

L7: Single-Image to 3D

Hao Su

Slides prepared by Dr. Songfang Han

Agenda

- Task
- Synthesis-for-Learning Pipeline
- Single-image to Depth Map
- Single-image to Point Cloud
- Single-image to Mesh

Task

Review: Multi-View Stereo

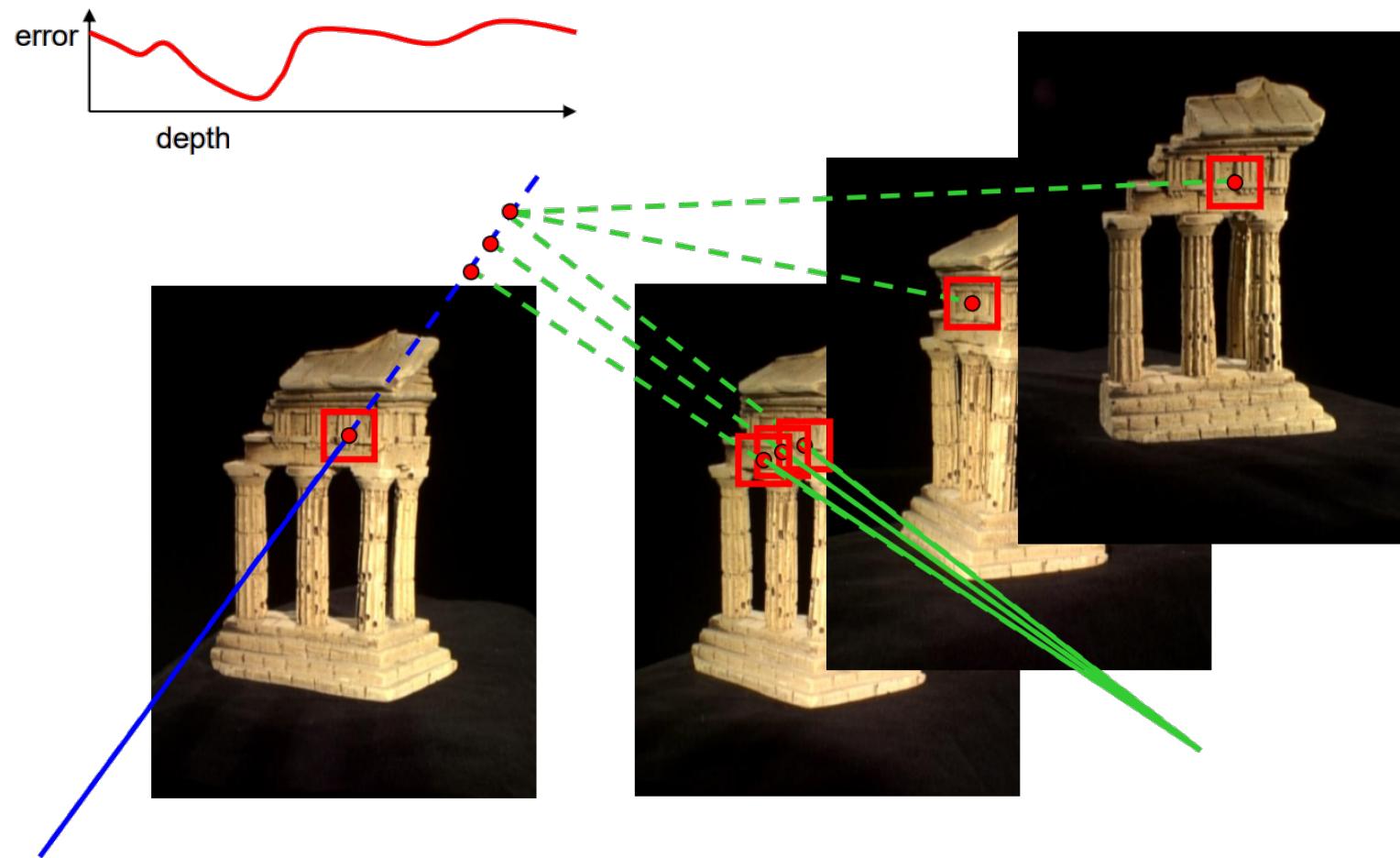
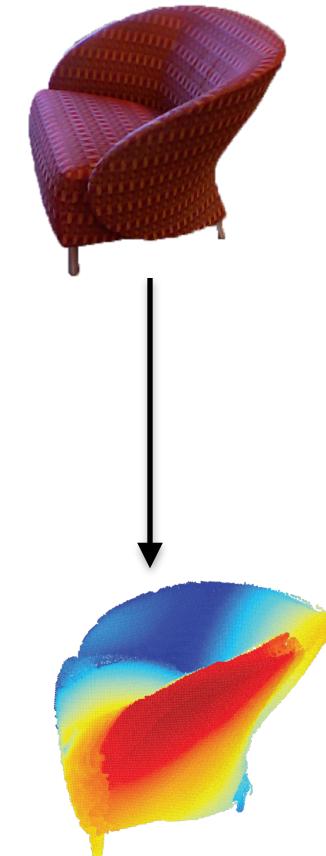
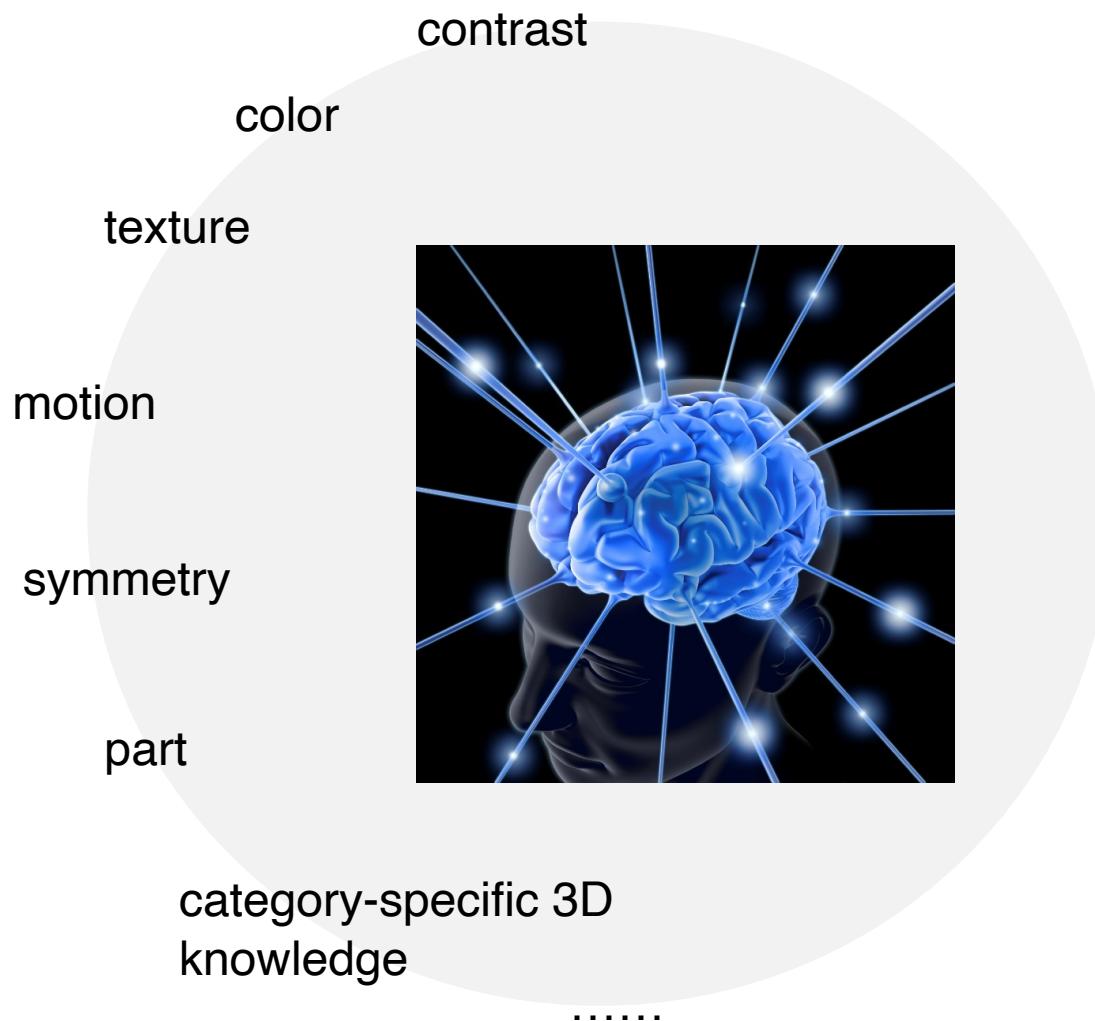


Image source: UW CSE455

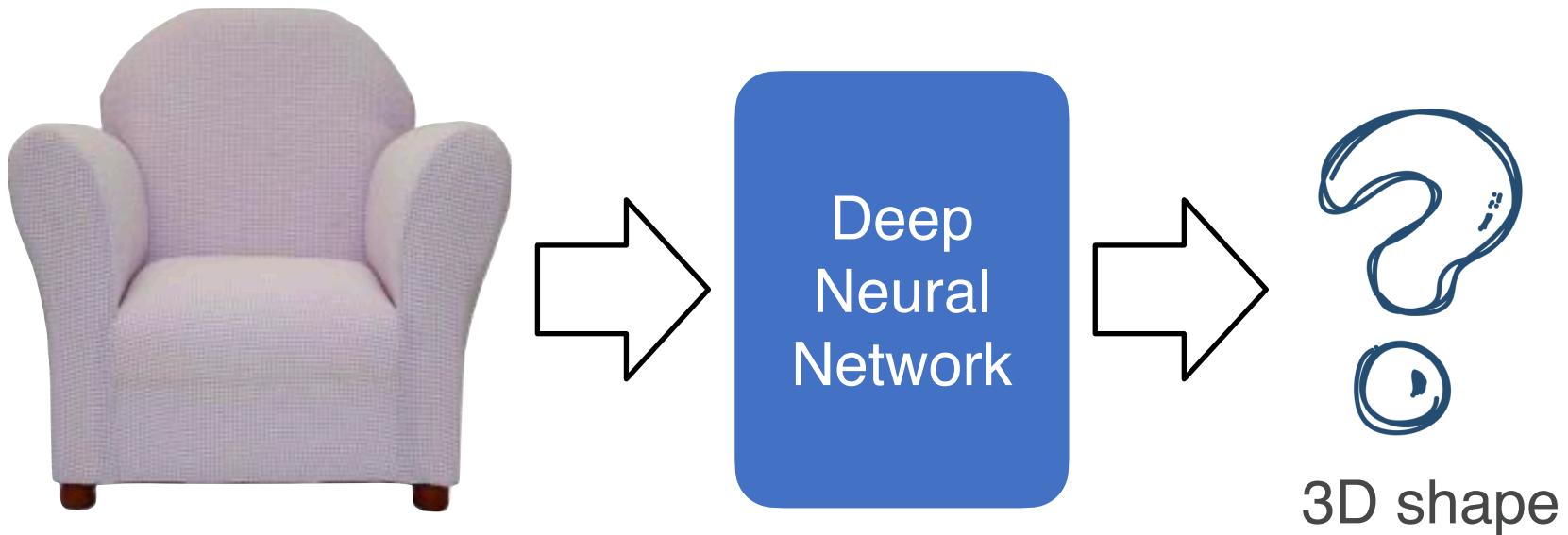
Can We Infer 3D from just
a **Single** Image?



Many Cues That Allow 3D Estimation



Learning-based 3D Reconstruction



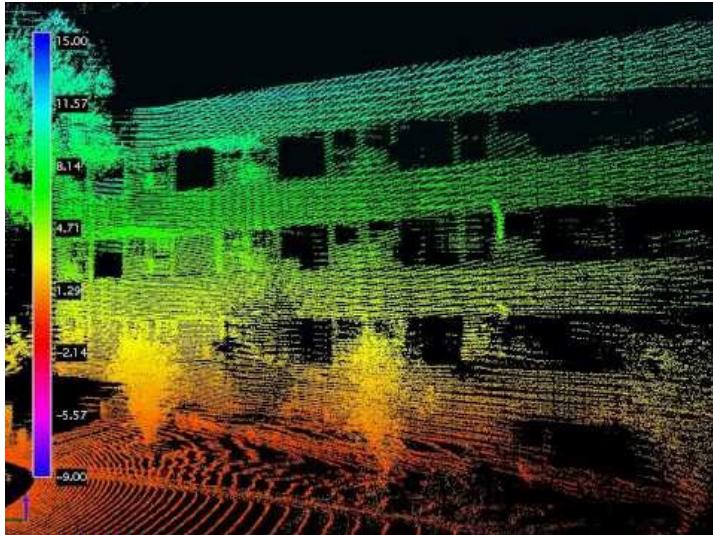
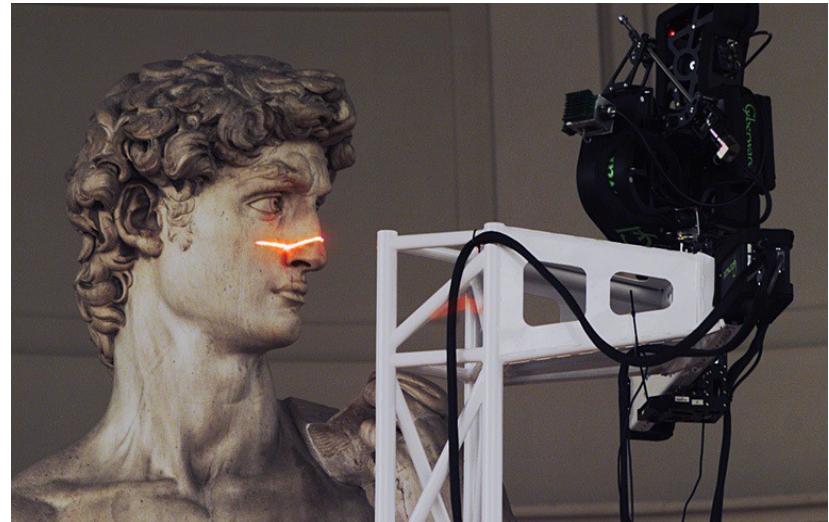
Synthesis-for-Learning Pipeline

Where Are My Training Data?

- In general, training deep networks needs **a lot of data with labels!**
- In our case, we need many image-3D shape pairs...
- Before talking about learning algorithms, obtaining training data is already a challenge!

