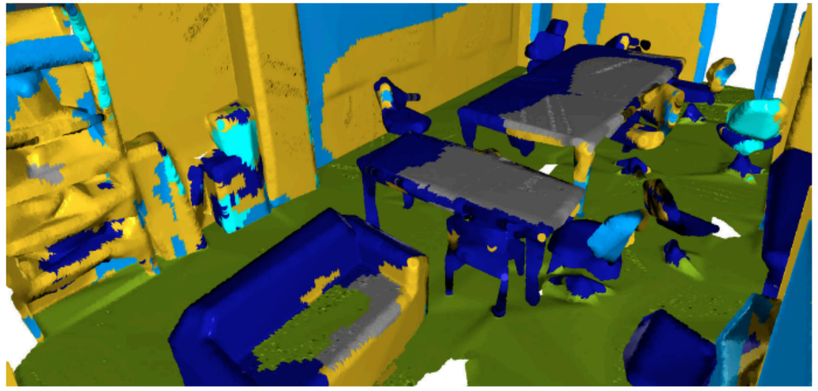
Color



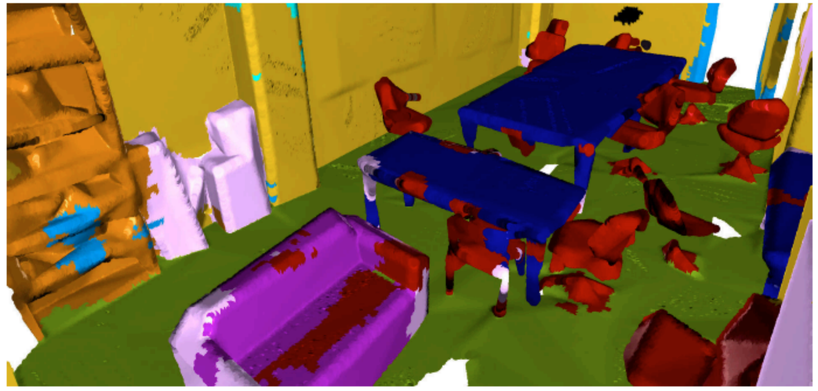
PointNet [39]



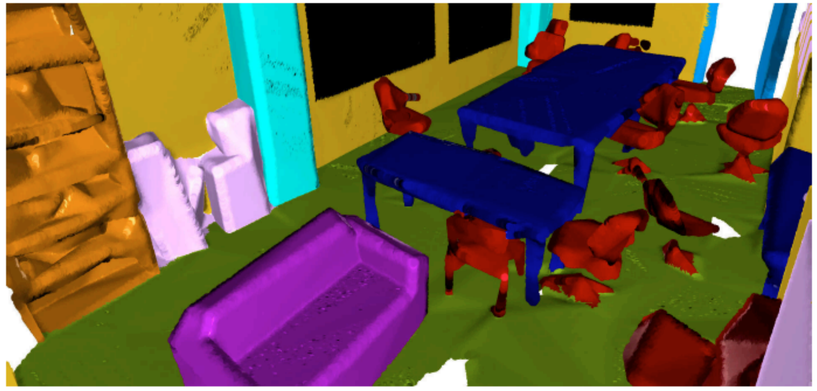
ScanNet [10]



OctNet [43]



Ours (DHNRGB)



Ground truth

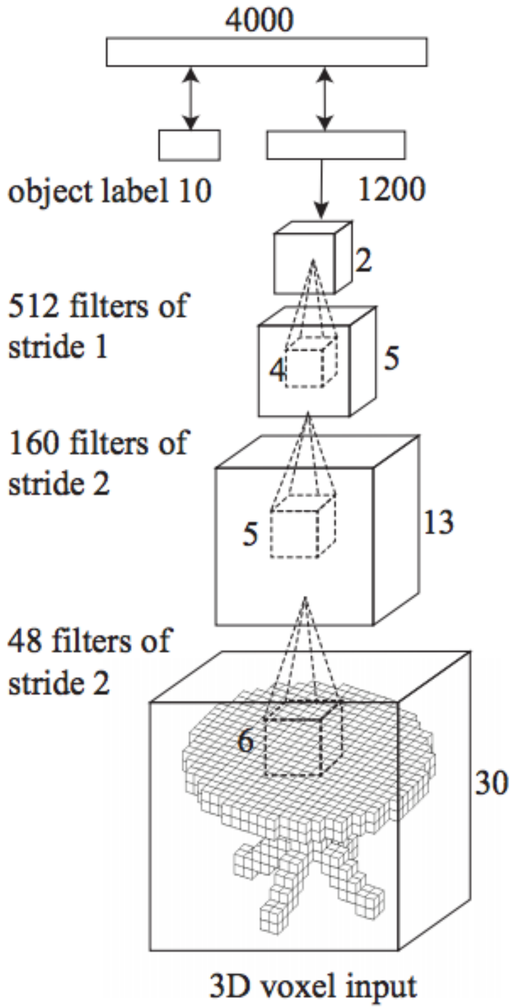
Representation for 3D

- **Image-based**
  - **PROS**: directly use image networks, good performance
  - **CONS**: rendering is slow and memory-heavy, not very geometric
- Volumetric
- Point-based
- Surface-based

Representation for 3D

- Image-based
- **Volumetric**
- Surface-based
- Point-based

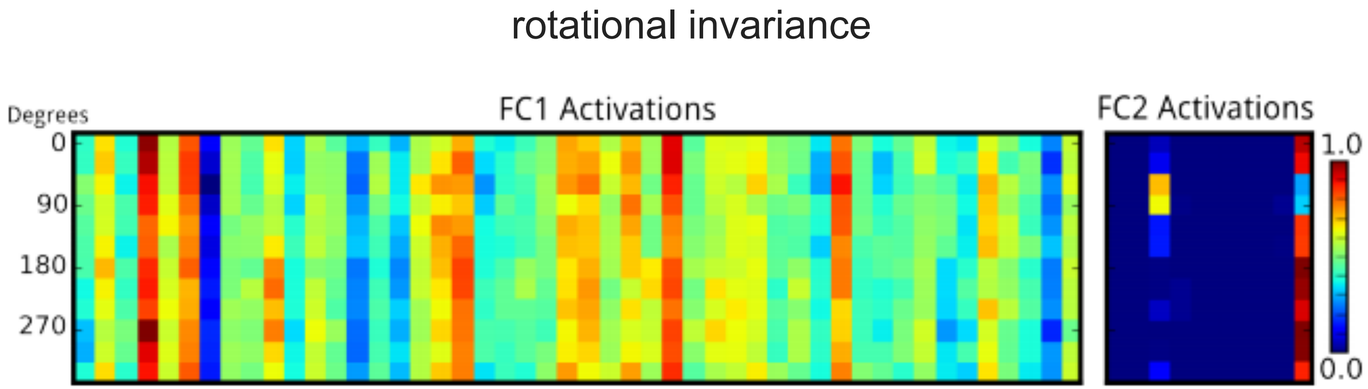
3D CNNs : Direct Approach

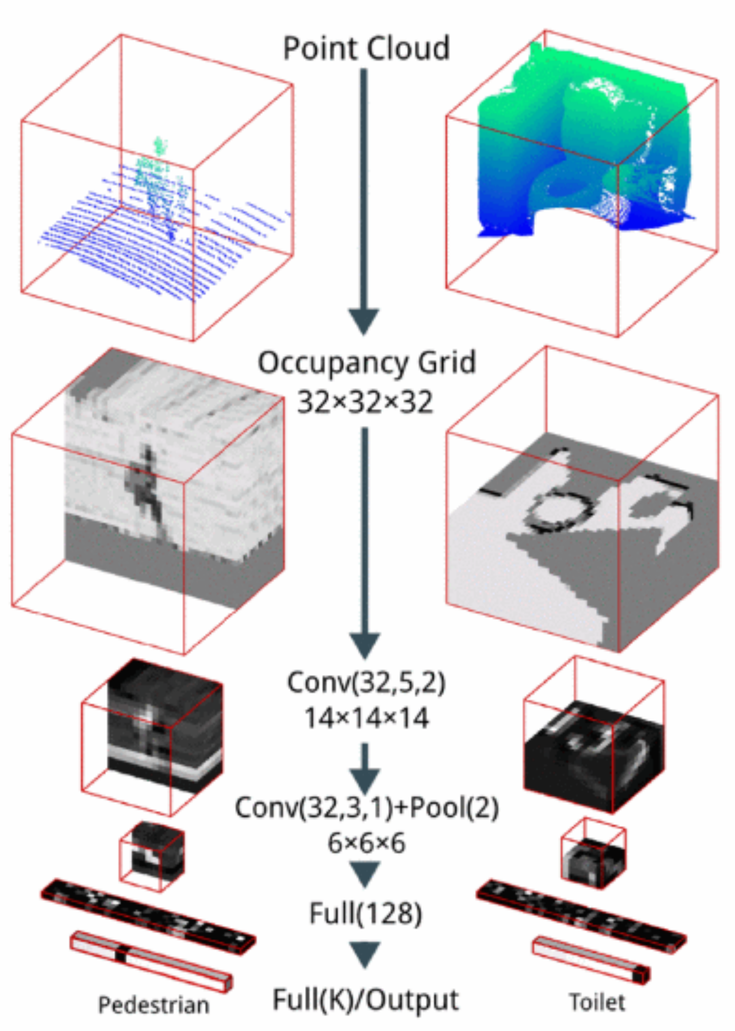


[Xiao et al. 2014]

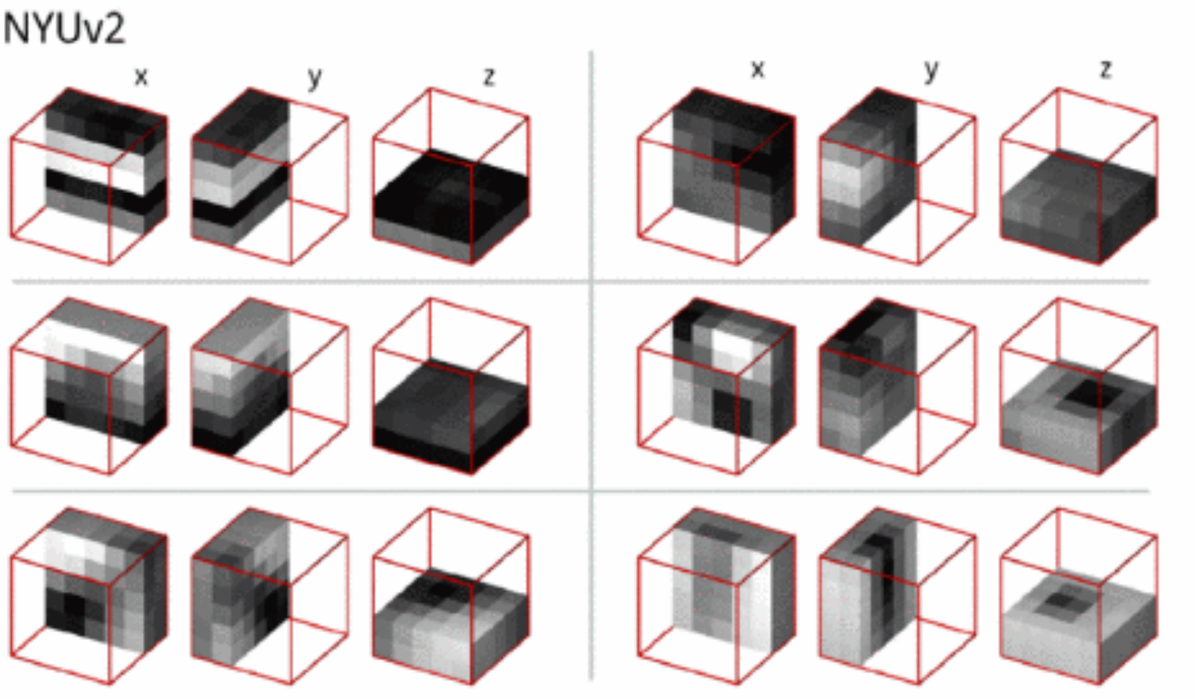
VoxNet [Maturana et al. 15]

- Binary occupancy, density grid, etc.

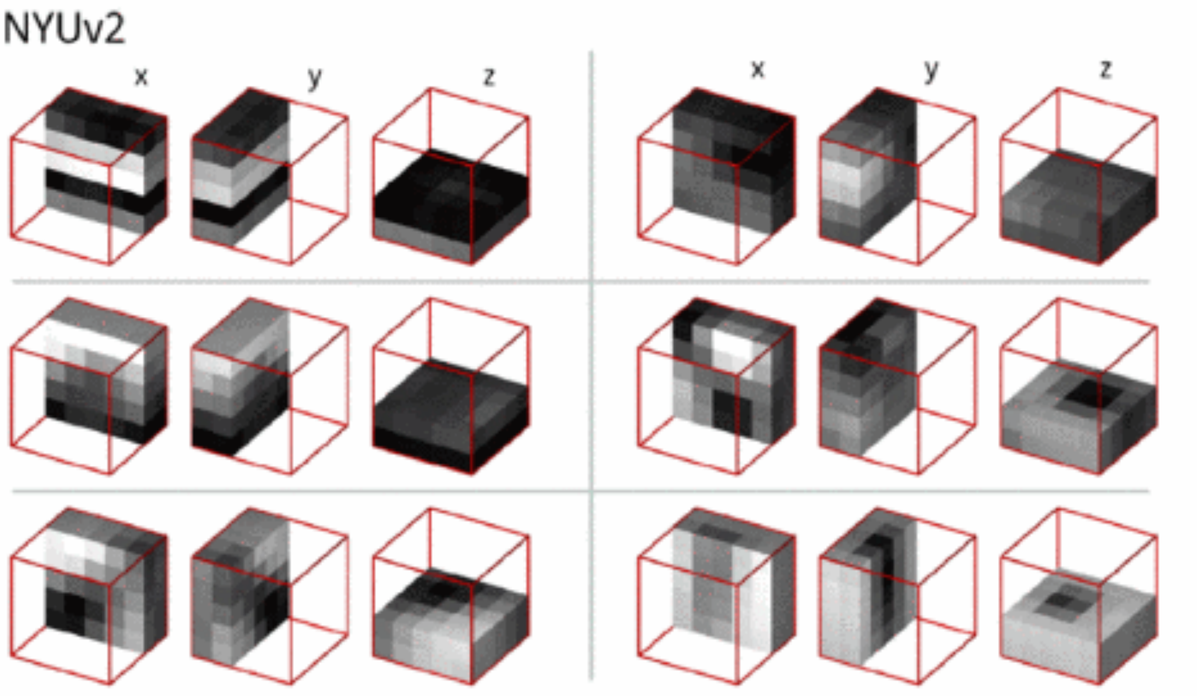


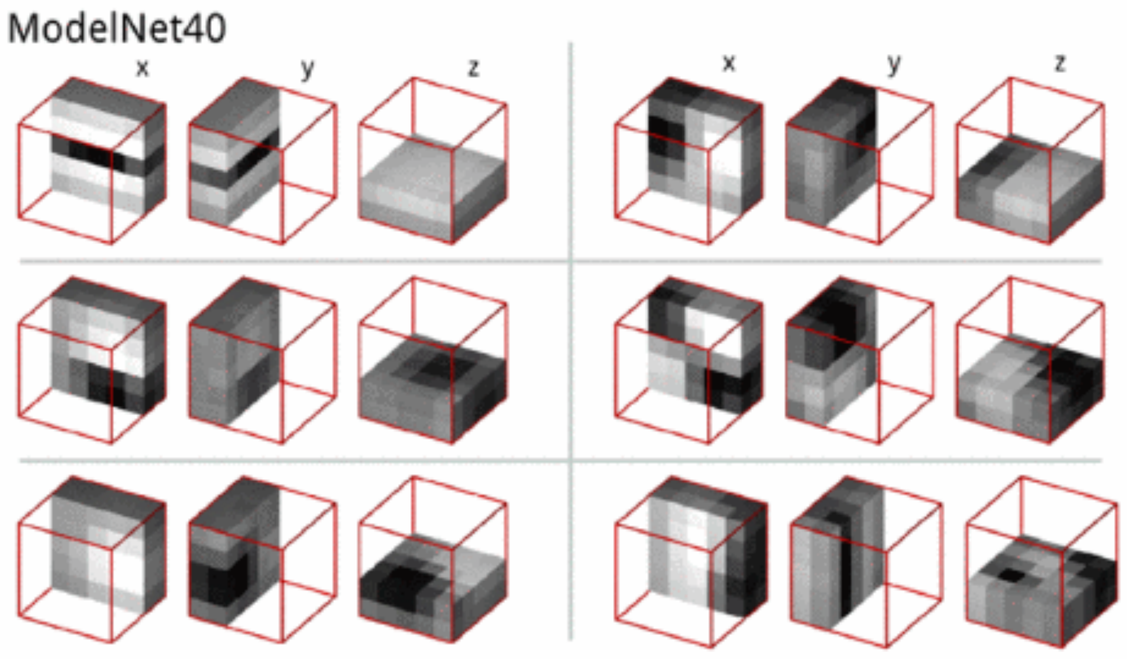


Visualization of First Level Filters

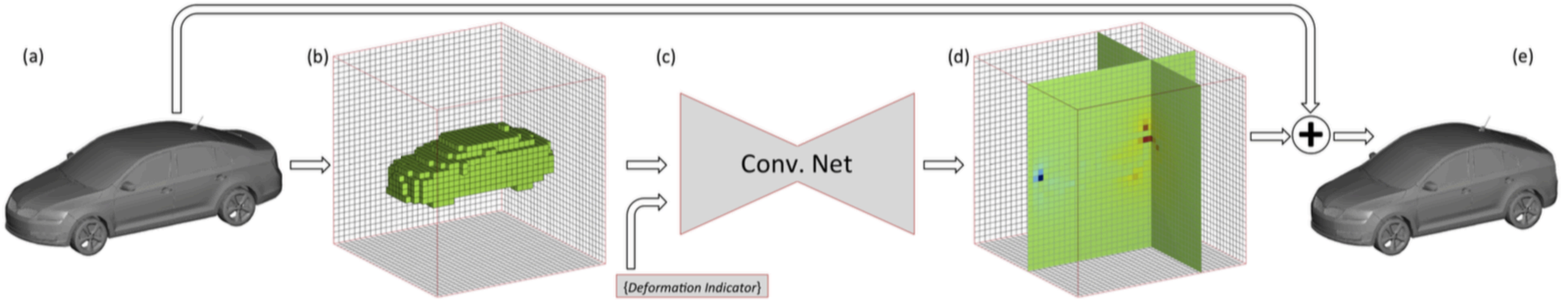


Visualization of First Level Filters



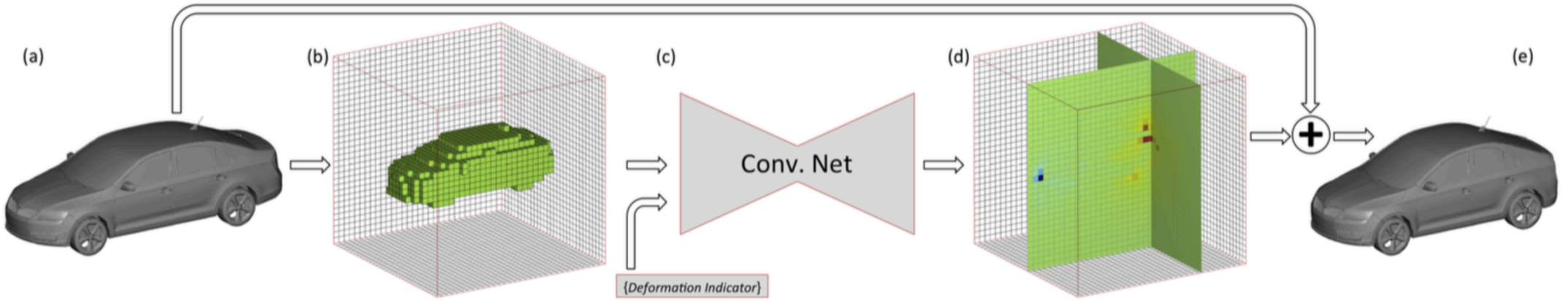


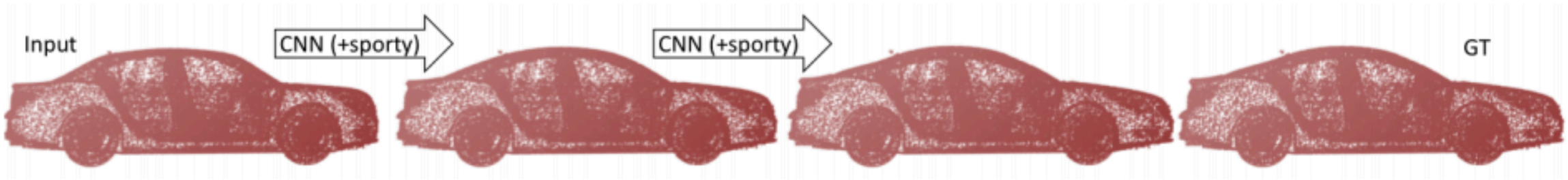
Representation for 3D: Volumetric Deformation



