Cont. of Single-Image to 3D

Hao Su

Image to Surfaces:

Brief Introduction to the Progress of Mesh Editing

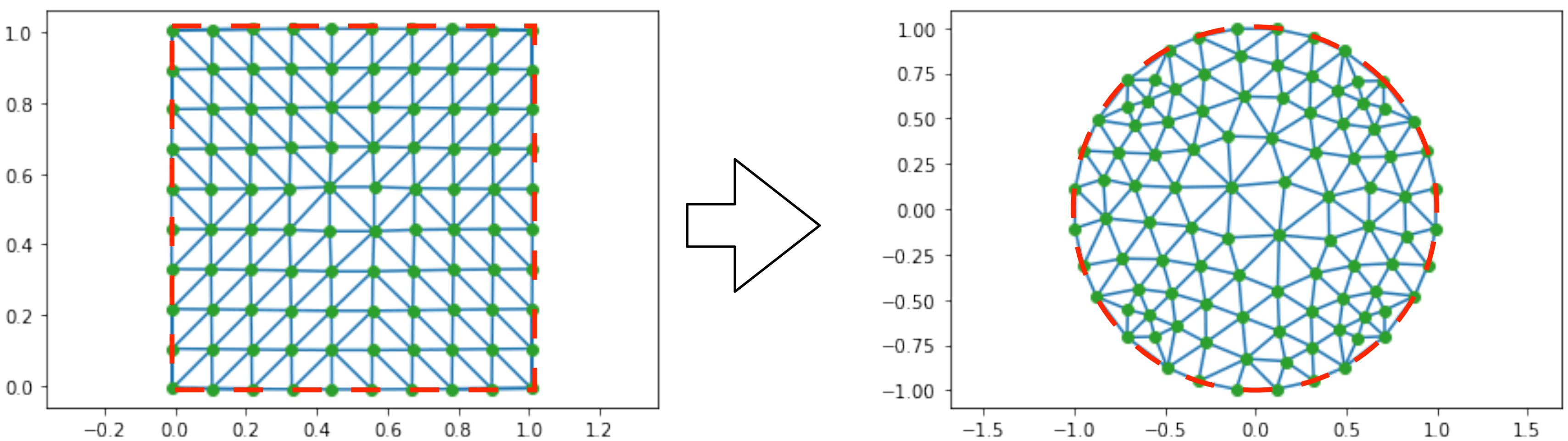
Loss II: Uniform Vertices Distribution

- Penalizes the flying vertices and overlong edges to guarantee the high quality of recovered 3D geometry
- Encourage equal edge length between vertices

$$L_{\text{unif}} = \sum_{p} \sum_{k \in N(p)} \|p - k\|_2^2$$

$$L_{\text{unif}} = \sum_{p} \sum_{q \in N(p)} \|p - q\|_2^2$$

Effect of minimizing l when fixing topology and setting boundary points to the new positions



How to Implement?

- Laplacian matrix:
  - A: adjacency matrix ($n \times n$)
  - $D = \mathrm{diag}(A \cdot 1)$ (diagonal matrix, $n \times n$)
  - $L = D - A$
- Let $X = [p_1, p_2, \ldots, p_n]^T$ (an $n \times 3$ matrix) whose each column is a point coordinate, and denote the block matrix of $X = [X_1, X_2, X_3]$, then:
  - $\sum_{p} \sum_{q \in N(p)} \|p - q\|^2 = \mathrm{tr}(X^T L X) = \sum_{i=1}^{3} X_i^T L X_i$

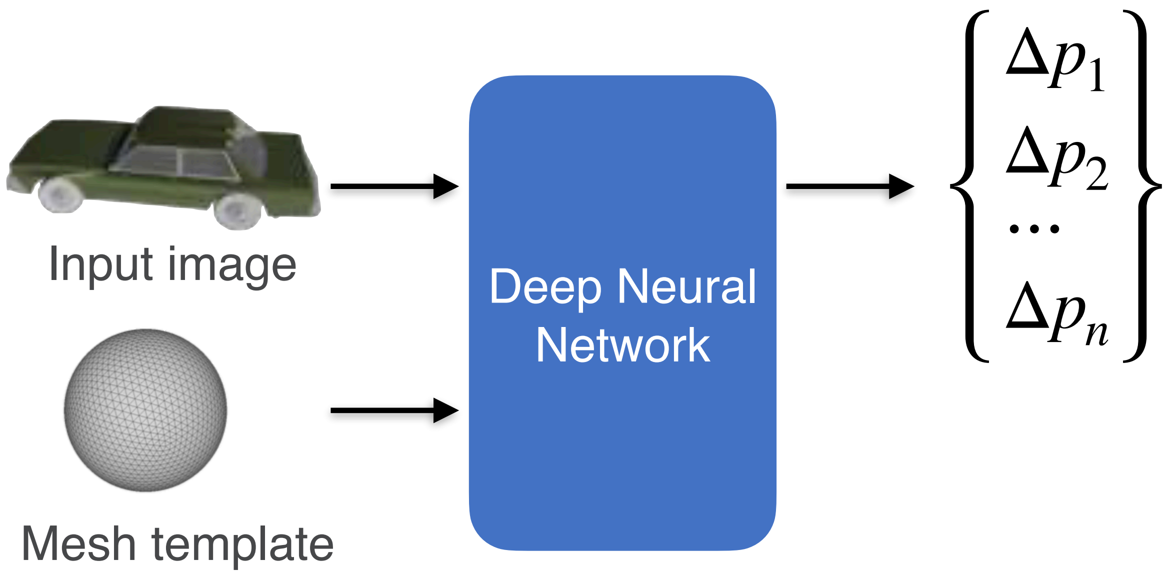
How to Implement?

$$\text{minimize}_{\{X_i\}} \quad \sum_{i=1}^{3} X_i^T L X_i + \text{other losses (e.g, CD or EMD)}$$

$$\text{subject to} \quad A_i X_i = b_i \ \forall i \text{ (boundary conditions)}$$

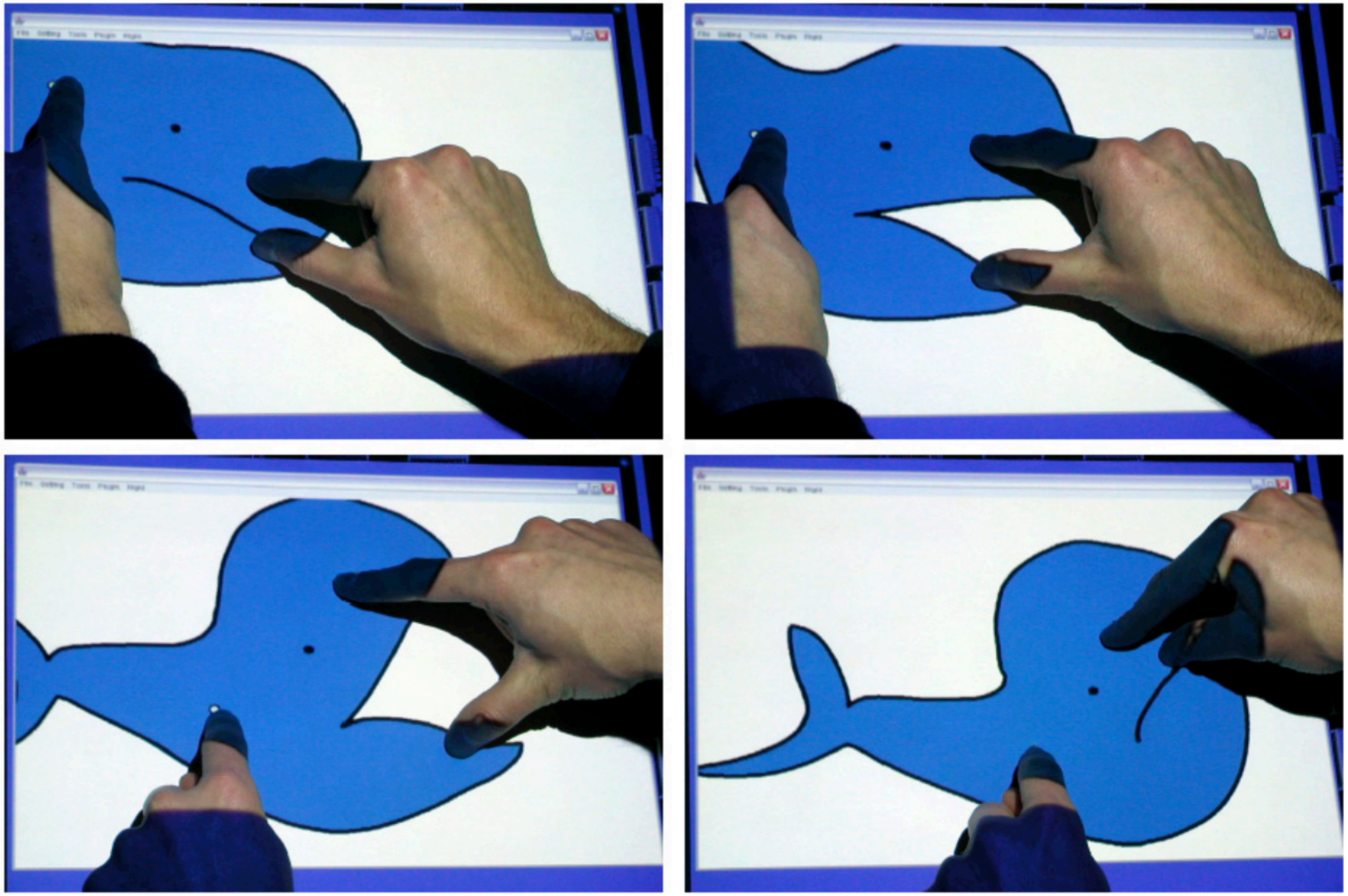
Challenges of Mesh Editing (I)

- In deformation based method, how do we parameterize the movement of vertices of a template mesh?



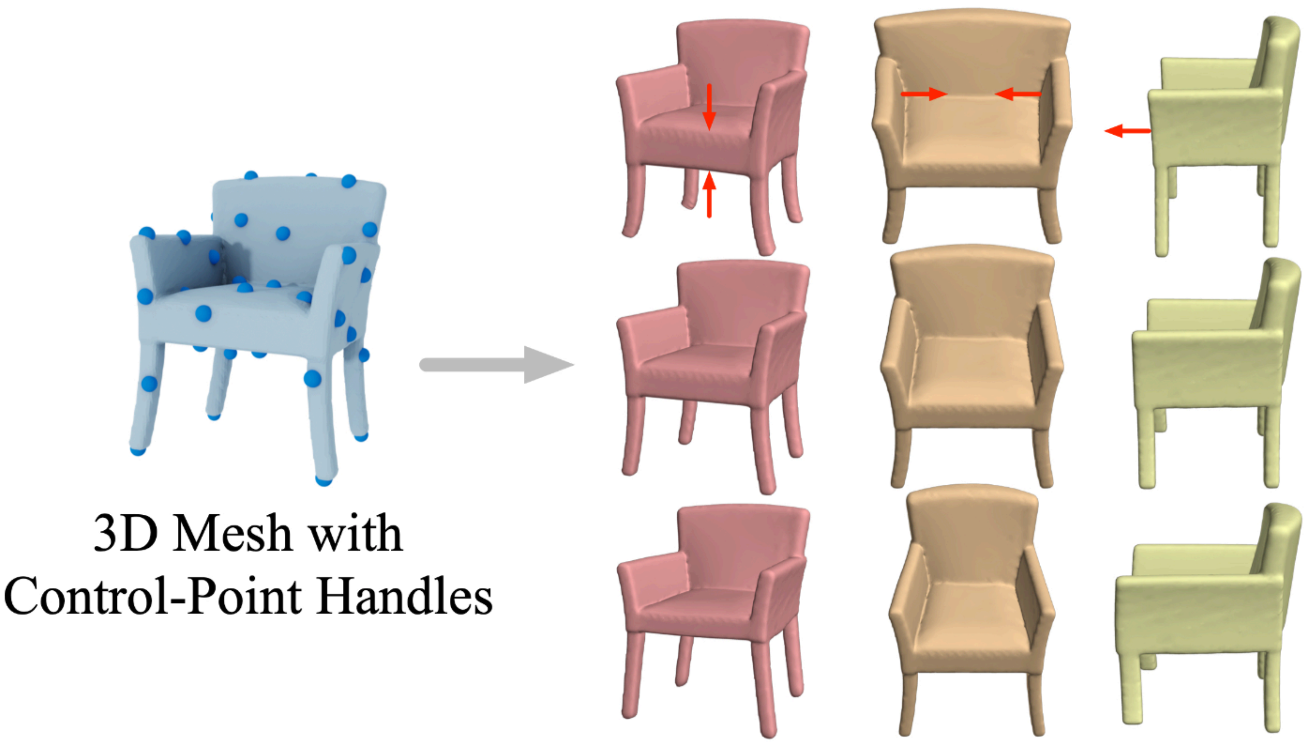
Challenges of Mesh Editing (I)

- While one can control at vertices/edges level, we may expect to find low-dimensional control handles



Deformation Field

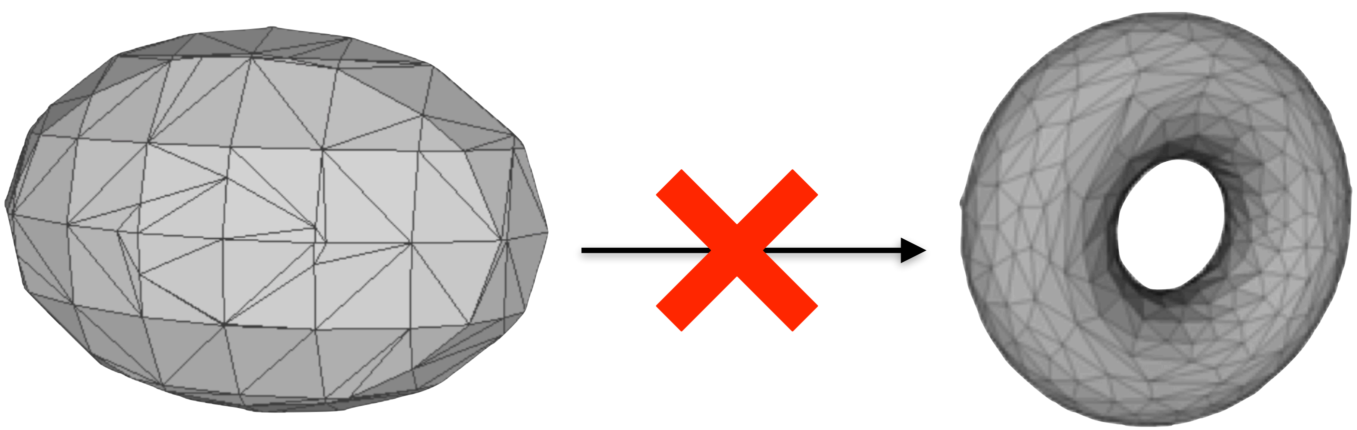
- The movement of the control points warps the space, hence warps any matter in the space