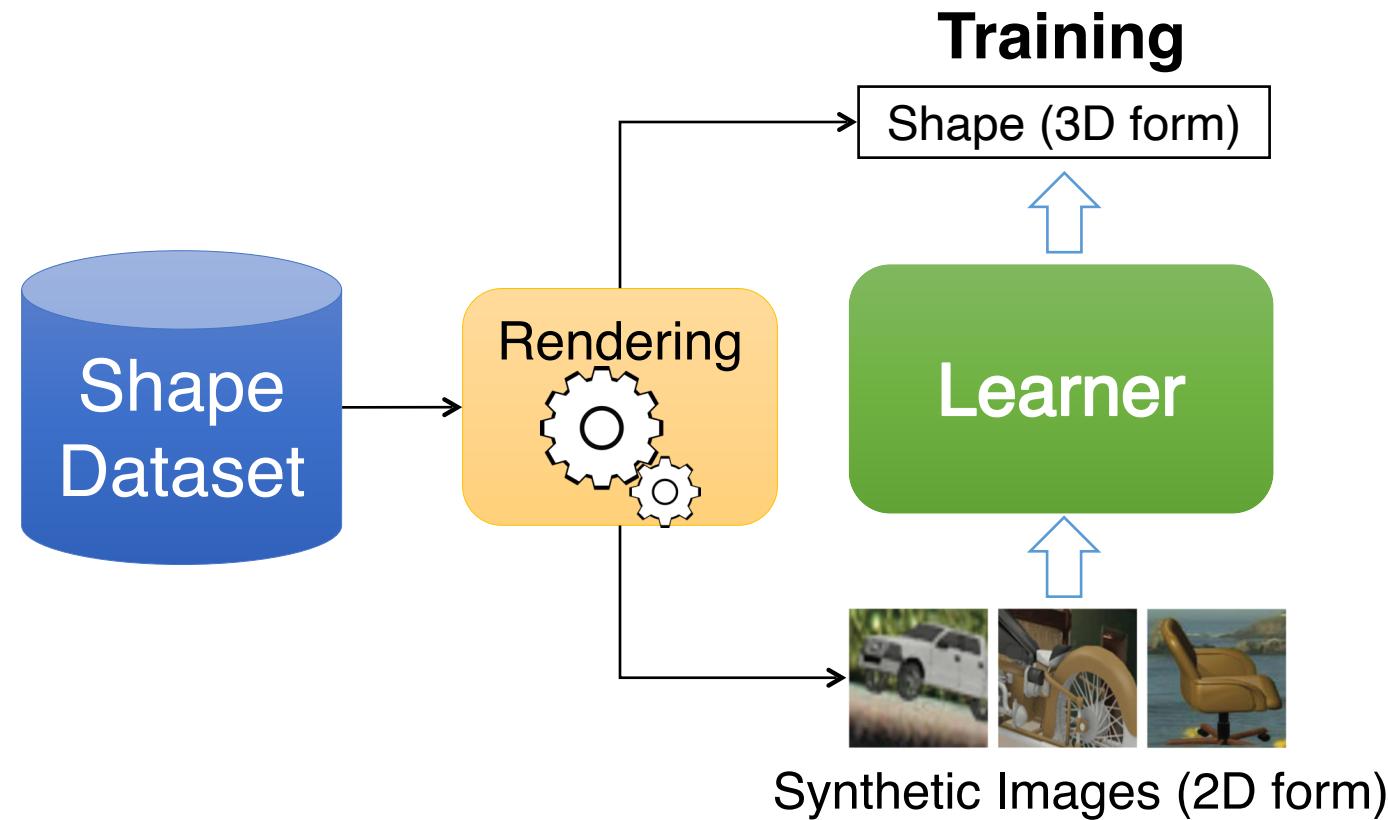
Source I: Real Data

- Many techniques
 - Indoor: ToF or stereo sensors (Kinect, RealSense, ...)
 - Outdoor: LiDAR

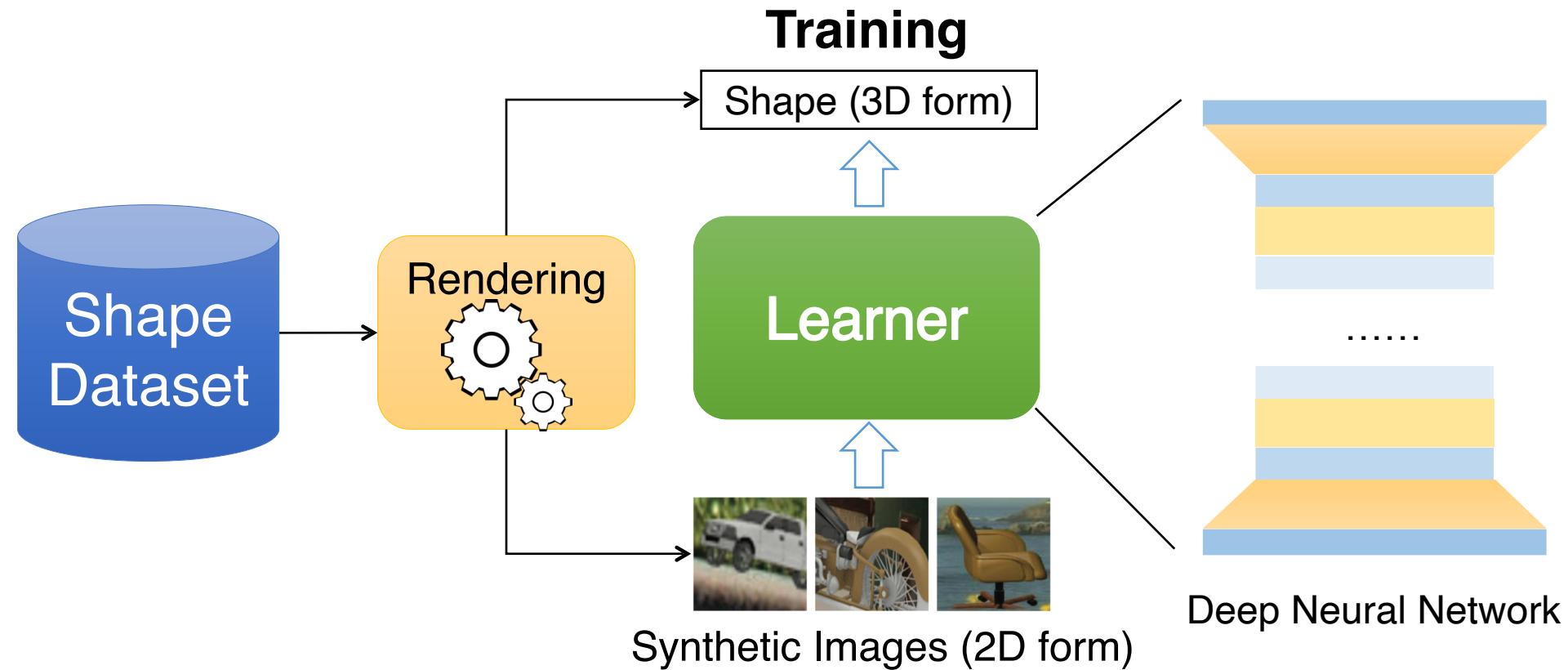


- The amount of real data is increasing quickly

Source II: Synthesis for Learning

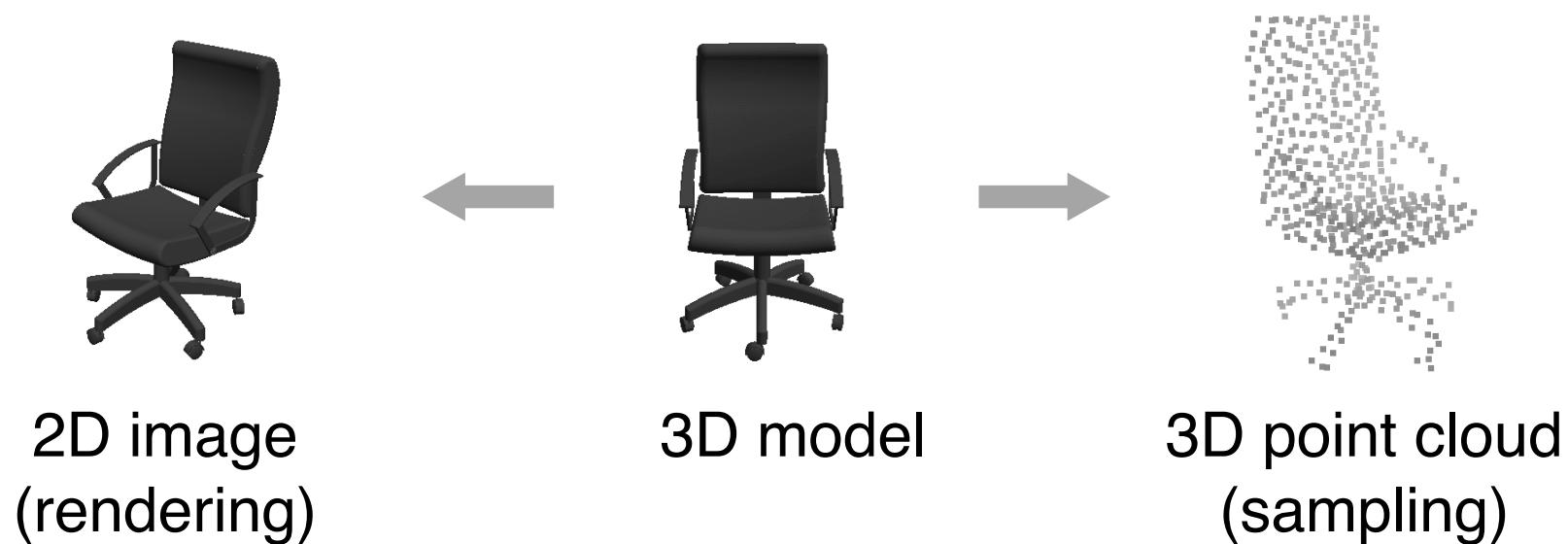


Source II: Synthesis for Learning



Source II: Synthesis for Learning

- For example, image → point cloud



Large-Scale Synthetic 3D Dataset

- For example,
 - ShapeNet: <http://www.shapenet.org>



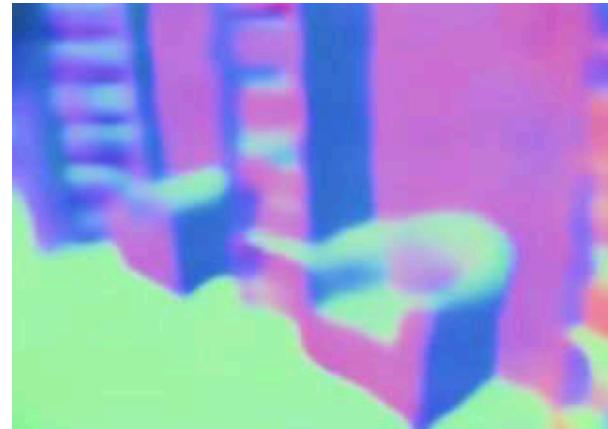
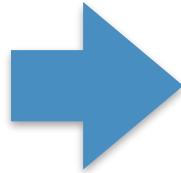
A Very Coarse Literature Review

Literature: to Depth Map

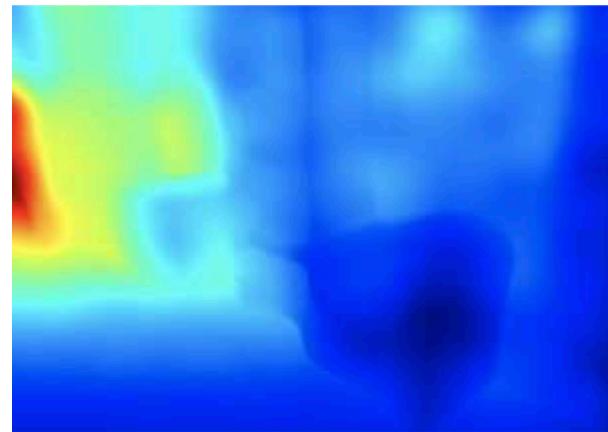
- Fully-convolutional



Input image

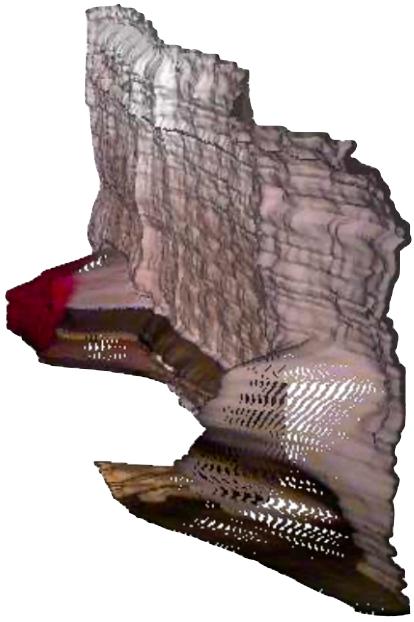


(a) Normal map



(b) Depth map

Recall: Issue of L_p Depth Loss



Prediction



Groundtruth

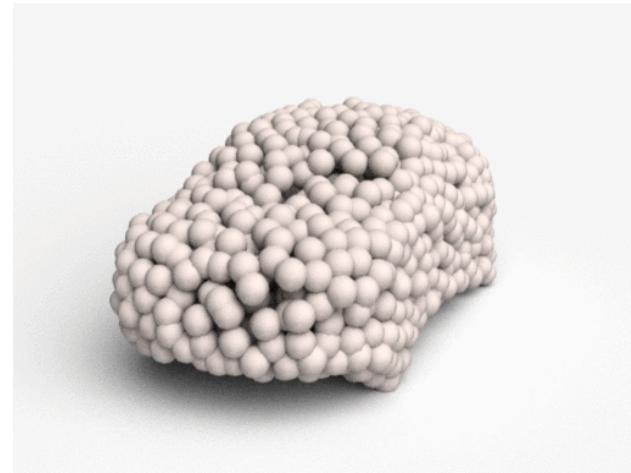
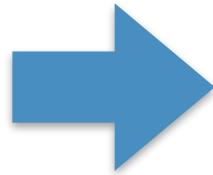
- Common strategy: Depth-Normal consistency
- Review last lecture
- Limitation: partial 3D info from camera view

Literature: to Point Cloud

- From a single image to 3D point cloud generation.



Input image



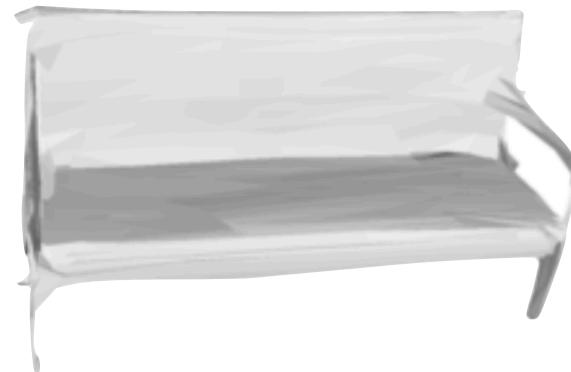
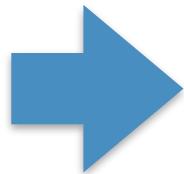
Reconstructed 3D point cloud

Literature: to Mesh

- From a single image to mesh surface.



Input image



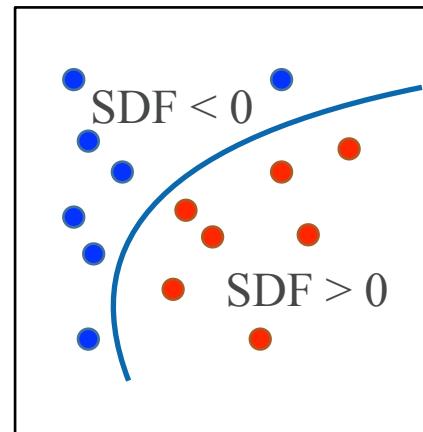
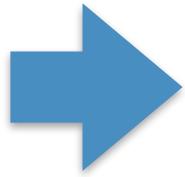
Reconstructed 3D mesh

Literature: to Implicit Field Function

- From a single image to implicit field function.



Input image



Implicit field function



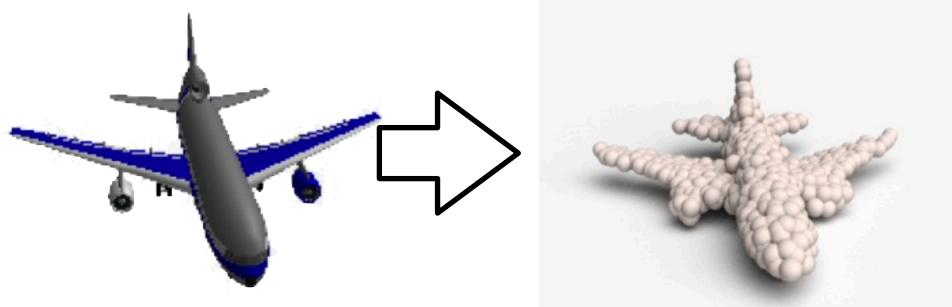
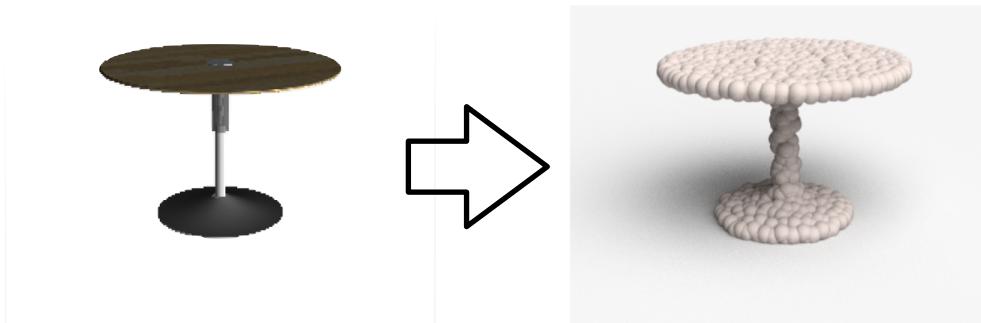
$$F(x) = 0$$

Mescheder et al., “Occupancy networks: Learning 3d reconstruction in function space”, CVPR 2019

Image to Point Cloud

Why Point Representation?

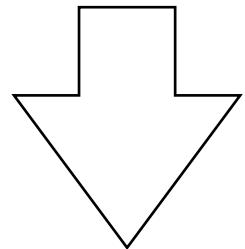
- Previous depth map covers only visible area.
- A flexible representation
 - A few thousands of points can model a great variety of shapes.