[Yumer and Mitra 2016]

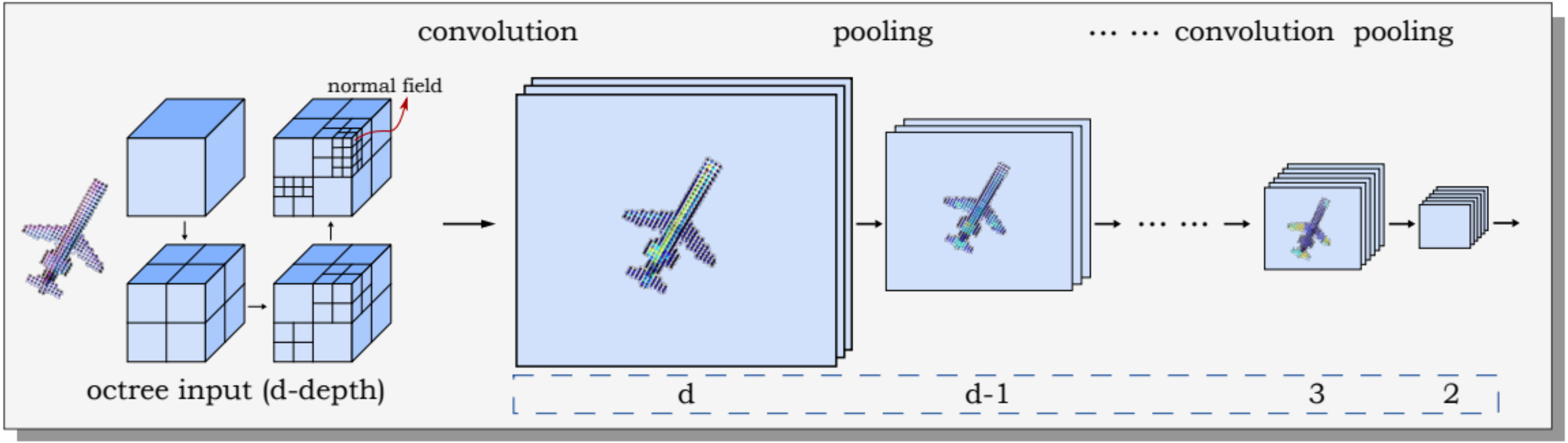
Representation for 3D: Volumetric Deformation





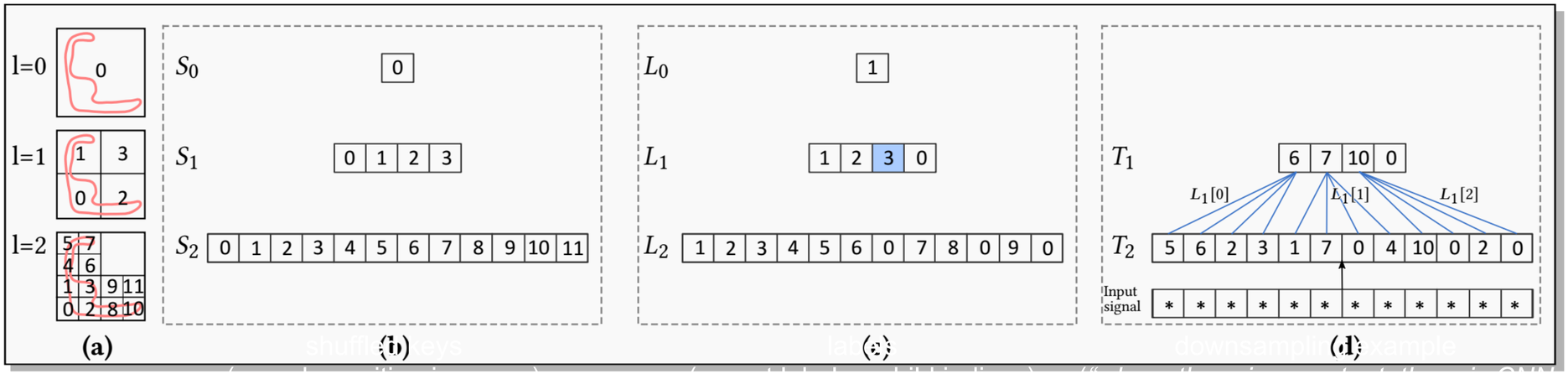
[Yumer and Mitra 2016]

Efficient Volumetric Datastructures

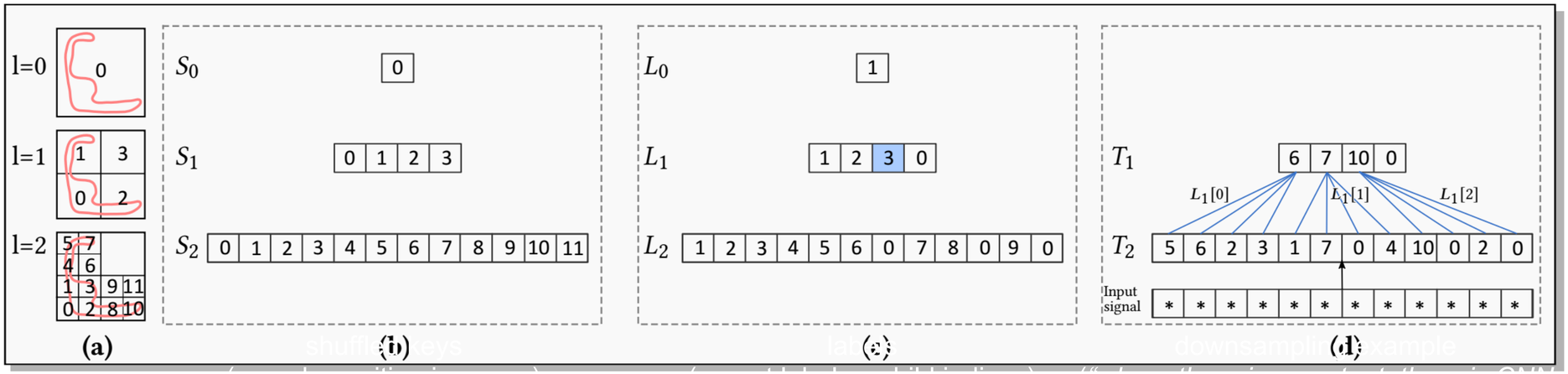


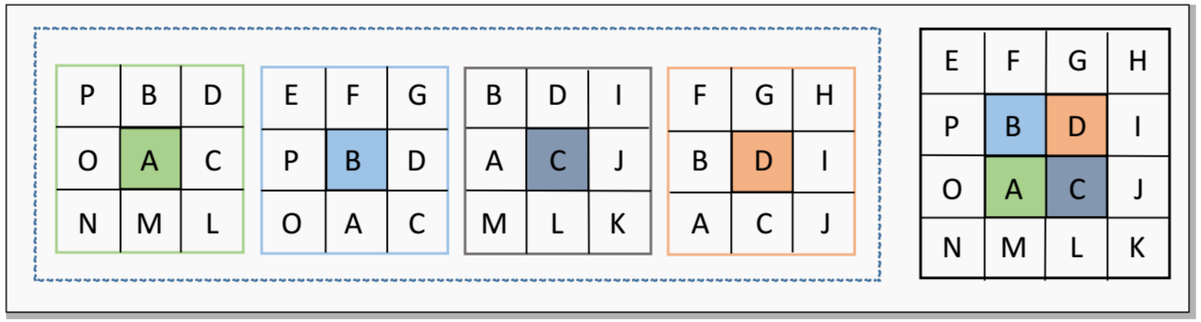
[Wang et al. 2017]

Data Structure and CNN Operations

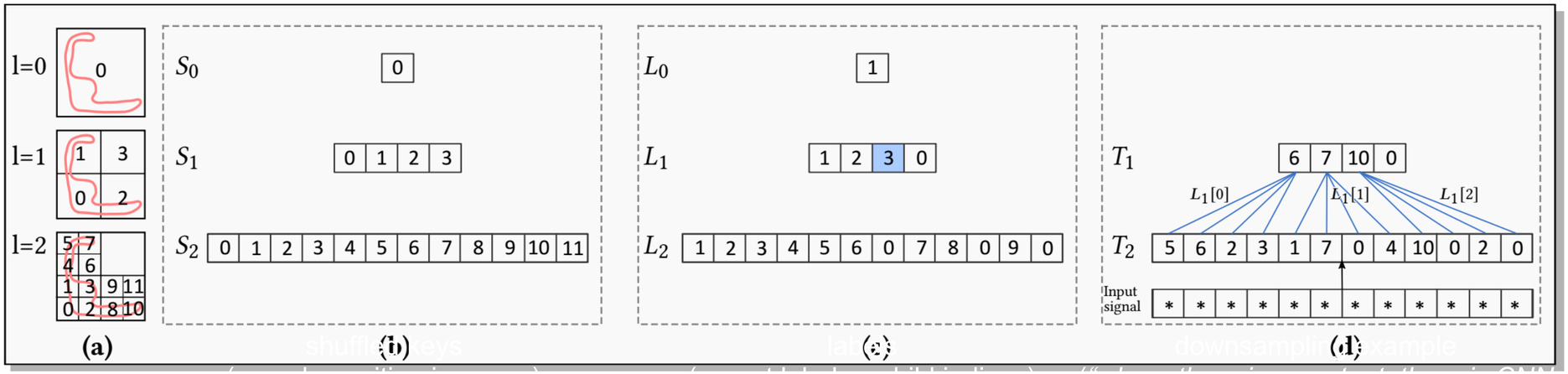


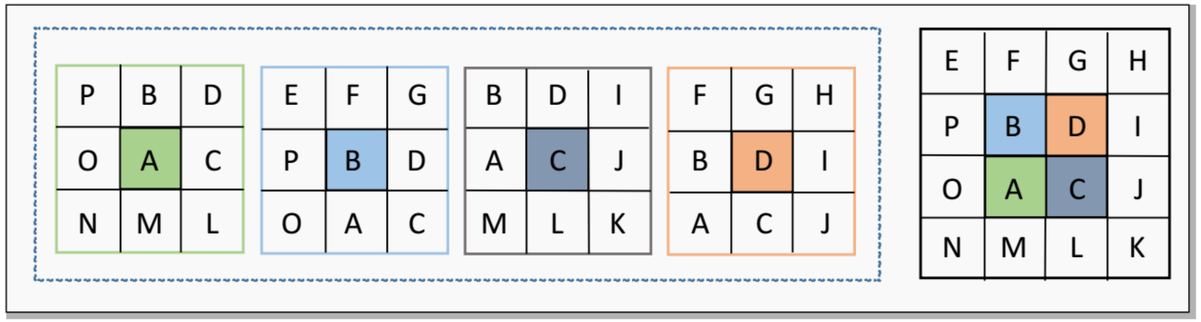
Data Structure and CNN Operations

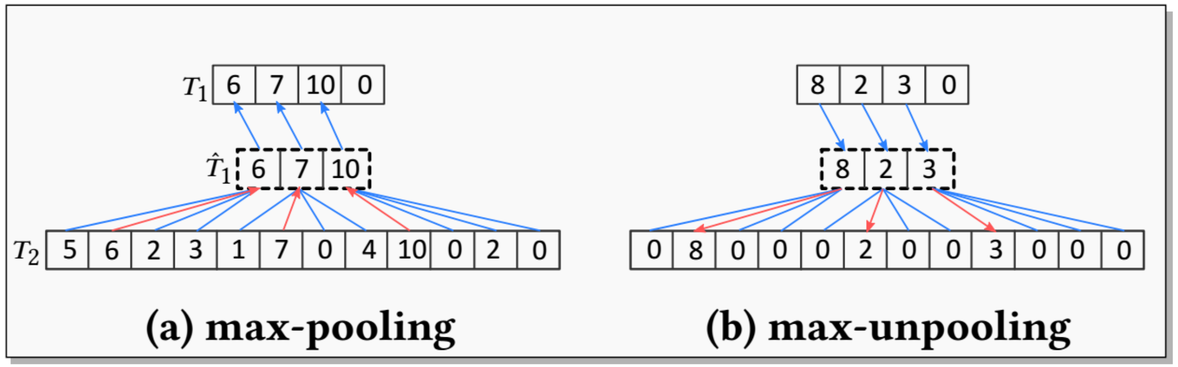




Data Structure and CNN Operations

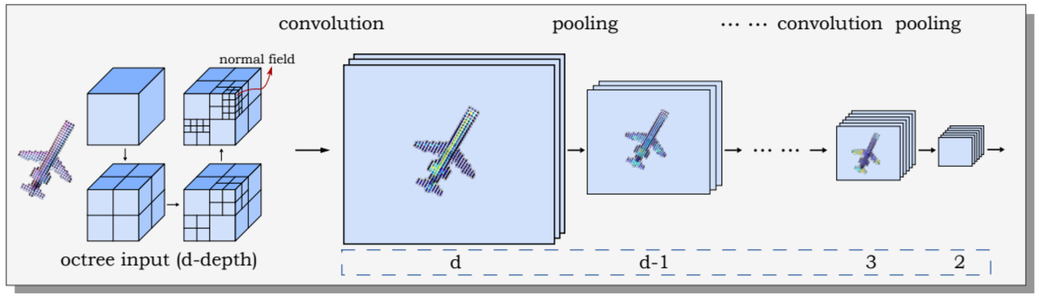






Efficient Volumetric Datastructures

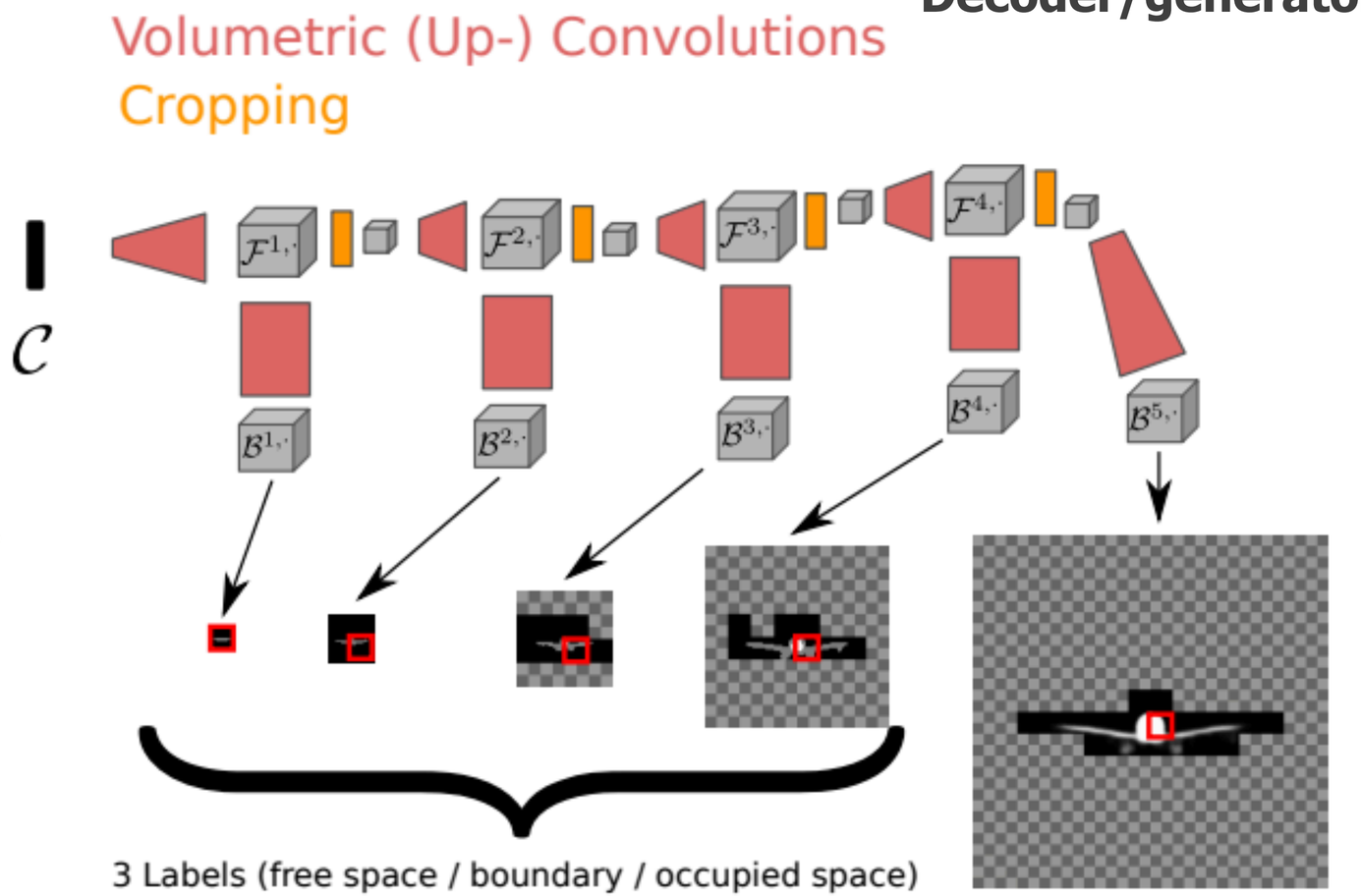
Encoder



Wang et al. 2017

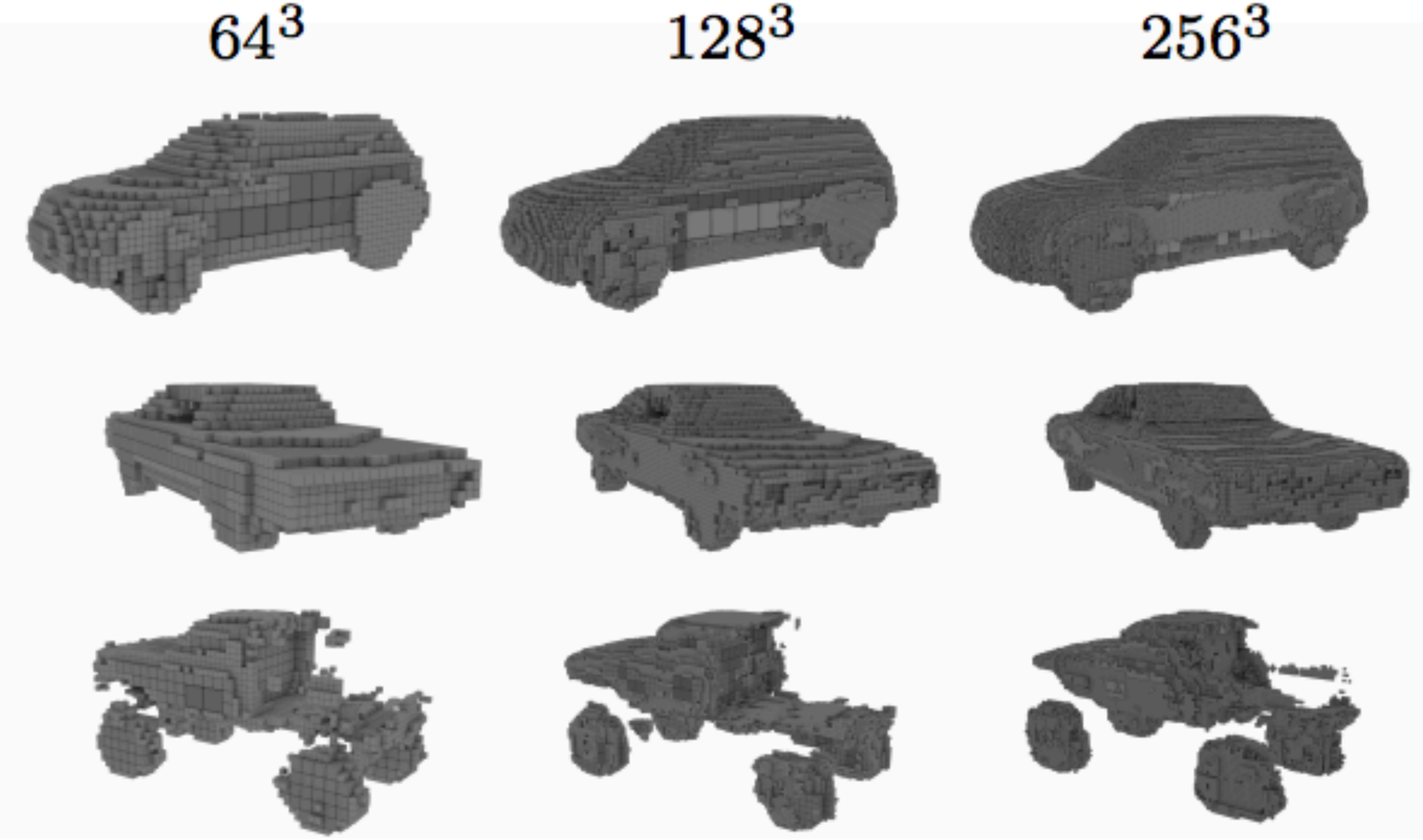
only generate non-empty voxels

Decoder/generator



[Hane et al. 2018]