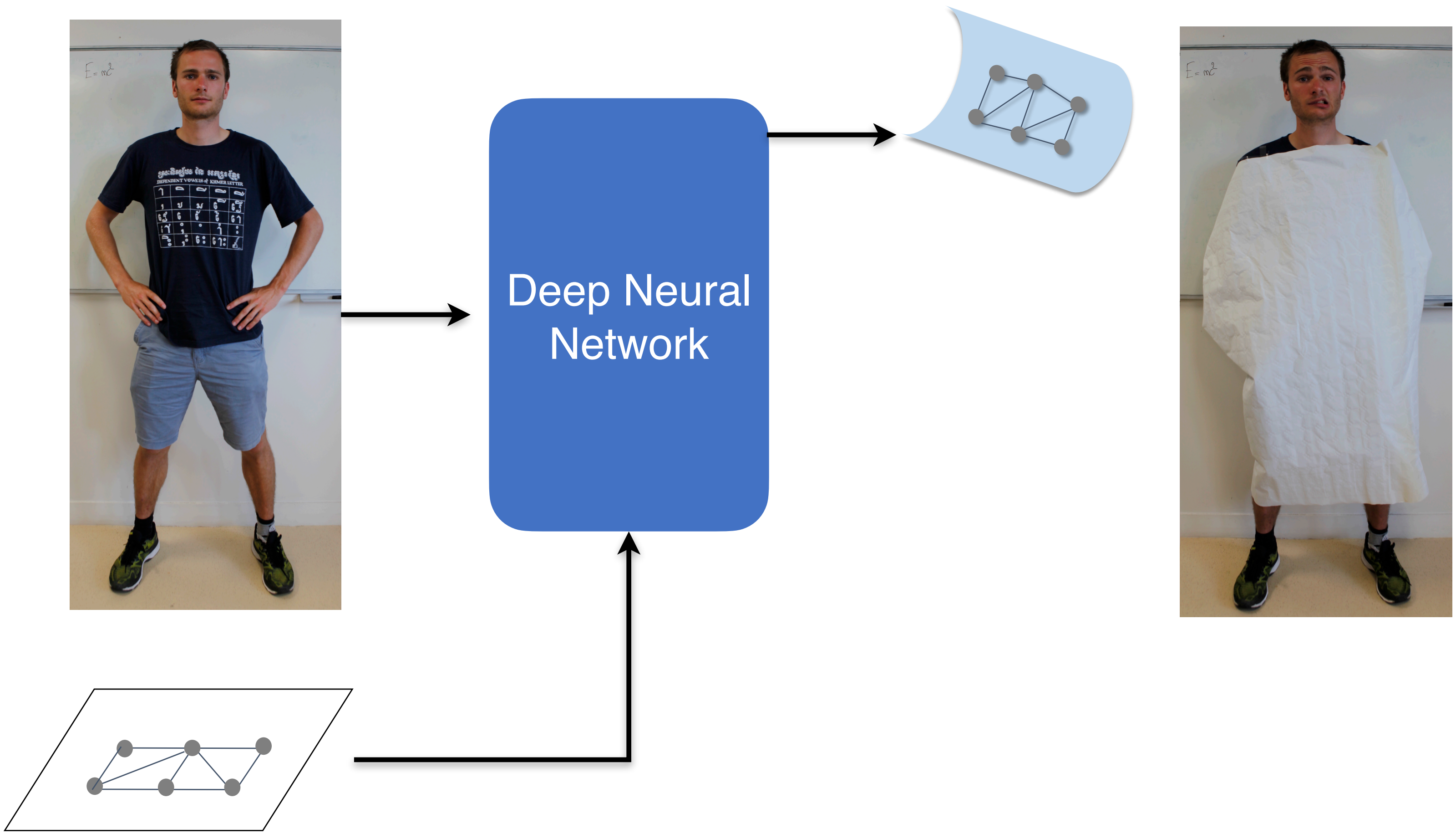
Defer to later lecture

Challenges of Mesh Editing (II)

- Continuous deformation alone is NOT able to change the topology of template mesh.



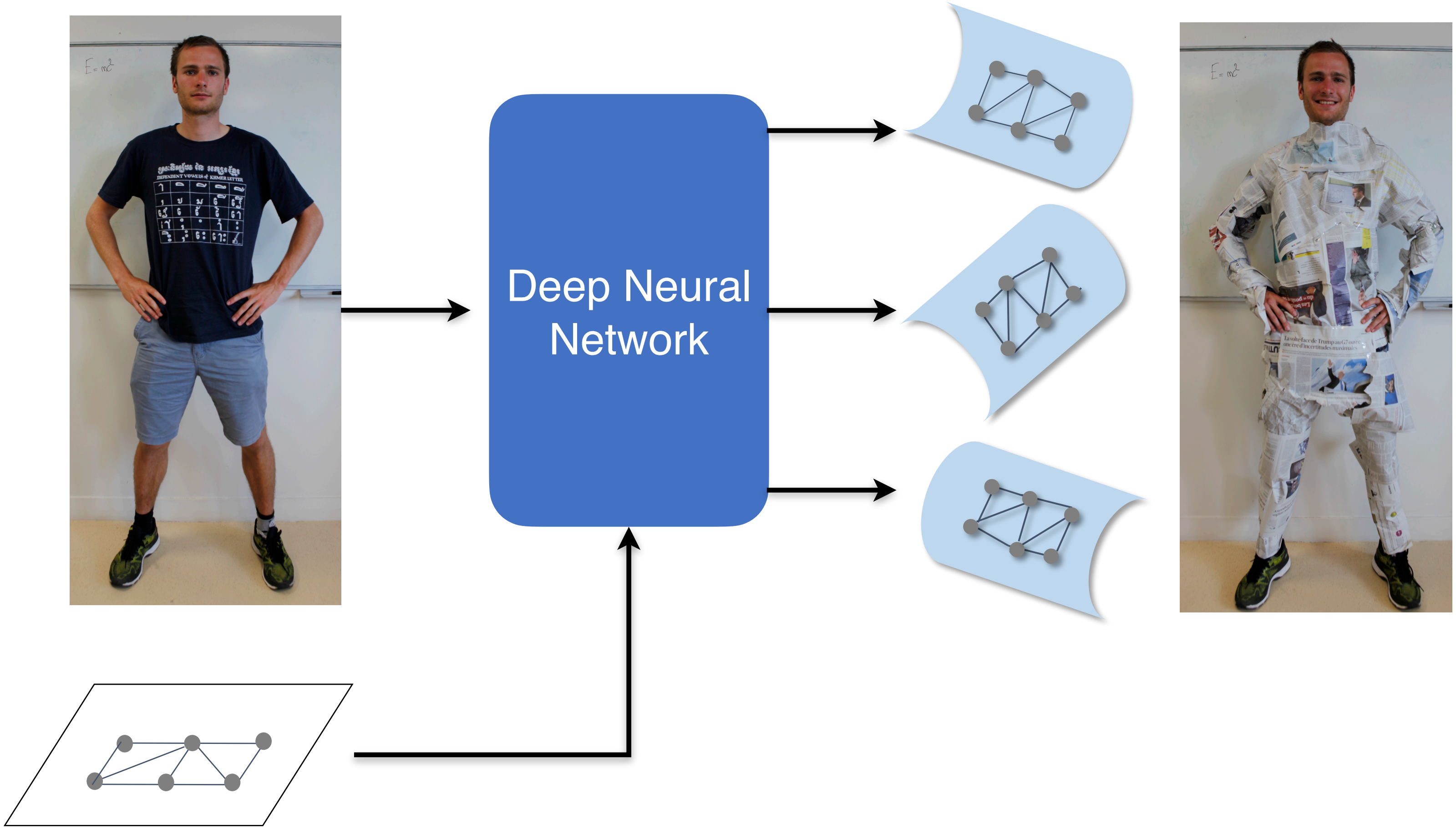
Possible Way: Stitching Multiple Surfaces?



- Idea 1: Multiple template mesh.

Groueix et al, **AtlasNet:A papier-mâché approach to learning 3d surface generation**, CVPR 2018

Possible Way: Stitching Multiple Surfaces?



- Idea 1: Multiple template mesh.

Groueix et al, **AtlasNet:A papier-mâché approach to learning 3d surface generation**, CVPR 2018

Possible Way: Stitching Multiple Surfaces?

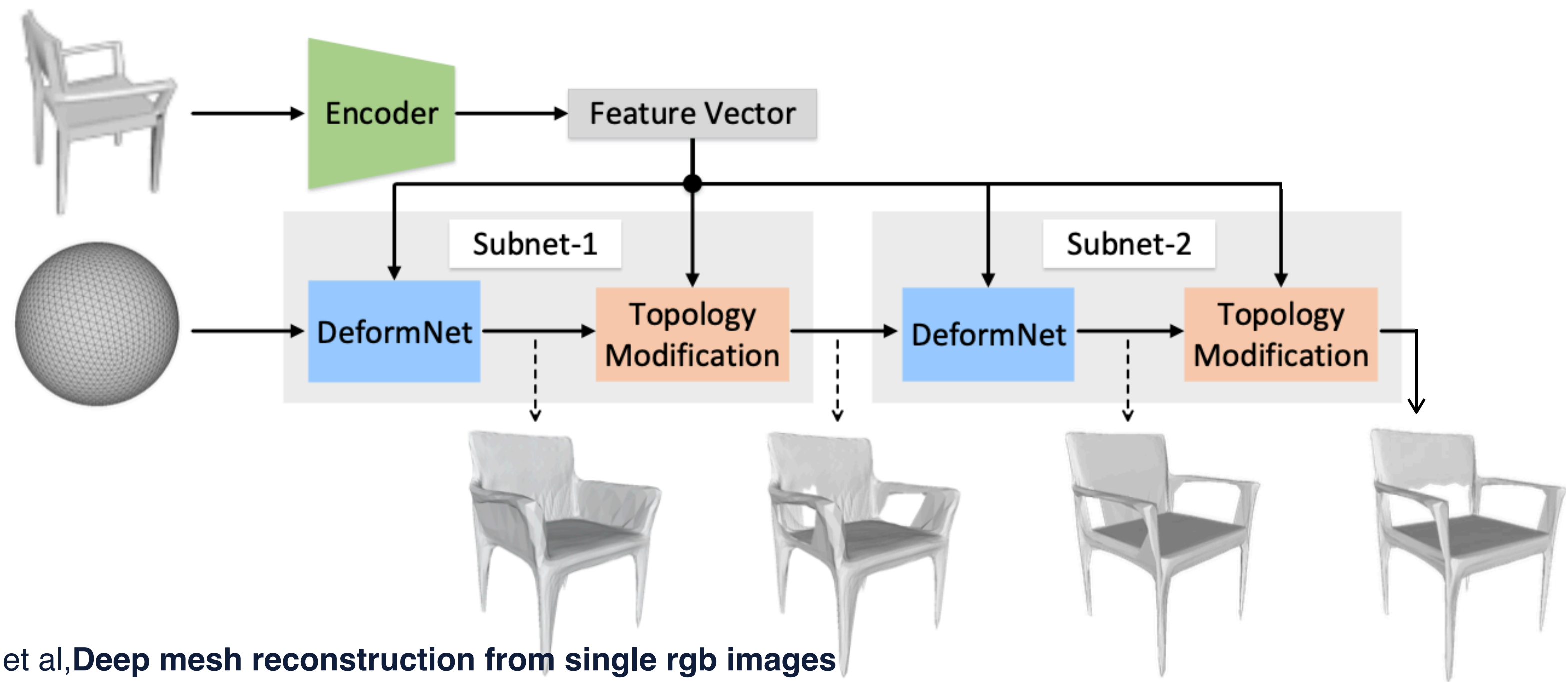
- Issues of multiple templates.
  - Self-intersections and overlaps caused by multiple disconnected patches.
  - Hard to generate a proper deformation that can cover the surface with low distortion.

Groueix et al, **AtlasNet:A papier-mâché approach to learning 3d surface generation**, CVPR 2018

Possible Way: Modifying Shape Topology?

- Idea 1: Multiple template mesh.

- Idea 2: Modify template topology by removing mesh faces.



Pan et al,**Deep mesh reconstruction from single rgb images via topology modification networks**, ICCV 2019.

Possible Way: Modifying Shape Topology?

- Problems of modify template topology by removing mesh faces:
  - Nontrivial to determine a proper pruning threshold.
  - Open boundaries introduced by the face pruning.
  - Hard to generate a proper face pruning for complex shapes.

Summary

- Often have issues with local minimums in optimization

- Adding topology constraints is hard

L8: 3D Networks

Hao Su

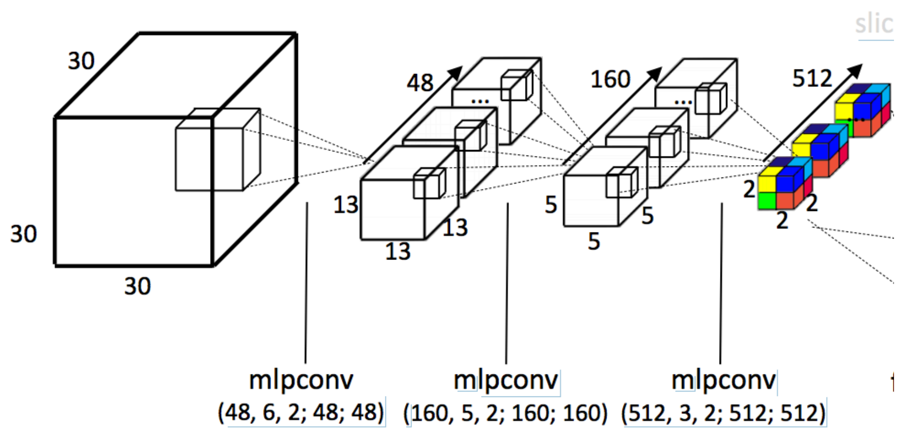
Volumetric CNN

Voxelization

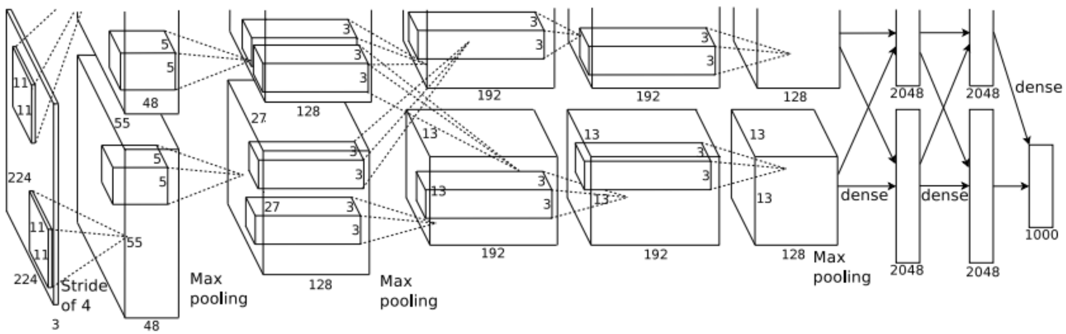
Represent the occupancy of regular 3D grids