Point Cloud as a Set



3D mesh

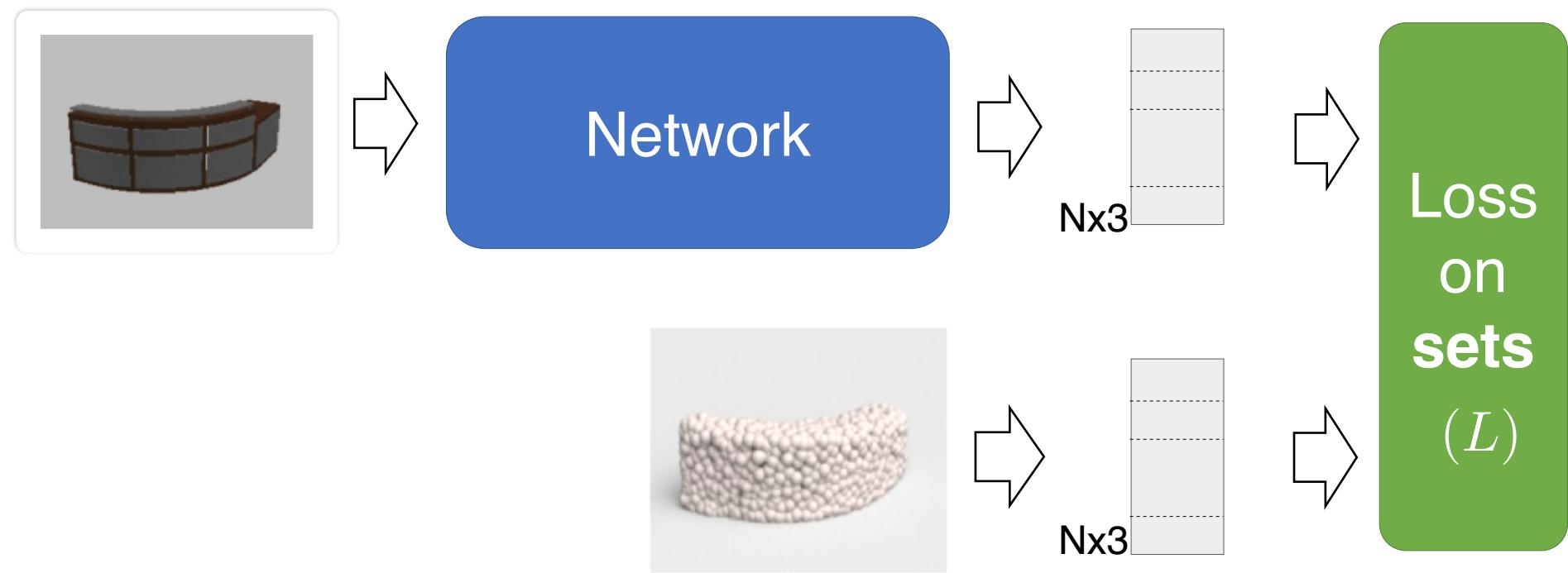


sampling



as $\left\{ \begin{array}{l} (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ \dots \\ (x_n, y_n, z_n) \end{array} \right\}$

Pipeline



Real-world Results

Some results

input

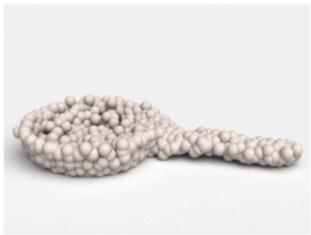
observed view

90°

input

observed view

90°



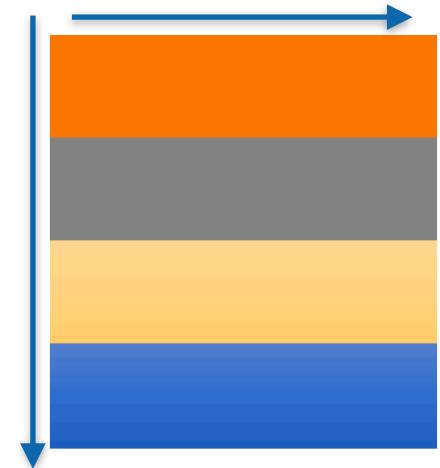
Differentiable Loss for Point Clouds

Permutation Invariance

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



represents the same **set** as

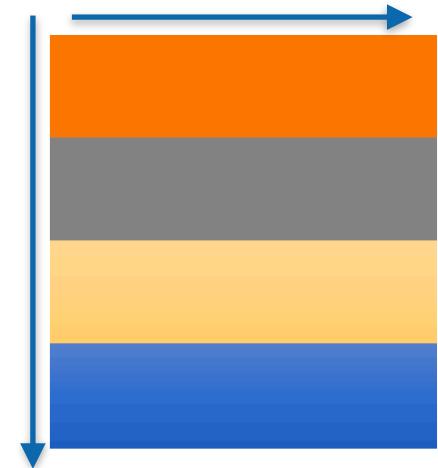


Permutation Invariance

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



represents the same **set** as



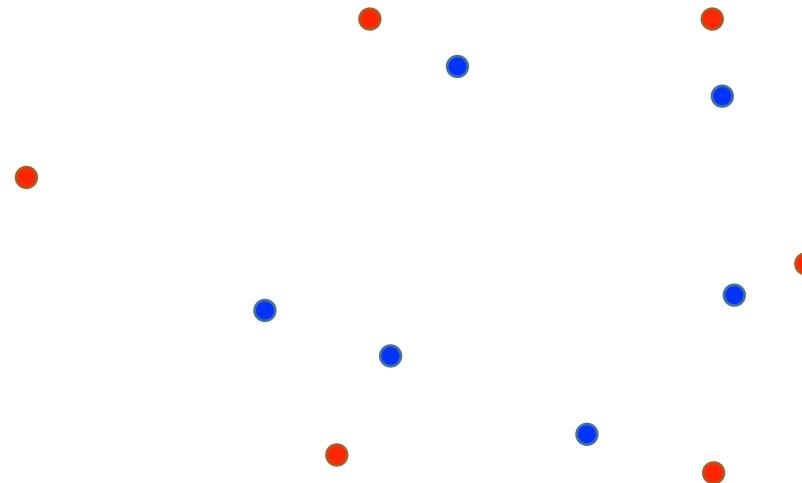
Loss needs to be invariant to ordering of points!

Metric for Point Clouds

- L2 loss does not work for point cloud.
- Need a metric to measure distance between two point sets
- Two popular choices
 - Earth Mover's Distance
 - Chamfer Distance

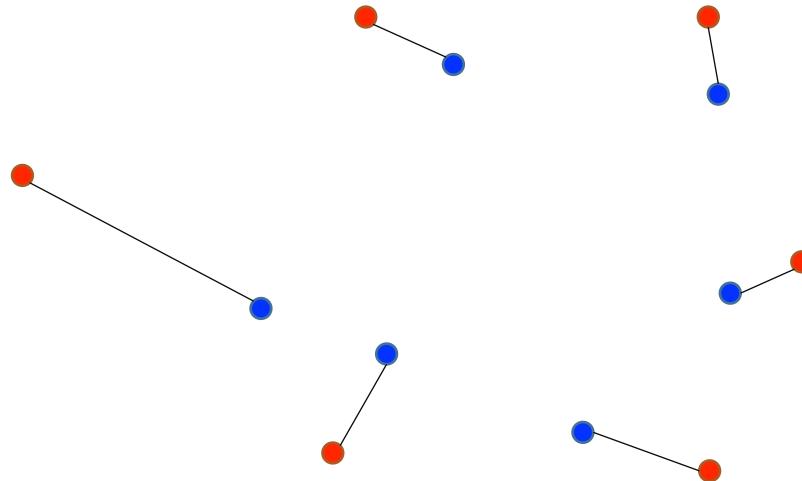
Earth Mover's Distance

- Find a 1-1 correspondence between point sets



Earth Mover's Distance

- Find a 1-1 correspondence between point sets



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

where $\phi : S_1 \rightarrow S_2$ is a bijection

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

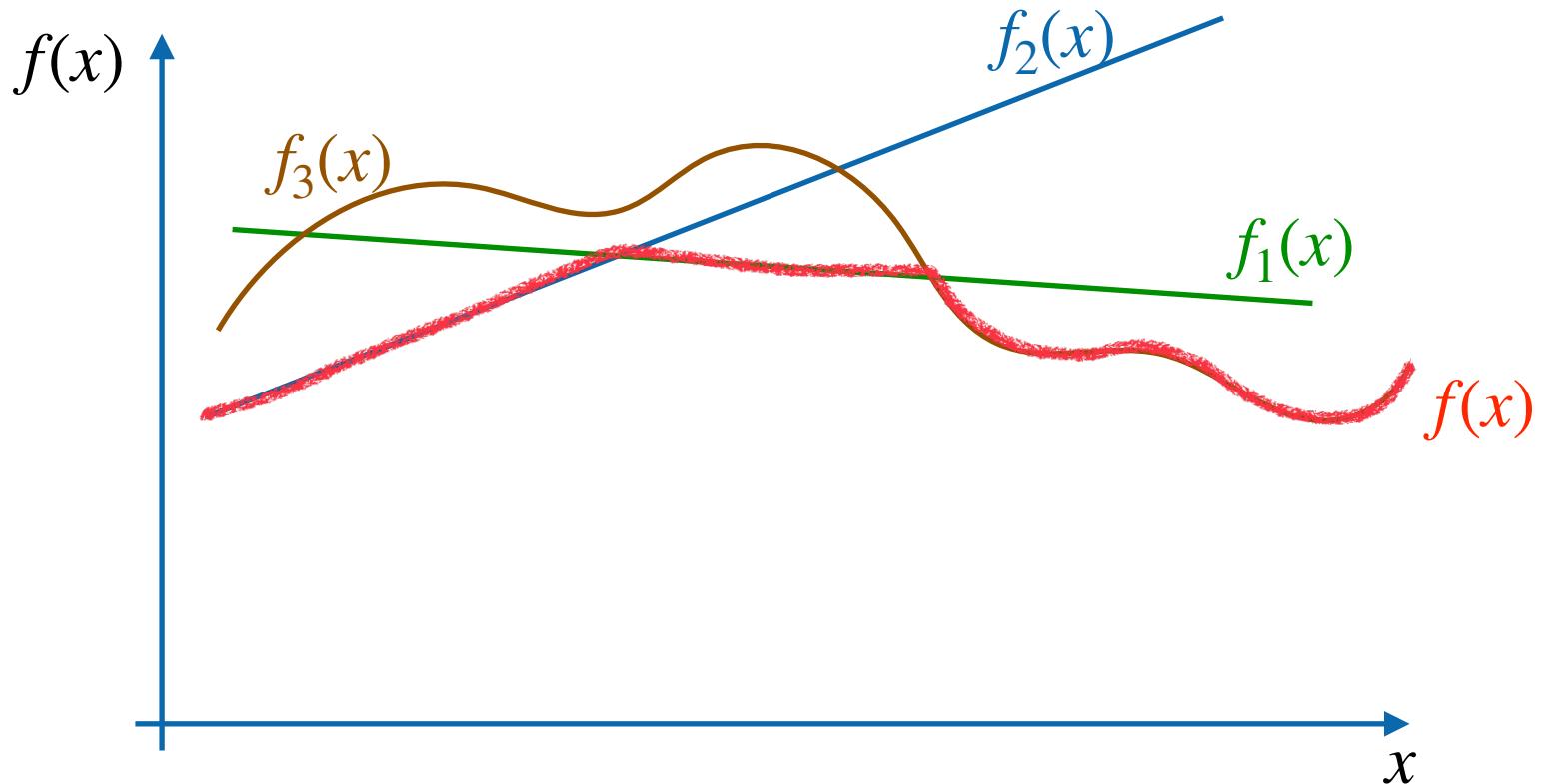
where $\phi : S_1 \rightarrow S_2$ is a bijection

Question:

Viewing $d_{EMD}(S_1, S_2)$ as a function of point coordinates in S_1 , is this function **continuous**?

Lemma

- For a family of continuous functions $\{f_i(x)\}$, the pointwise minimum $f(x) = \min_i \{f_i(x)\}$ is continuous.



Continuity of $d_{EMD}(S_1, S_2)$

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

where $\phi : S_1 \rightarrow S_2$ is a bijection

- $\phi(x)$ defines a point-wise correspondence ($n!$ possibilities, $n = \text{size of } S_1$).
- For a fixed ϕ , define $f_\phi(S_1) = \sum_{x \in S_1} \|x - \phi(x)\|_2$, and $f_\phi(S_1)$ is obviously continuous
- $d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \rightarrow S_2} f_\phi(S_1)$ is thus continuous!

Differentiable?

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

where $\phi : S_1 \rightarrow S_2$ is a bijection

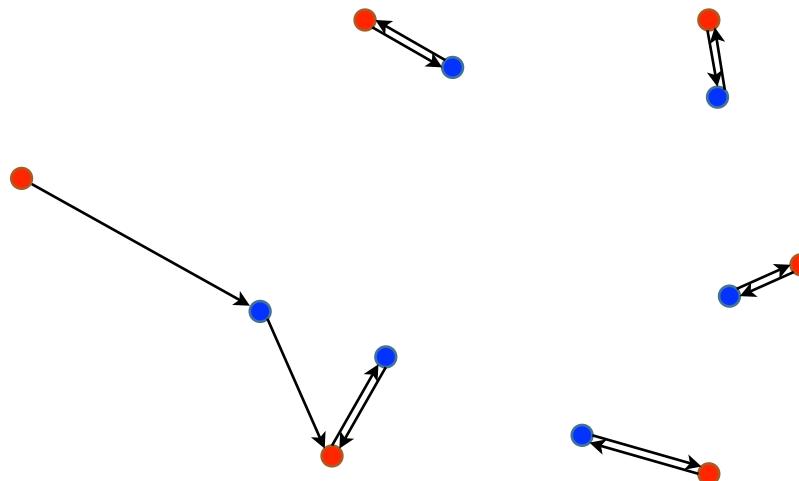
- From the example, we see that $d_{EMD}(S_1, S_2)$ can be constructed in a piece-wise manner
- Inside each piece, it is $f_{\phi_i}(S_1)$ by some ϕ_i , which is obviously differentiable (as $\phi_i(x)$ is a constant)
- $d_{EMD}(S_1, S_2)$ is differentiable except for a finite cases!

Implementation

- Many algorithmic study on fast EMD computation (a specific bipartite matching problem)
- There exists parallelizable implementation of EMD on CUDA
- A fast implementation (approximated EMD): <https://github.com/Colin97/MSN-Point-Cloud-Completion> (by courtesy Minghua Liu)

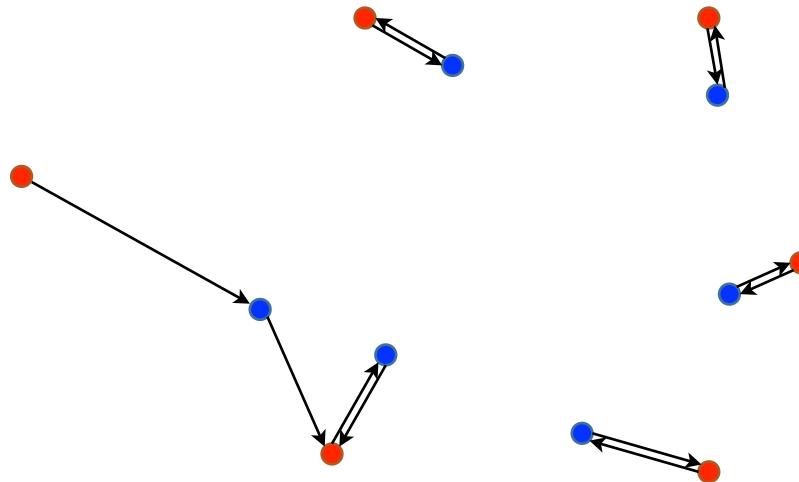
Chamfer Distance

- Nearest neighbor correspondence for each point



Chamfer Distance

- Nearest neighbor correspondence for each point



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

Differentiability?

- Similar argument as EMD computation.

How Distance Metric Affect Learning?

- A fundamental issue: inherent ambiguity in 2D-3D dimension lifting.



How Distance Metric Affect Learning?

- A fundamental issue: inherent ambiguity in 2D-3D dimension lifting.



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

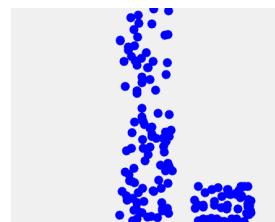
Distance Metrics Affect Mean Shapes

- The mean shape carries characteristics of the distance metric.