Efficient Volumetric Datastructures



[Hane et al. 2018]

Lower Memory Footprint

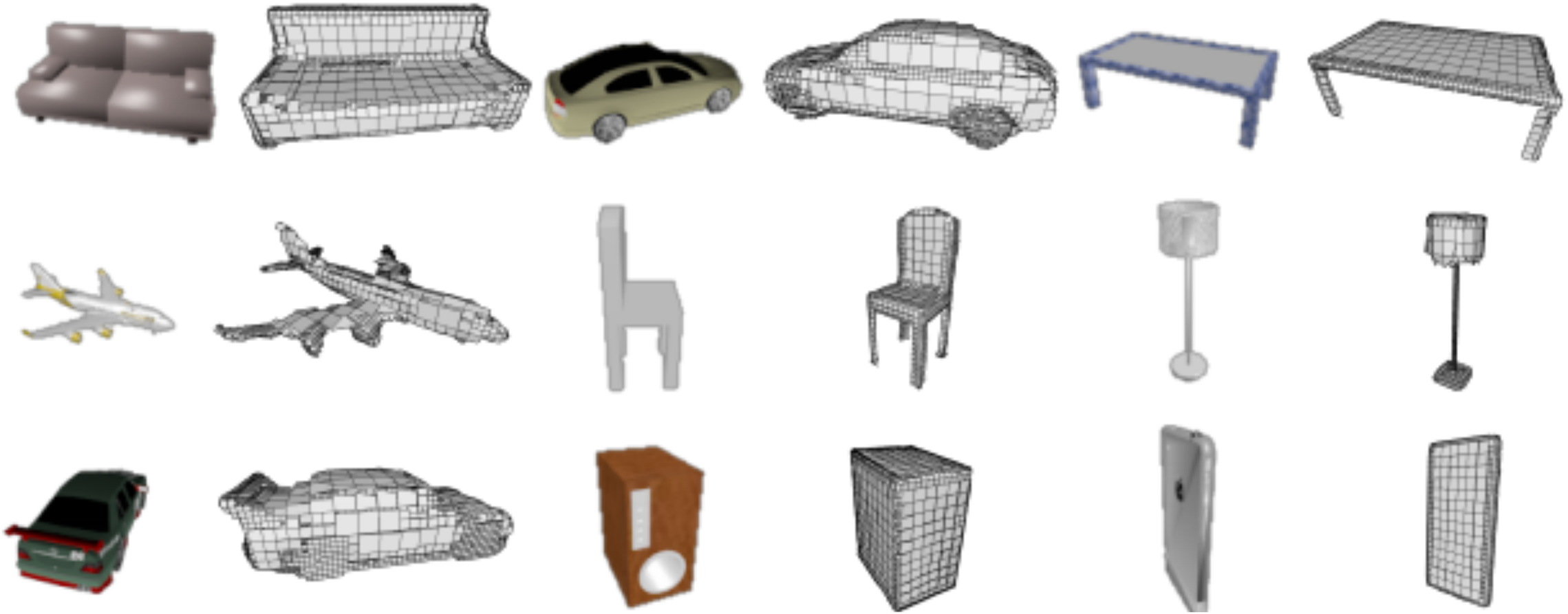
Method	16^3	32^3	64^3	128^3	256^3
O-CNN	0.32GB	0.58GB	1.1GB	2.7GB	6.4GB
full voxel+binary	0.23GB	0.71GB	3.7GB	Out of memory	Out of memory
full voxel+normal	0.27GB	1.20GB	4.3GB	Out of memory	Out of memory

Table 3. Comparisons on GPU-memory consumption. The batch size is 32.

Method	16^3	32^3	64^3	128^3	256^3
O-CNN	17ms	33ms	90ms	327ms	1265ms
full voxel+binary	59ms	425ms	1648ms	-	-
full voxel+normal	75ms	510ms	4654ms	-	-

Table 4. Timings of one backward and forward operation in milliseconds. The batch size is 32.

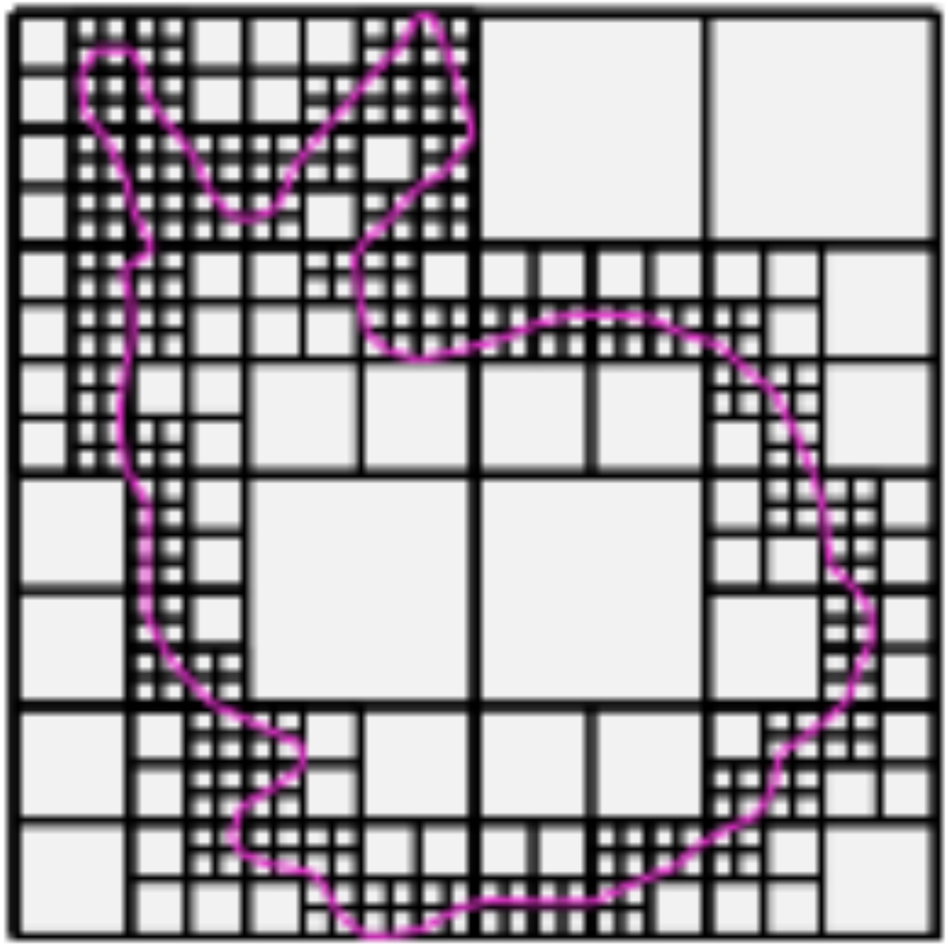
Adaptive O-CNN



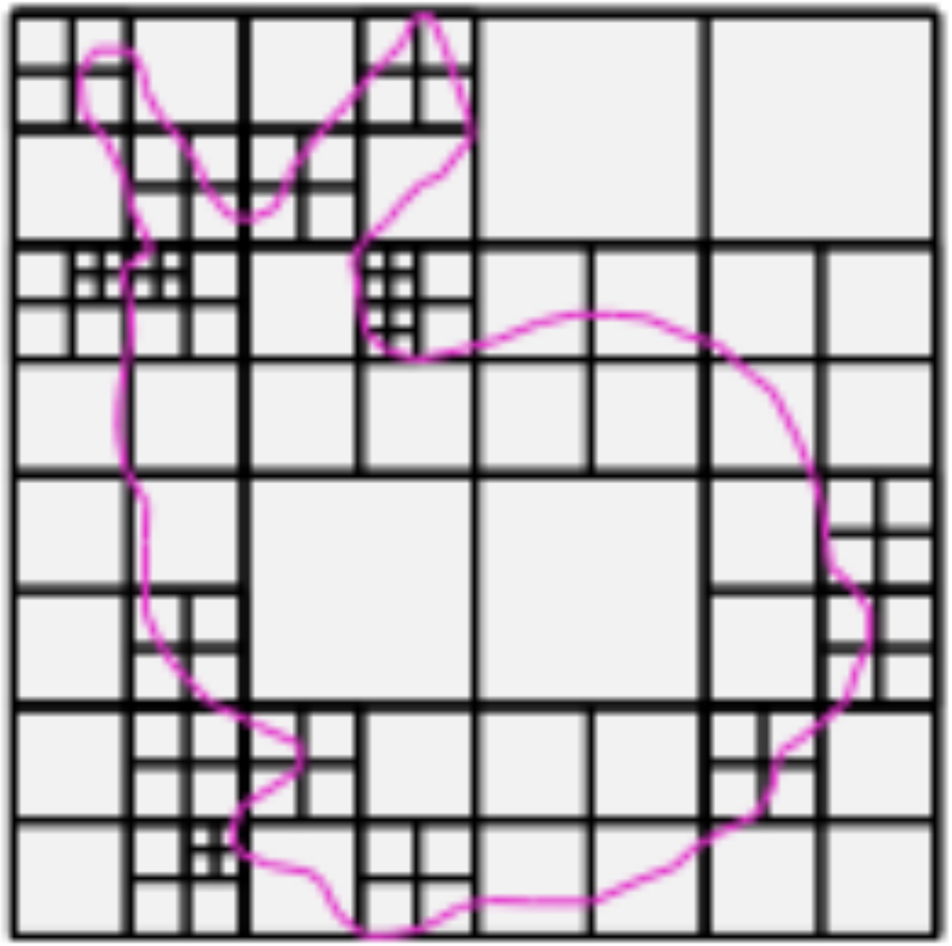
[Wang et al. 2018]

image to planar patch-based shapes

First-order Patches



OCNN

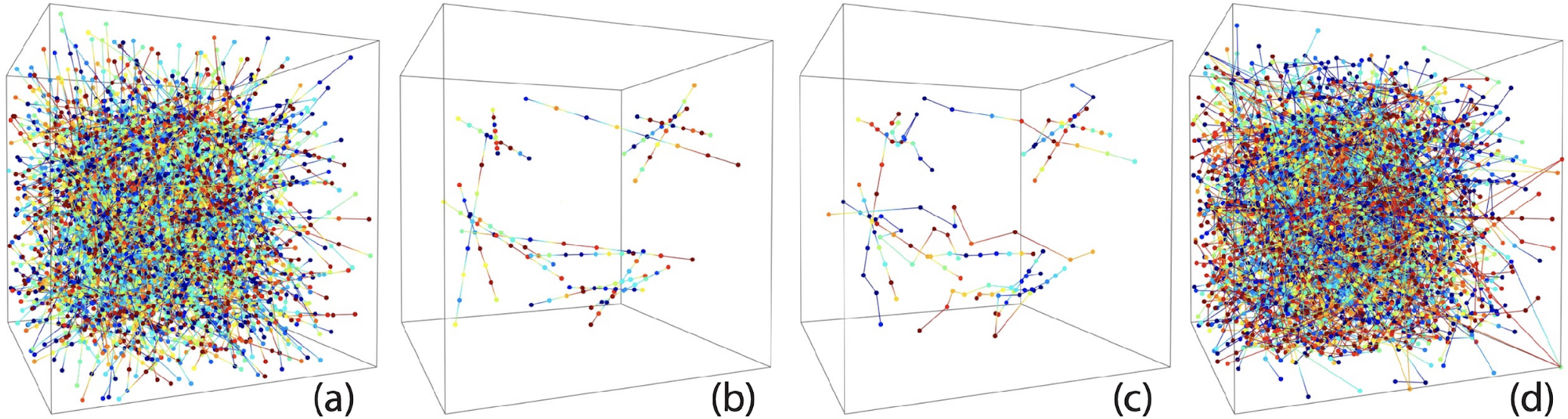


Adaptive OCNN

Field Probing Neural Networks for 3D Data

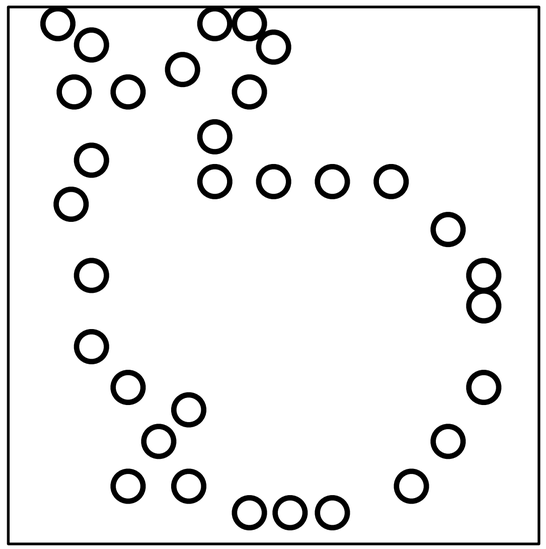
[Li et al. 2016]

Field Probing Neural Networks for 3D Data

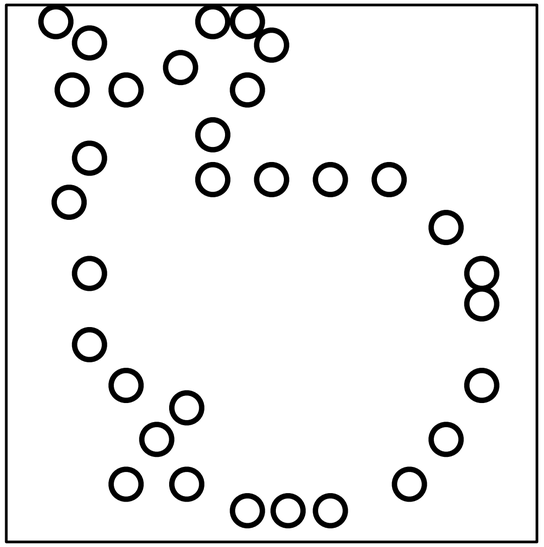


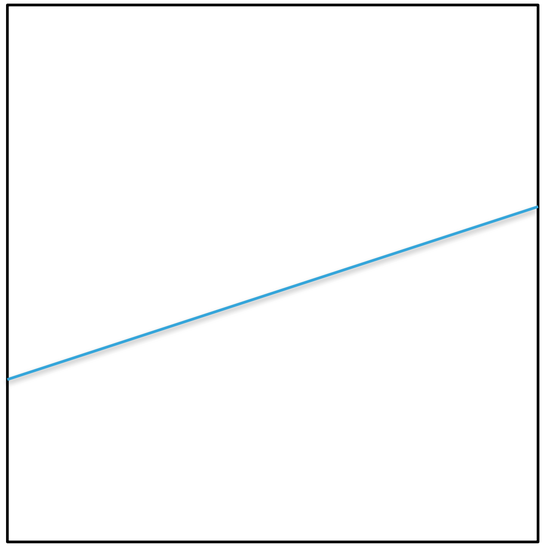
[Li et al. 2016]