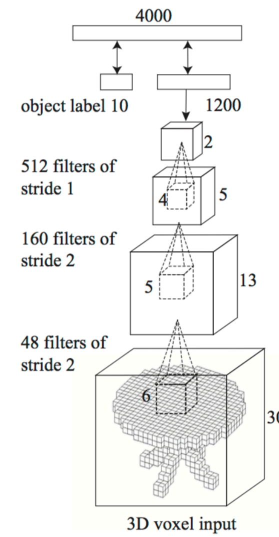
3D CNN on Volumetric Data

3D convolution uses 4D kernels



Complexity Issue





AlexNet, 2012

Input resolution: 224x224

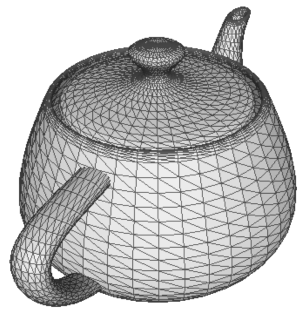
224x224=50176

3DShapeNets, 2015

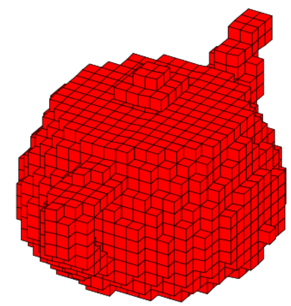
Input resolution: 30x30x30

224x224=27000

Complexity Issue



Polygon Mesh

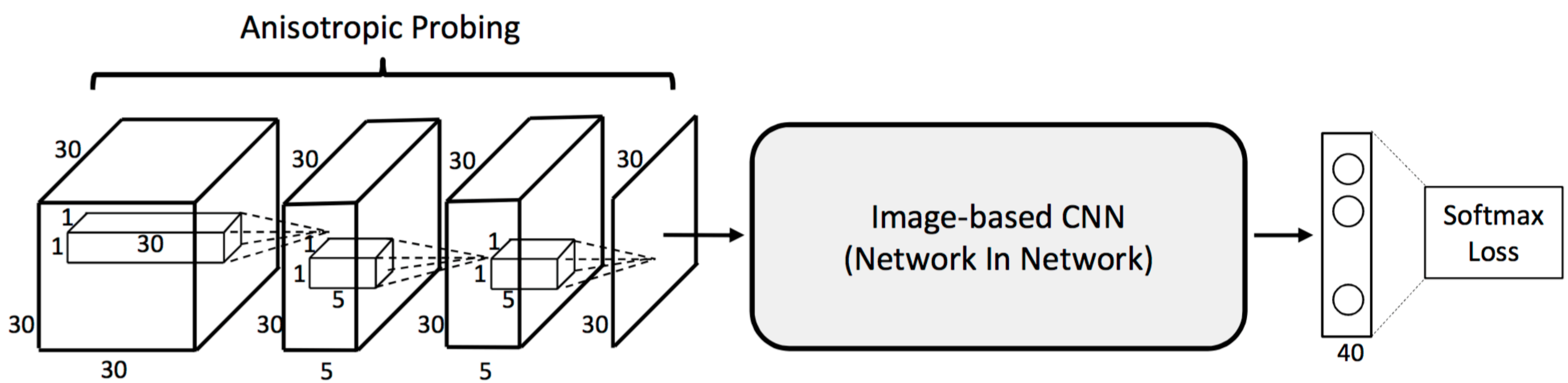


Occupancy Grid
30x30x30

Information loss in voxelization

Idea 1: Learn to Project

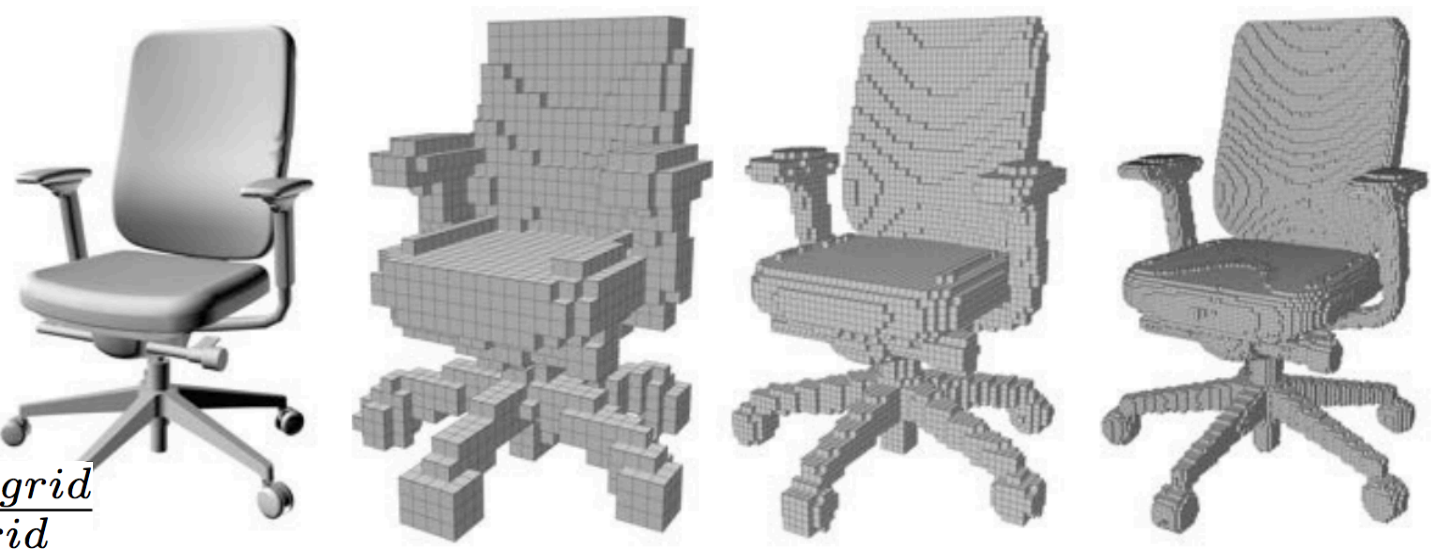
Idea: "X-ray" rendering + Image (2D) CNNs
very low #param, very low computation



Su et al., "**Volumetric and Multi-View CNNs for Object Classification on 3D Data**", *CVPR 2016*

Many other works in autonomous driving that uses **bird's eye view** for object detection

More Principled: Sparsity of 3D Shapes



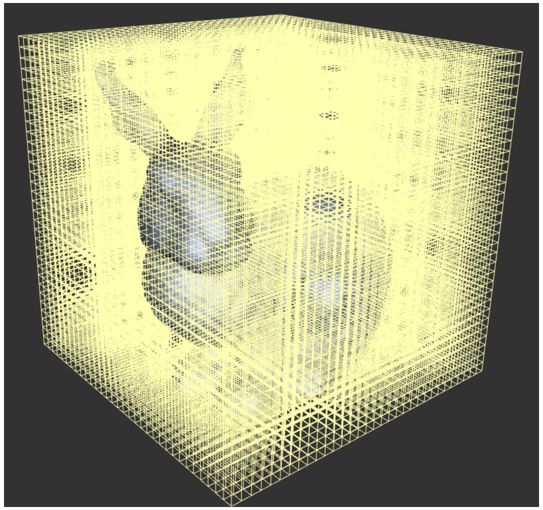
$$\frac{\#occupied\ grid}{\#total\ grid}$$

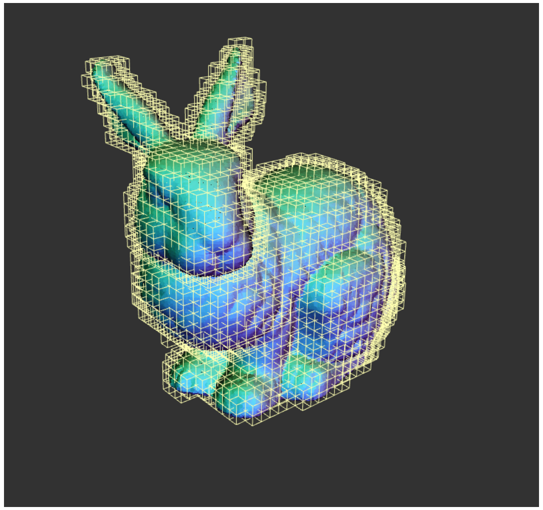
Occupancy: 10.41% 5.09% 2.41%

Resolution: 32 64 128

Store only the Occupied Grids

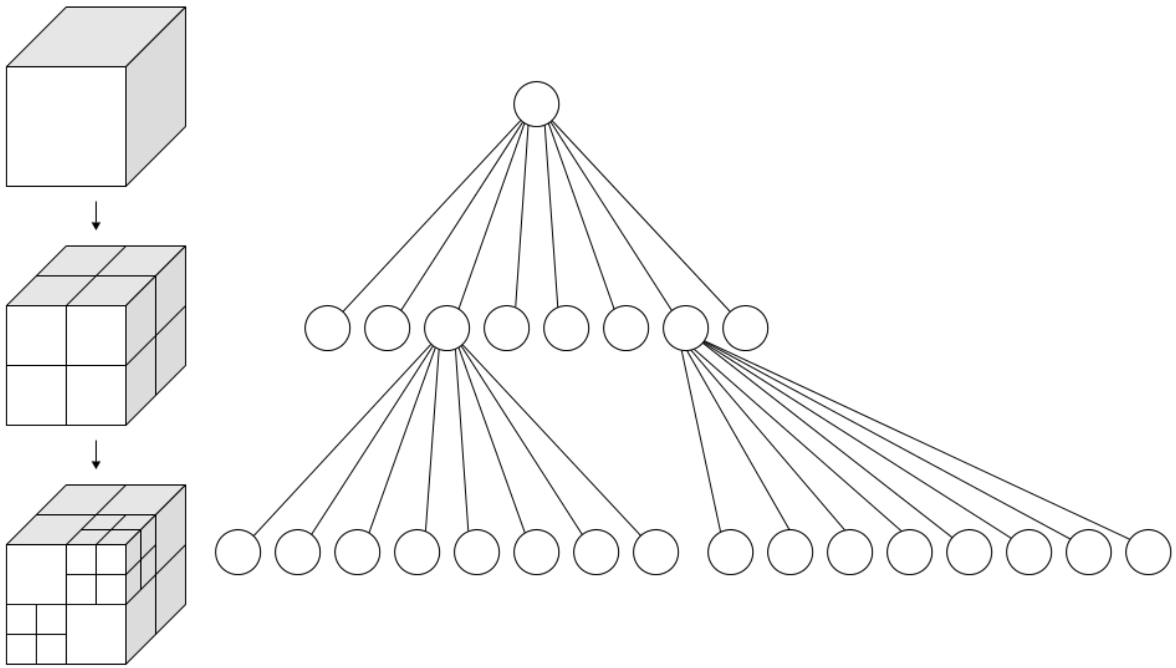
- Store the sparse surface signals
- Constrain the computation near the surface



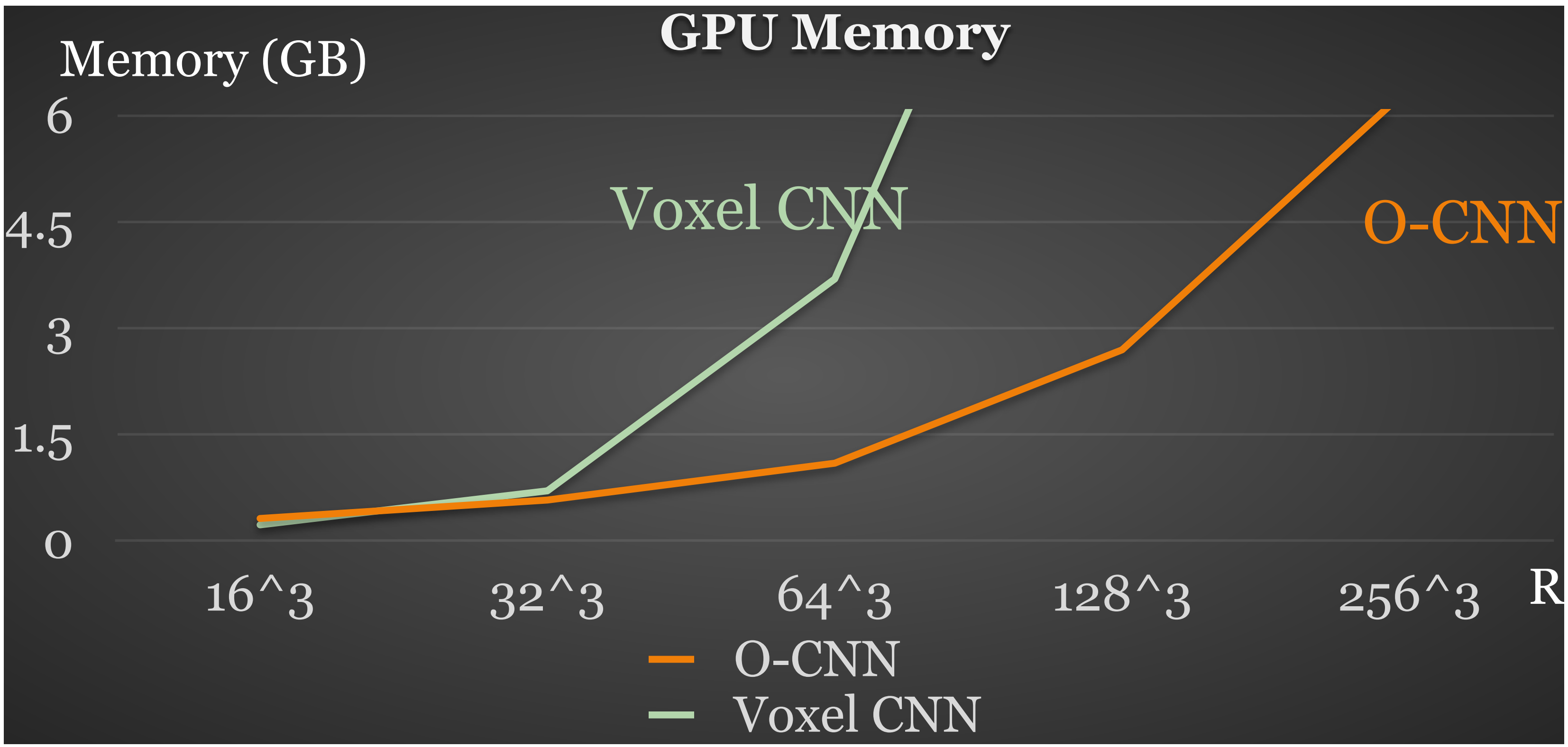


Octree: Recursively Partition the Space

- Each internal node has exactly eight children
- Neighborhood searching: Hash table