Continuous
hidden variable
(radius)



Discrete
hidden variable
(add-on location)



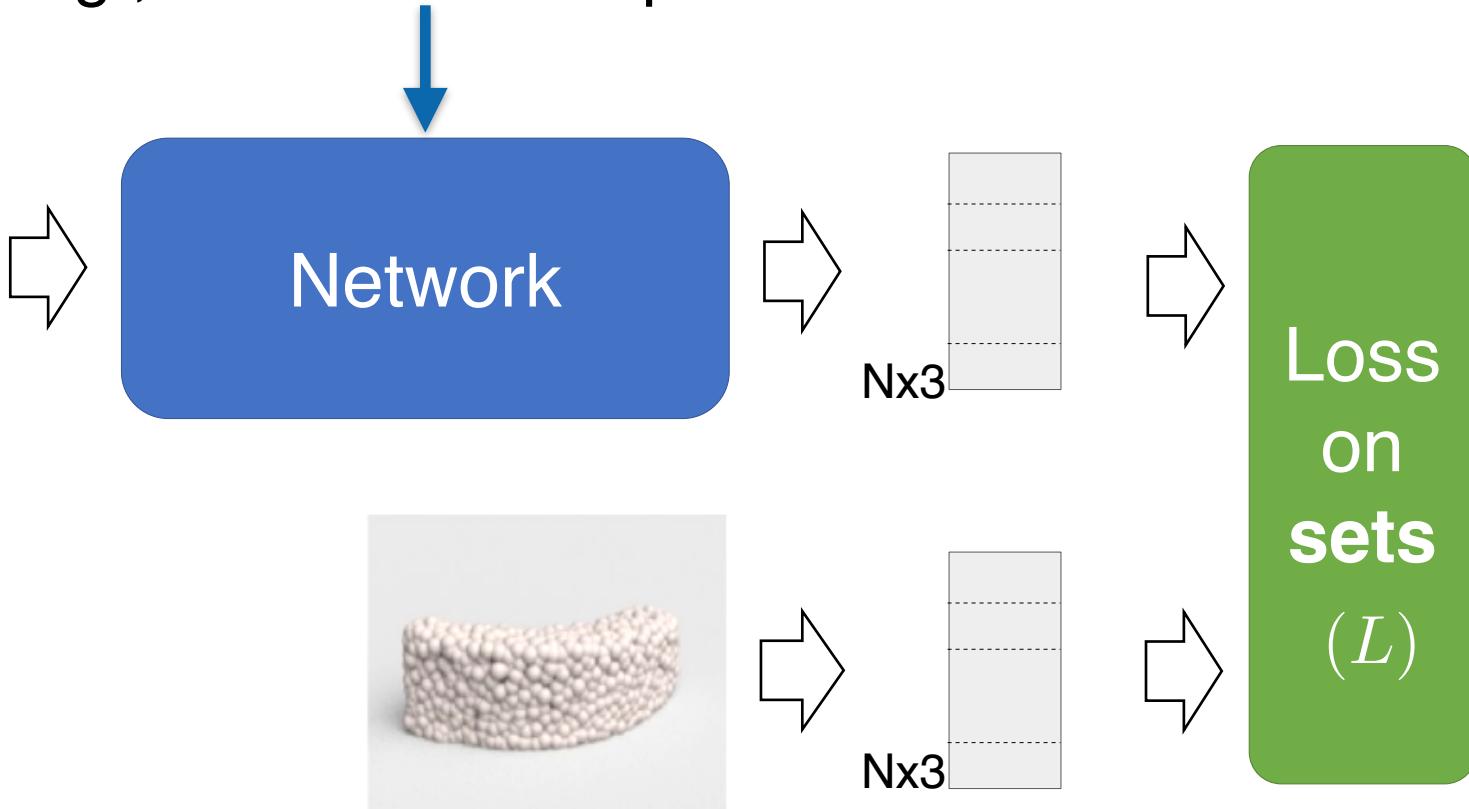
Input

EMD mean

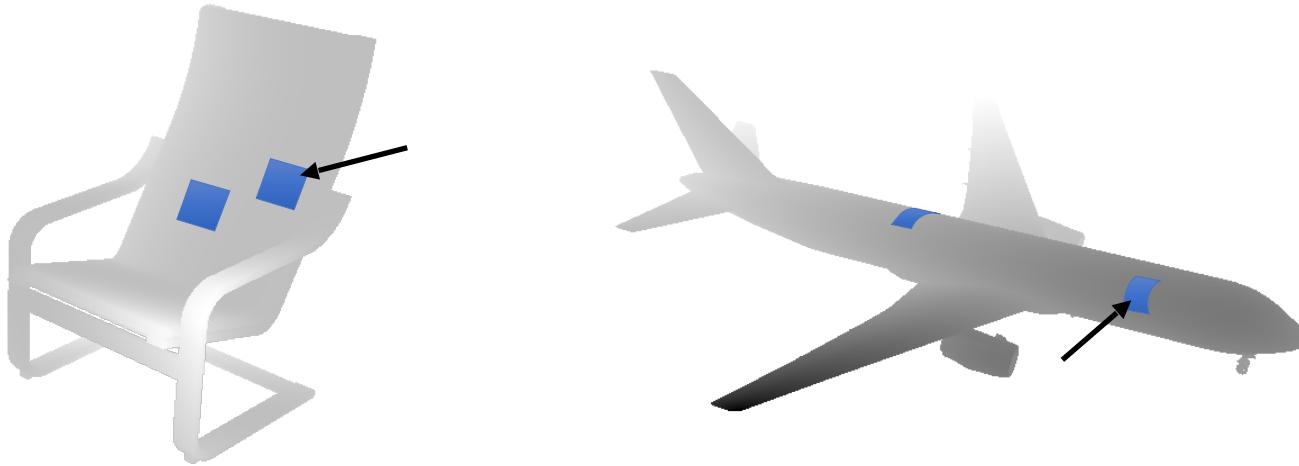
CD mean

Network Choice: Certain Tricks

E.g., ConvNet+FC/UpConv



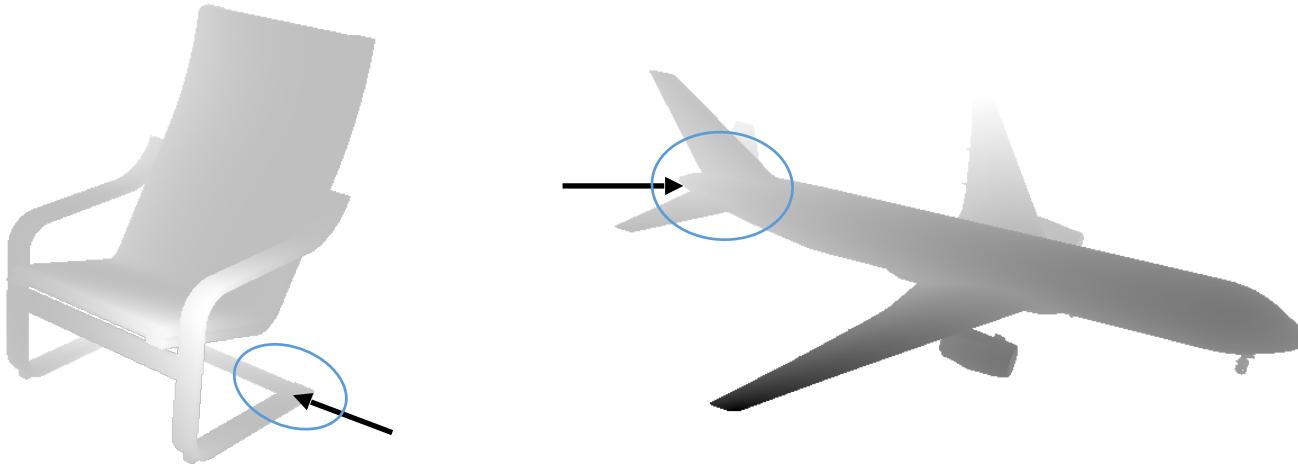
Network Design: Respect Natural Statistics of Geometry



- Many local structures are common

Read by Yourself

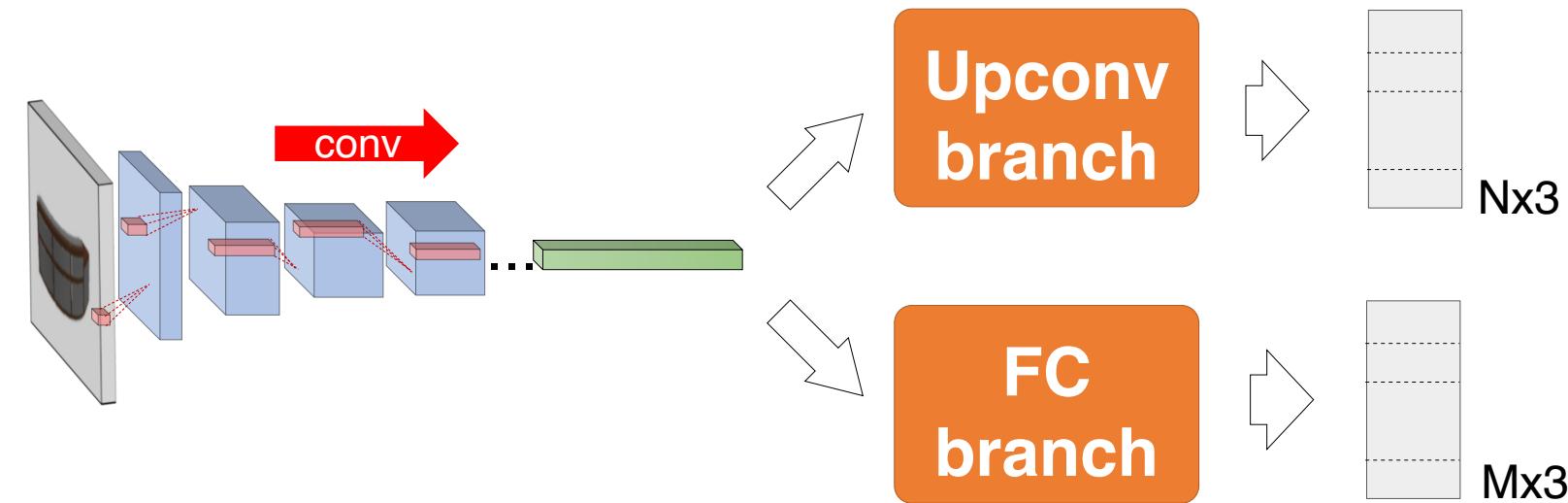
Network Design: Respect Natural Statistics of Geometry



- Many local structures are common
- Also some intricate structures

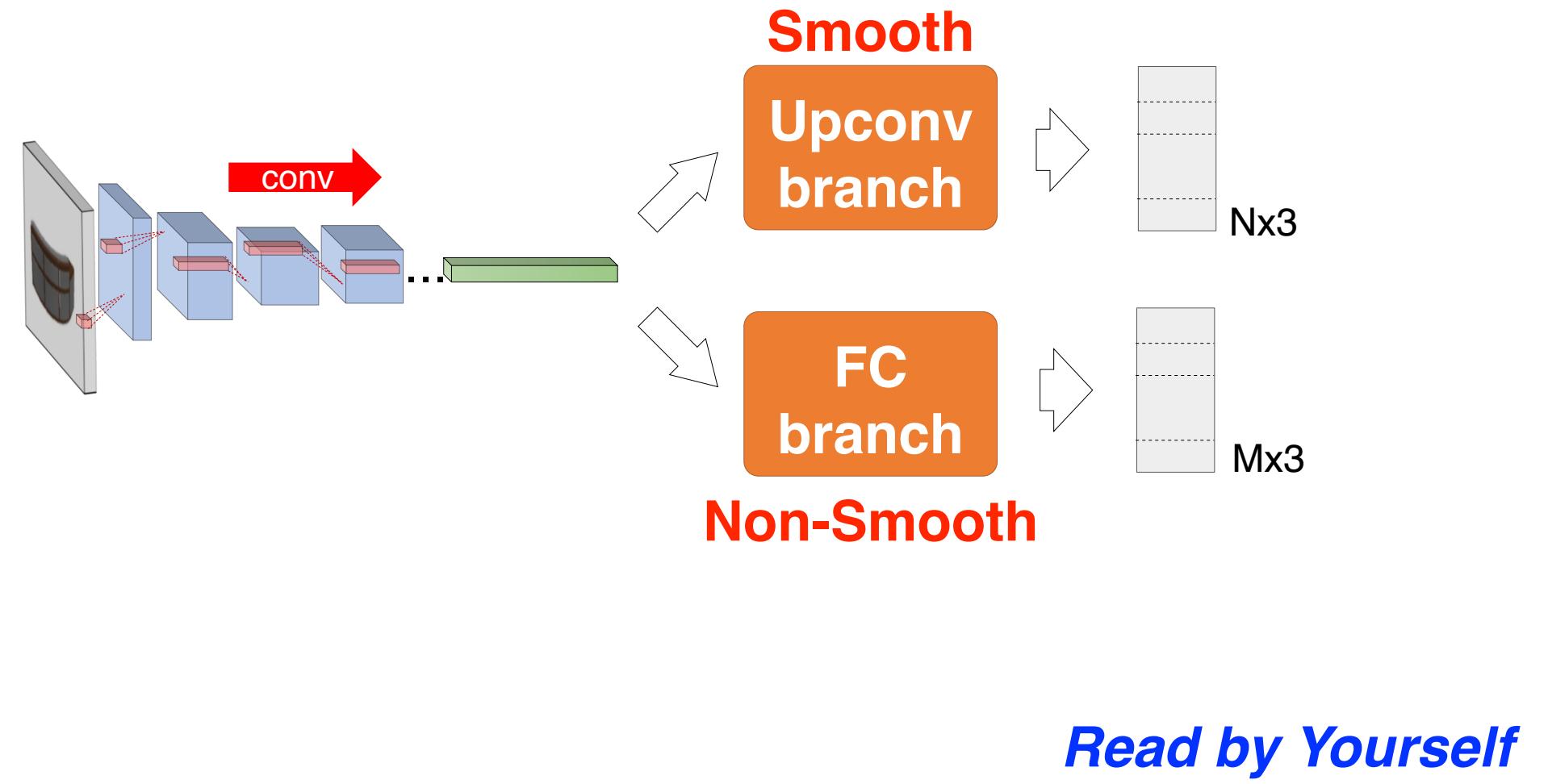
Read by Yourself

Two-Branch Architecture

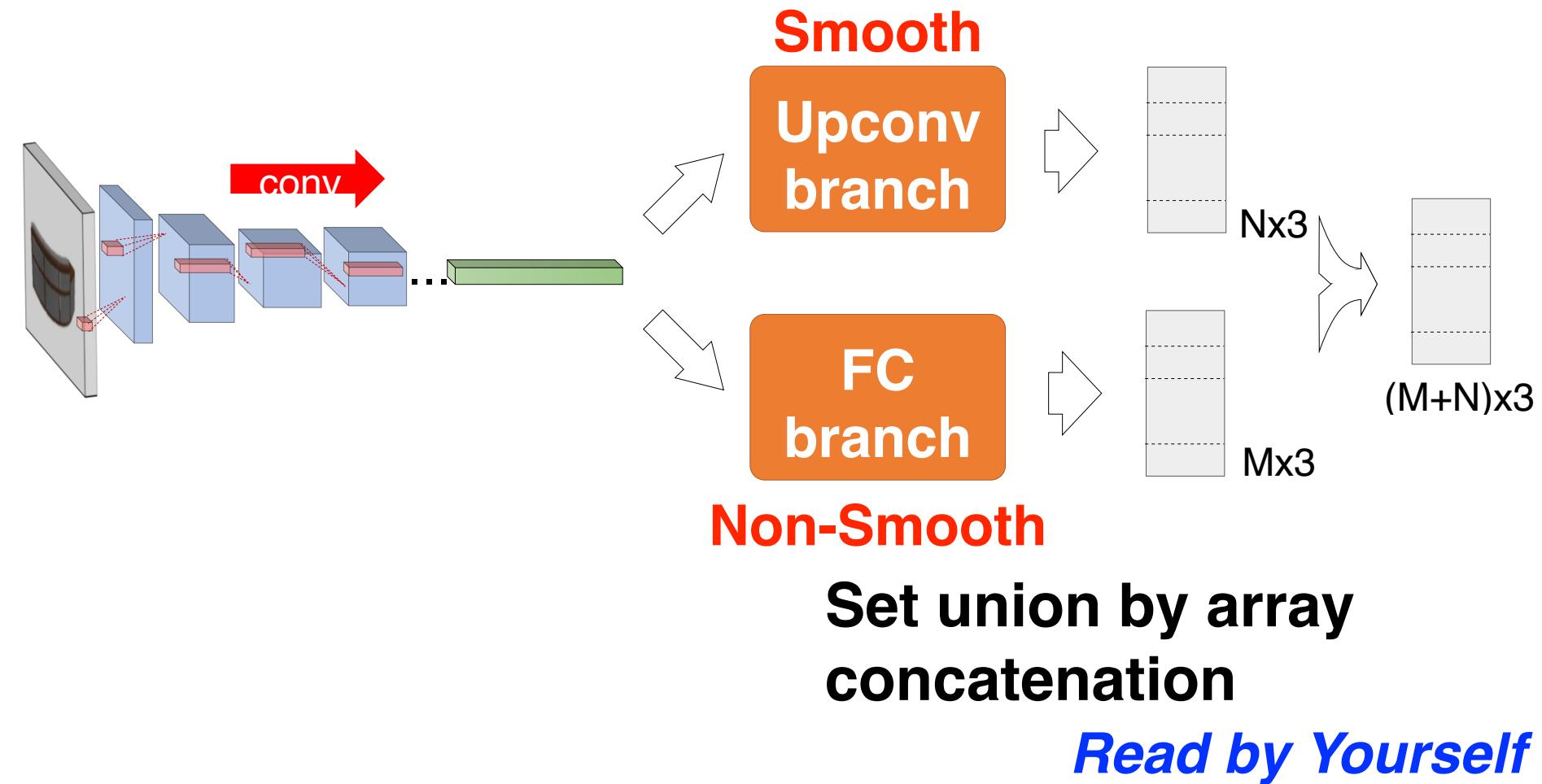


Read by Yourself

Two-Branch Architecture



Two-Branch Architecture

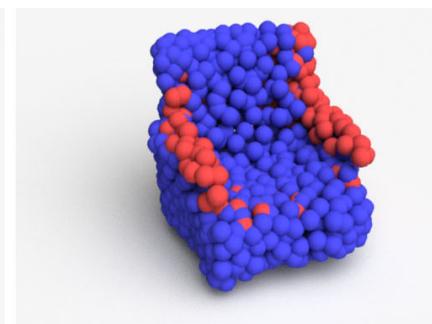
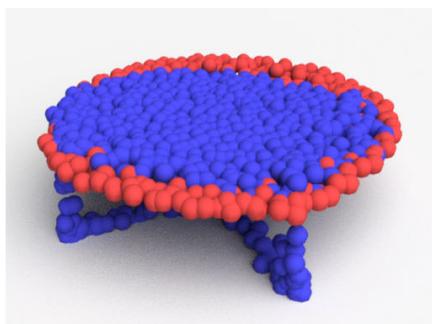


Which color corresponds to the upconv branch? FC branch?