Spatial Probes

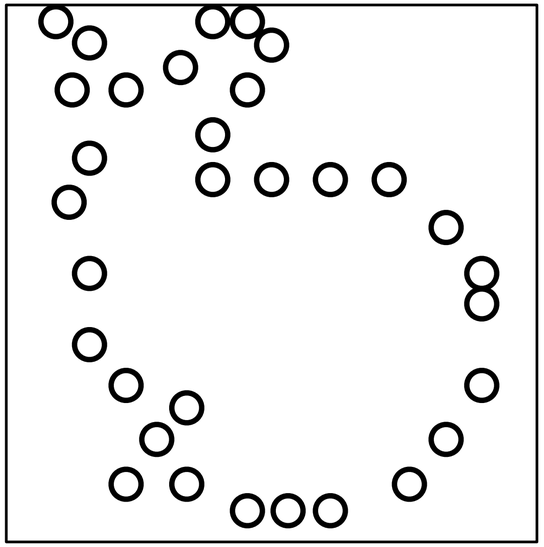


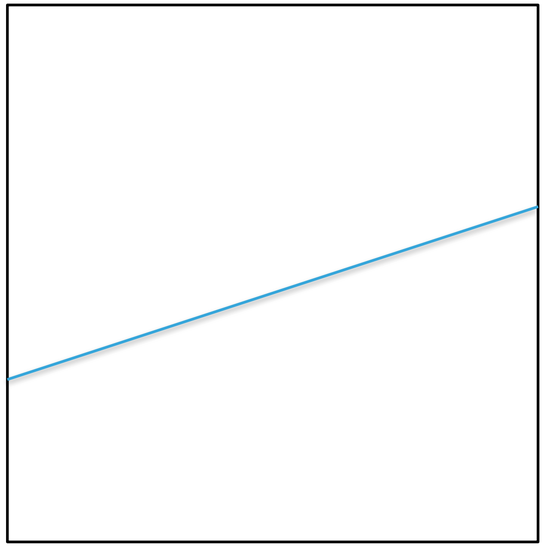
Spatial Probes

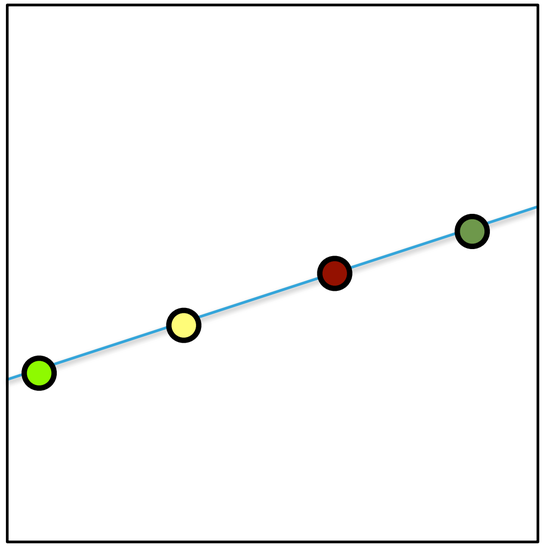




Spatial Probes

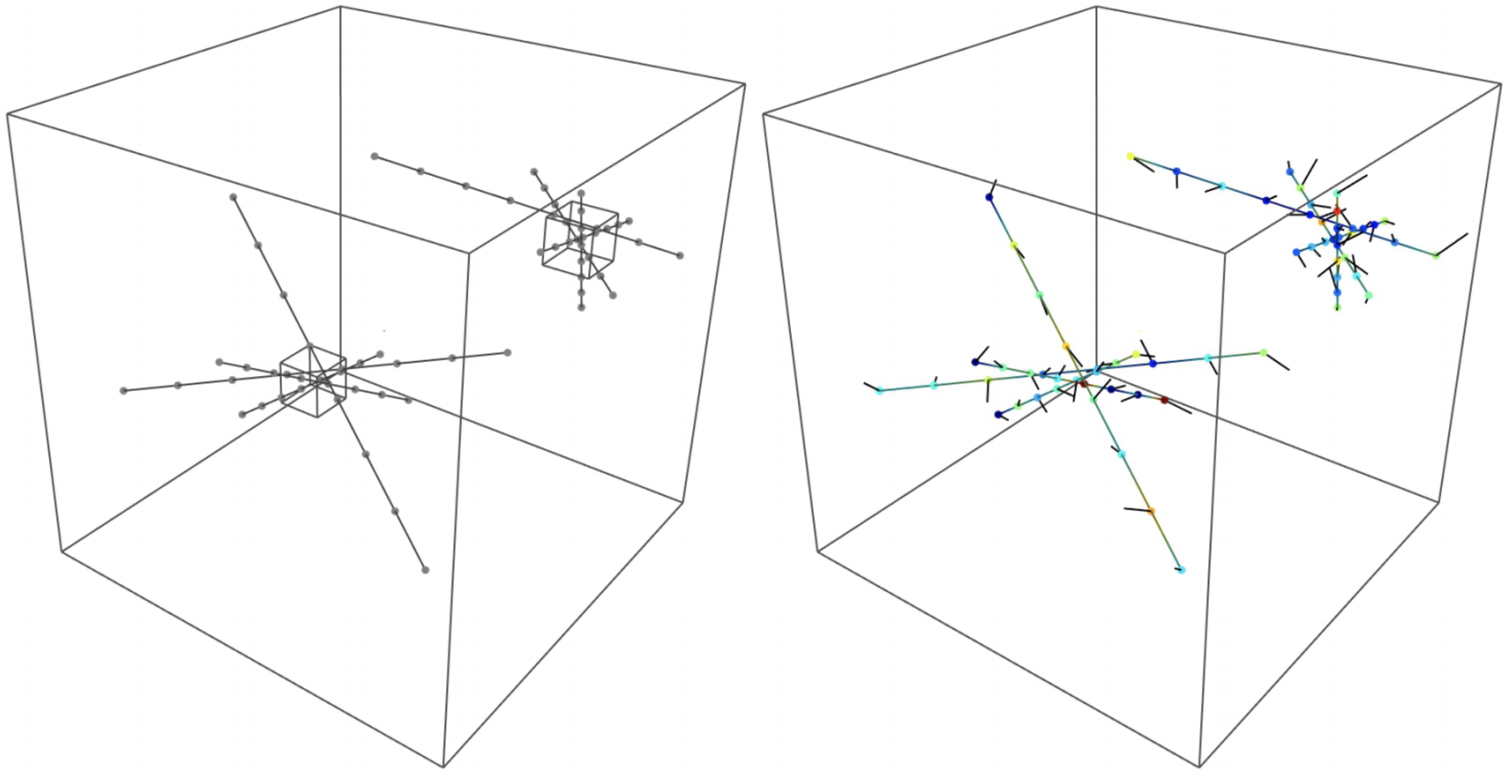




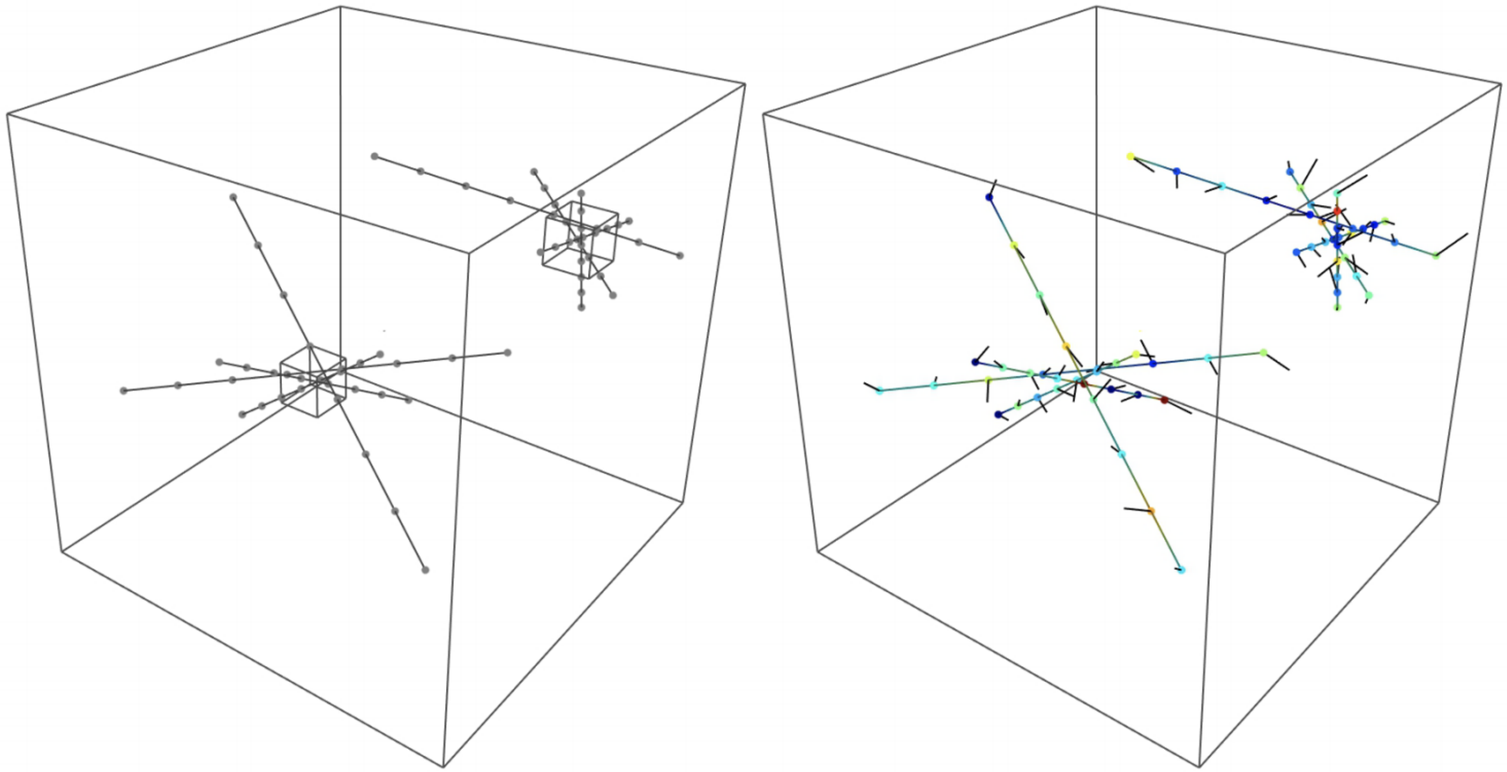


Method Details

Method Details

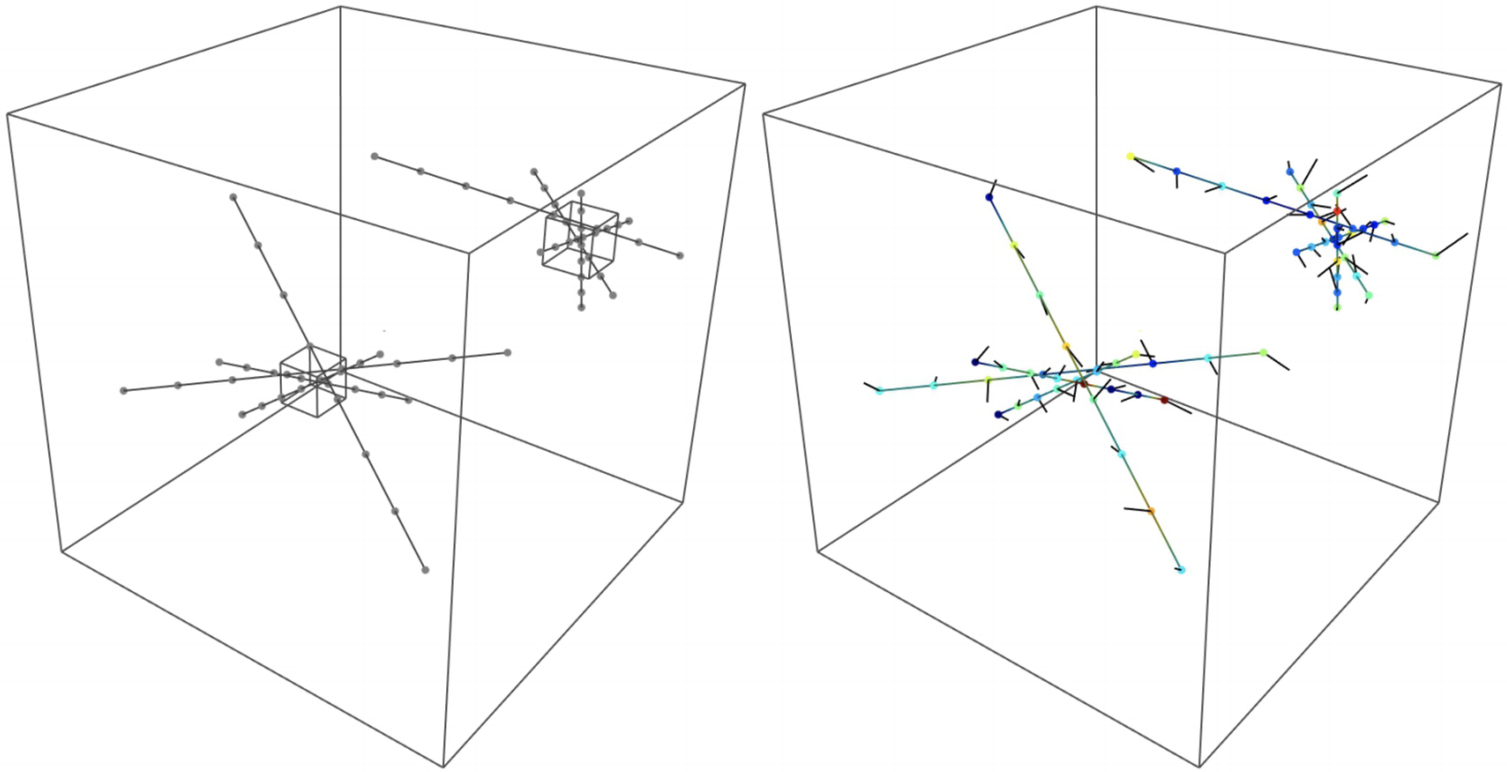


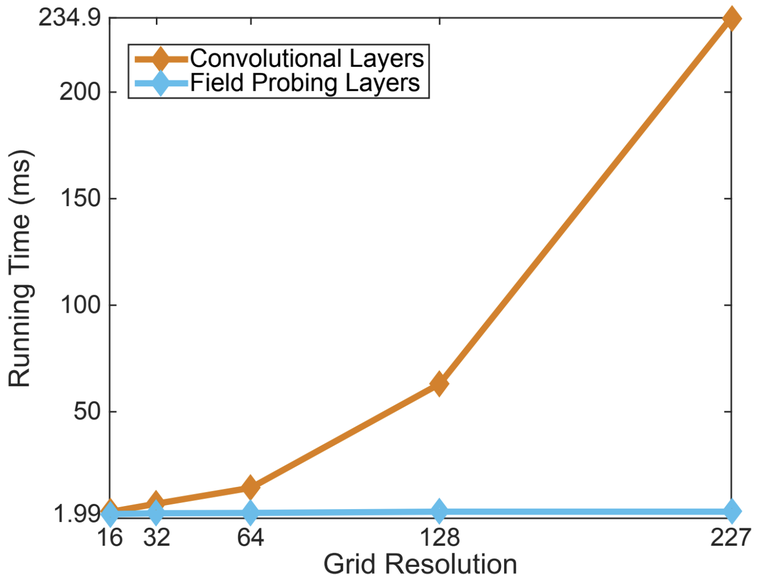
Method Details



$$v_c = v(\{p_{c,i,j}\}, \{w_{c,i,j}\}) = \sum_{\substack{i=1,\ldots,N \\ j=1,\ldots,T}} p_{c,i,j} \times w_{c,i,j}$$

Method Details





$$v_c = v(\{p_{c,i,j}\}, \{w_{c,i,j}\}) = \sum_{\substack{i=1,\ldots,N \\ j=1,\ldots,T}} p_{c,i,j} \times w_{c,i,j}$$

Representation for 3D

- Image-based

- Volumetric
  - **PROS**: adaptations of image networks
  - **CONS**: special layers for hierarchical datastructures, still too coarse

- Surface-based

- Point-based

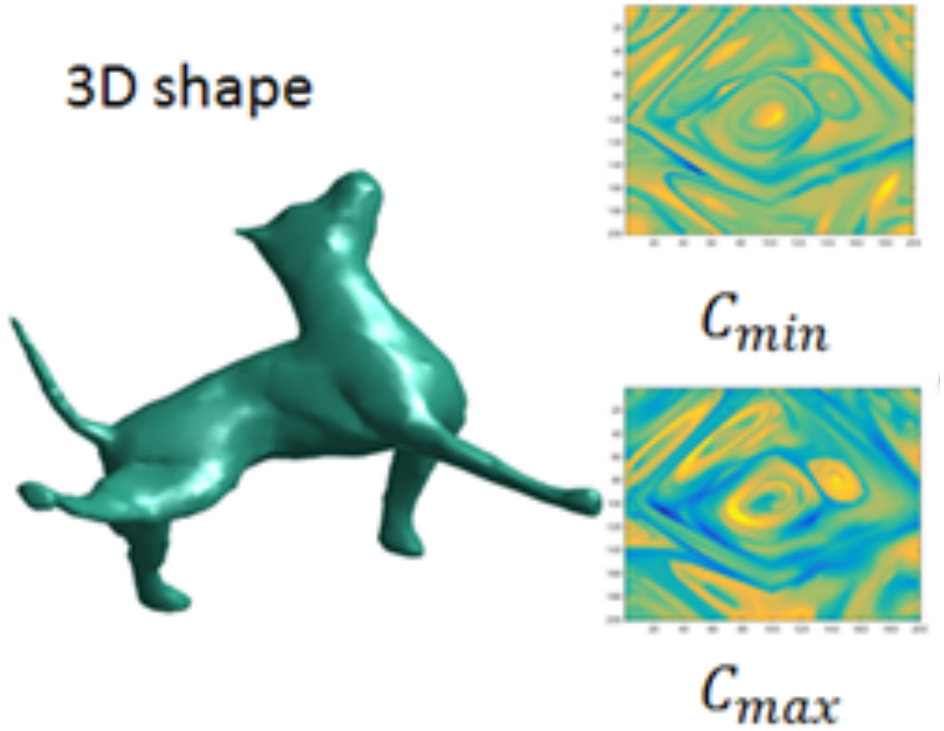
Representation for 3D

- Image-based
- Volumetric
- **Surface-based**
- Point-based

Local/Global Parameterizations

Local/Global Parameterizations

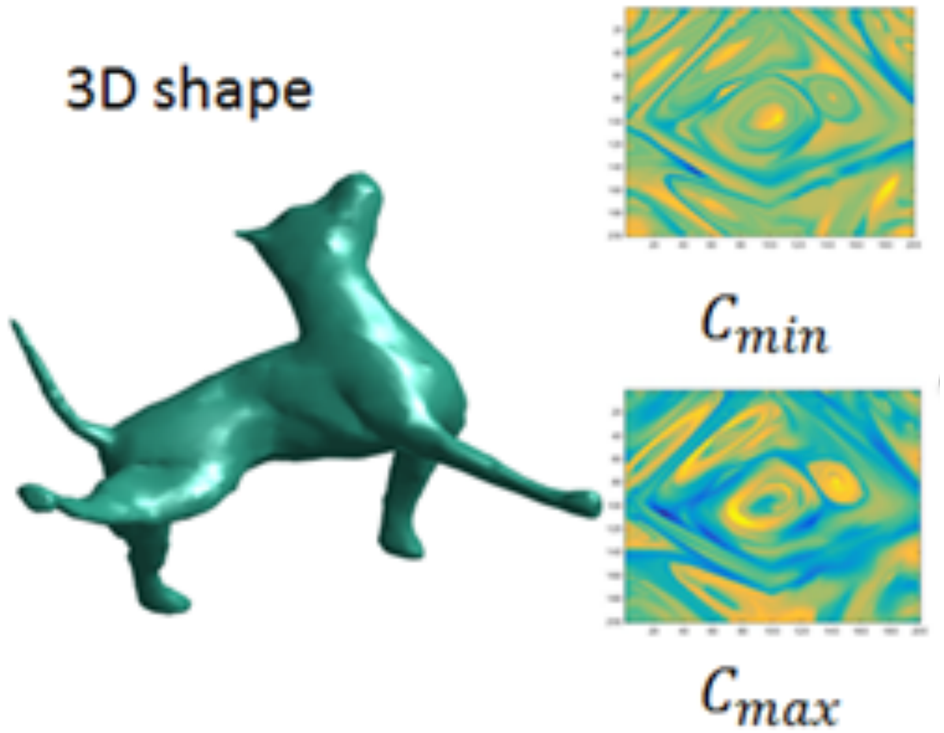
Geometry Image



[Sinha et al. 2016]

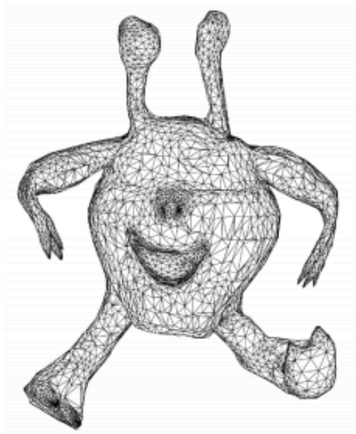
Local/Global Parameterizations

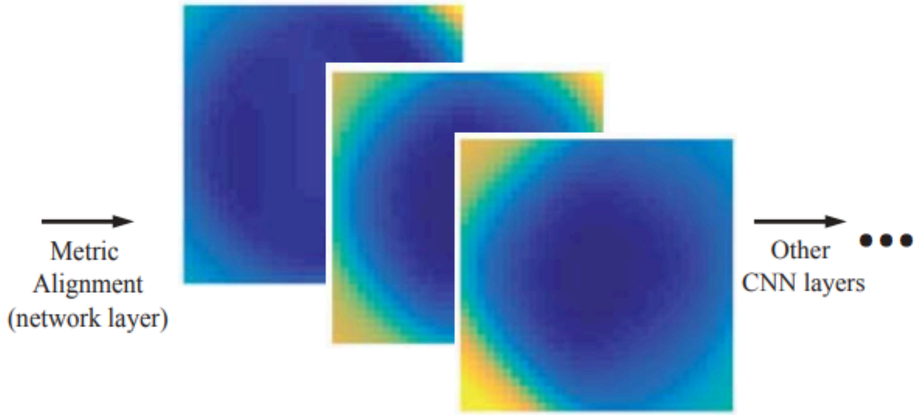
Geometry Image



[Sinha et al. 2016]

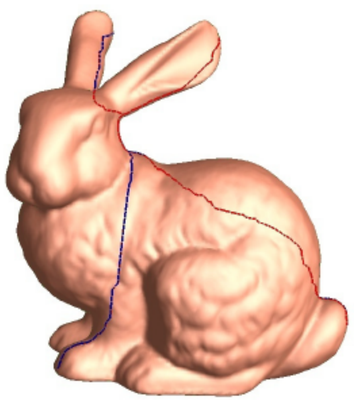
Metric Alignment (GWCNN)



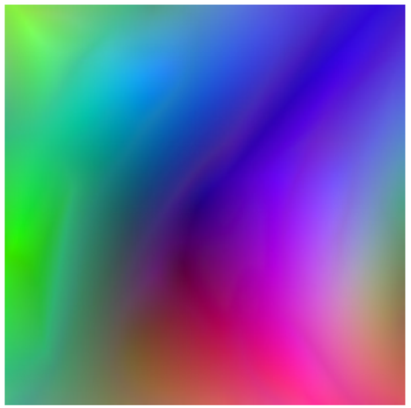


[Ezuz et al. 2017]

Shape Surfaces using Geometry Images

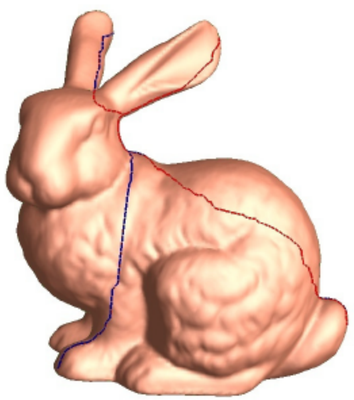


(a) Original mesh with cut
70K faces; genus 0

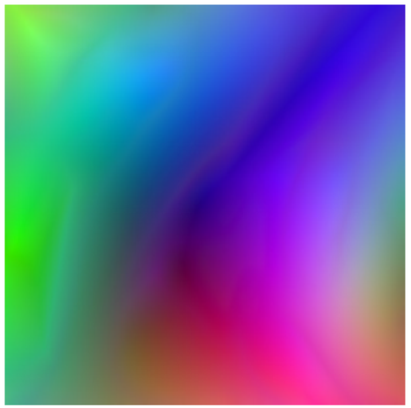


(b) Geometry image 257×257
(b∗) Compr. to 1.5KB (not shown)

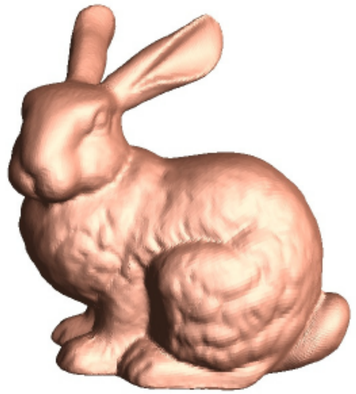
Shape Surfaces using Geometry Images



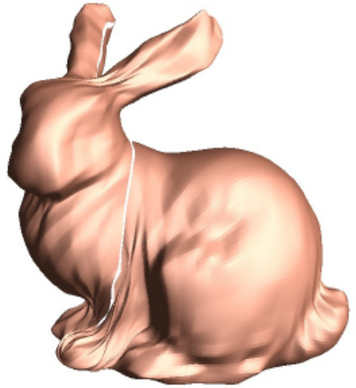
(a) Original mesh with cut
70K faces; genus 0