Memory Efficiency

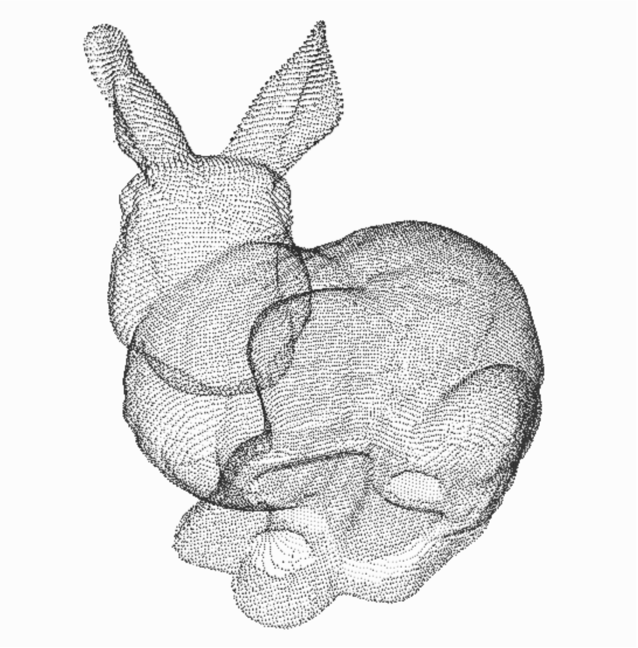


Implementation

- SparseConvNet
  - https://github.com/facebookresearch/SparseConvNet
  - Uses ResNet architecture
  - State-of-the-art for 3D analysis
  - Takes time to train

Graham et al., “**Submanifold Sparse Convolutional Networks**”, *arxiv*

Point Networks

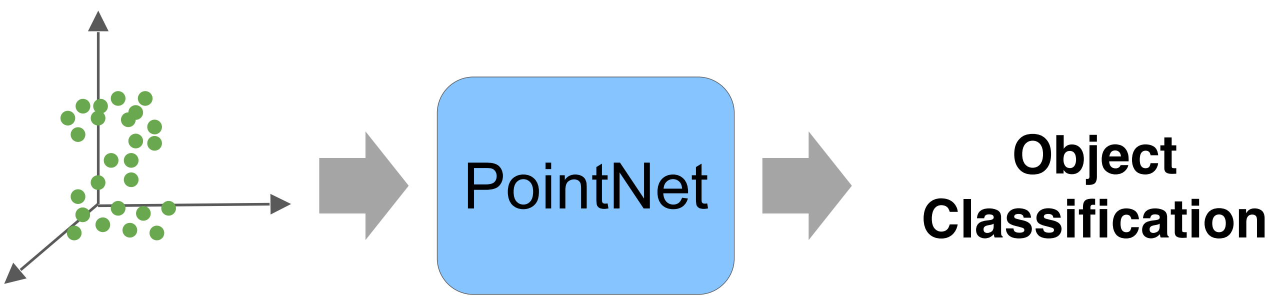


Point cloud

(The most common 3D sensor data)

Directly Process Point Cloud Data

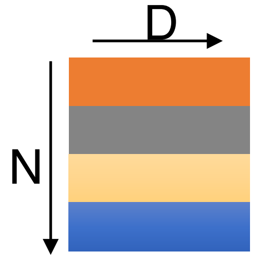
End-to-end learning for **unstructured, unordered** point data



Qi, Charles R., et al. "**Pointnet: Deep learning on point sets for 3d classification and segmentation**", CVPR 2017
Zaheer, Manzil, et al. "**Deep sets**", NeurIPS 2017

Properties of a Desired Point Network

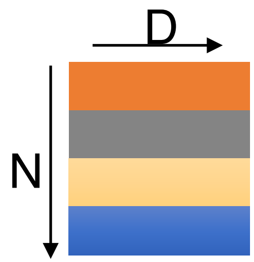
Point cloud: N **orderless** points, each represented by a D dim coordinate



2D array representation

Properties of a Desired Point Network

Point cloud: N **orderless** points, each represented by a D dim coordinate



2D array representation

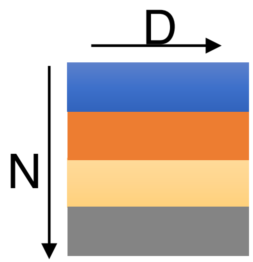
Permutation invariance

Transformation invariance

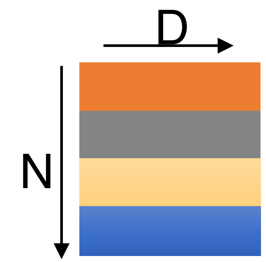
Permutation Invariance of PointNet

Permutation Invariance

Point cloud: N **orderless** points, each represented by a D dim coordinate



represents the same **set** as



2D array representation

Permutation Invariance: Symmetric Function

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

Examples:

$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$

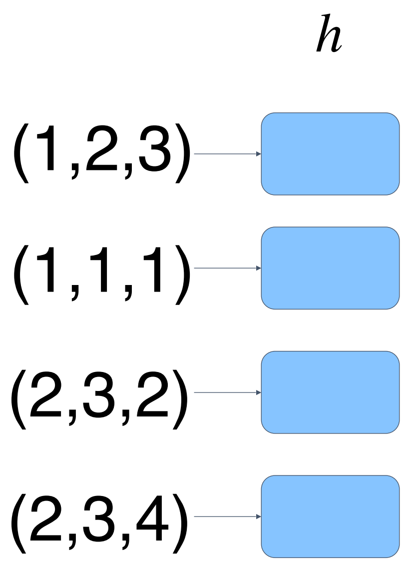
$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

$\ldots$

Construct a Symmetric Function

Observe:

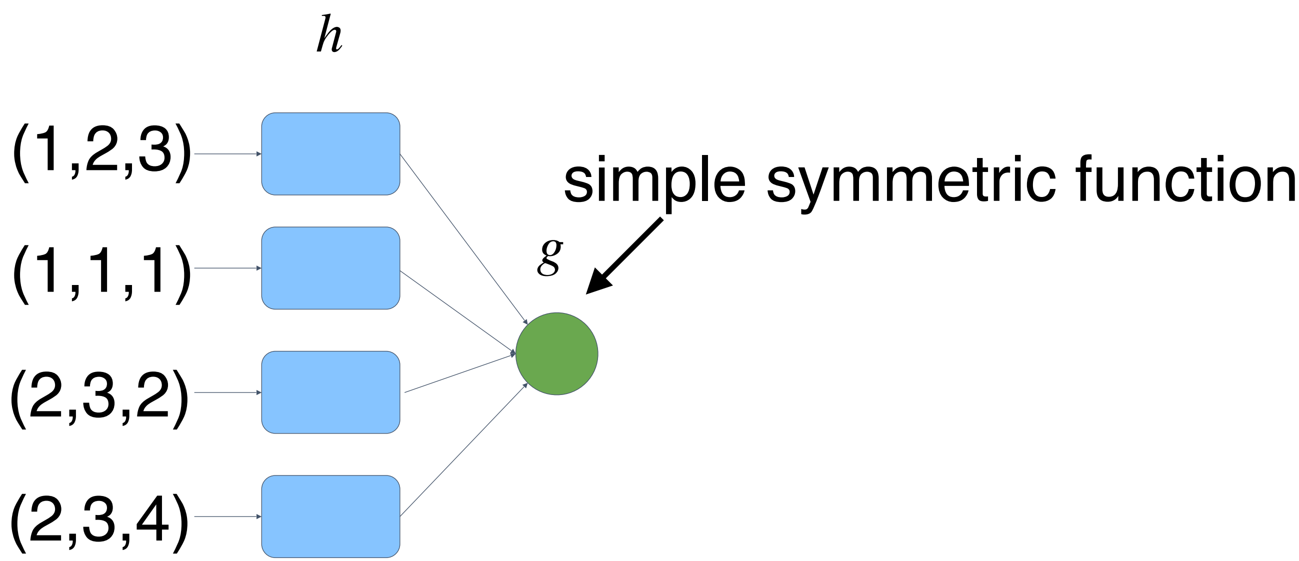
$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric



Construct a Symmetric Function

Observe:

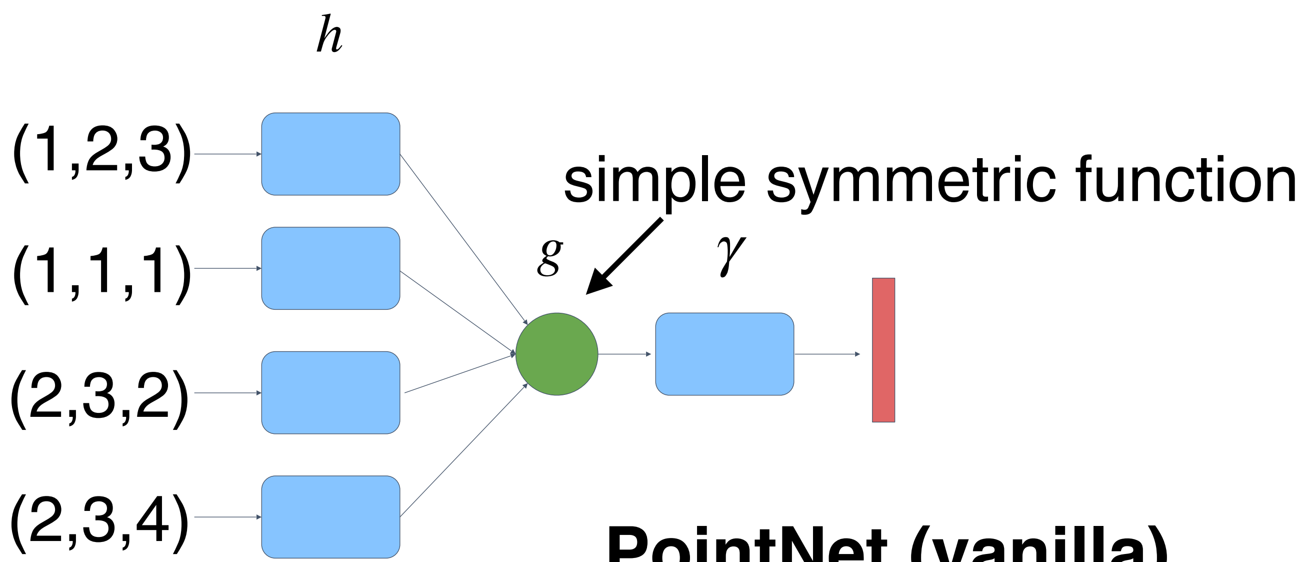
$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric



Construct a Symmetric Function

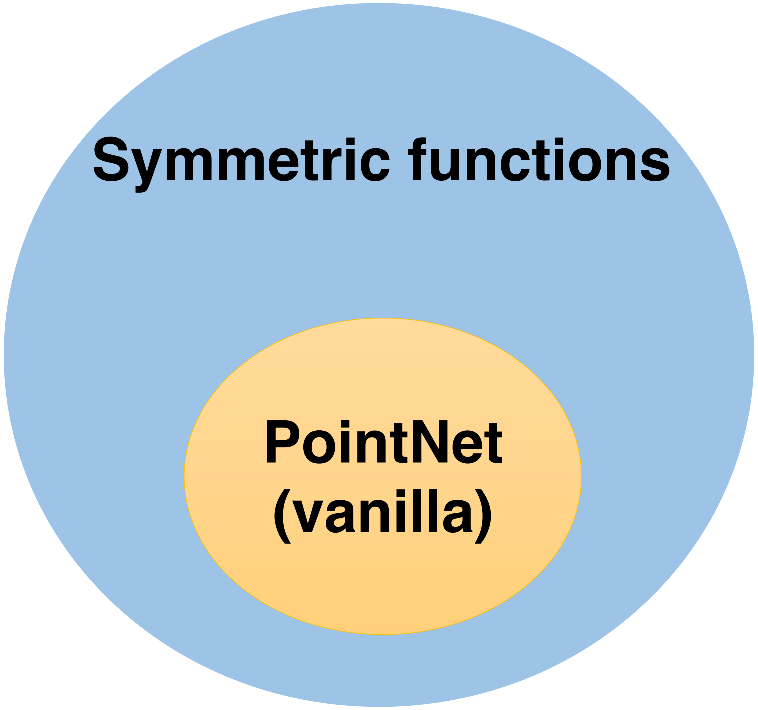
Observe:

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



PointNet (vanilla)

Q: What Symmetric Functions Can Be Constructed by PointNet?



Universal Approximation Theorem

- Can approximate any "continuous" functions over sets
- "Continuous": A function value would change by little if the point positions vary by little

$$\left| f(S) - \boxed{\gamma \left(\underset{x_i \in S}{\mathrm{MAX}} \{h(x_i)\} \right)} \right| < \epsilon$$

$S \subseteq \mathbb{R}^d$,

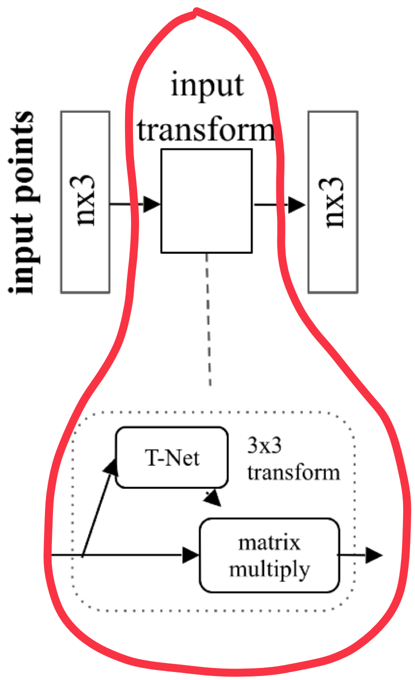
PointNet (vanilla)