Input

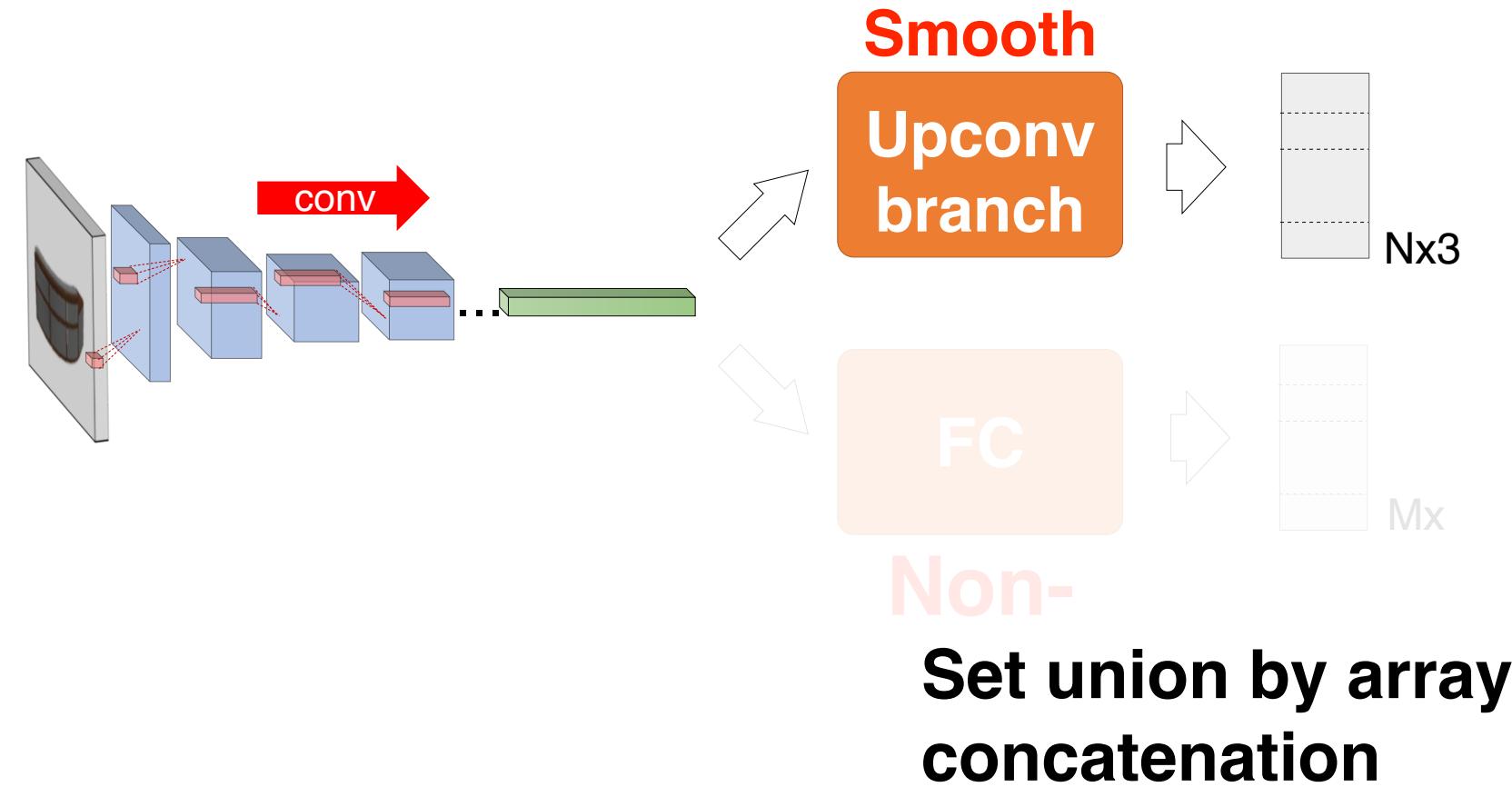


Prediction



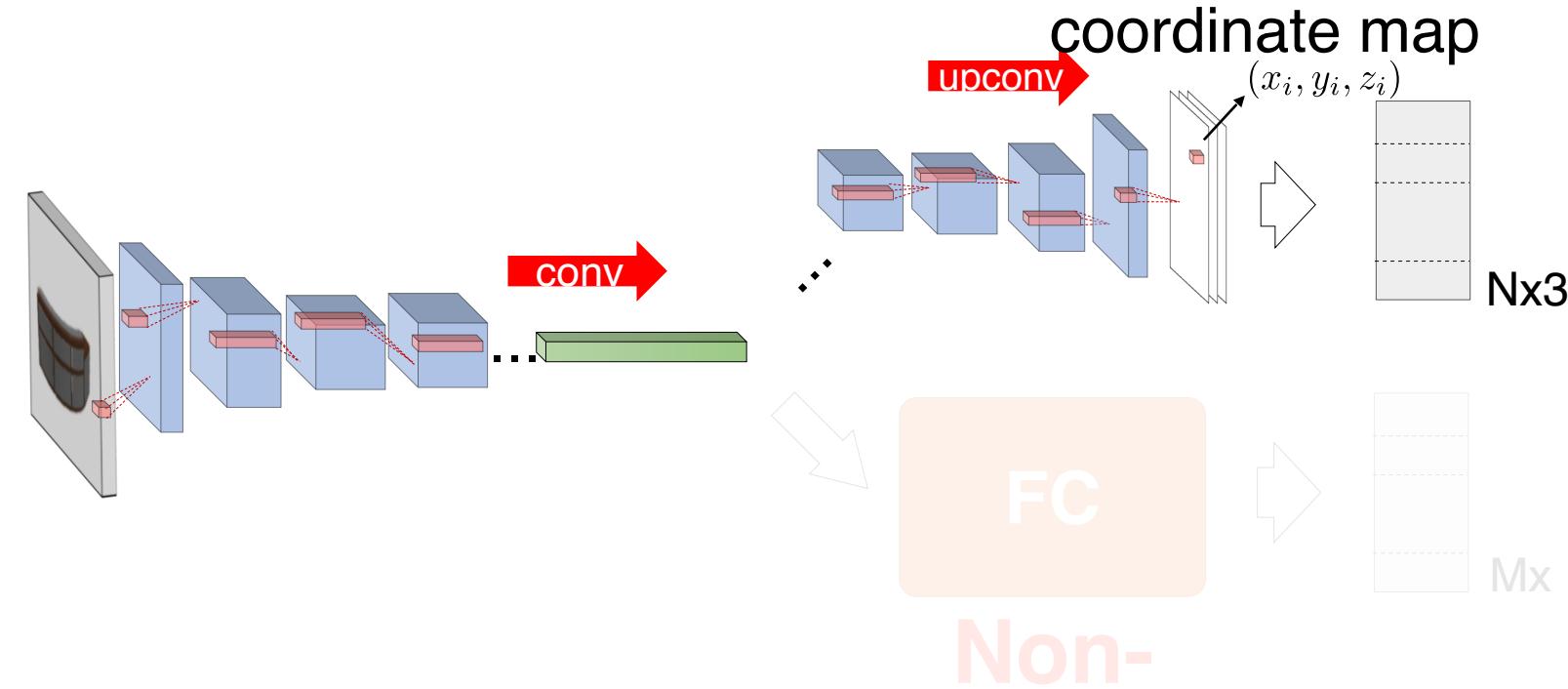
Read by Yourself

Design of Upconvolution Branch



Read by Yourself

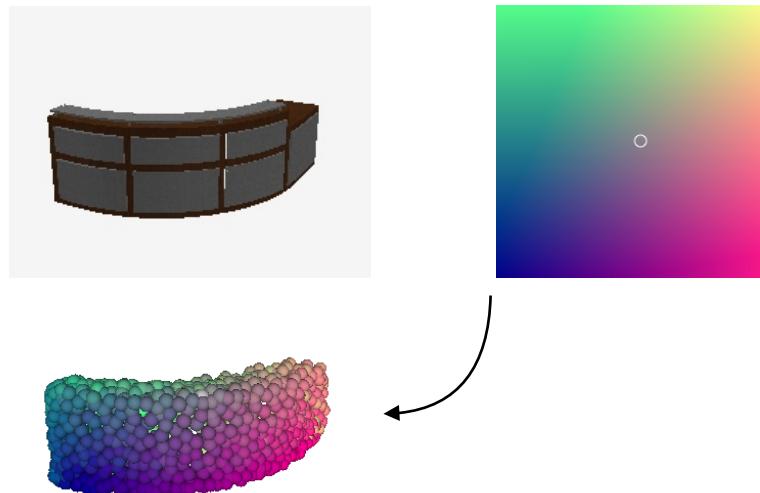
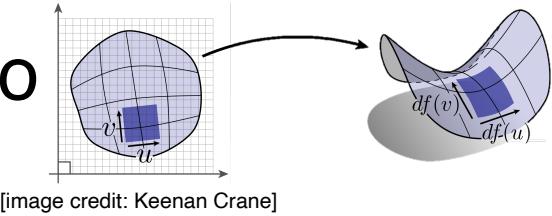
Design of Upconvolution Branch



Read by Yourself

Learns a Surface Parameterization

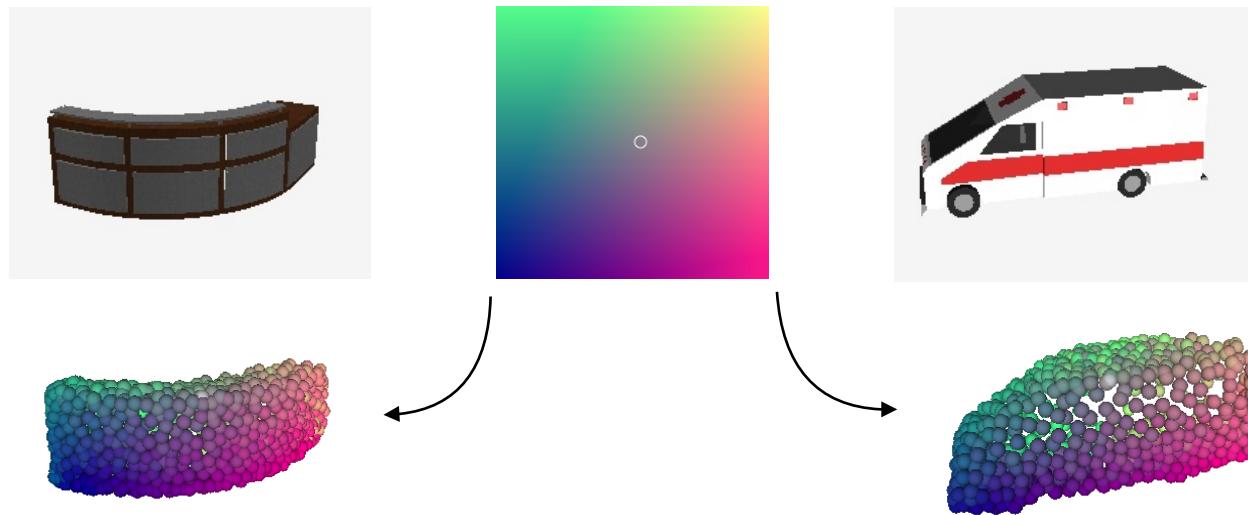
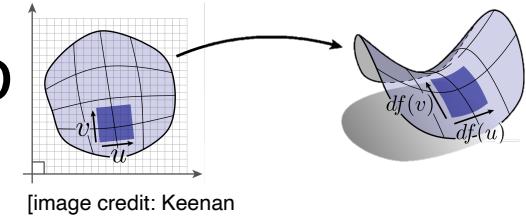
Smooth parameterization from 2D to



Read by Yourself

Learns a Surface Parameterization

Smooth parameterization from 2D to
Consistent across objects

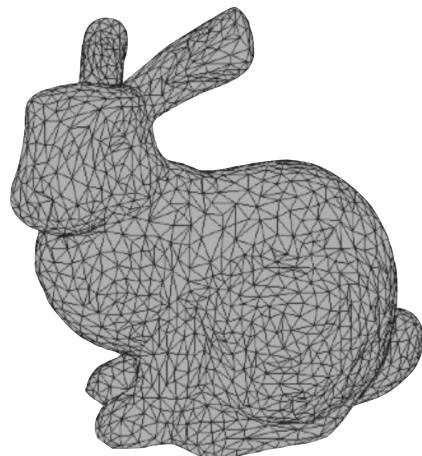


Read by Yourself

Image to Surfaces

Mesh Representation

- Previous point representation predicts only geometry without point connectivity.
- Mesh elements include mesh connectivity and mesh geometry $G = (V, E)$.



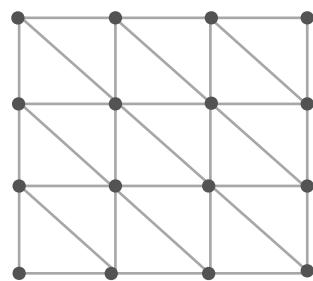
Mesh

Topology Ambiguity

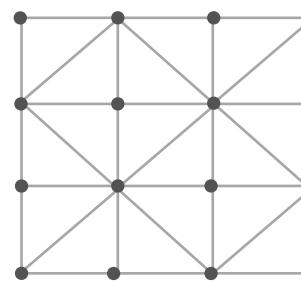
- Can we regress the vertices **and edges** from neural network?
 - Estimate vertices as a set of points. 
 - Estimate edges?

Designing Loss for Edge Prediction is Hard

- **Key observation:** given vertices, there are many possible ways to connect them and represent the same underlying surface:



$$G = (V, E)$$



$$G = (V, E')$$

Image → Intermediate Repr. → Mesh

- One option is to first build a high-resolution intermediate representation, and then convert the point cloud to mesh
- Intermediate representations:
 - Voxel
 - Implicit surface
 - Point cloud

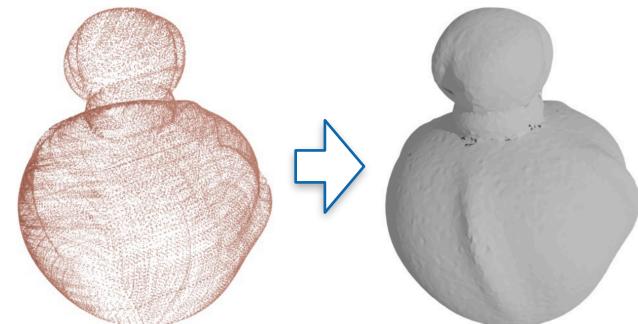
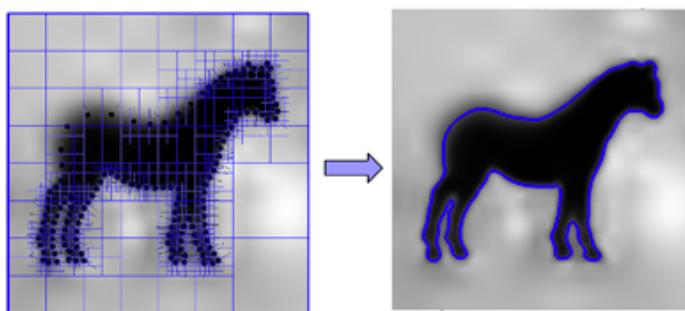
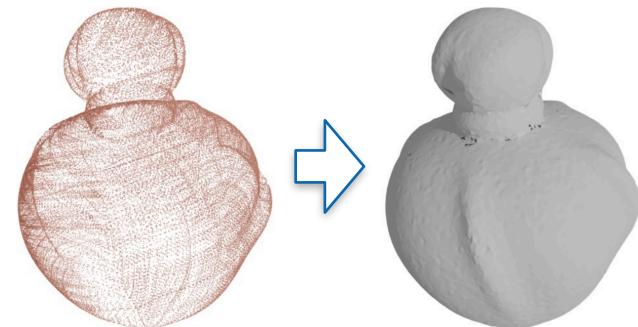
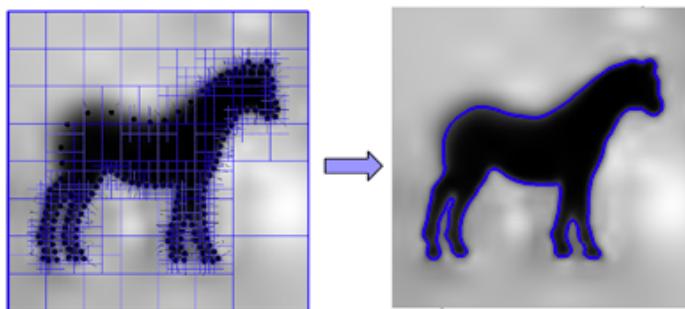


Image → Intermediate Repr. → Mesh

- One option is to first build a high-resolution intermediate representation, and then convert the point cloud to mesh
- Intermediate representations:
 - Voxel
 - Implicit surface
 - Point cloud

Defer to a later lecture!