(b) Geometry image 257×257
(b∗) Compr. to 1.5KB (not shown)

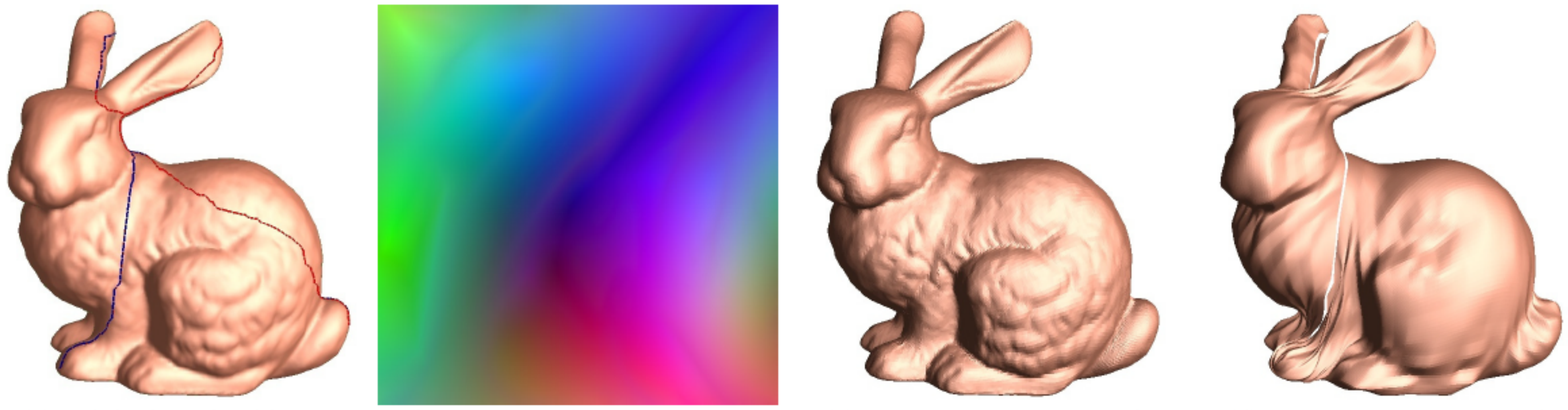


(c) Geometry reconstructed
entirely from b



(d) Geometry reconstructed
entirely from b∗

Shape Surfaces using Geometry Images

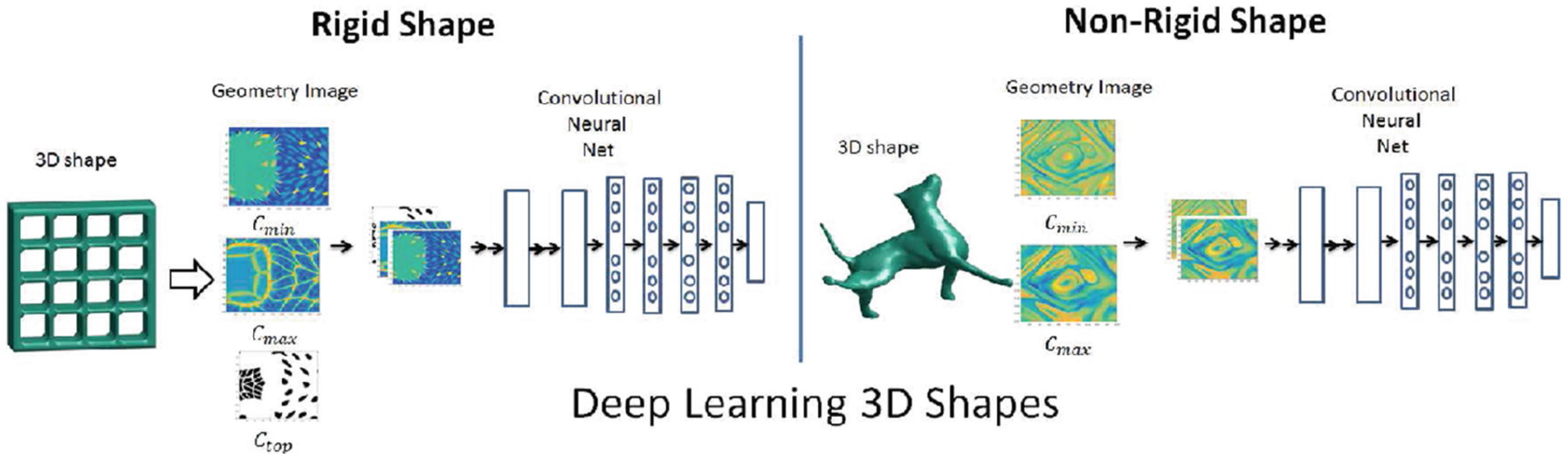


(a) Original mesh with cut 70K faces; genus 0

(b) Geometry image 257×257 (b∗) Compr. to 1.5KB (not shown)

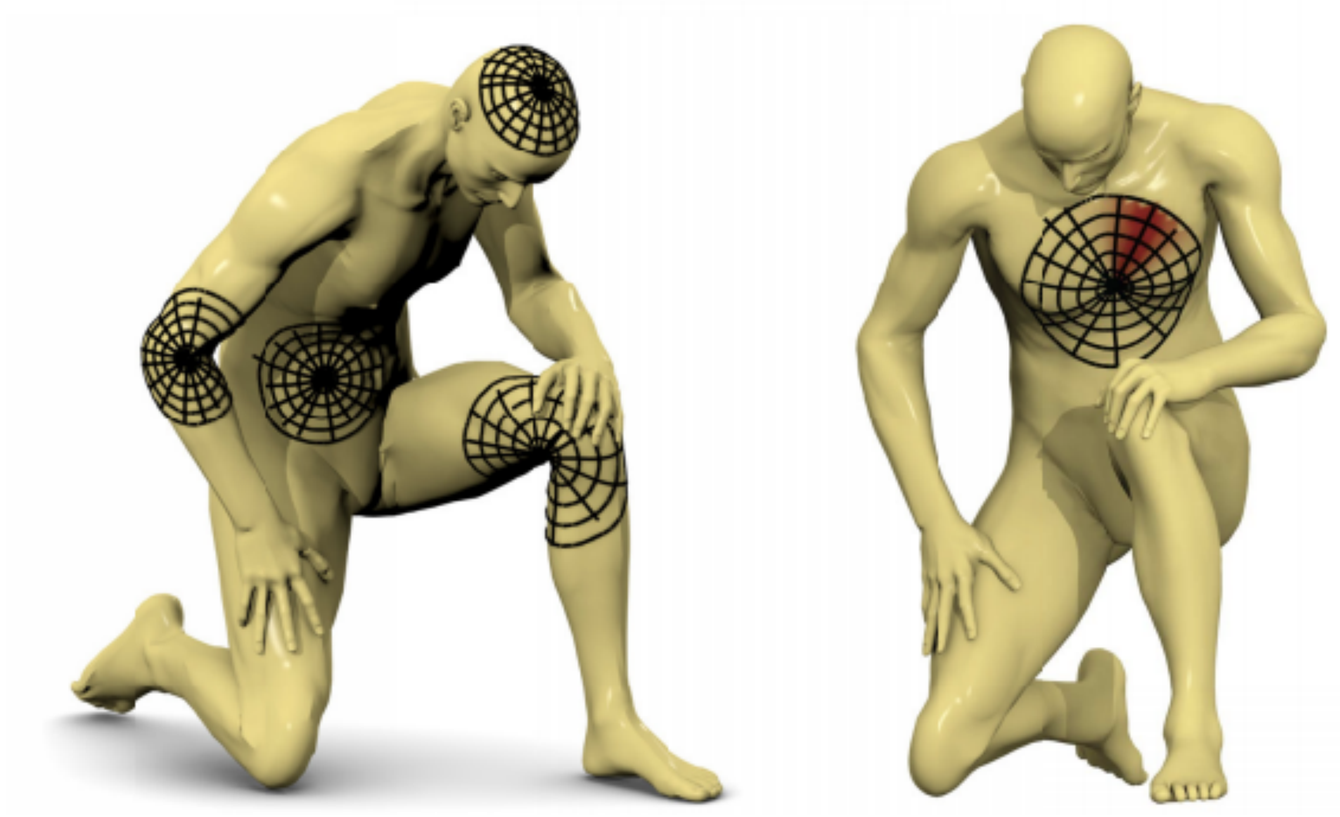
(c) Geometry reconstructed entirely from b

(d) Geometry reconstructed entirely from b∗



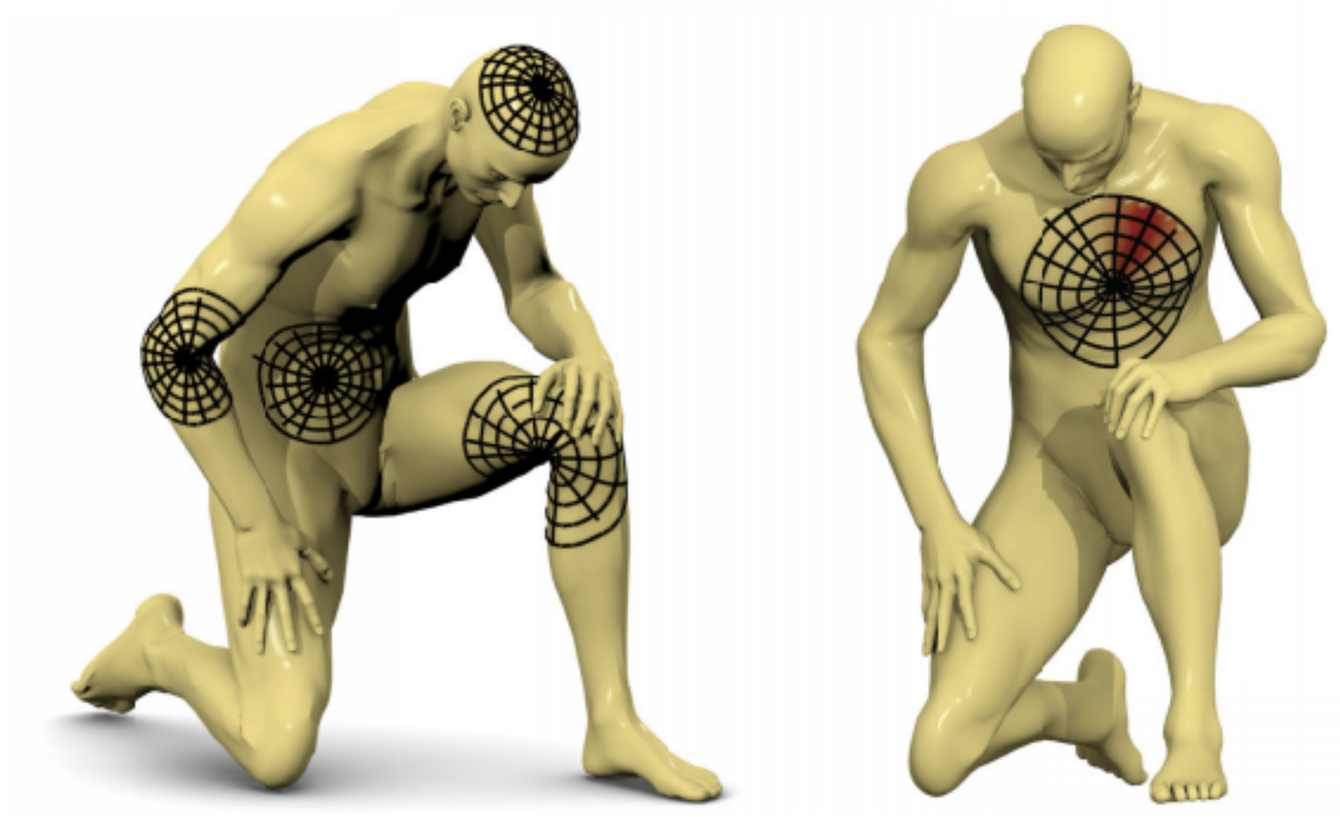
Deep Learning 3D Shapes

Using Geodesic Patches: GCNN



[Masci et al. 2015]

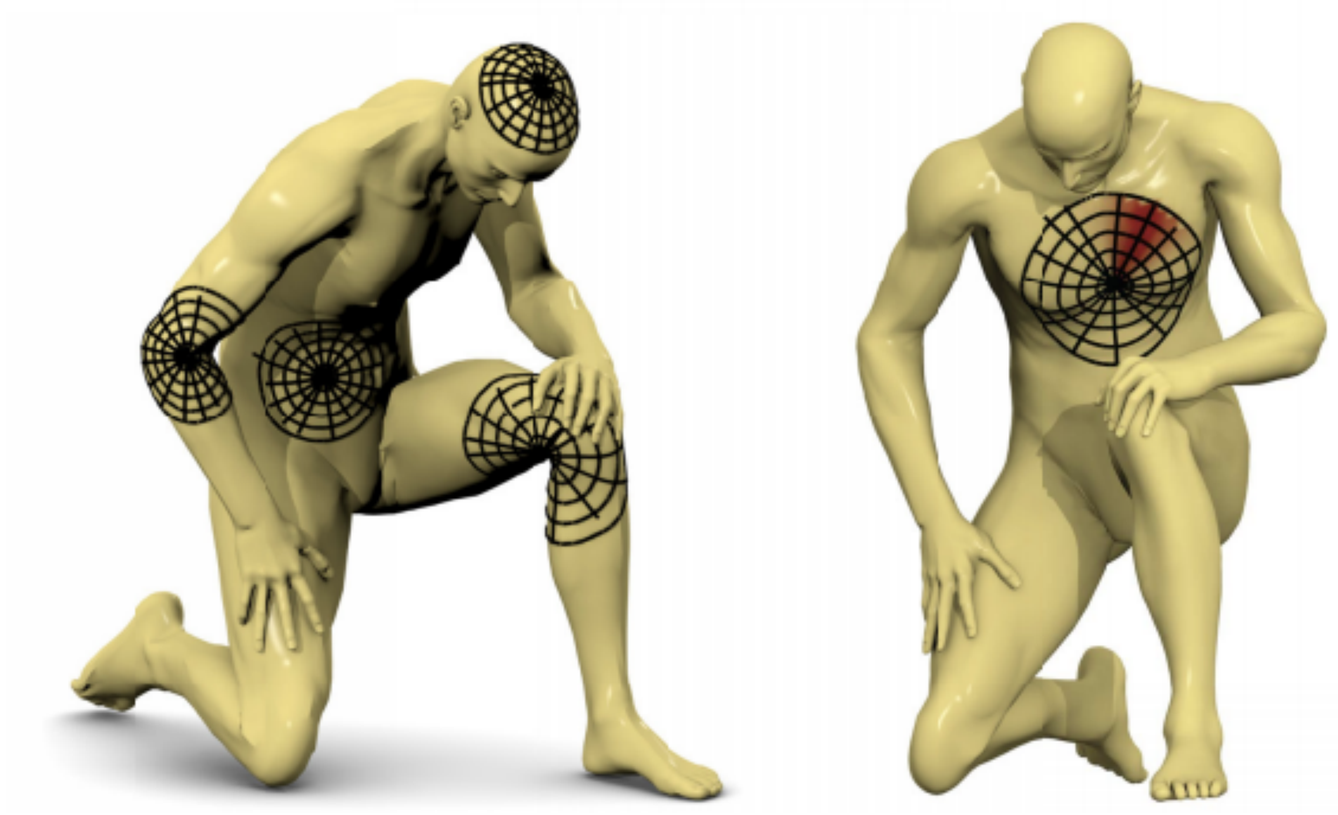
Using Geodesic Patches: GCNN

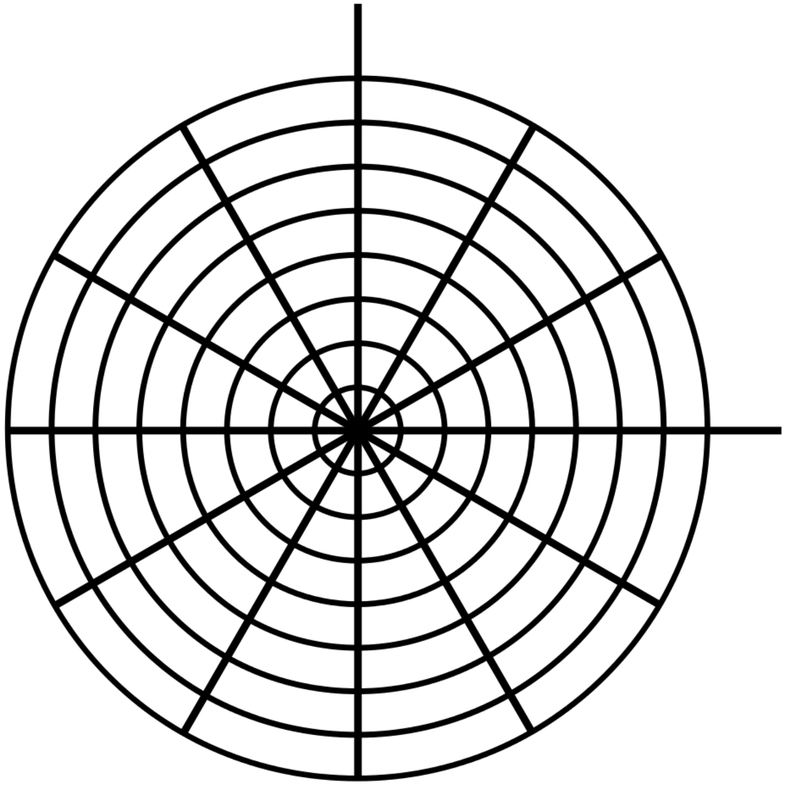


$$(f \star a)(x) := \sum_{\theta,r} a(\theta + \Delta\theta, r)(D(x)f)(r,\theta)$$

[Masci et al. 2015]