A Detailed Implementation of PointNet

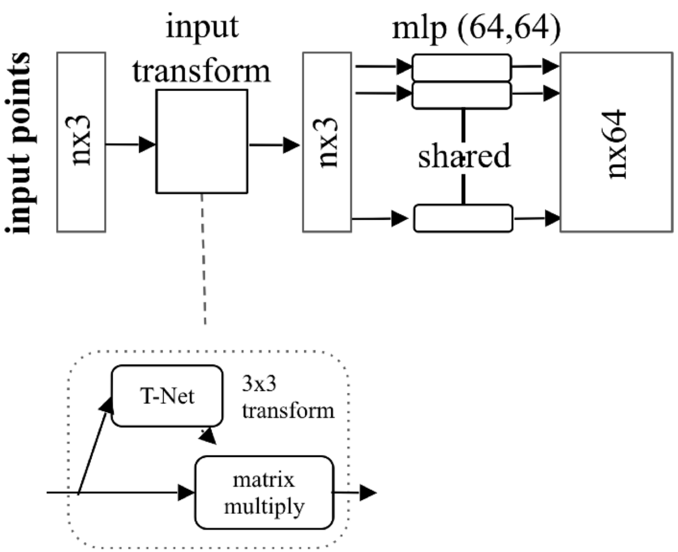
PointNet Classification Network

PointNet Classification Network

Marginally helpful when dataset is big

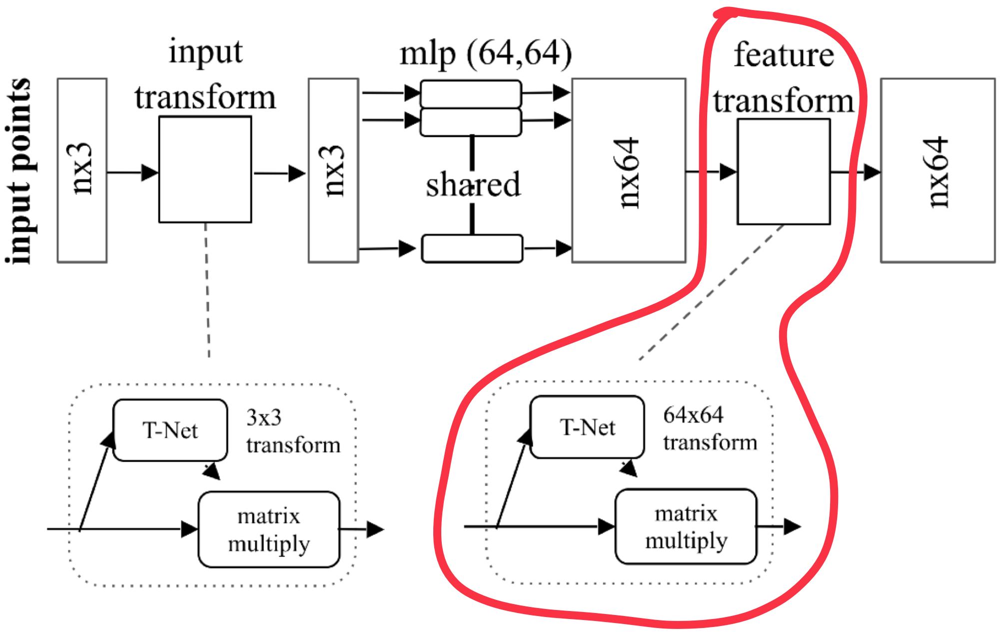


PointNet Classification Network

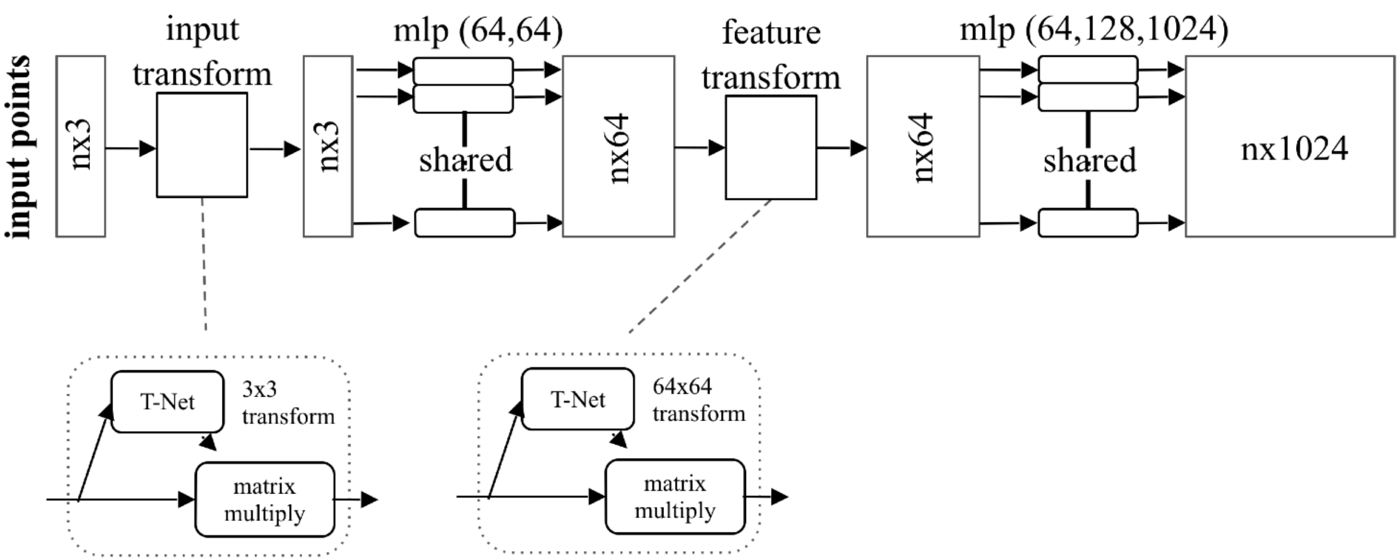


PointNet Classification Network

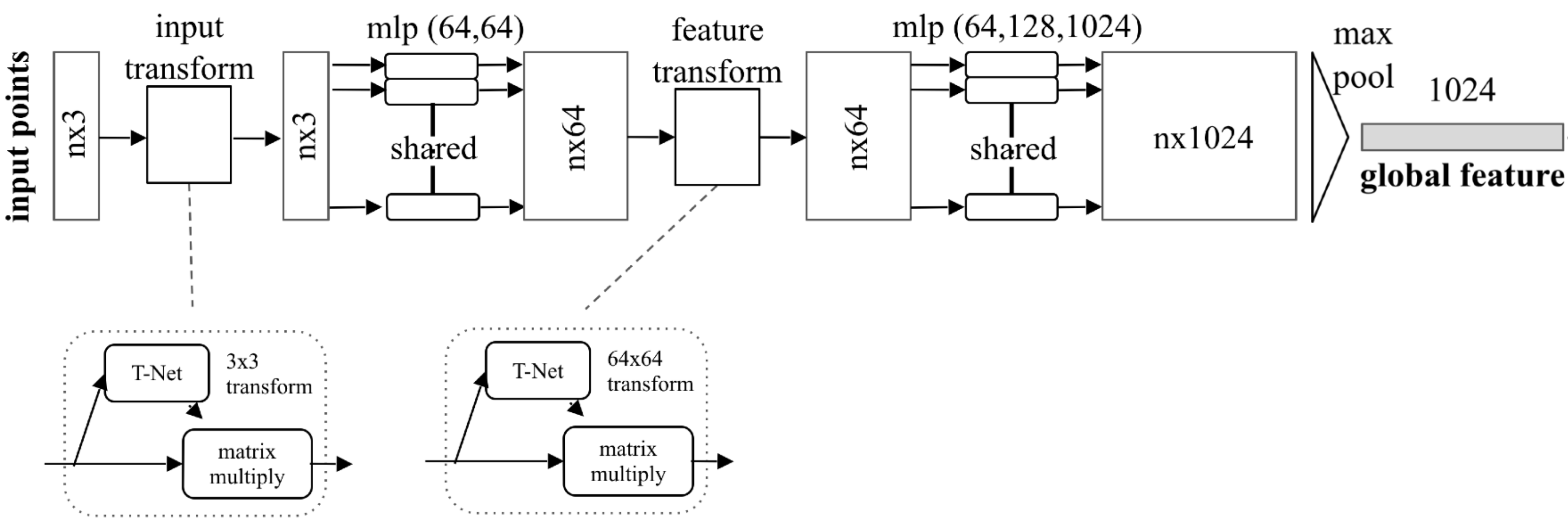
Marginally helpful when dataset is big



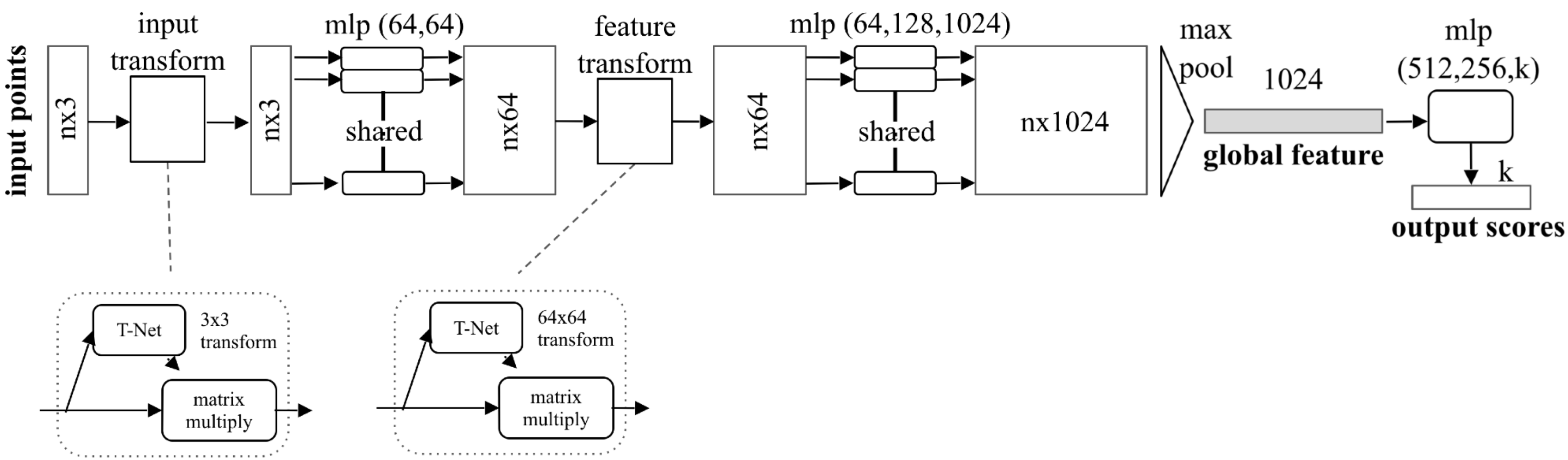
PointNet Classification Network



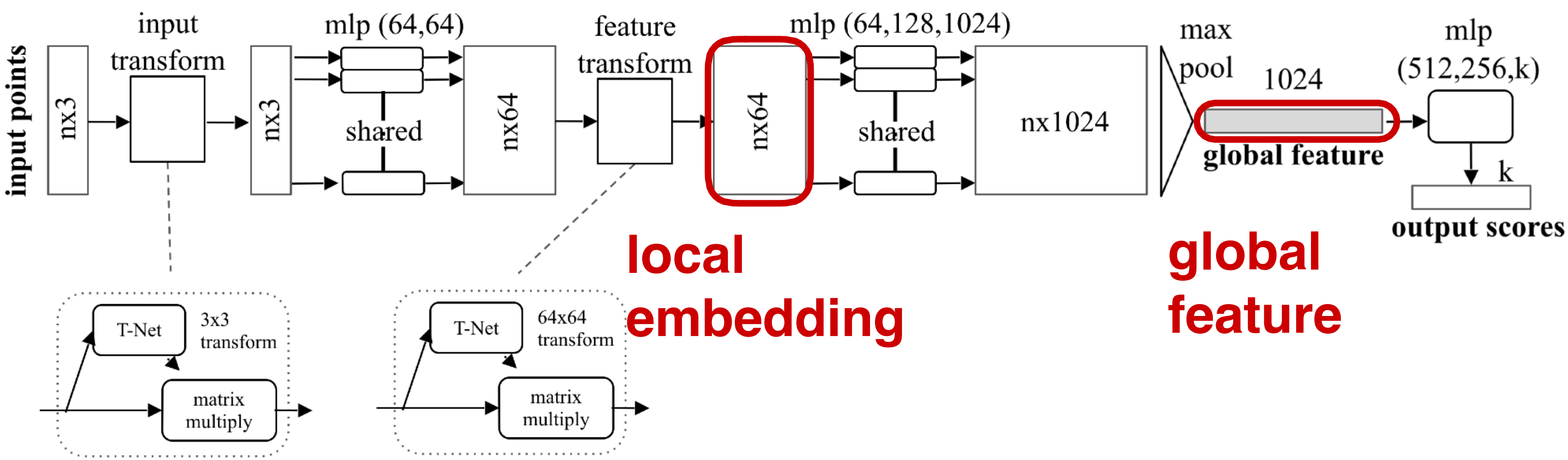
PointNet Classification Network



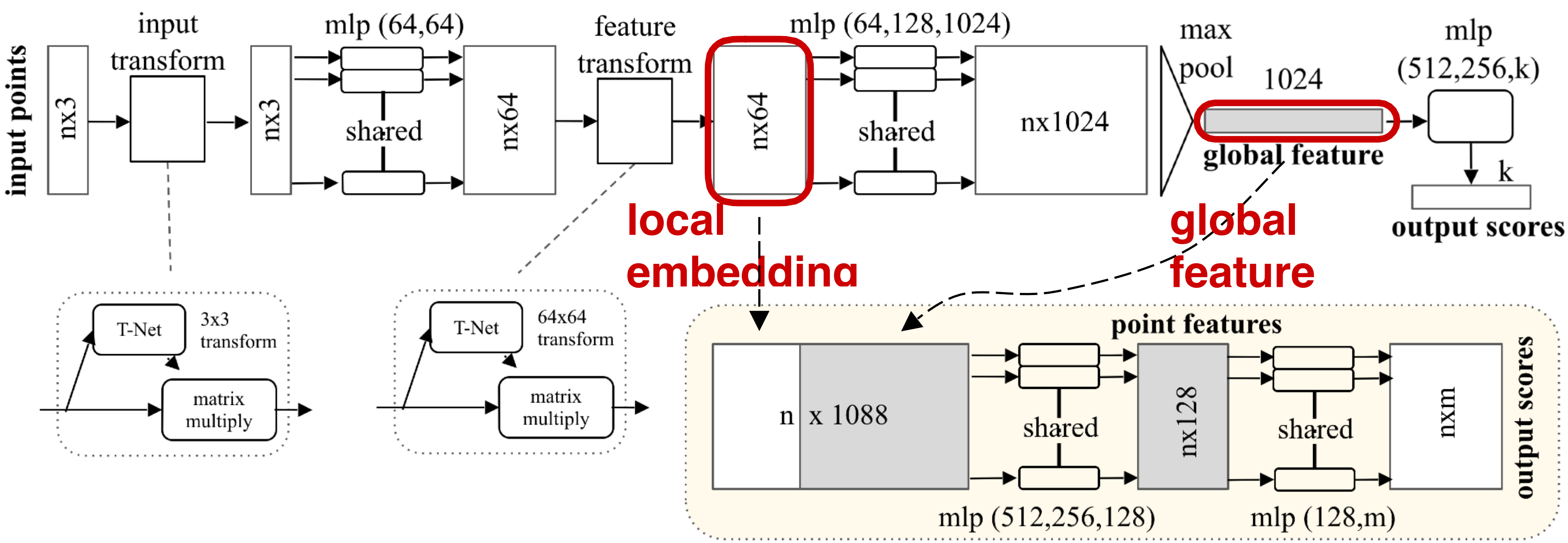
PointNet Classification Network



Extension to Segmentation Network



Extension to Segmentation Network



Robustness to Data Corruption

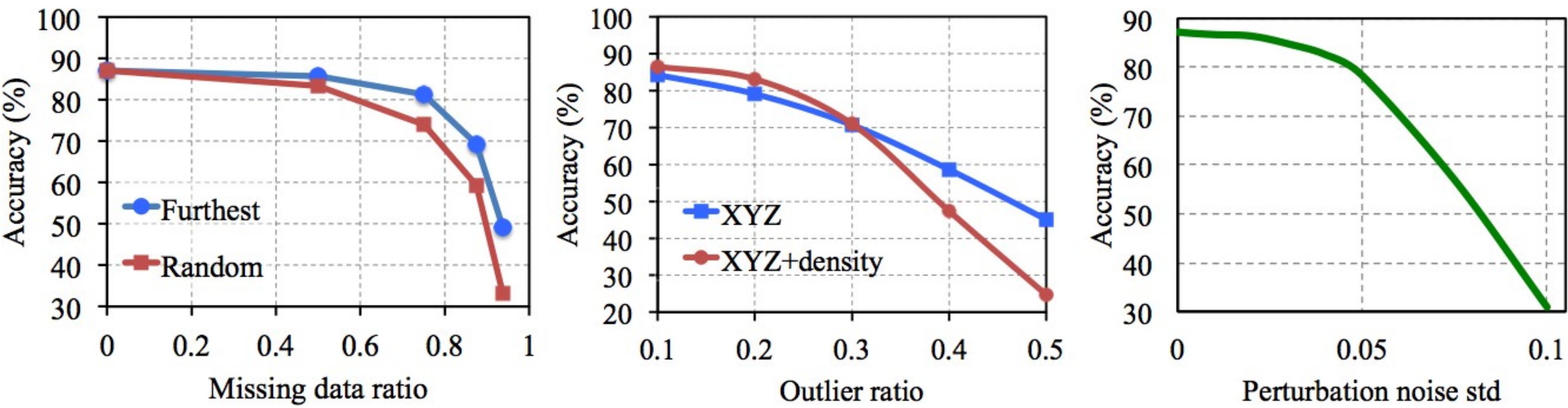
dataset: ModelNet40; metric: 40-class classification accuracy (%)