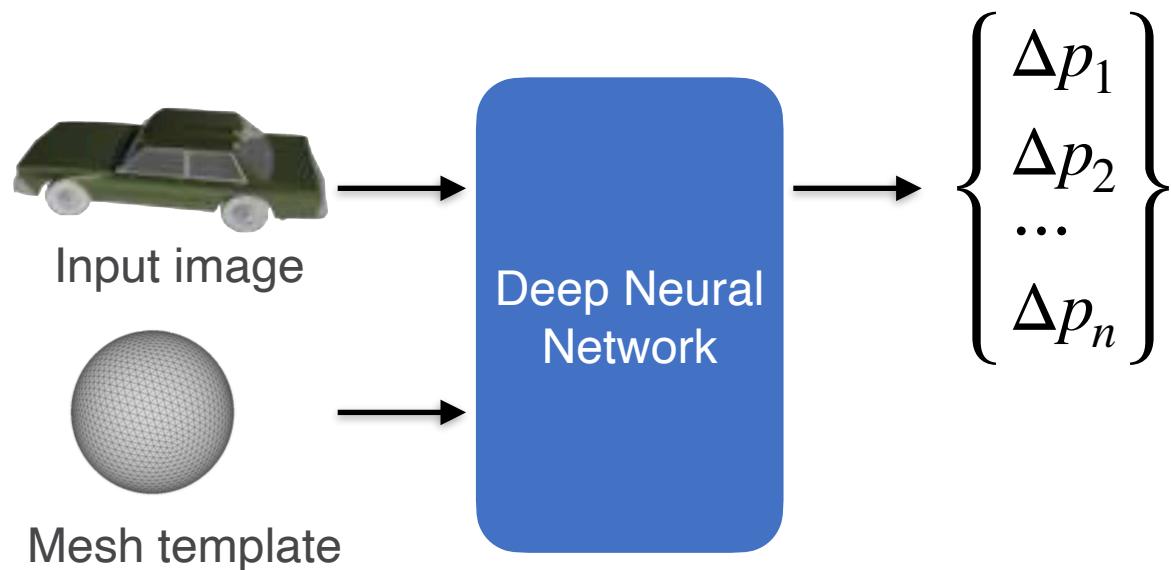
Editing-based Mesh Modeling

- Can we model mesh without predicting edges?

**Mesh Editing-based
Methods**

Editing-based Mesh Modeling

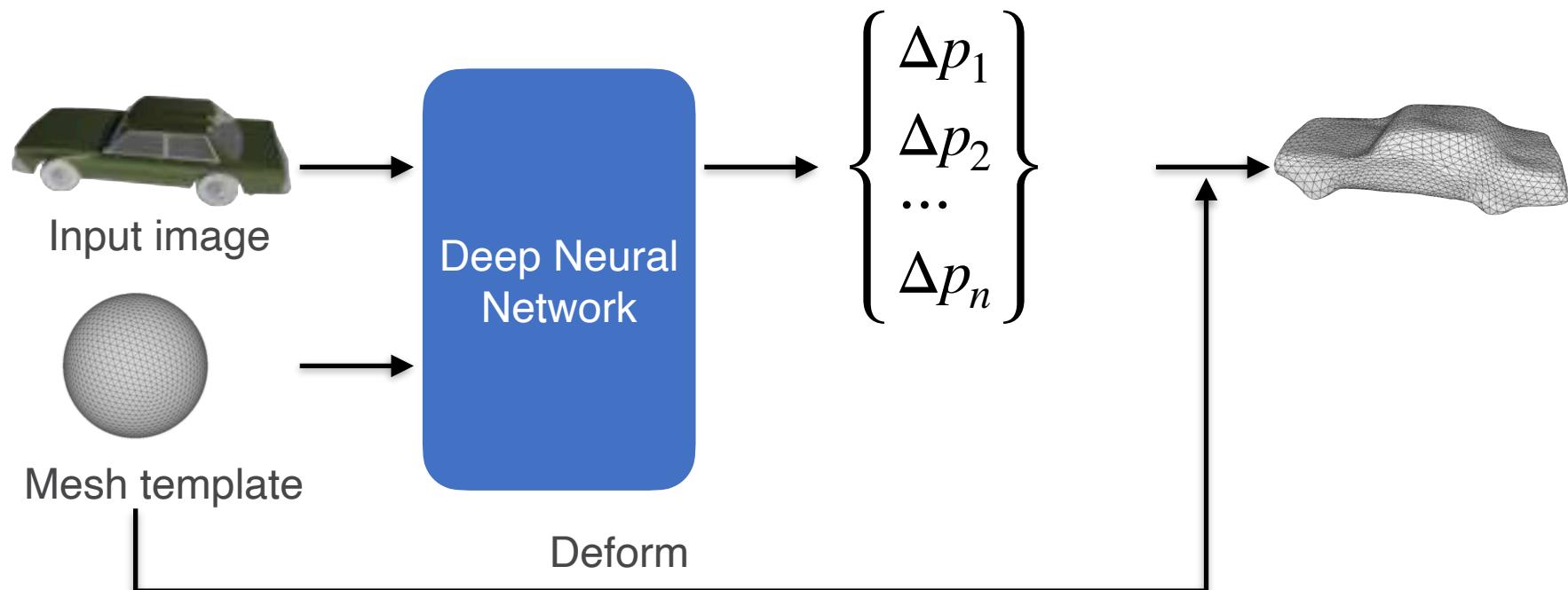
- Key idea: starting from an established mesh and modify it to become the target shape



Editing-based Mesh Modeling

- Key idea: starting from an established mesh and modify it to become the target shape

For example, deformation-based modeling:



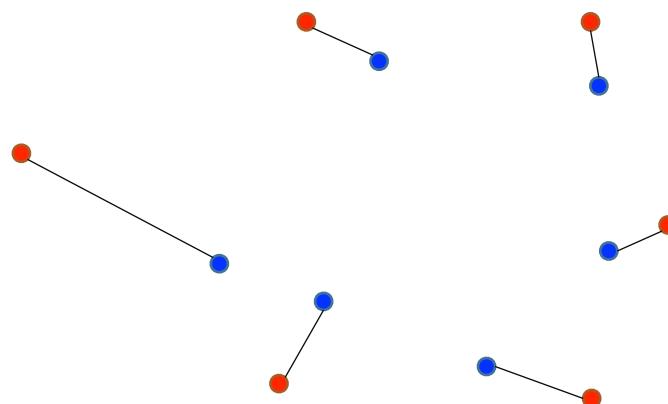
Losses for Mesh Editing

Some Example Losses

- Vertices distance.
 - Vertices point set distance.
- Uniform vertices distribution.
 - Edge length regularizer.
- Mesh surface smoothness.
- Normal Loss.

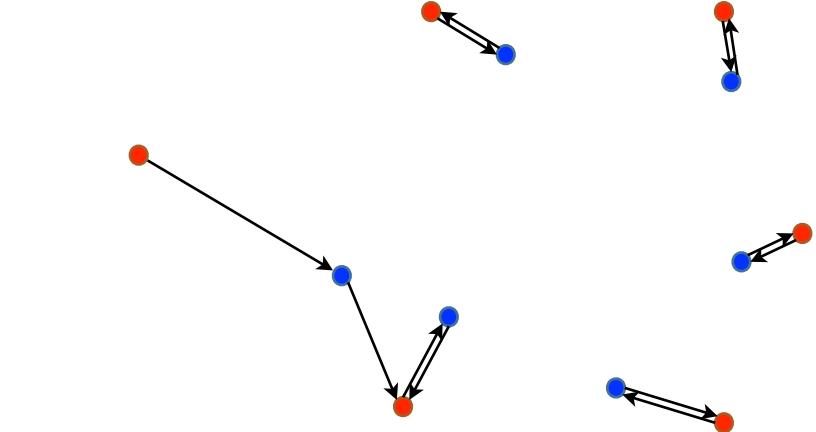
Loss I: Set Distance between Vertices

- Vertices are a set of points
- Recall the metrics for point clouds



Earth Mover's distance

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

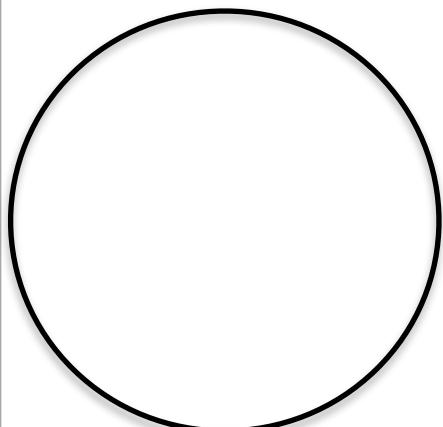


Chamfer distance

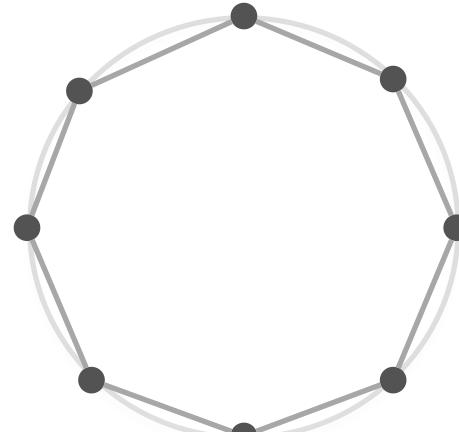
$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Loss II: Uniform Vertices Distribution

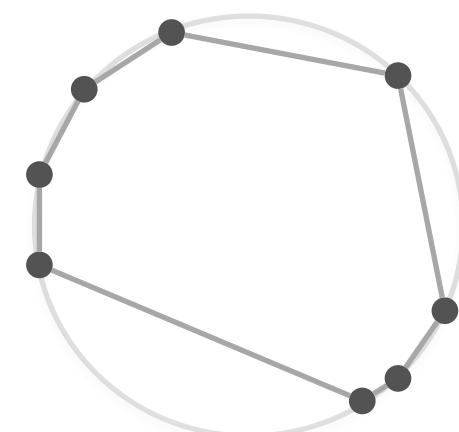
- However, vertices distance is not enough to guarantee mesh quality!



Surface



Mesh 1



Mesh 2

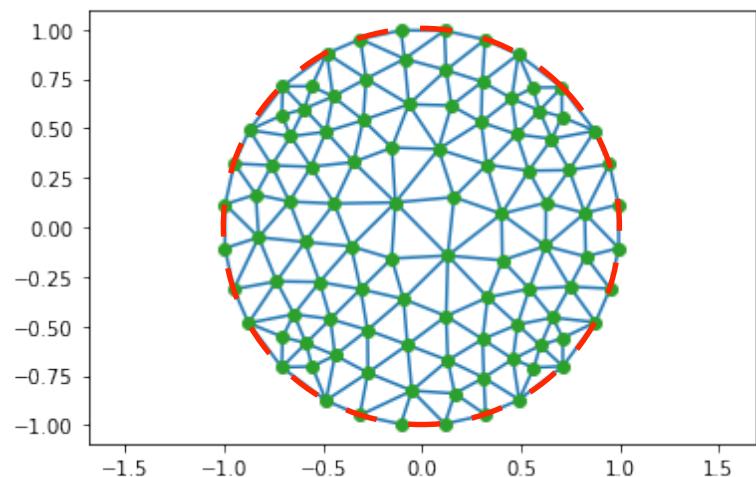
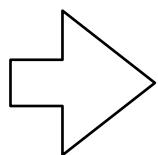
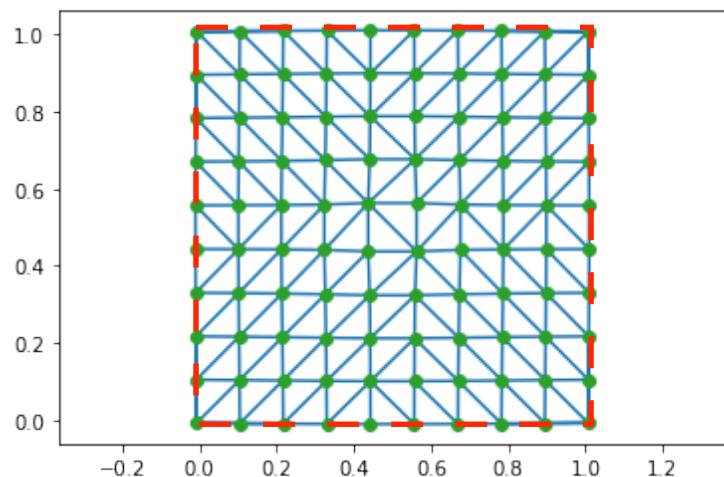
Loss II: Uniform Vertices Distribution

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

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

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

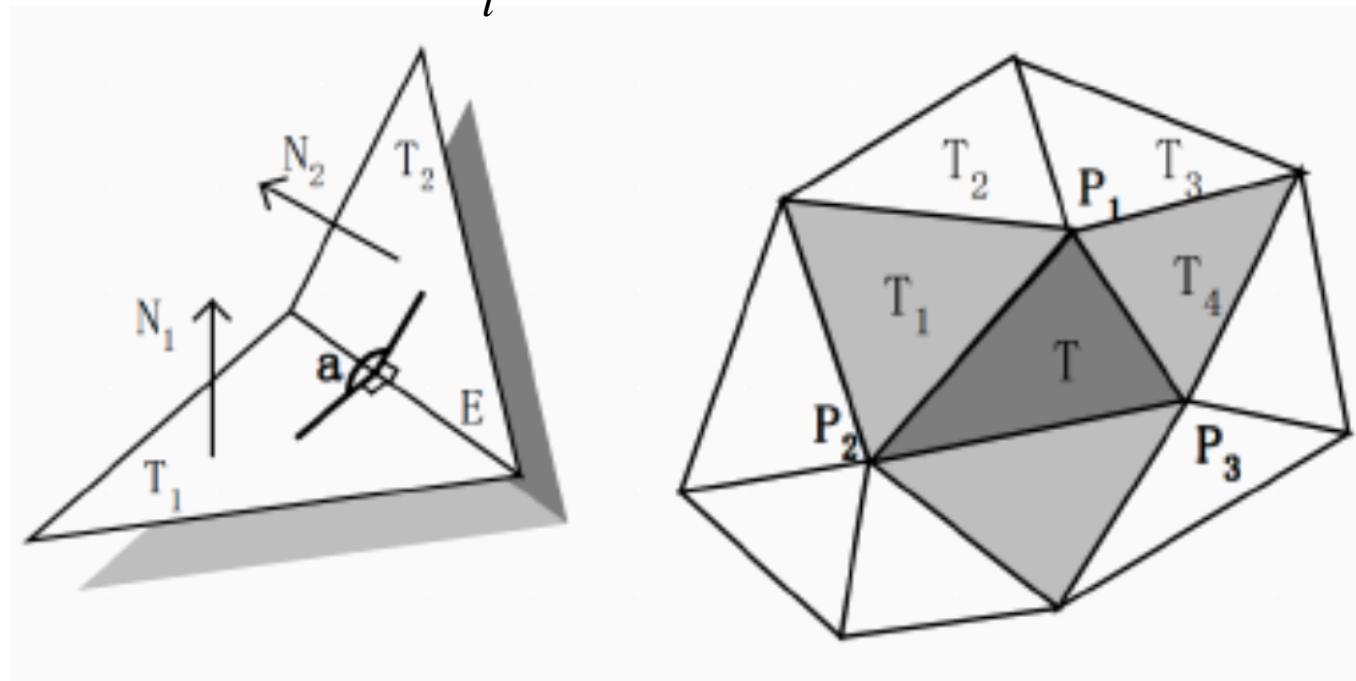
Effect of minimizing l when fixing topology and setting boundary points to the new positions