Using Geodesic Patches: GCNN

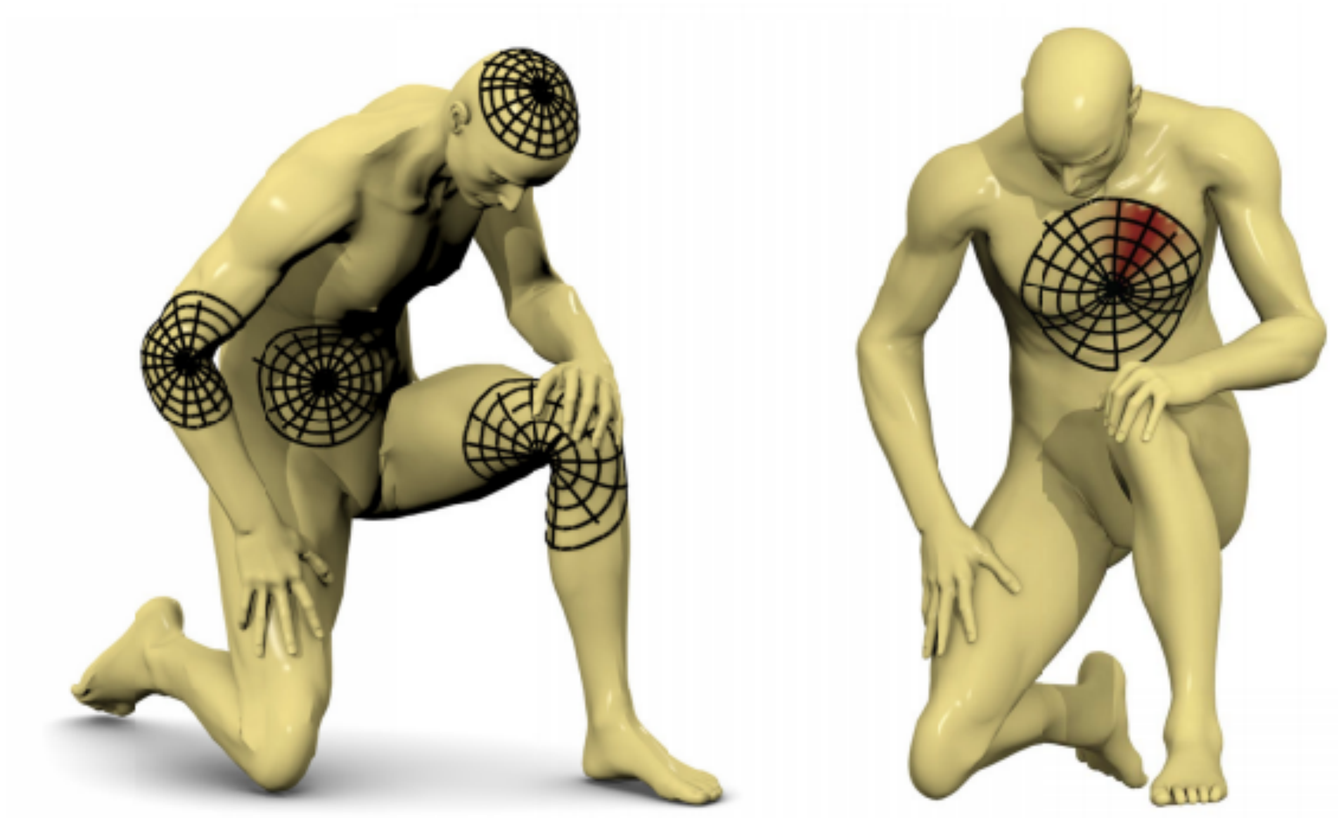


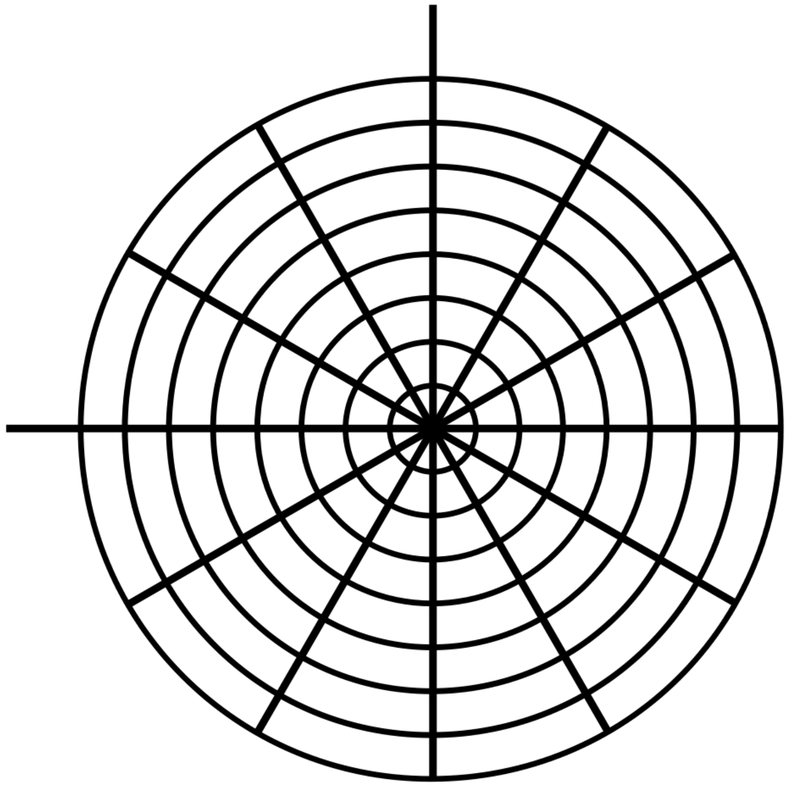


$$(f \star a)(x) := \sum_{\theta,r} a(\theta + \Delta\theta, r)(D(x)f)(r,\theta)$$

[Masci et al. 2015]

Using Geodesic Patches: GCNN

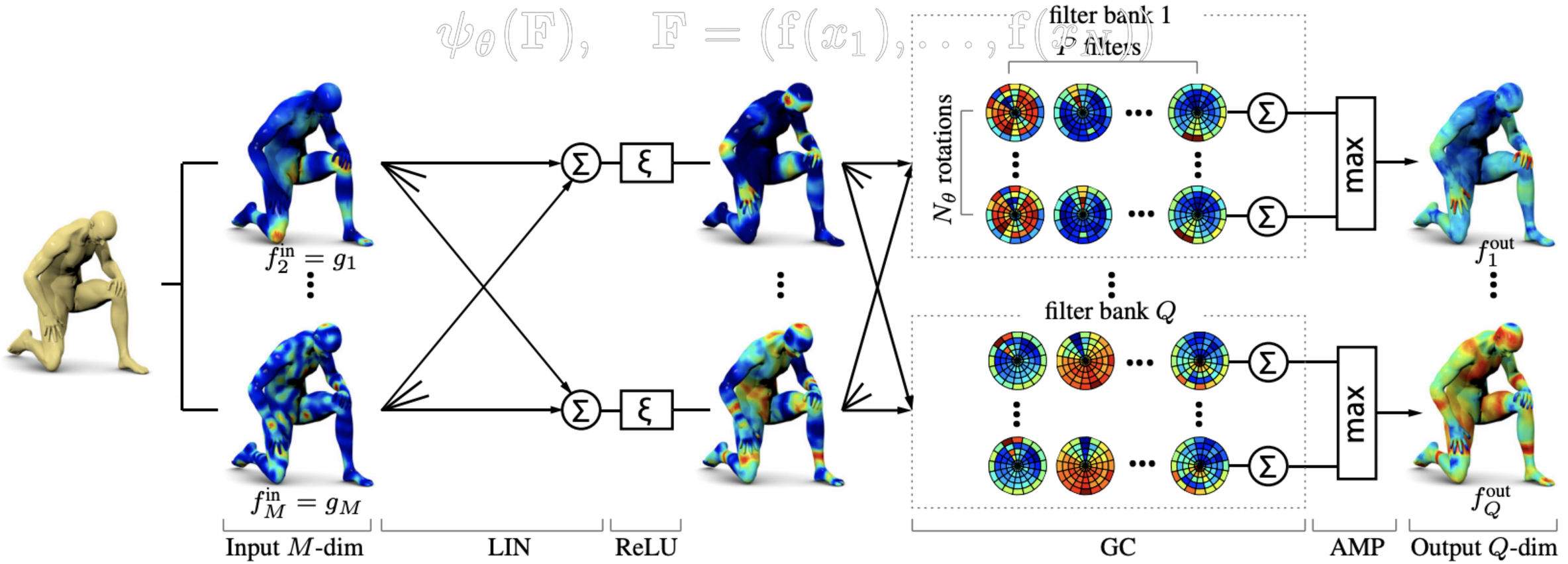




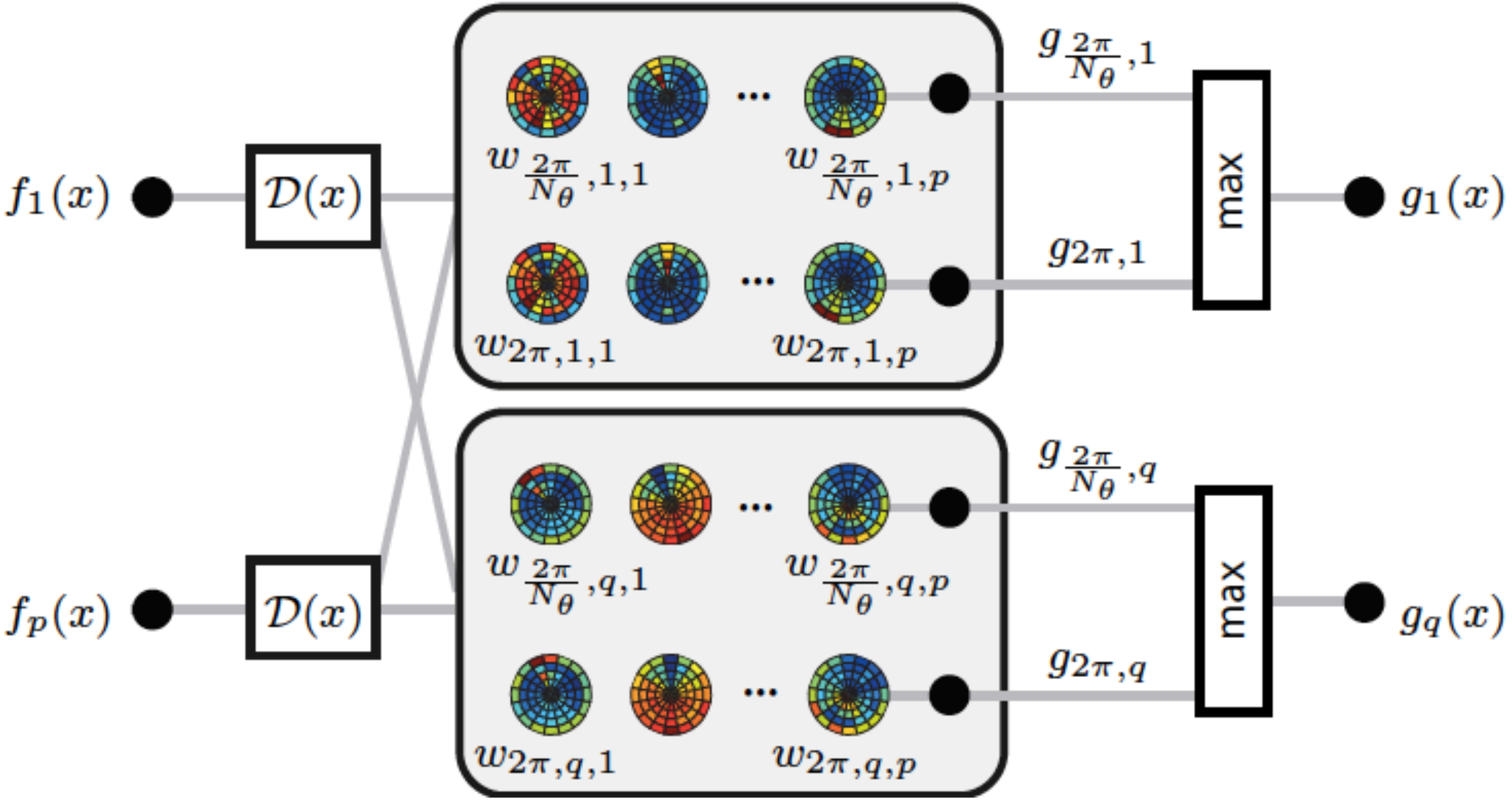
$$(f \star a)(x) := \sum_{\theta,r} a(\theta + \Delta\theta, r)(D(x)f)(r,\theta)$$

[Masci et al. 2015]

GCNN Architecture



Handling Rotational Ambiguity



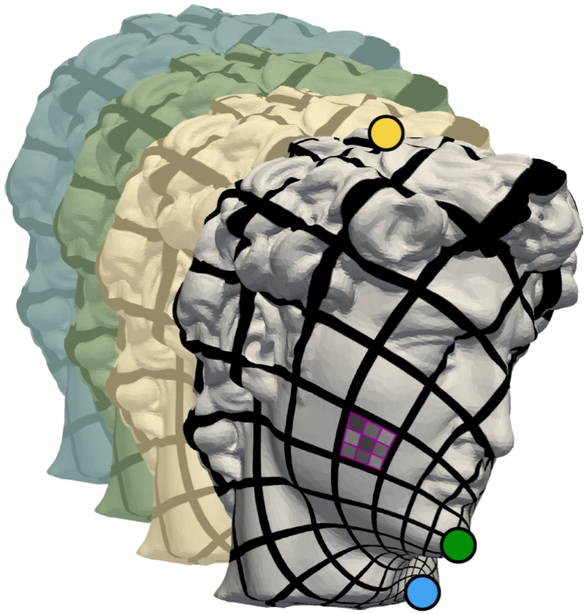
Parameterization for Surface Analysis

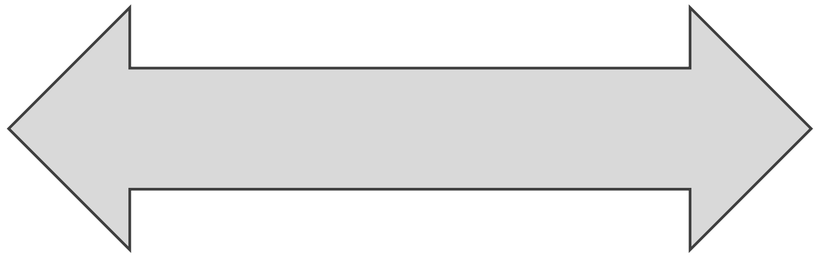
map 3D surface to 2D domain

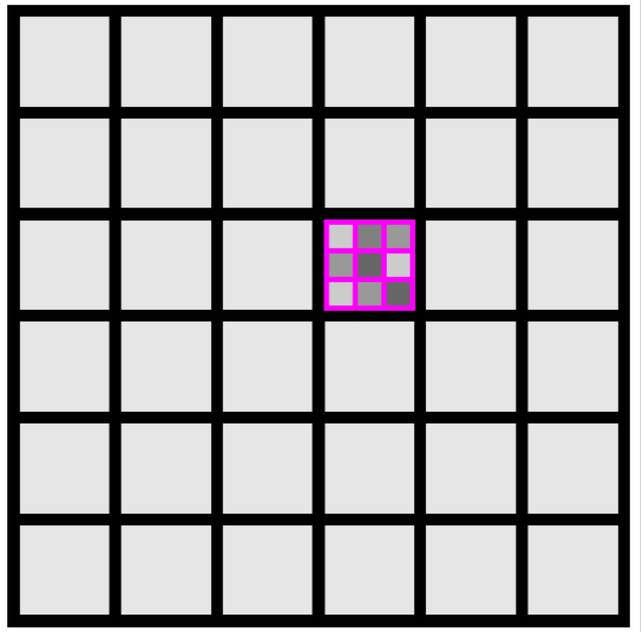
[Maron et al. 2017]

Parameterization for Surface Analysis

map 3D surface to 2D domain

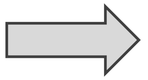


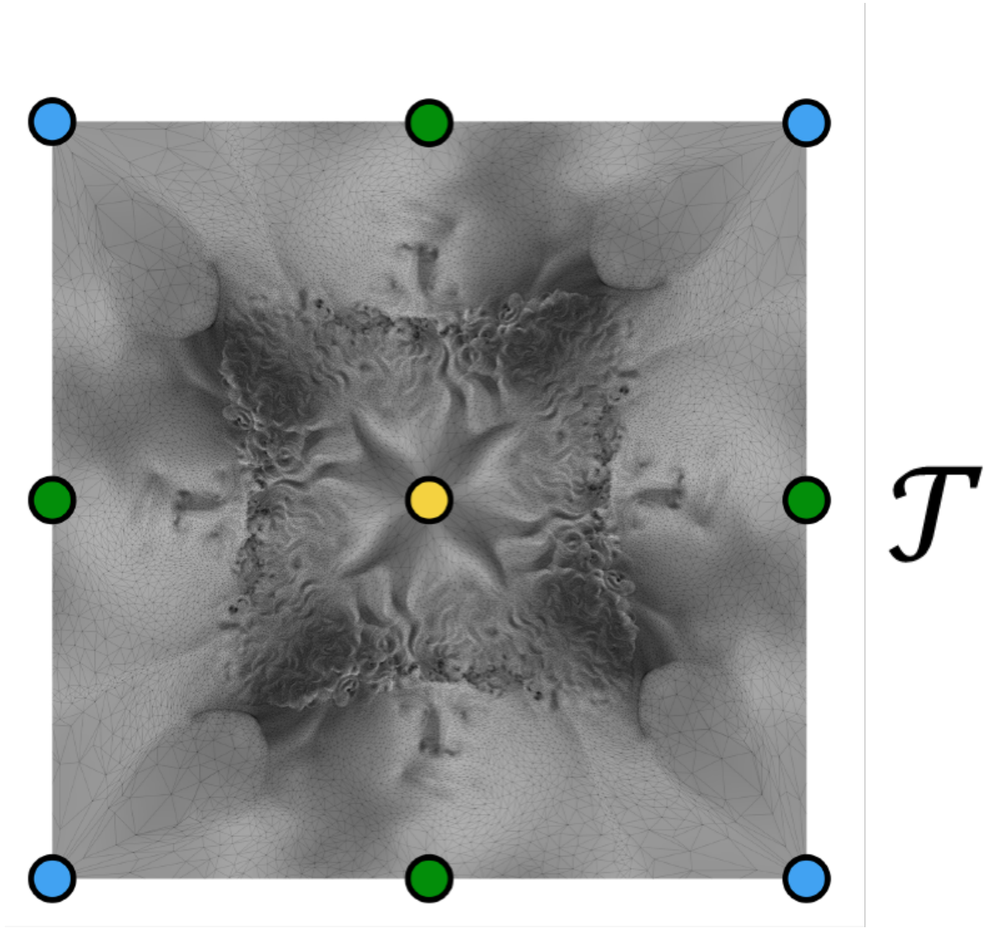




[Maron et al. 2017]

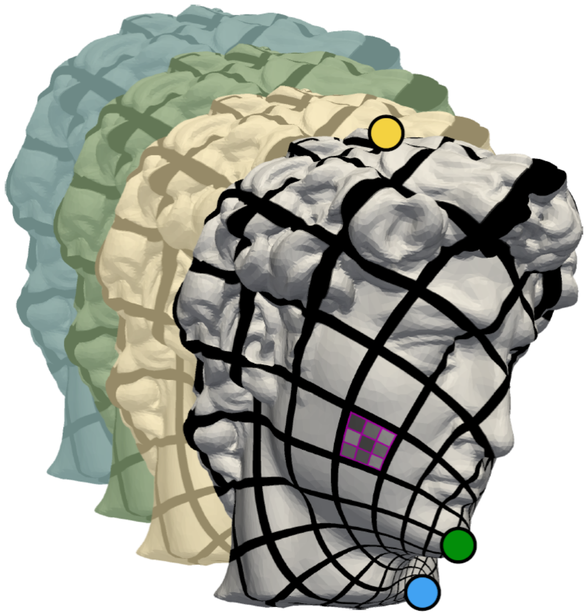
Parameterization for Surface Analysis

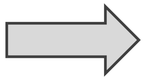


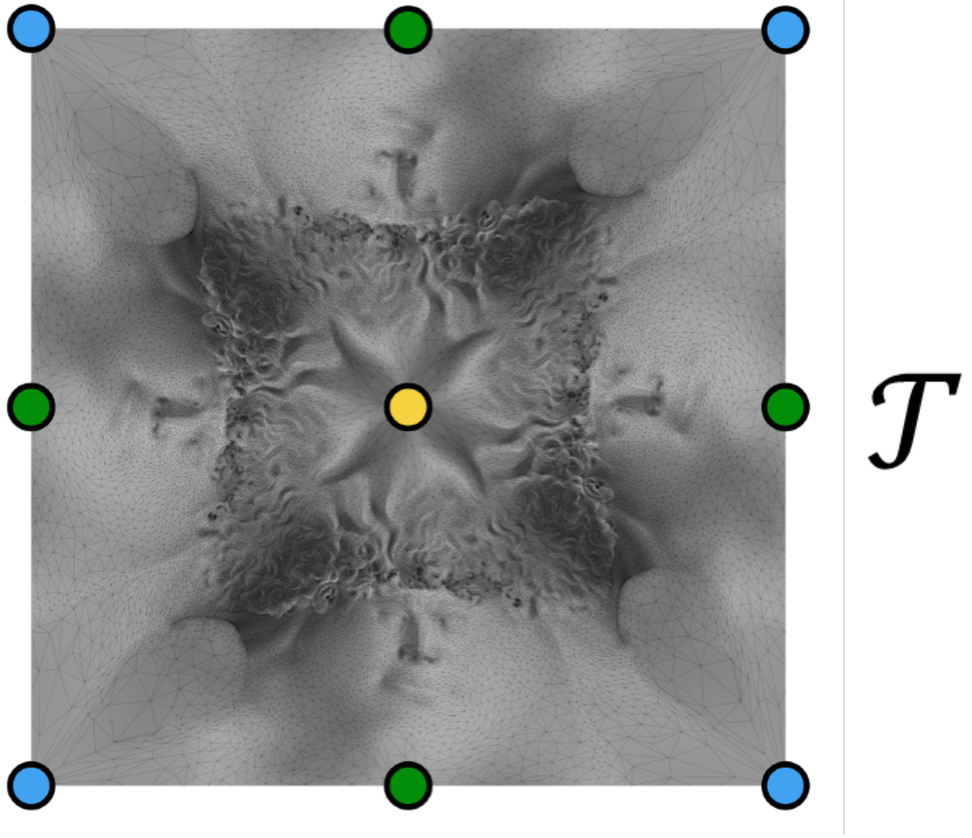


[Maron et al. 2017]