Robustness to Data Corruption

Less than 2% accuracy drop with 50% missing data

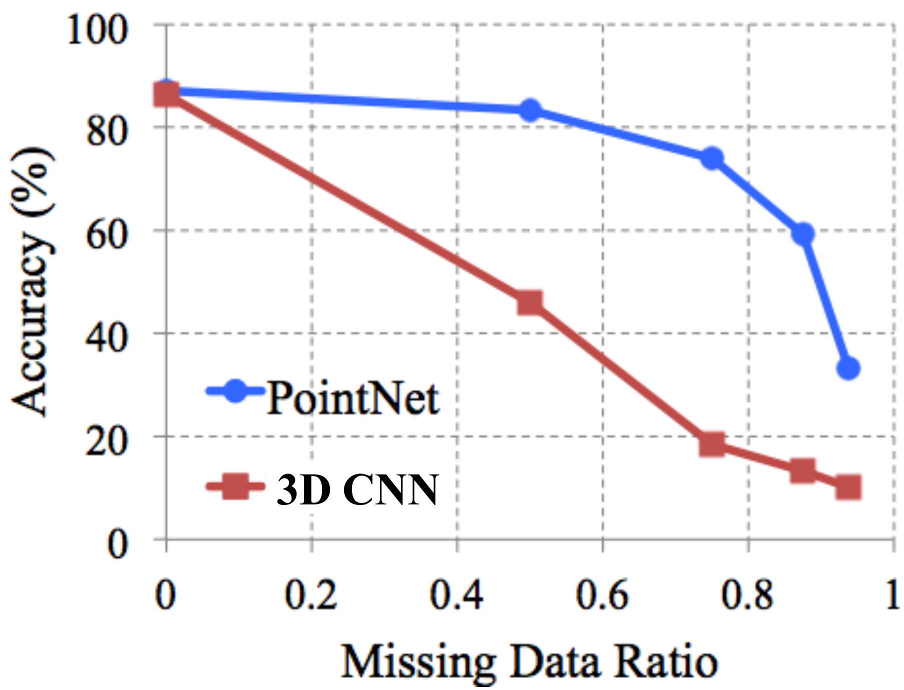
dataset: ModelNet40; metric: 40-class classification accuracy (%)

Robustness to Data Corruption



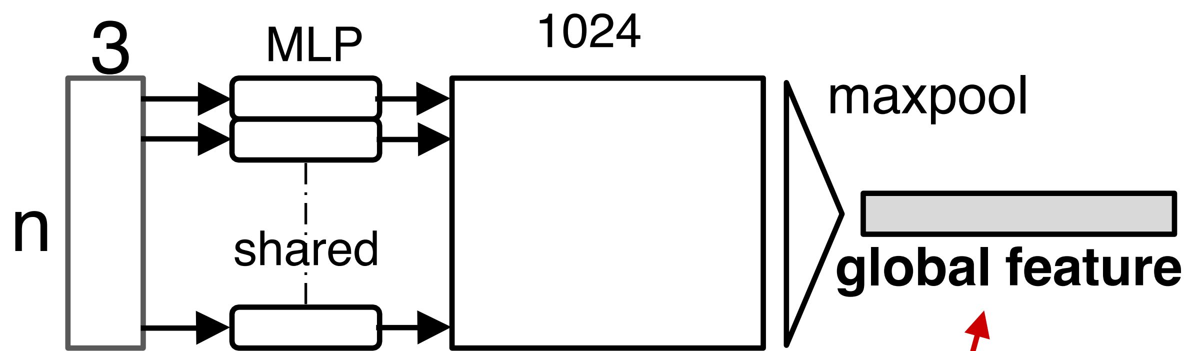
dataset: ModelNet40; metric: 40-class classification accuracy (%)

Robustness to Data Corruption



Why is PointNet so robust to missing data?

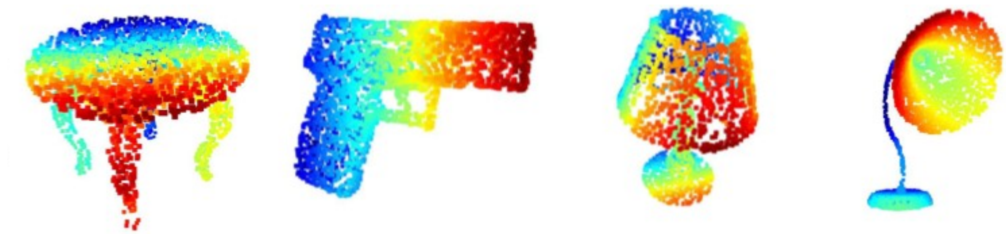
Visualizing Global Point Cloud Features



Which input points are contributing to the global feature?
(critical points)

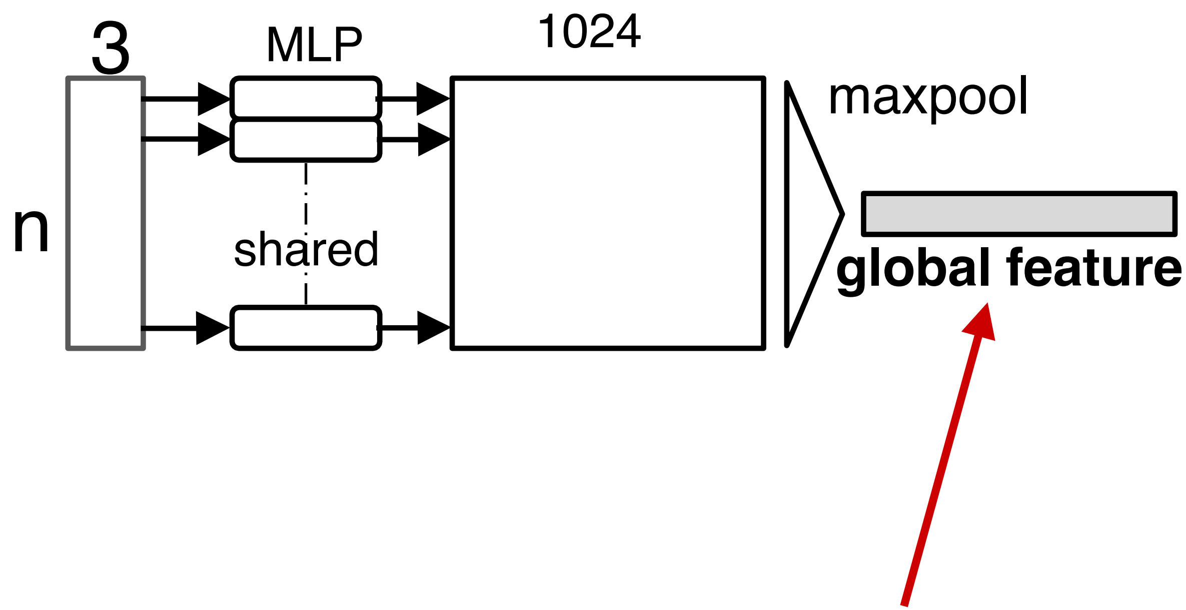
Visualizing Global Point Cloud Features

Original Shape:



Critical Point Set:

Visualizing Global Point Cloud Features



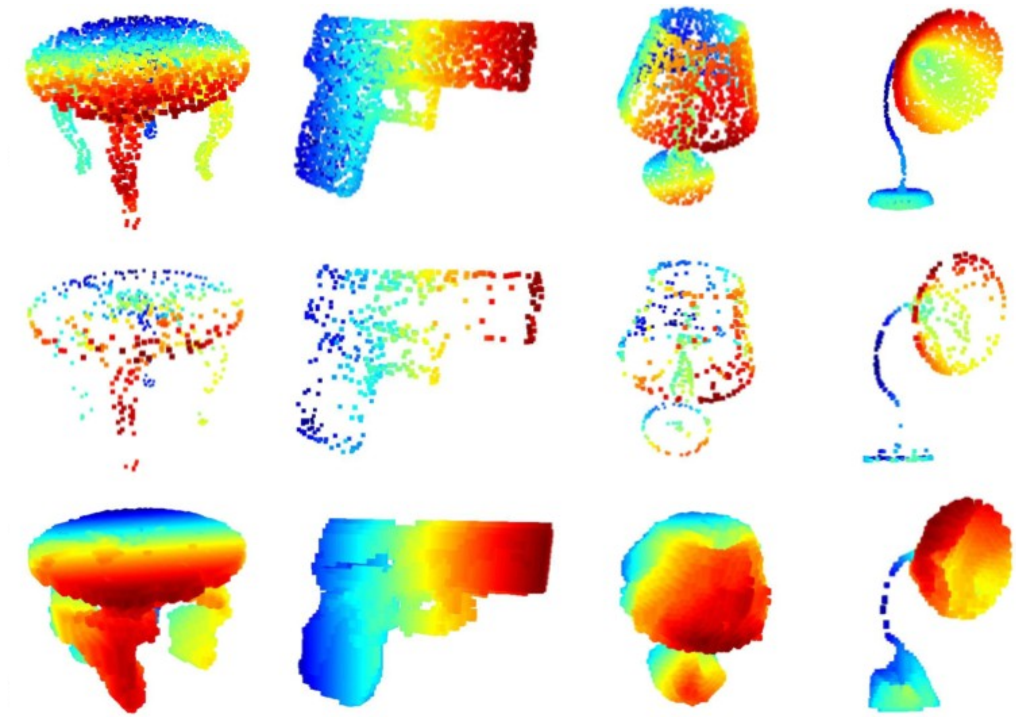
Which points won't affect the global feature?

Visualizing Global Point Cloud Features

Original Shape:

Critical Point Set:

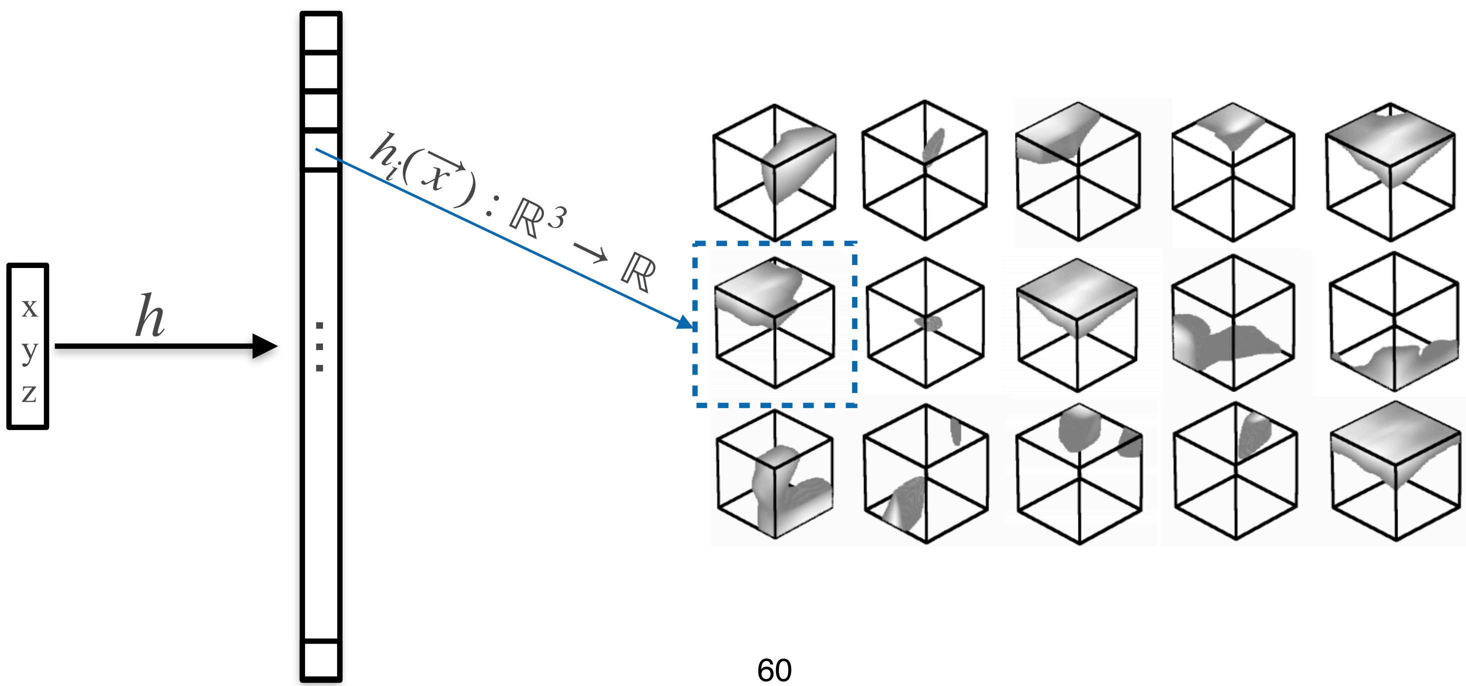
Upper bound set:



Interpretation to “First Layer”

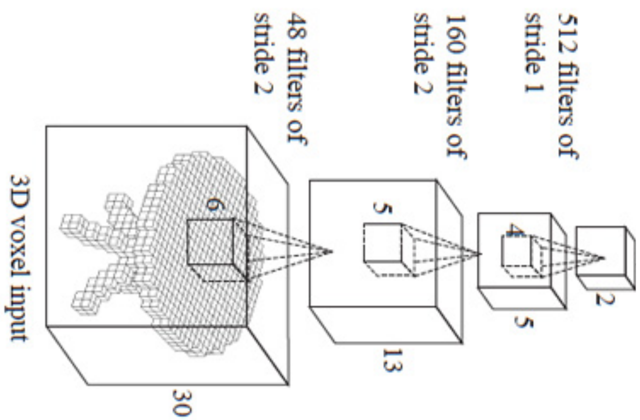
- Think of each dimension as a “binary” variable (the truth is a soft version)
- It encodes whether the point is in a certain spatial region
- The shape of the spatial region is learned

3D voxels of irregular boundaries!