Loss III: Mesh Smoothness

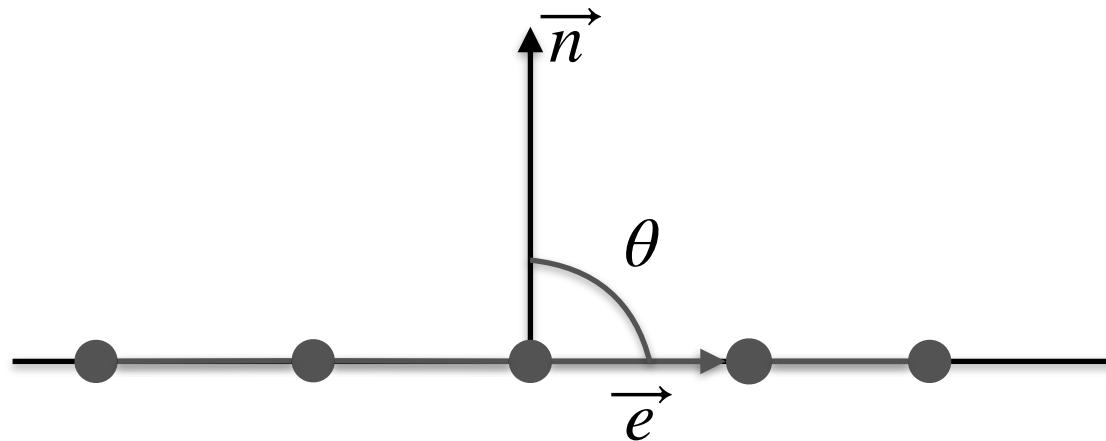
- L_{smooth} encourages that intersection angles of faces are close to 180 degrees.

$$L_{smooth} = \sum_i (\cos \theta_i + 1)^2$$



Loss IV: Normal Loss

- **Key assumption:** vertices within a local neighborhood lie on the same tangent plane.
- Regularize edge to be perpendicular to the underlying groundtruth surface normal

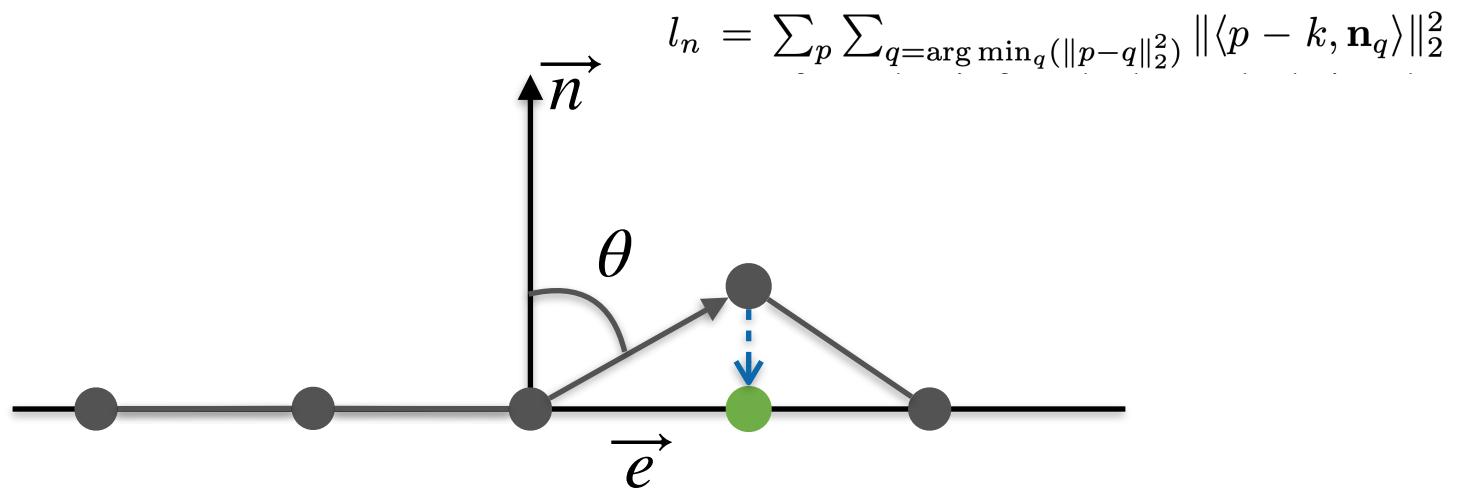


Loss IV: Normal Loss

- But how to find the vertices normal?
- One approach: use the nearest ground truth point normal as current vertex normal.

Loss IV: Normal Loss

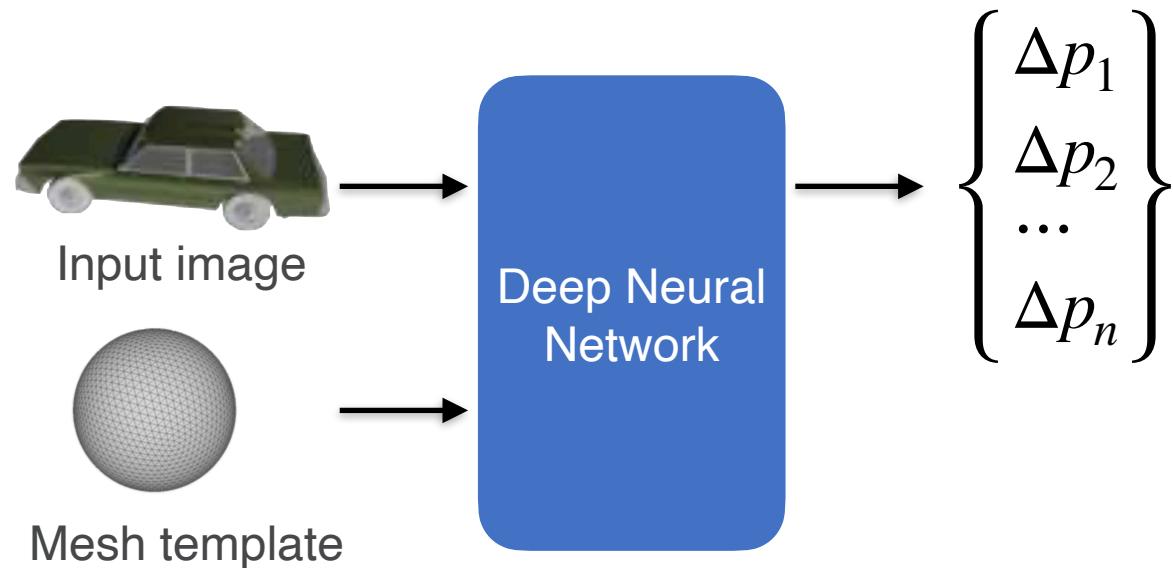
- But how to find the vertices normal?
- One approach: use the nearest ground truth point normal as current vertex normal.
- Penalize the edge direction to perpendicular to vertex normal.



Controlling Mesh Editing

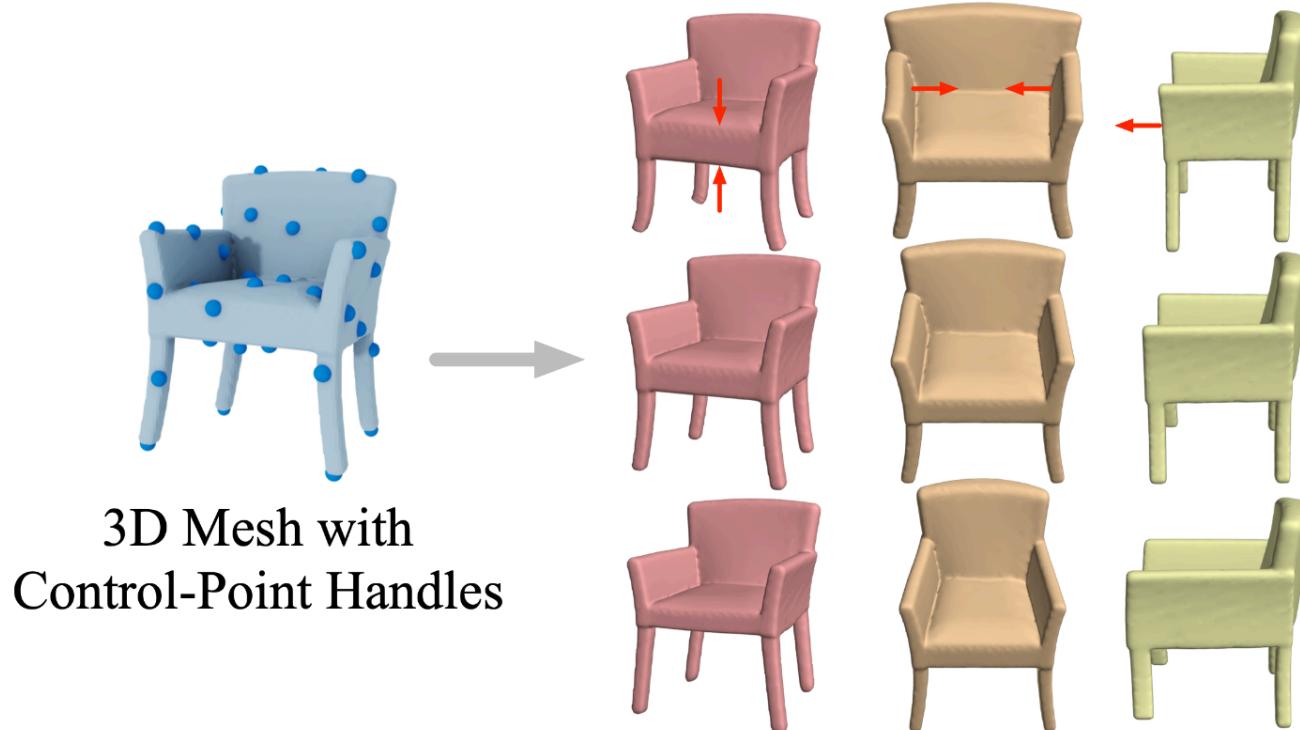
Deformation-based Mesh Modeling

- How do we parameterize movement of vertices of a template mesh?



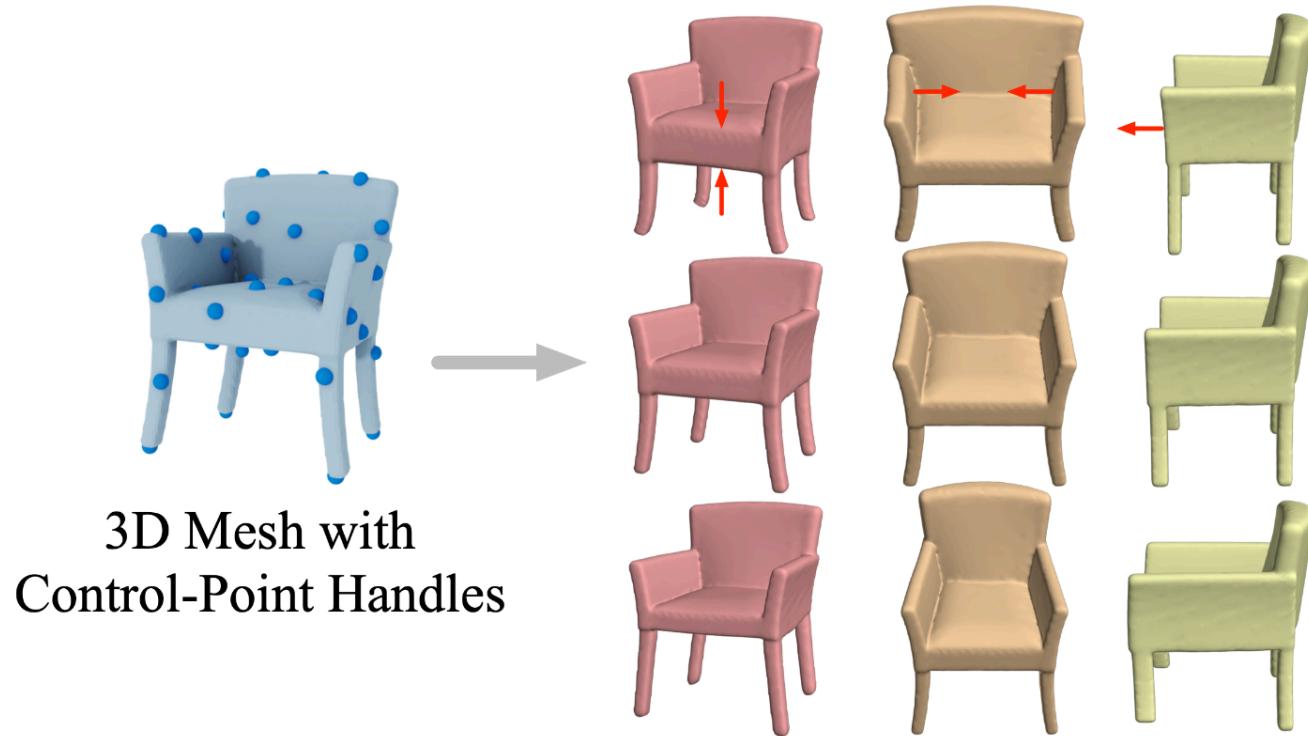
Challenges of Mesh Editing (I)

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



Possible Solution

- Deformation field

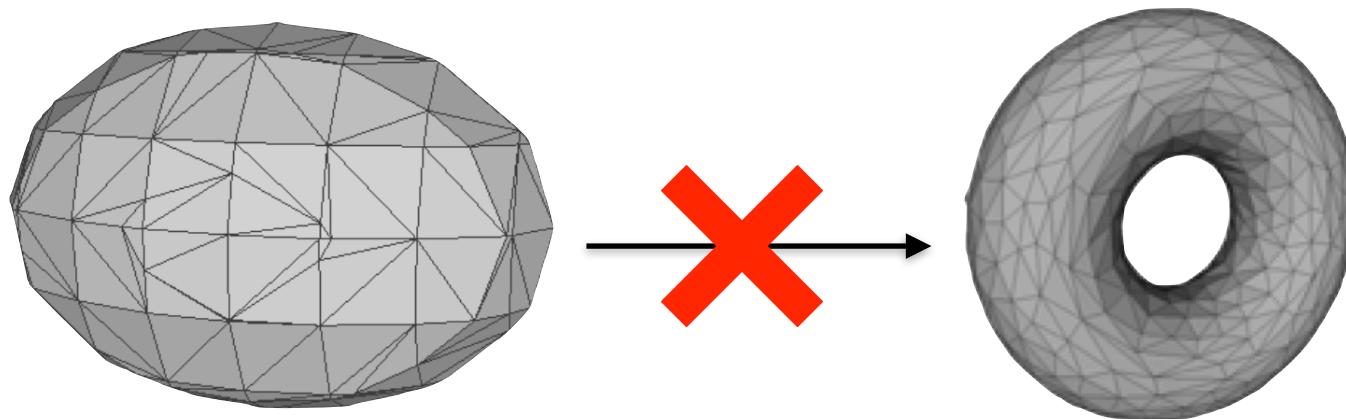


3D Mesh with
Control-Point Handles

Defer to later lecture

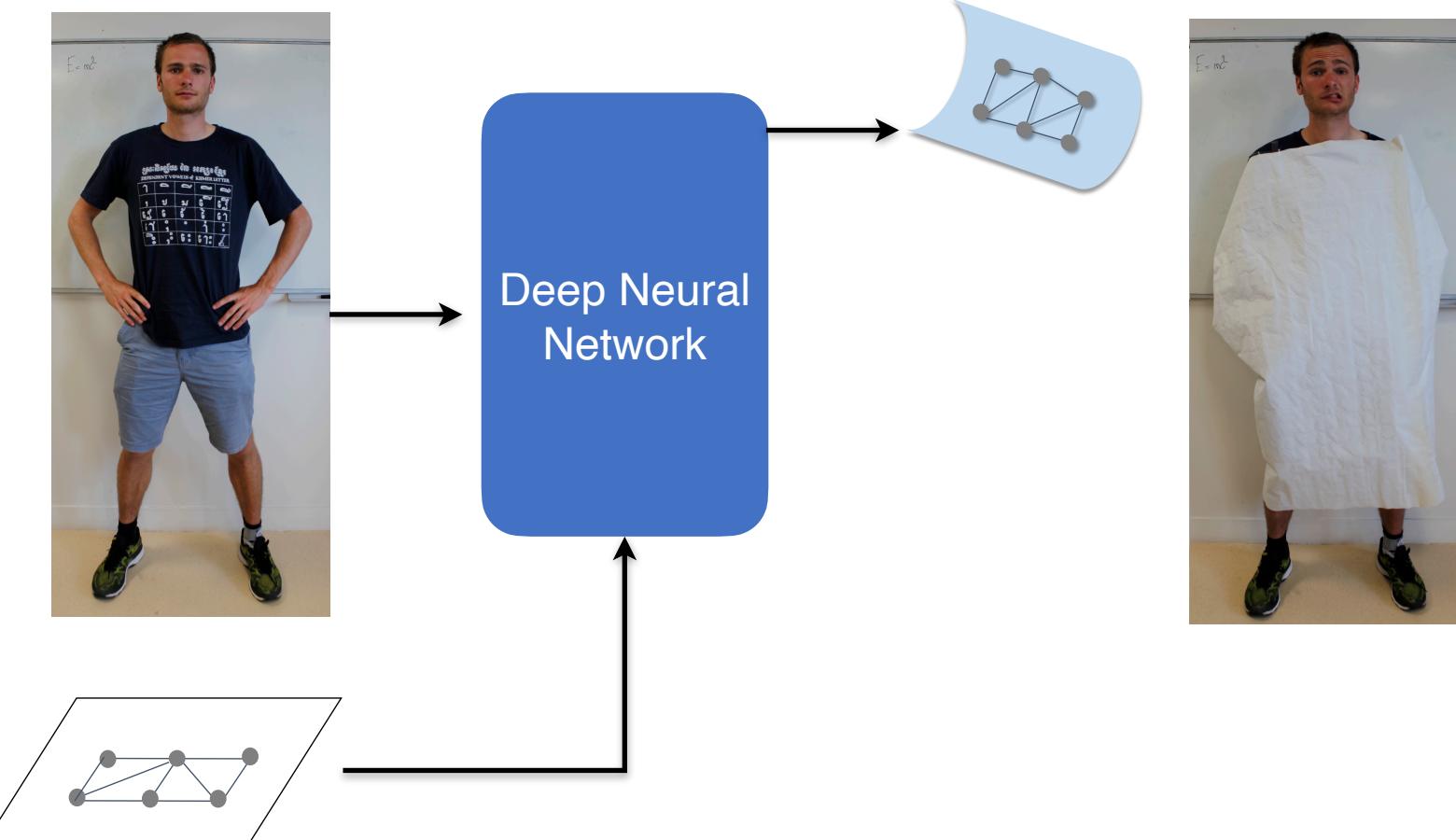
Challenges of Mesh Editing (II)