Parameterization for Surface Analysis







$\mathcal{T}$

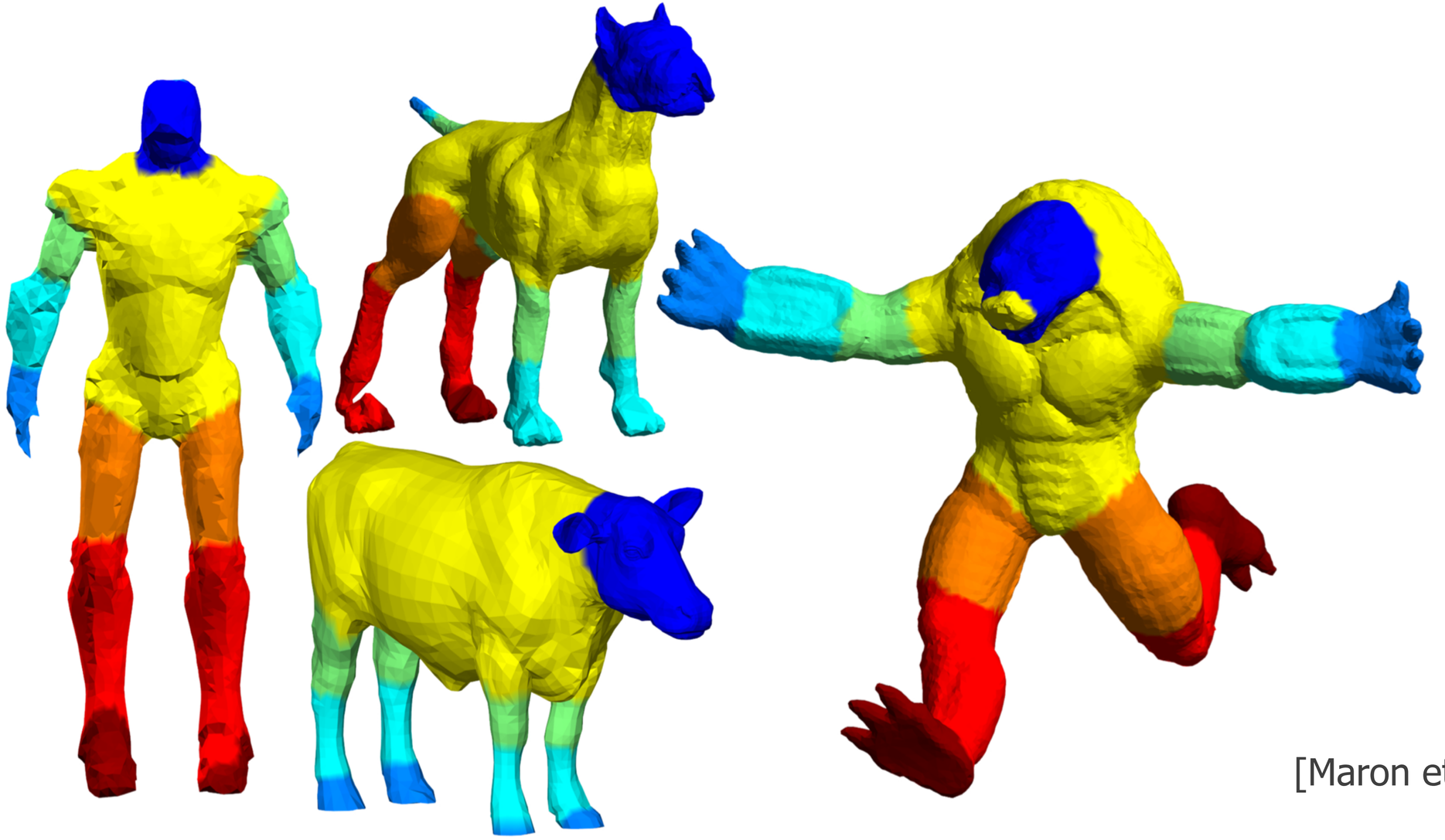
[Maron et al. 2017]

Parameterization for Surface Analysis

- Map 3D surface to 2D domain
  - One such mapping: **flat torus** (seamless => translation-invariant)
  - Many mappings exists: sample a few and average result
  - Which functions to map?
    XYZ, normals, curvature, …

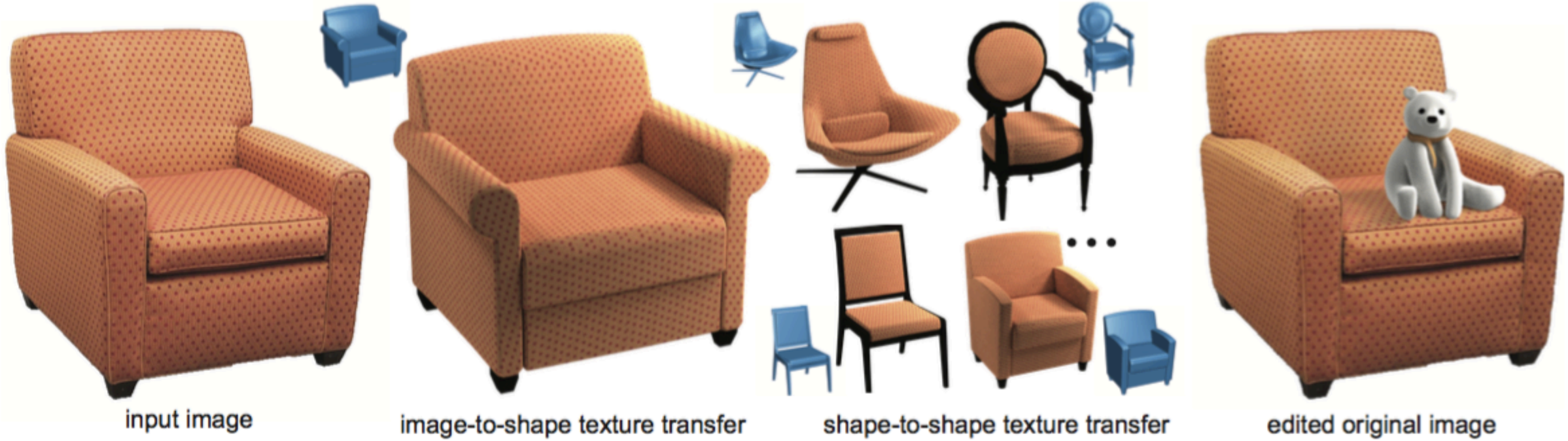
[Maron et al. 2017]

Parameterization for Surface Analysis



[Maron et al. 2017]

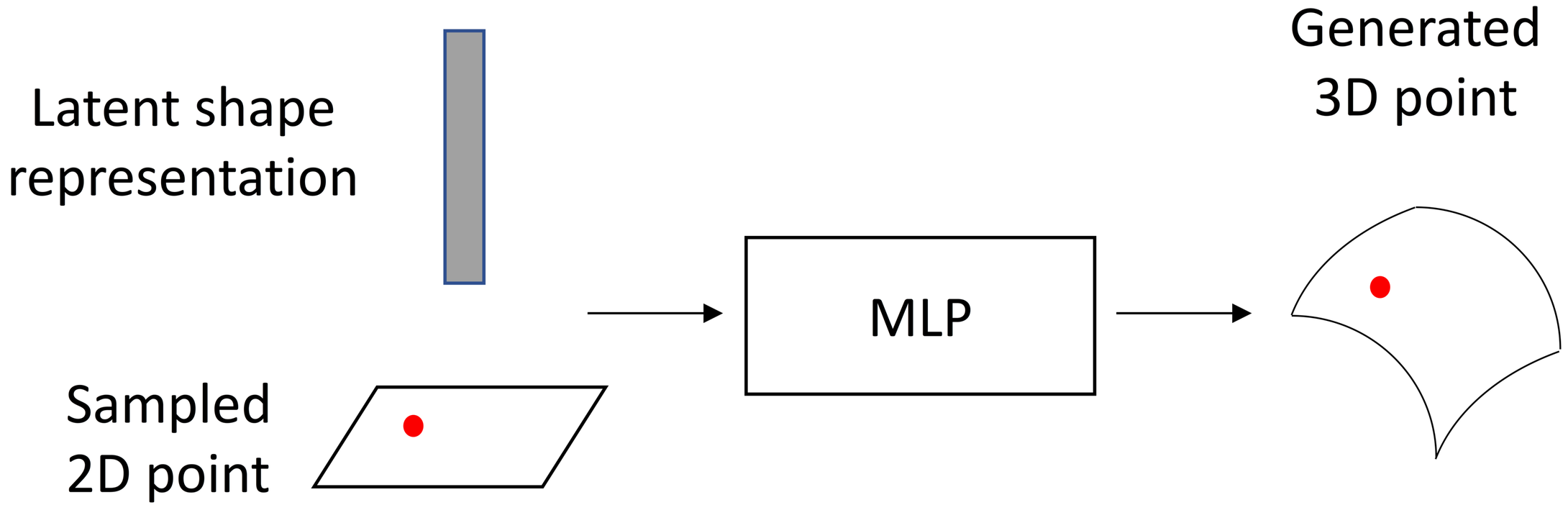
Texture Transfer (Parameterization + Alignment)



[Wang et al. 2016]

AtlasNet for Surface Generation

condition decoded points on 2D patches

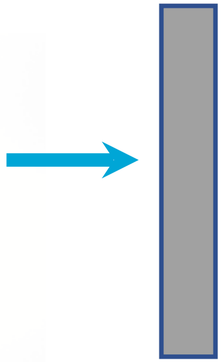


[Groueix et al. 2018]

AtlasNet for Surface Generation

2D Image

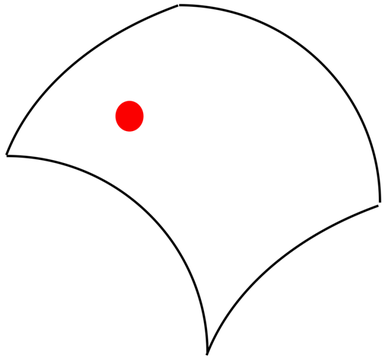
3D Point Cloud



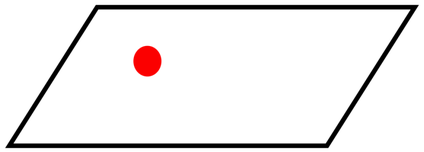
Latent representation can be inferred from images or point clouds

Generated 3D point

MLP



Sampled 2D point



[Groueix et al. 2018]

AtlasNet for Surface Generation

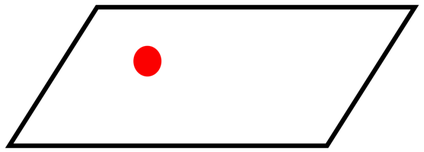
Quad Mesh is generated by mapping a regular grid in 2D domain to 3D points

2D Image

3D Point Cloud

Latent representation can be inferred from images or point clouds

Sampled 2D point



MLP

[Groueix et al. 2018]

AtlasNet for Surface Generation

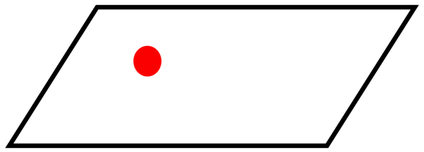
texture coordinates come for free!!

2D Image

3D Point Cloud

Latent representation can be inferred from images or point clouds

Sampled 2D point



MLP

Representation for 3D

- Image-based
- Volumetric
- Surface-based
  - **PROS**: parameterize + image networks (instrinsic representation)
  - **CONS**: suffers from parameterisation artefacts (local versus global distortion), requires good quality mesh
- Point-based

Representation for 3D

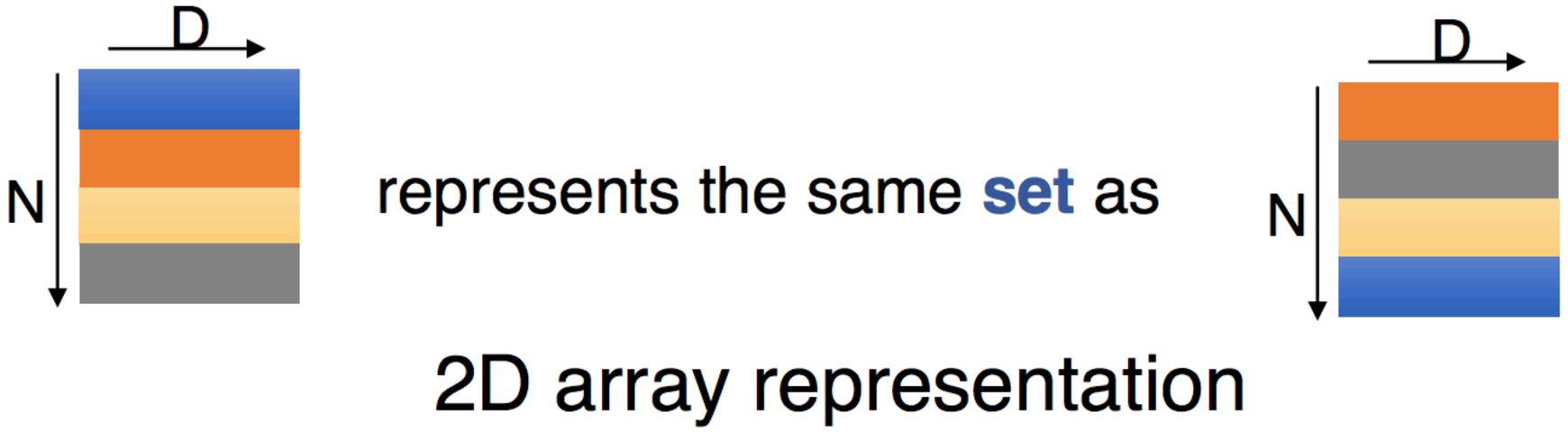
- Image-based
- Volumetric
- Surface-based
- **Point-based**

Representation for 3D: Point-based

- Common representation: native representation
- Easy to obtain from meshes, depth scans, laser scans

In Original Representation

- Common representation
- Easy to obtain from meshes, depth scans, laser scans
- **Unstructured** (e.g., any permutation of points gives same shape!)



[Qi et al. 2017]

PointNet for Point Cloud Analysis

$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

permutation-invariant functions

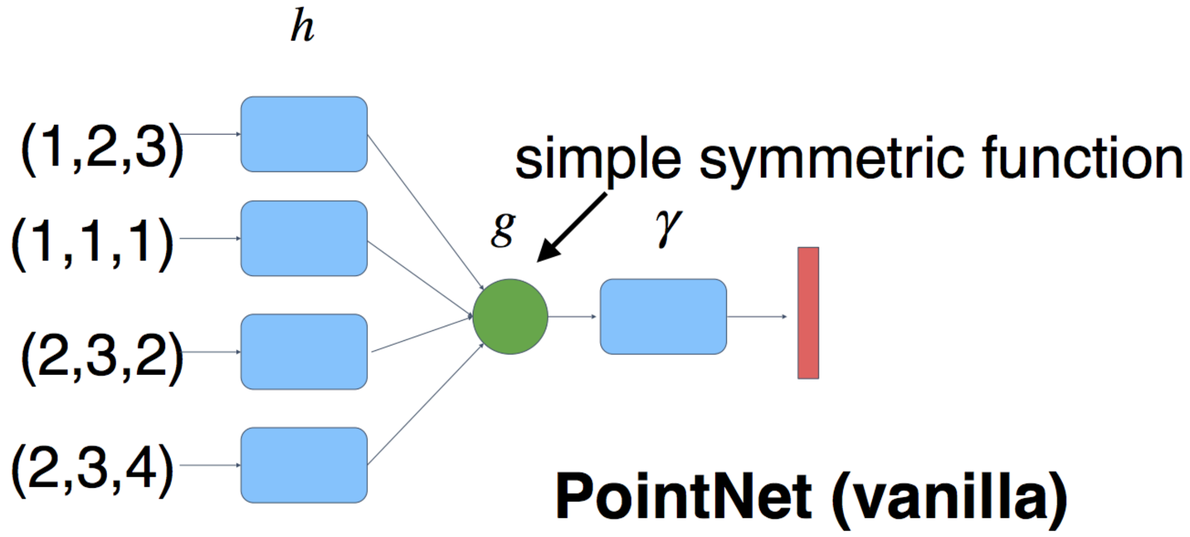
[Qi et al. 2017]

PointNet for Point Cloud Analysis

Use **MLPs** (h) and **max-pooling** (g) as simple symmetric functions

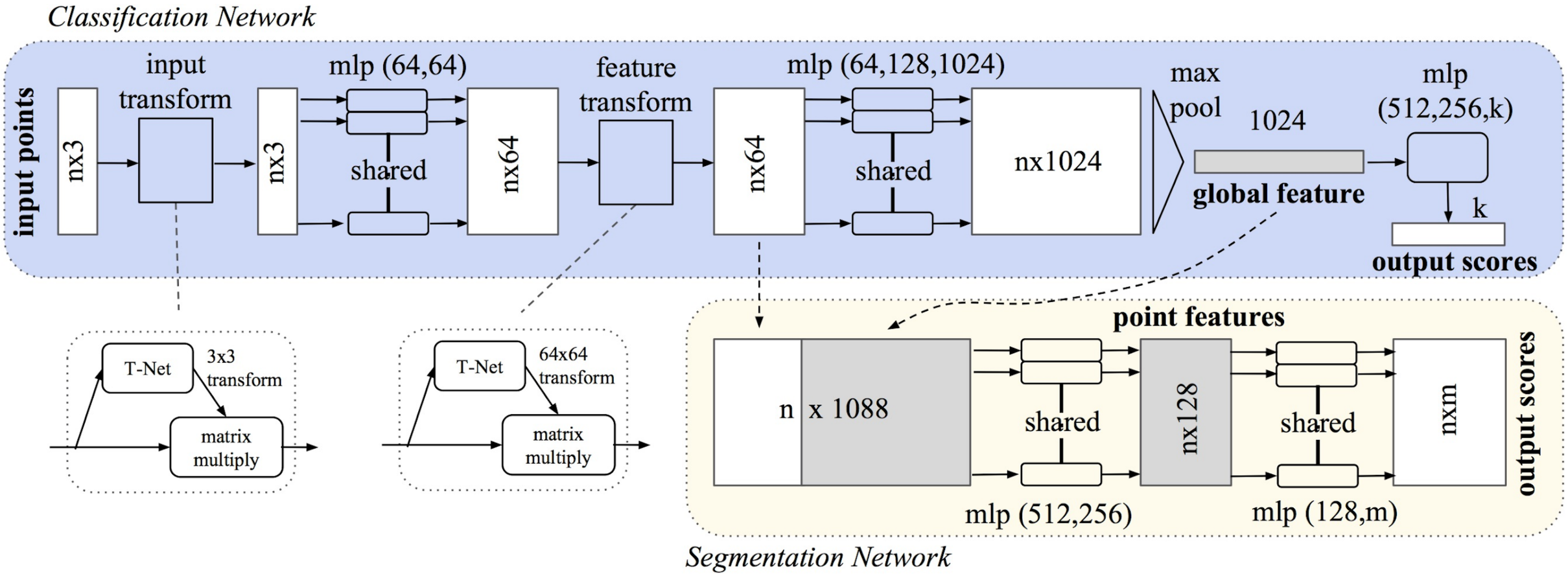
$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

$$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$$



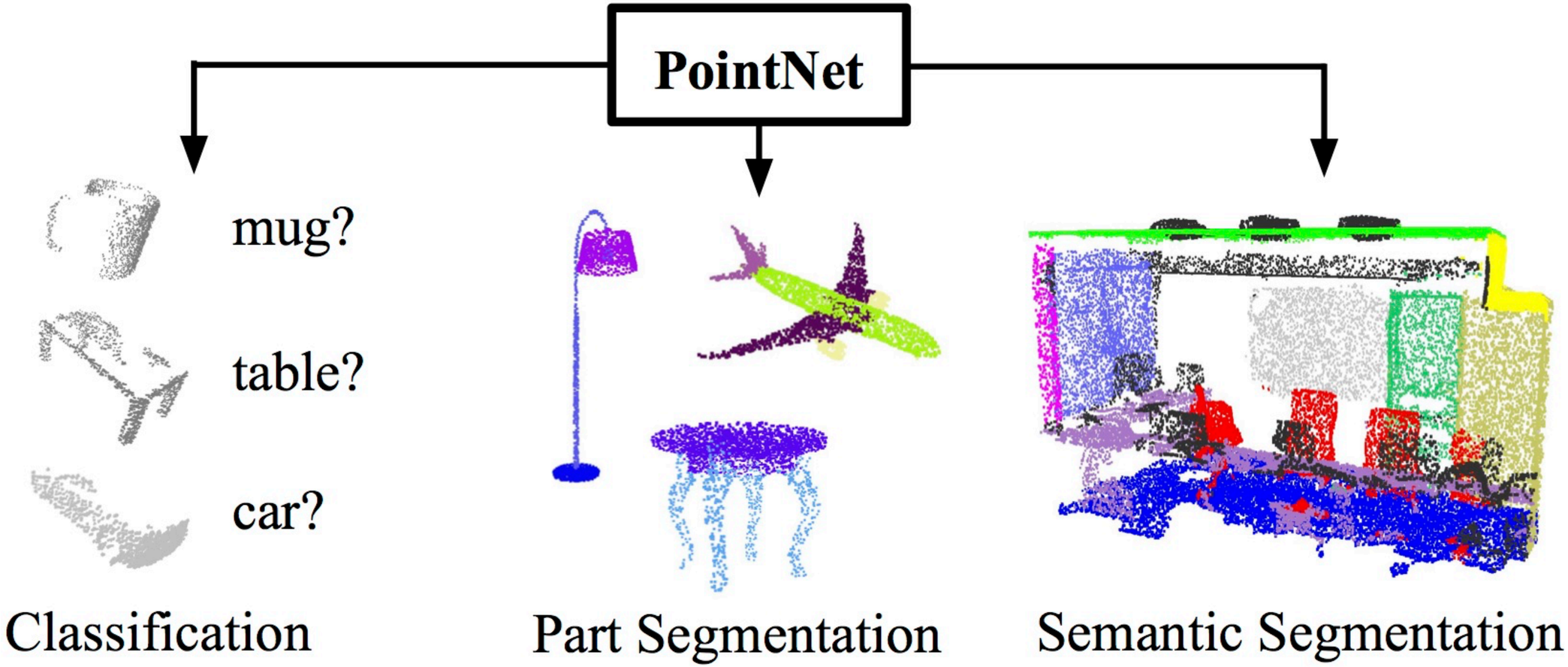
[Qi et al. 2017]