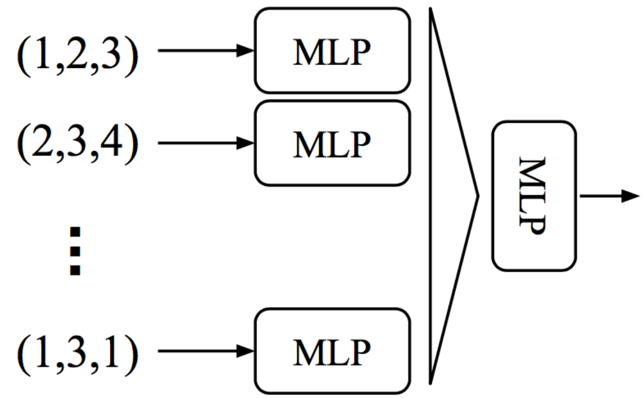
Limitations of PointNet

Hierarchical feature learning
Multiple levels of abstraction



3D CNN (Wu et al.)

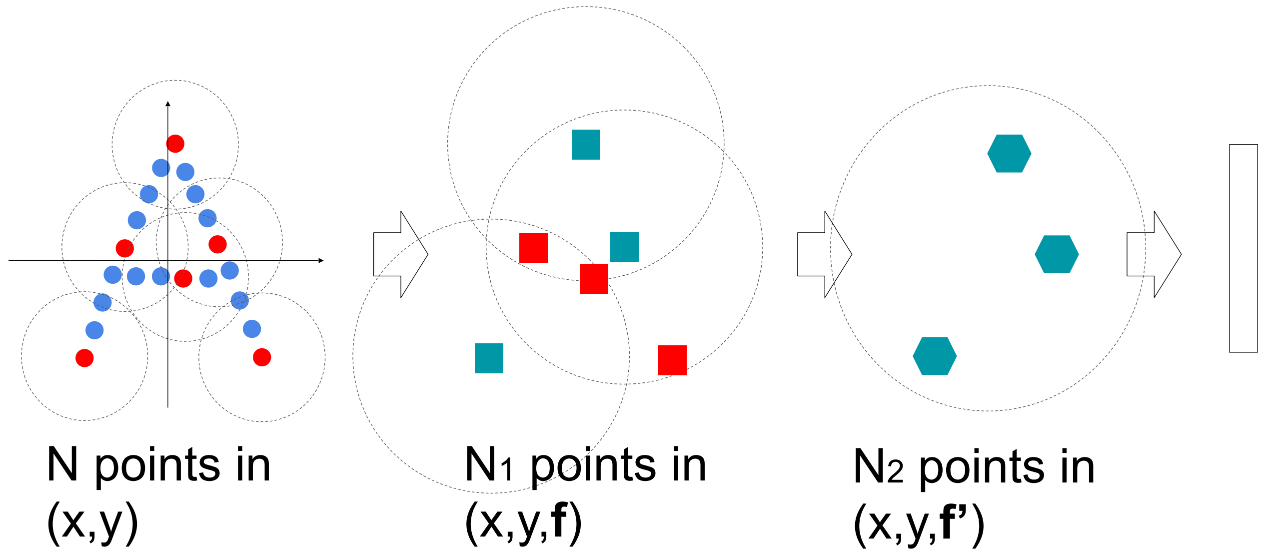
Global feature learning
Either one point or all points



PointNet (vanilla) (Qi et al.)

- No local context for each point!
- Global feature depends on absolute coordinate. Hard to generalize to unseen scene configurations!

PointNet v2.0: Multi-Scale PointNet

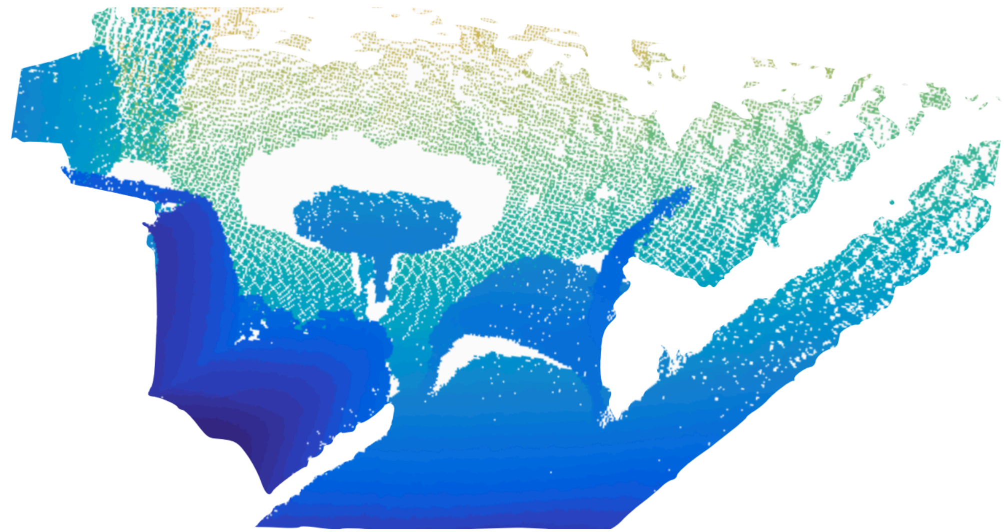


Repeat

- Sample anchor points by FPS
- Find neighborhood of anchor points
- Apply PointNet in each neighborhood to mimic convolution

Overcoming Non-Uniform Surface Sampling Issue

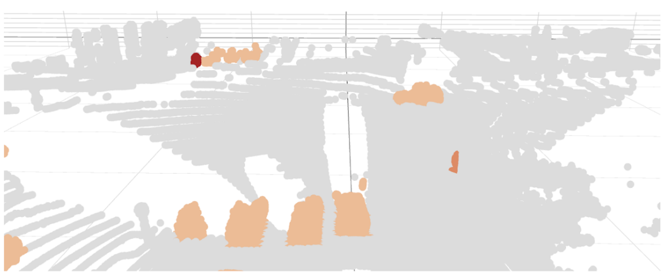
Real Point Clouds are Non-Uniform

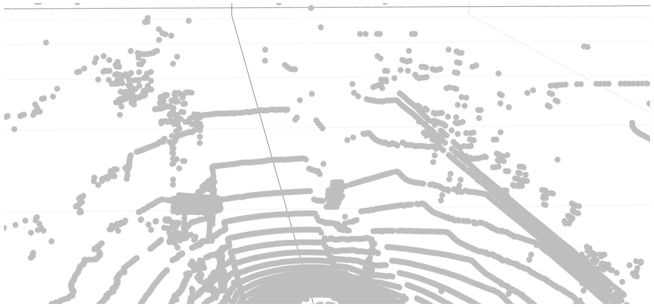


Sampling Caused Domain Gap

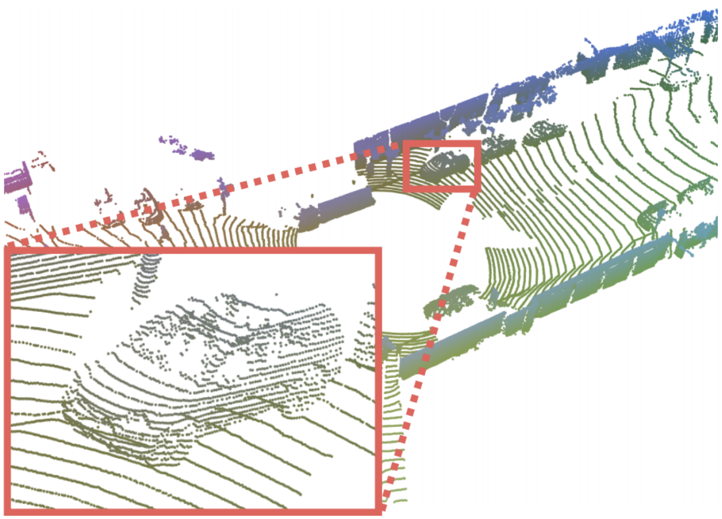
Labeled source domain

Unlabeled target domain

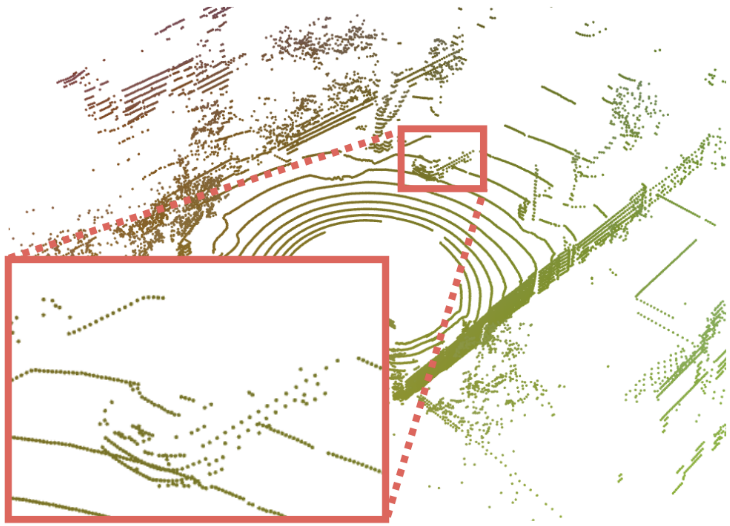




Sampling Caused Domain Gap



(a) captured by a 64-beam LiDAR



(b) captured by a 32-beam LiDAR

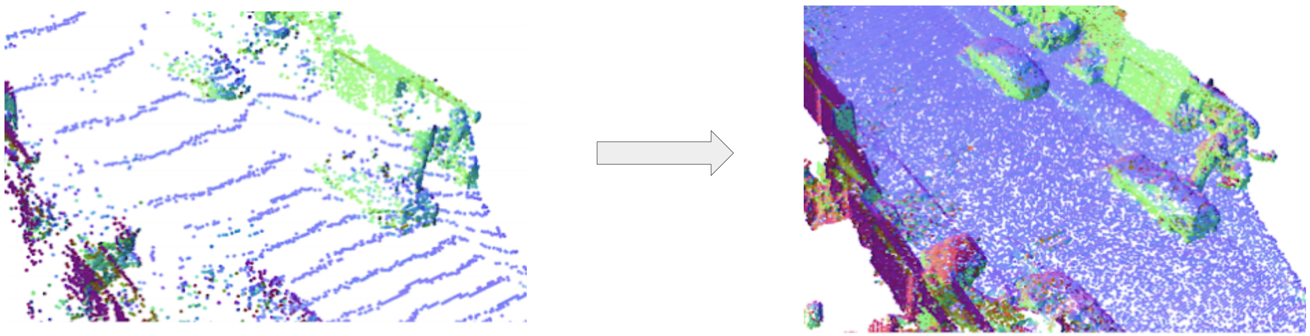
Some Directions to Address the Issue

1. Randomly throw away some points in the training data by a dropout layer (as in PointNet++)

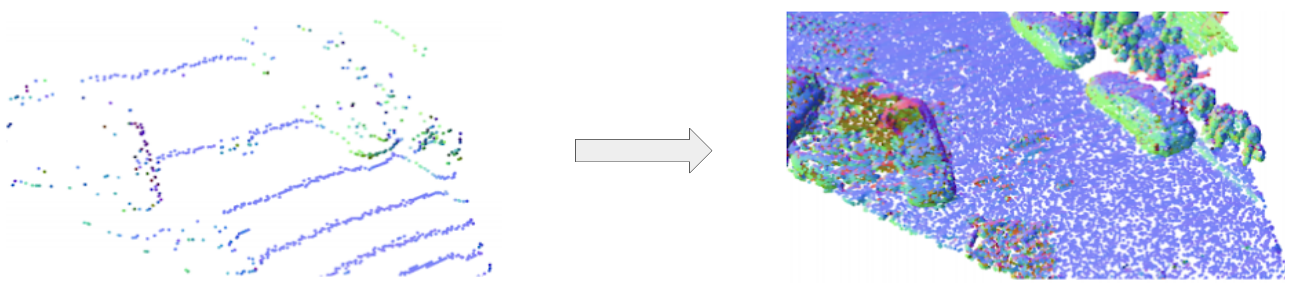
2. Learn to canonicalize the point cloud

3. Use an interpolatable kernel for convolution (as in KPConv)

Learn to Canonicalize the Point Cloud



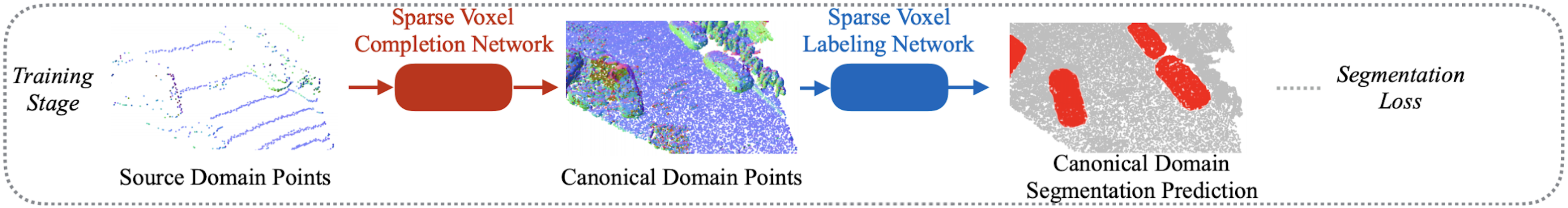
(a) captured by a 64-beam LiDAR



(b) captured by a 32-beam LiDAR

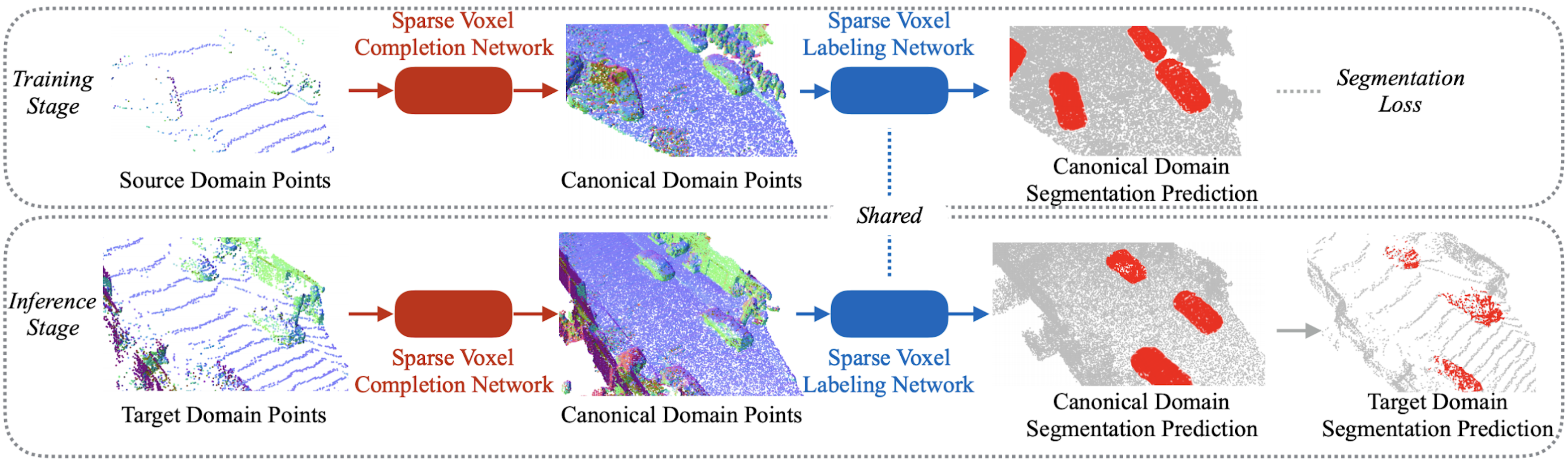
Yi et al, "***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***", arxiv, 2020