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



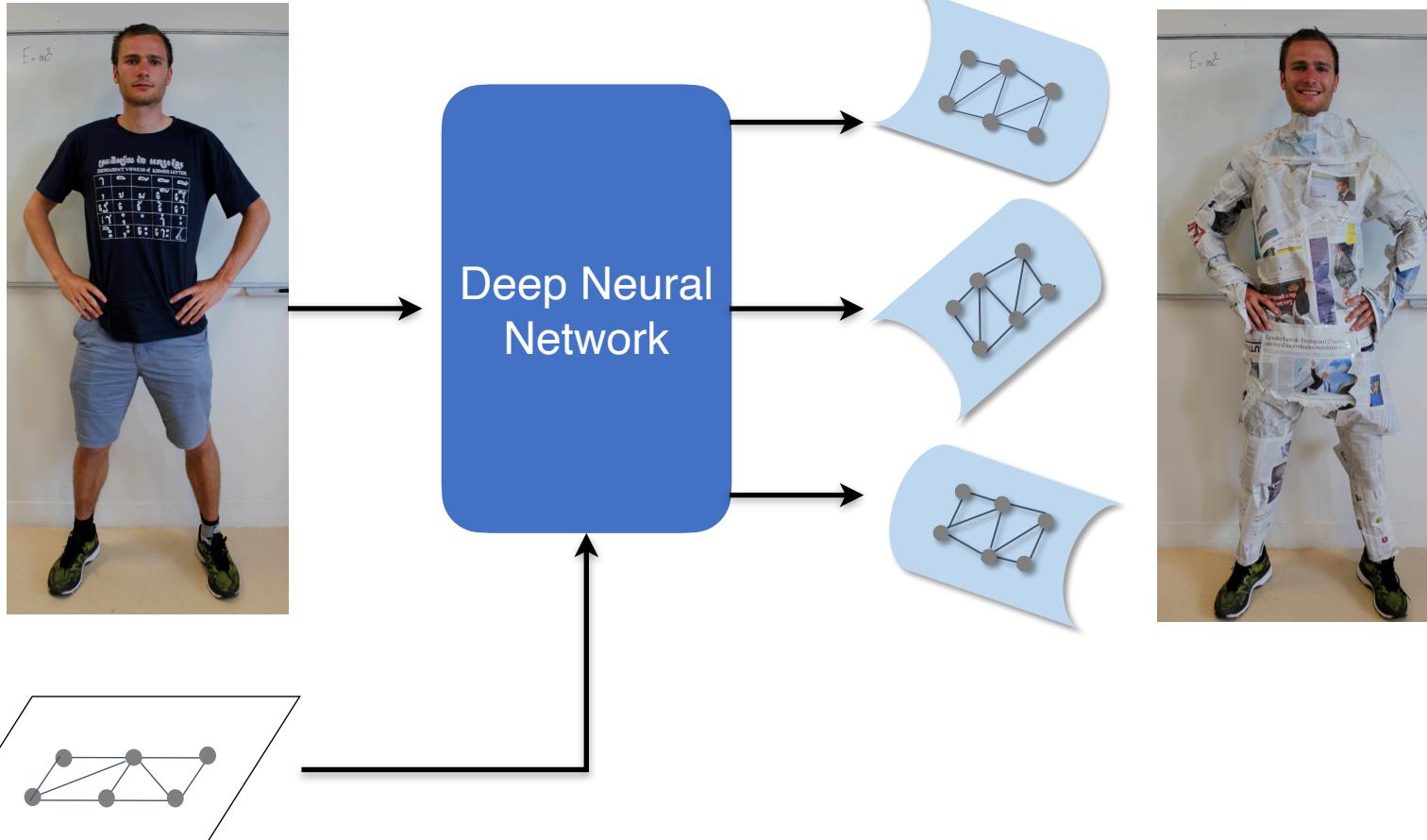
Possible Solutions

- Idea 1: Multiple template mesh.



Possible Solutions

- Idea 1: Multiple template mesh.



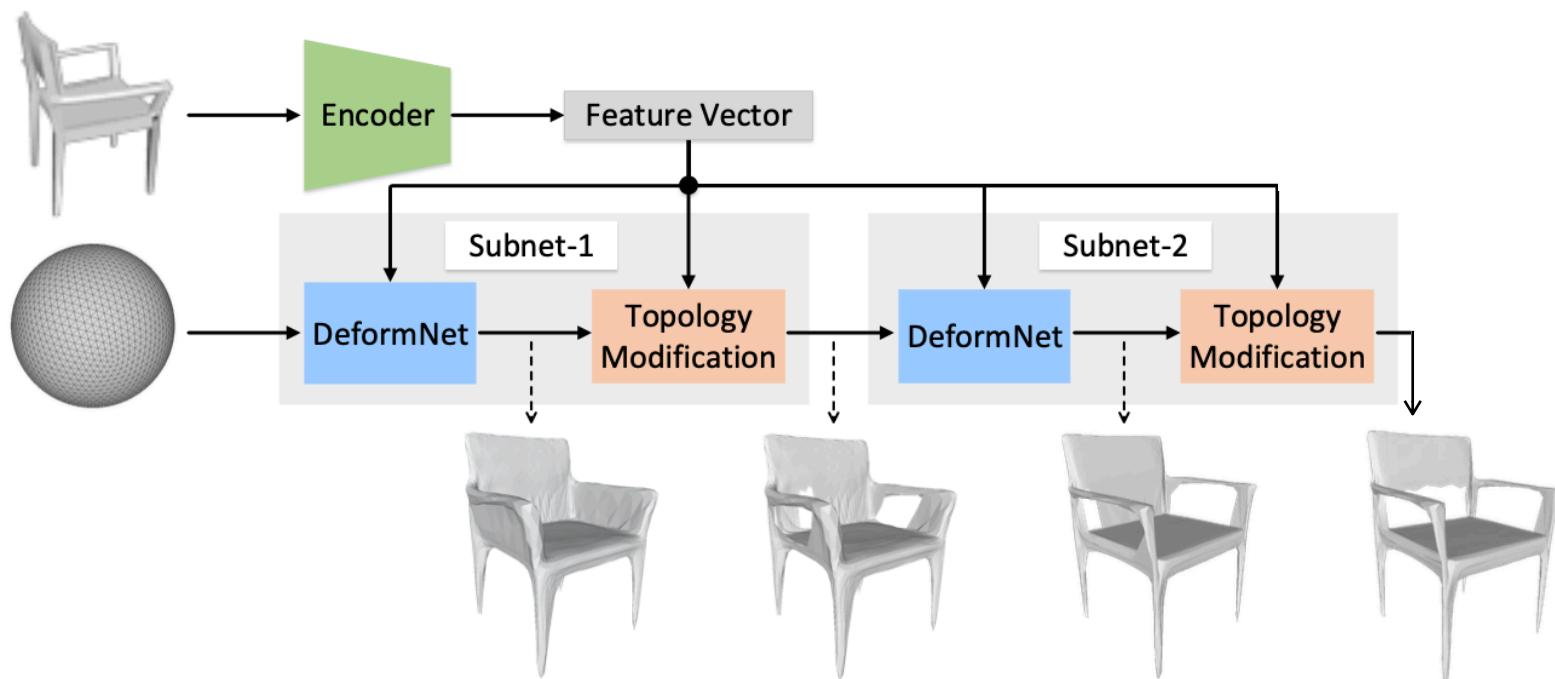
Possible Solutions

- 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.



Possible Solutions

- Idea 1: Multiple template mesh.
- Idea 2: Modify template topology by remove mesh faces.



Complex Mesh Topology

- Problems of modifying topology by removing 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

- Usually many losses have to be added
- Often have issues with local minimums in optimization
- Adding topology constraints is hard

Implicit Neural Representations

3D representations

- Voxel, point, and mesh representations \Rightarrow Discrete
- Implicit Neural Representation \Rightarrow Continuous

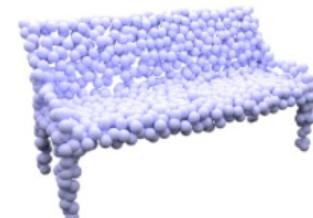
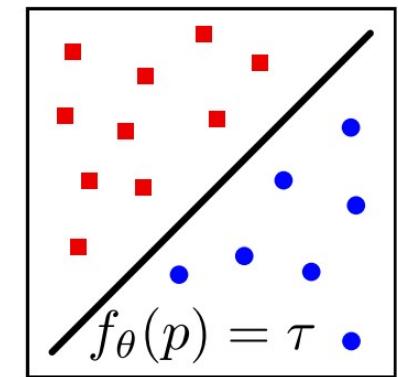
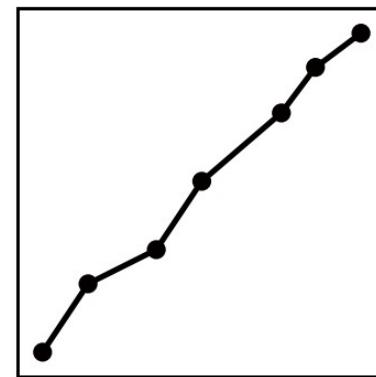
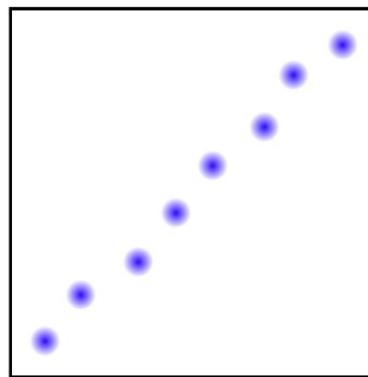
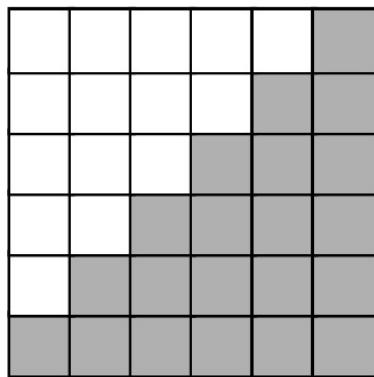


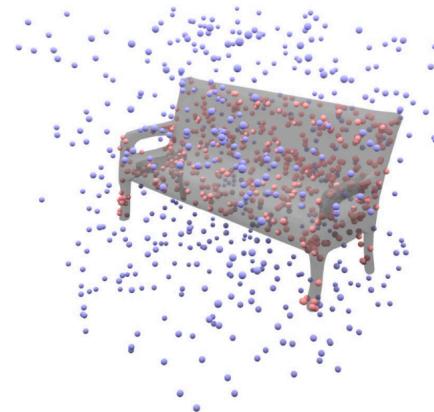
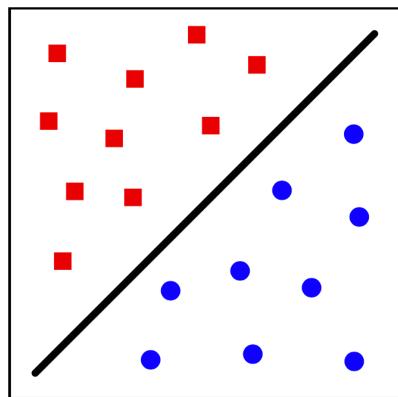
Figure from occupancy network.

Implicit Field Representation

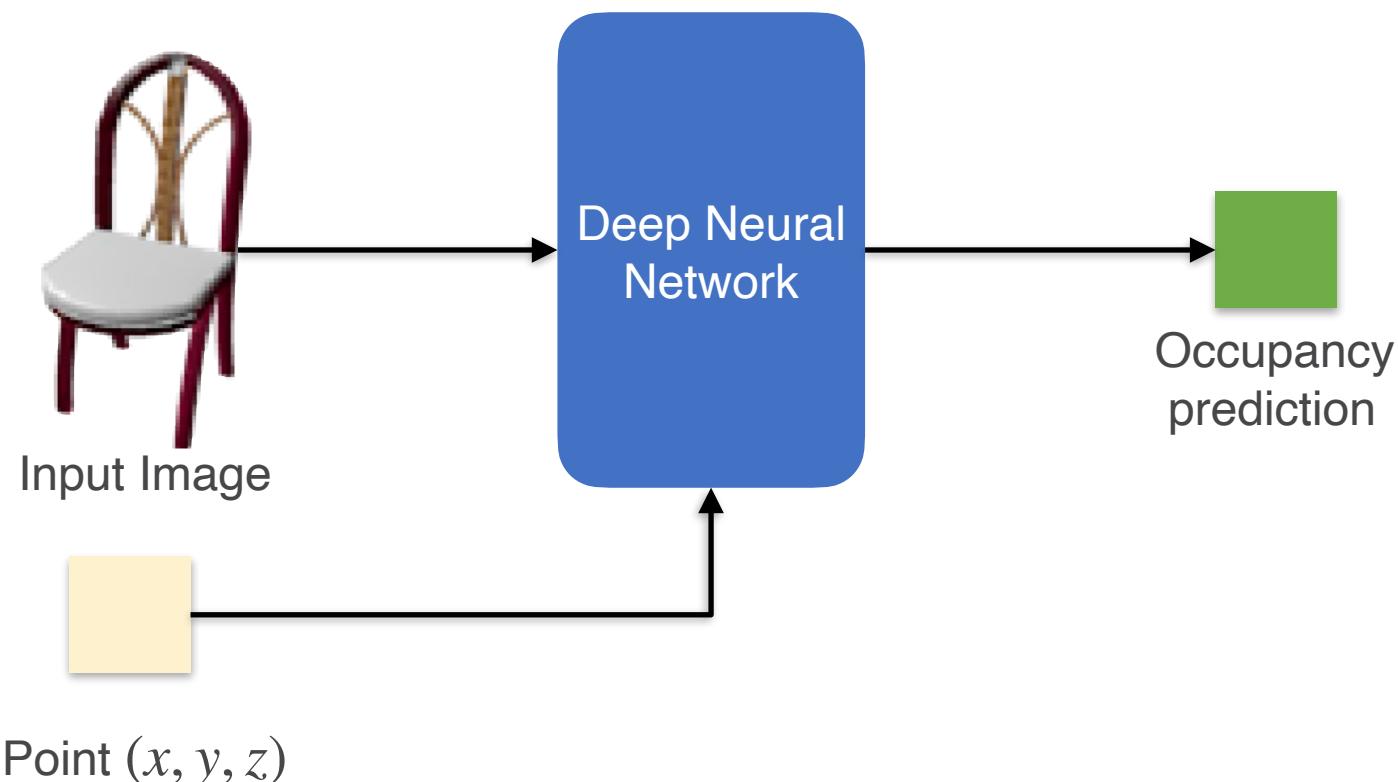
- **Key idea:** consider surface implicitly as decision boundary of a non-linear classifier:

$$f_{\theta} : \mathbb{R}^3 \times \mathcal{X} \rightarrow [0, 1]$$

↑ ↑ ↑
3D Location Condition (eg, Image) Occupancy Probability



Implicit Field Network Architecture



Limitation and Future Work

- Few-shot?
- How to model the distribution of multiple options of invisible area?
- How to add structure constraint? Example: model a shape with several holes.