PointNet Architecture

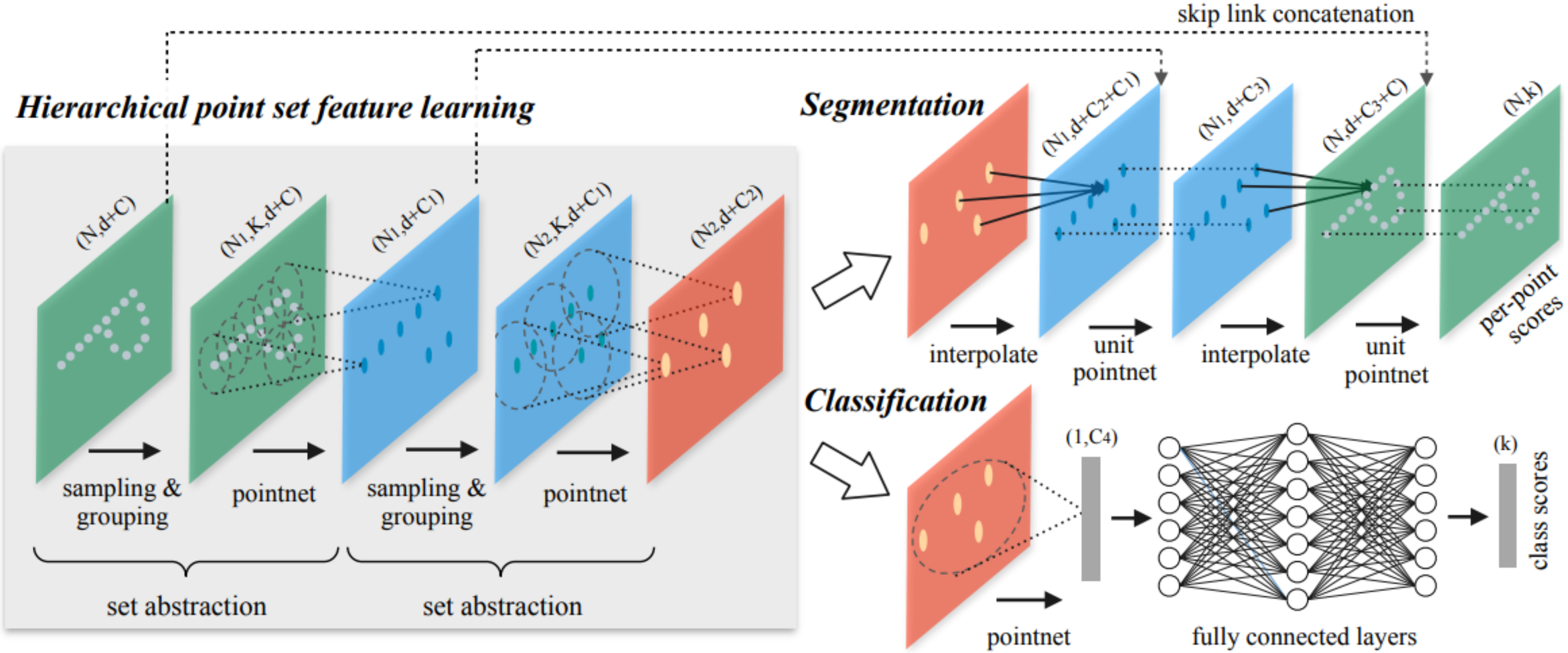


[Qi et al. 2017]

PointNet for Point Cloud Analysis

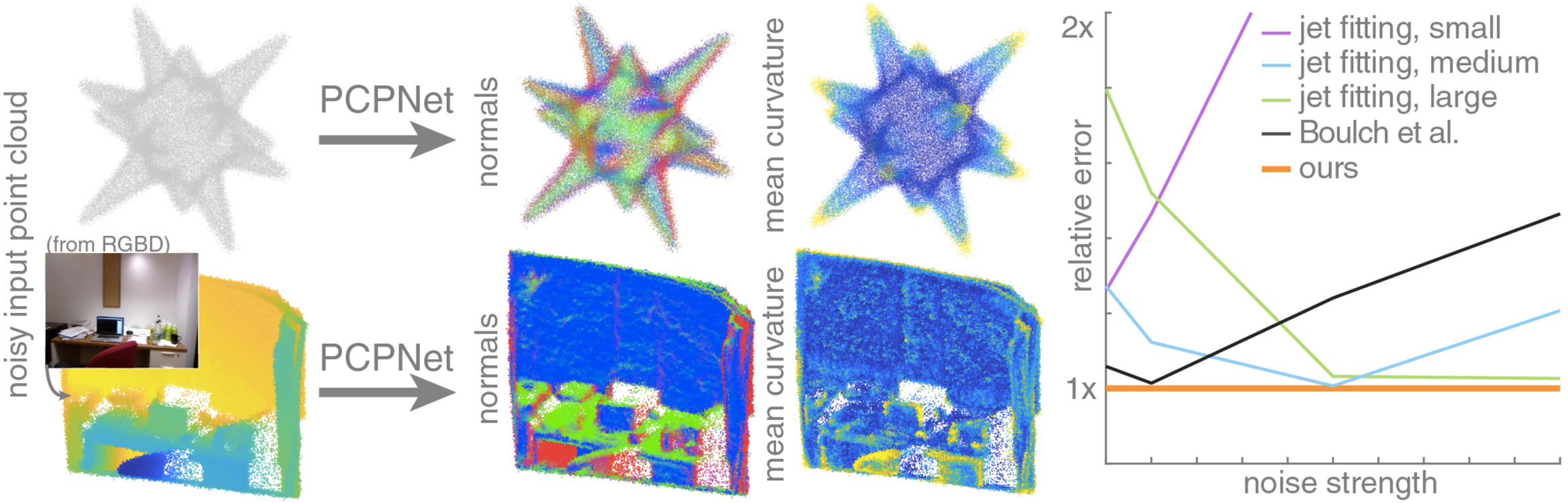


PointNet++



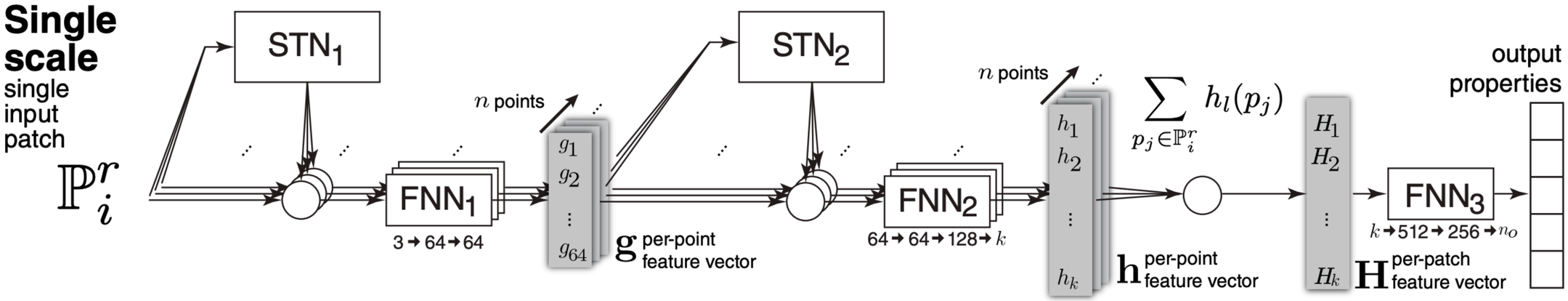
[Qi et al. 2018]

PCPNet for Local Point Cloud Analysis

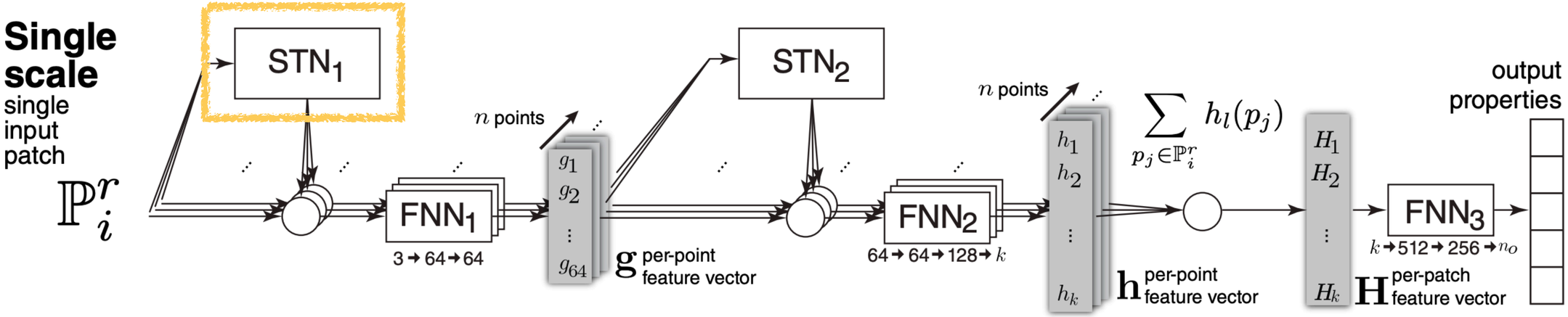


[Guerrero et al. 2018]

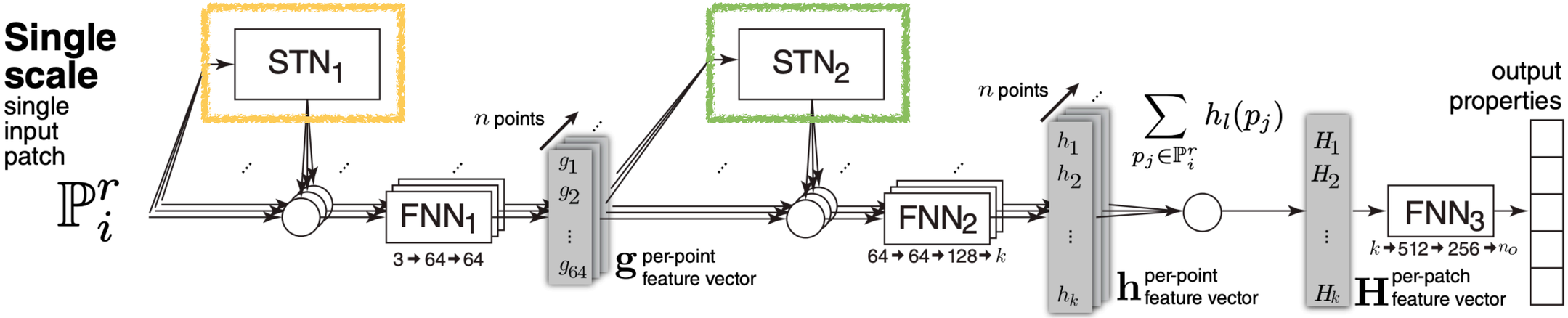
PCPNet Architecture



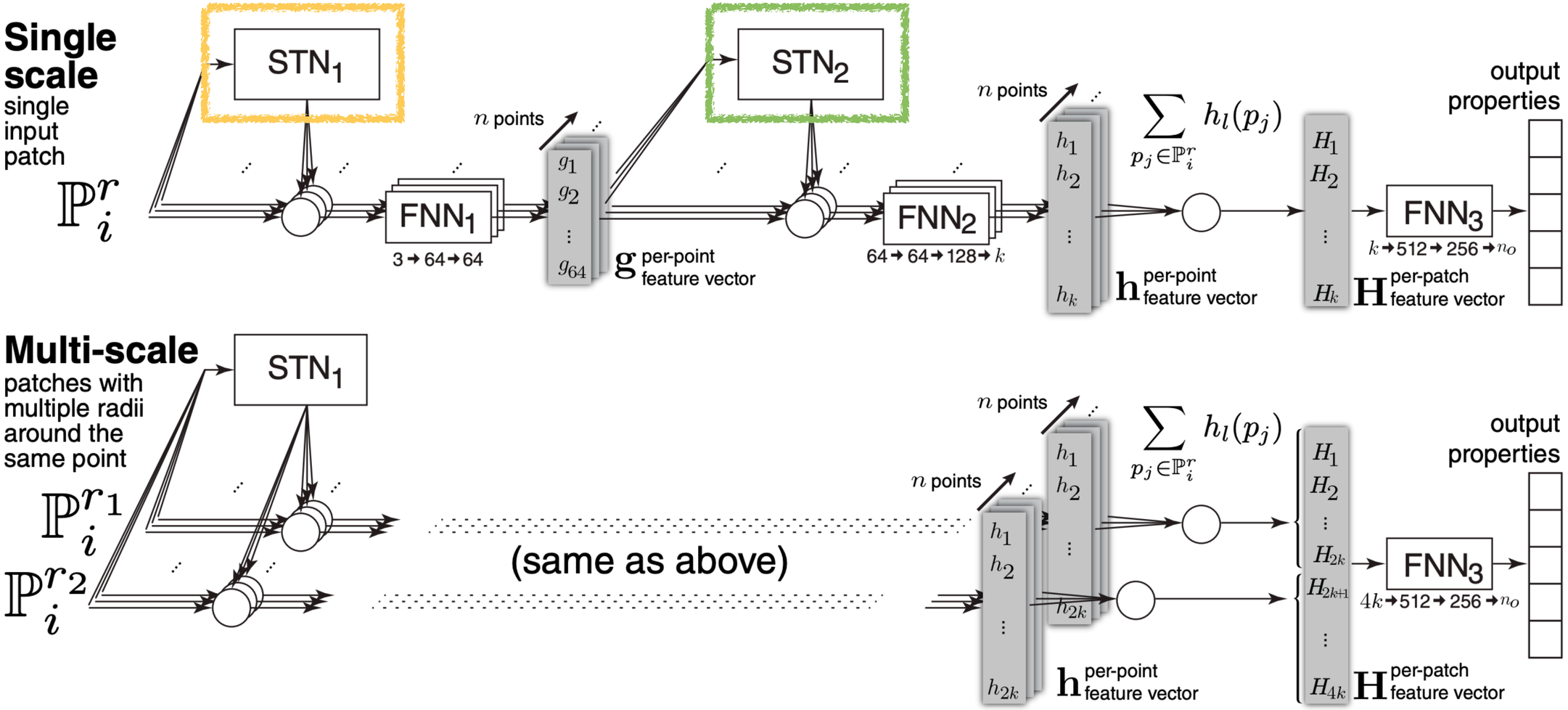
PCPNet Architecture



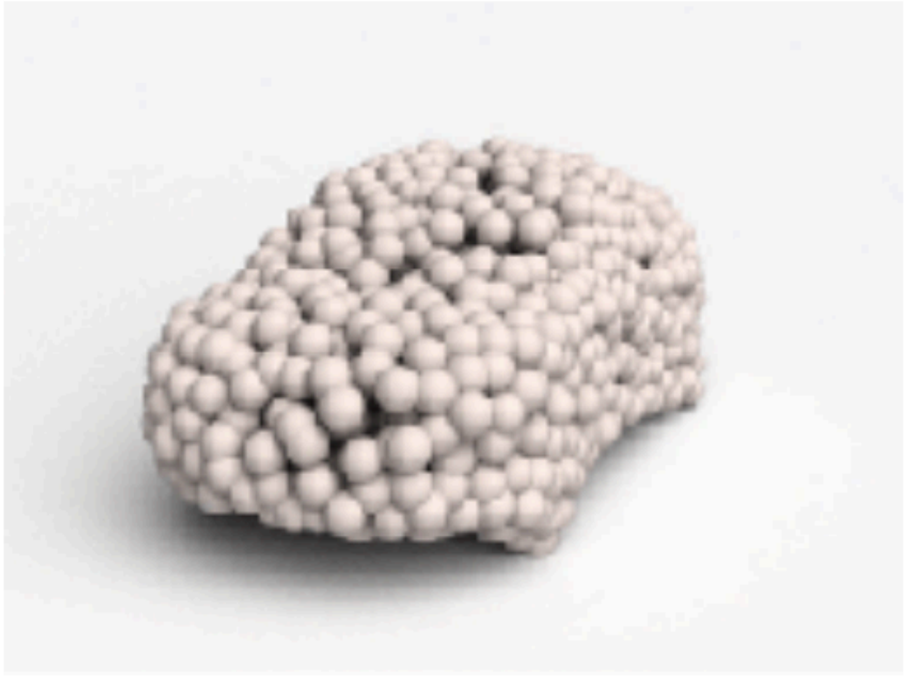
PCPNet Architecture



PCPNet Architecture

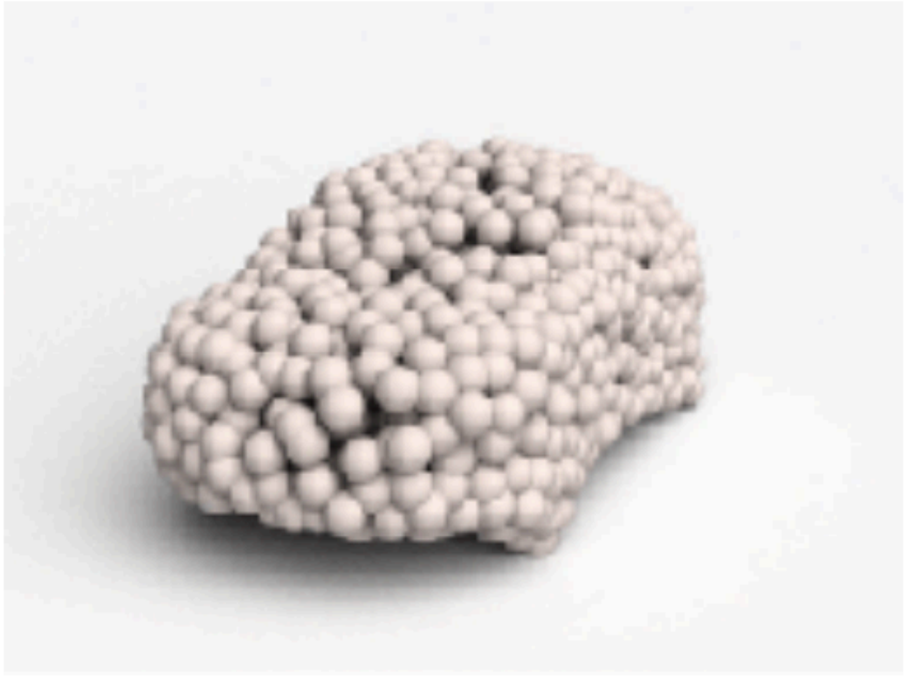


PointNet for Point Cloud Synthesis



[Su et al. 2017]

PointNet for Point Cloud Synthesis

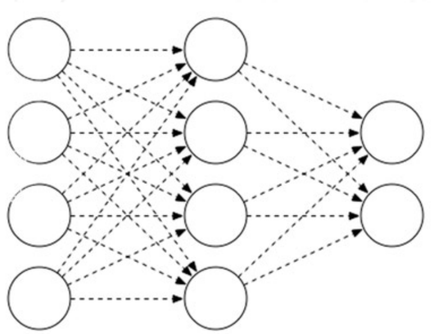


Earth Mover Distance as loss function

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

[Su et al. 2017]

Course Information (slides/code/comments)



http://geometry.cs.ucl.ac.uk/creativeai/