Method Overview



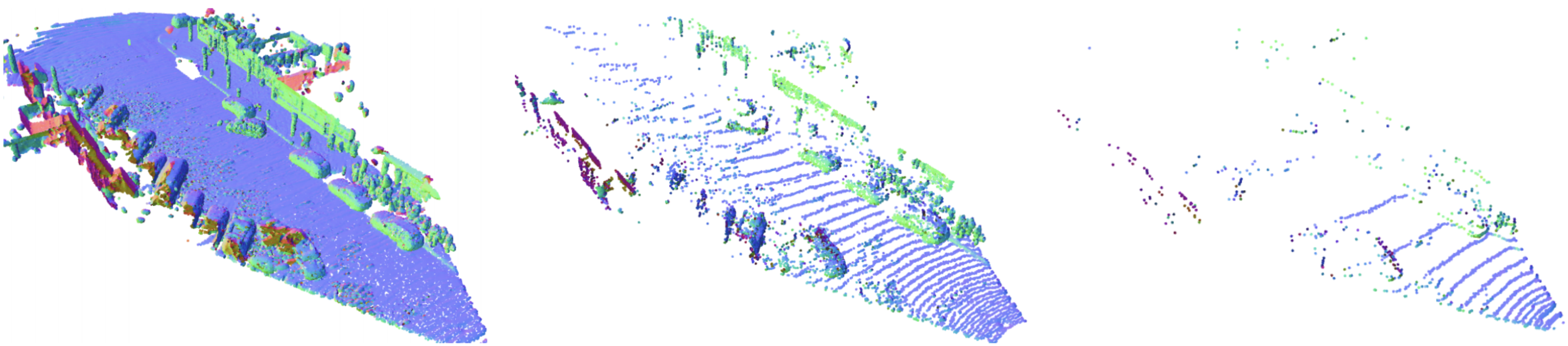
Yi et al, "***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***", arxiv, 2020

Method Overview



Yi et al, “***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***”, arxiv, 2020

Training Data Preparation

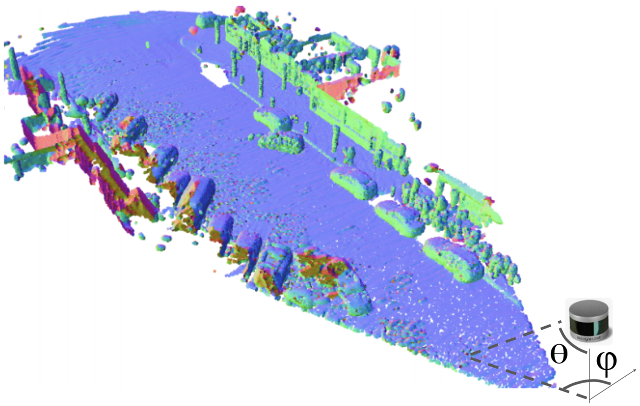


(a) complete scene point cloud

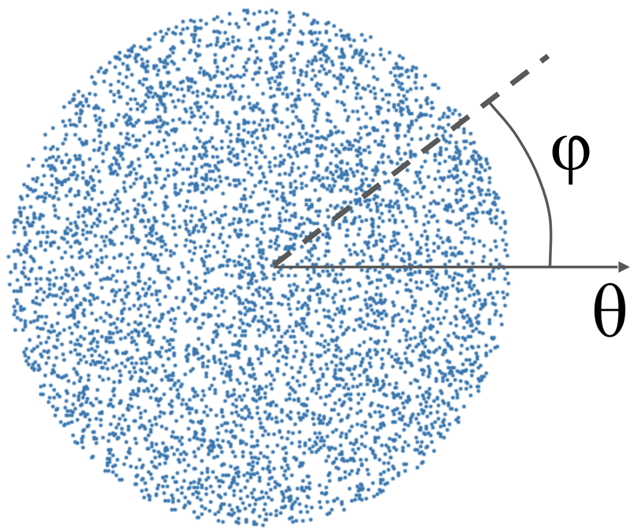
(b) simulated incomplete point cloud with sampling pattern transferred from Waymo

(c) simulated incomplete point cloud with sampling pattern transferred from nuScenes

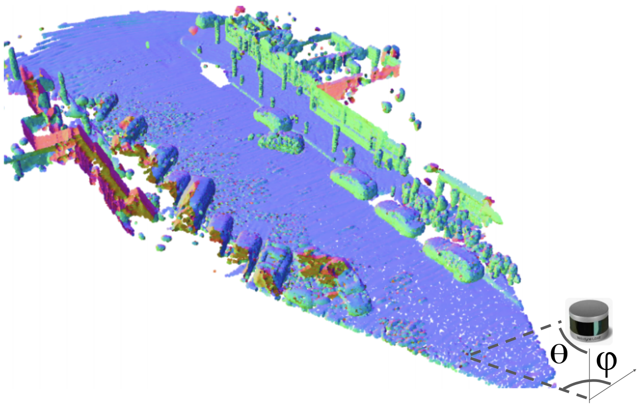
Yi et al, “***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***”, arxiv, 2020



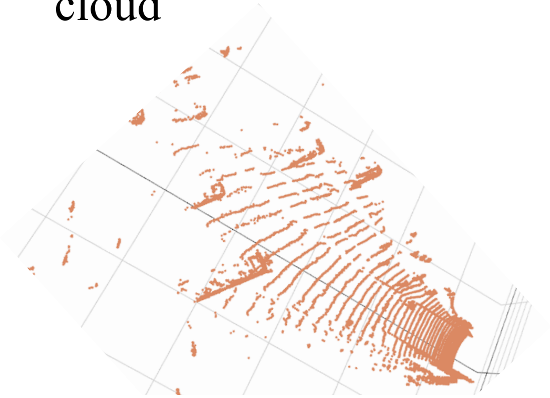
complete scene point cloud



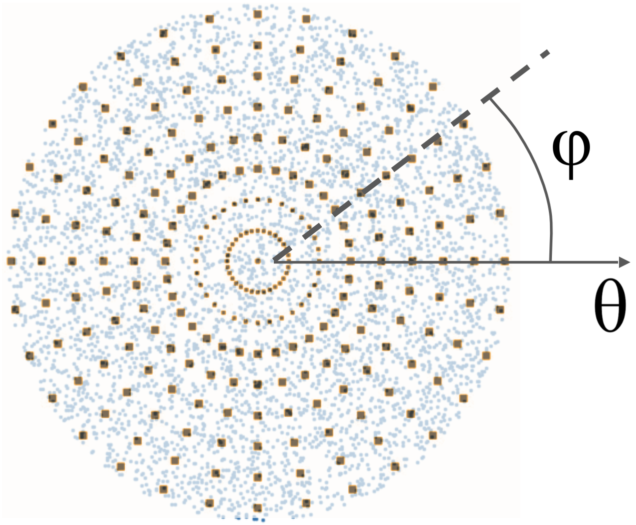
Yi et al, "***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***", arxiv, 2020



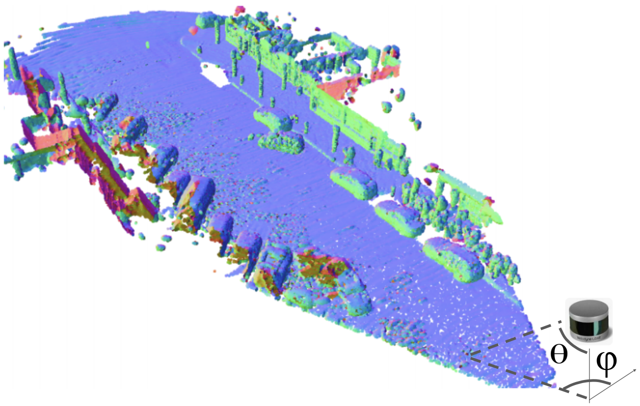
complete scene point cloud



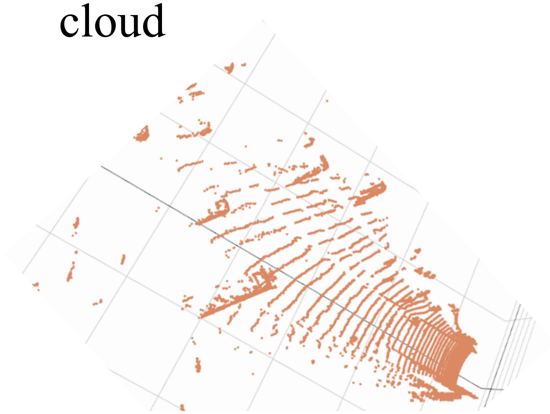
Reference frame



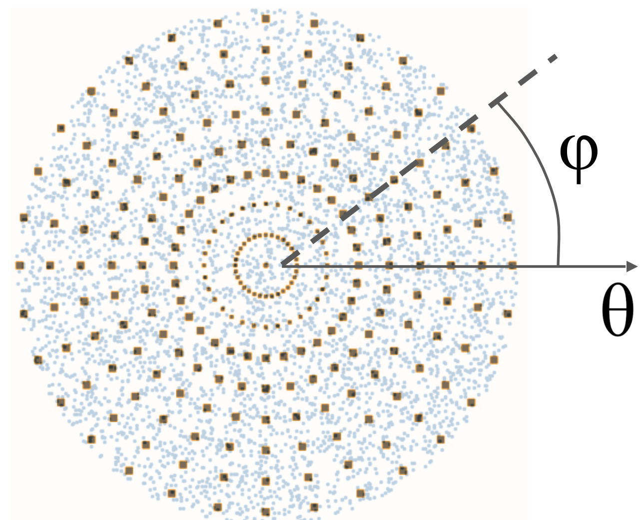
Yi et al, "***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***", arxiv, 2020

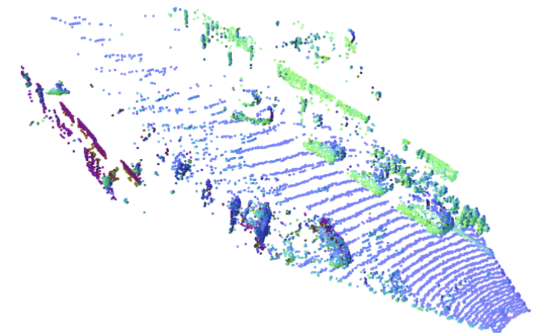


complete scene point cloud



Reference frame

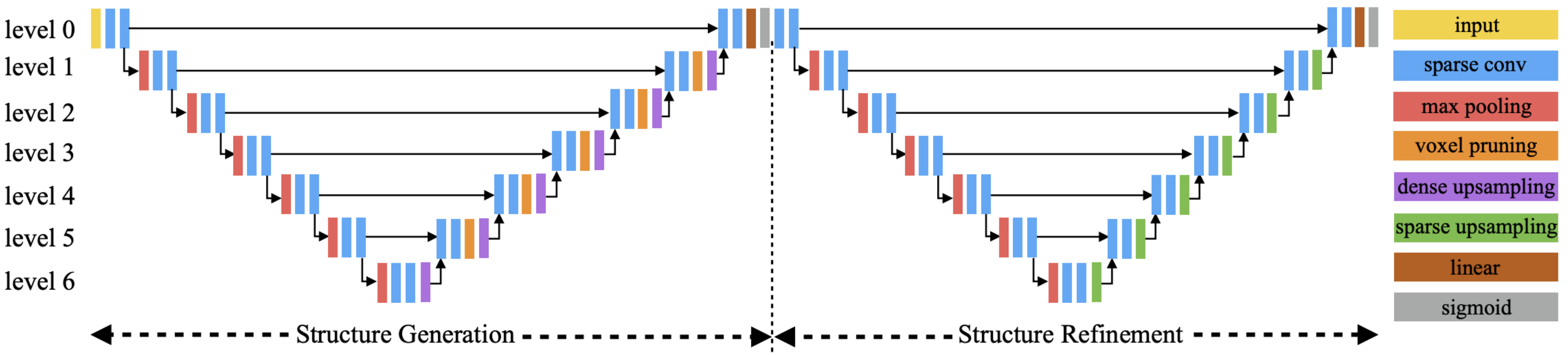




simulated samples

Yi et al, "***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***", arxiv, 2020

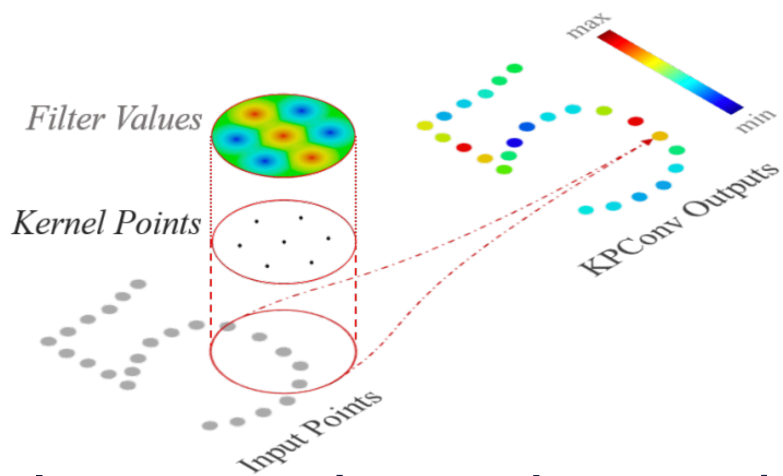
Sparse Voxel Completion Network (SVCN)



Yi et al, "***Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds***", arxiv, 2020

Interpolated Kernel for Convolution

- Continuous conv: $(\mathcal{F} * g)(x) = \int g(y - x) f(y) dy$
- Empirical conv: $(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$



- Learn kernel value at anchor points and interpolate to build continuous kernel

$$\kappa_{jm}(z) = \sum_l k_{ljm} \Phi(|z - y_l|)$$

Φ: RBF kernel

Atzmon et al., "**Point Convolutional Neural Networks by Extension Operators**", *Trans. on Graphics, 2018*

Thomas et al., "**KPConv: Flexible and Deformable Convolution for Point Clouds**", *ICCV 2019*

Check by yourself